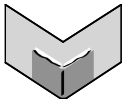
The Journal of Machine Learning Research
Volume 5 (2005)
Print-Archive Edition

Pages 1–800



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

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

The Journal of Machine Learning Research (JMLR) is an open access journal. All articles published in JMLR are freely available via electronic distribution. This Print-Archive Edition is published annually as a means of archiving the contents of the journal in perpetuity. The contents of this volume are articles published electronically in JMLR in late 2003 and 2004.

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

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

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

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

JMLR
Journal of Machine Learning Research Vol. 5 March 2005

Journal of Machine Learning Research 5 (2003) 1-25 Submitted 4/02; Revised 9/02; Published 12/03

Learning Rates for Q-learning

Eyal Even-Dar EVEND@CS.TAU.AC.IL
Yishay Mansour MANSOUR@CS.TAU.AC.IL
School of Computer Science
Tel-Aviv University
Tel-Aviv, 69978, Israel

Editor: Peter Bartlett

Abstract

In this paper we derive convergence rates for Q-learning. We show an interesting relationship between the convergence rate and the learning rate used in Q-learning. For a polynomial learning rate, one which is $1/t^{\omega}$ at time t where $\omega \in (1/2,1)$, we show that the convergence rate is polynomial in $1/(1-\gamma)$, where γ is the discount factor. In contrast we show that for a linear learning rate, one which is $1/t$ at time t, the convergence rate has an exponential dependence on $1/(1-\gamma)$. In addition we show a simple example that proves this exponential behavior is inherent for linear learning rates.

Keywords: Reinforcement Learning, Q-Learning, Stochastic Processes, Convergence Bounds, Learning Rates.

1. Introduction

In Reinforcement Learning, an agent wanders in an unknown environment and tries to maximize its long term return by performing actions and receiving rewards. The challenge is to understand how a current action will affect future rewards. A good way to model this task is with Markov Decision Processes (MDP), which have become the dominant approach in Reinforcement Learning (Sutton and Barto, 1998, Bertsekas and Tsitsiklis, 1996).

An MDP includes states (which abstract the environment), actions (which are the available actions to the agent), and for each state-action pair, a distribution of next states (the states reached after performing the action in the given state). In addition there is a reward function that assigns a stochastic reward for each state and action. The *return* combines a sequence of rewards into a single value that the agent tries to optimize. A *discounted return* has a parameter $\gamma \in (0,1)$ where the reward received at step k is discounted by γ^k.

One of the challenges of Reinforcement Learning is when the MDP is not known, and we can only observe the trajectory of states, actions and rewards generated by the agent wandering in the MDP. There are two basic conceptual approaches to the learning problem. The first is model based, where we first reconstruct a model of the MDP, and then find an optimal policy for the approximate model. The second approach is implicit methods that update the information after each step, and based on this derive an estimate to the optimal policy. The most popular of those methods is Q-learning (Watkins, 1989).

Q-learning is an off-policy method that can be run on top of any strategy wandering in the MDP. It uses the information observed to approximate the optimal function, from which one can

construct the optimal policy. There are various proofs that Q-learning does converge to the optimal Q function, under very mild conditions (Bertsekas and Tsitsiklis, 1996, Tsitsiklis, 1994, Watkins and Dyan, 1992, Littman and Szepesvári, 1996, Jaakkola et al., 1994, Borkar and Meyn, 2000). The conditions have to do with the exploration policy and the learning rate. For the exploration one needs to require that each state action be performed infinitely often. The learning rate controls how fast we modify our estimates. One expects to start with a high learning rate, which allows fast changes, and lowers the learning rate as time progresses. The basic conditions are that the sum of the learning rates goes to infinity (so that any value could be reached) and that the sum of the squares of the learning rates is finite (which is required to show that the convergence is with probability one).

We build on the proof technique of Bertsekas and Tsitsiklis (1996), which is based on convergence of stochastic iterative algorithms, to derive convergence rates for Q-learning. We study two models of updating in Q-learning. The first is the synchronous model, where all state action pairs are updated simultaneously. The second is the asynchronous model, where at each step we update a single state action pair. We distinguish between two sets of learning rates. The most interesting outcome of our investigation is the relationship between the form of the learning rates and the rate of convergence. A linear learning rate is of the form $1/t$ at time t, and a polynomial learning rate, which is of the form $1/t^{\omega}$, where $\omega \in (1/2,1)$ is a parameter.

We show for synchronous models that for a polynomial learning rate the convergence rate is polynomial in $1/(1-\gamma)$, while for a linear learning rate the convergence rate is exponential in $1/(1-\gamma)$. We also describe an MDP that has exponential behavior for a linear learning rate. The lower bound simply shows that if the initial value is one and all the rewards are zero, it takes $O((1/\varepsilon)^{1/(1-\gamma)})$ updates, using a linear learning rate, until we reach a value of ε.

The different behavior might be explained by the asymptotic behavior of $\sum_t \alpha_t$, one of the conditions that ensure that Q-learning converges from any initial value. In the case of a linear learning rate we have that $\sum_{t=1}^{T} \alpha_t = O(\ln(T))$, whereas using polynomial learning rate it behaves as $O(T^{1-\omega})$. Therefore, using polynomial learning rate each value can be reached by polynomial number of steps and using linear learning rate each value requires exponential number of steps.

The convergence rate of Q-learning in a batch setting, where many samples are averaged for each update, was analyzed by Kearns and Singh (1999). A batch setting does not have a learning rate and has much of the flavor of model based techniques, since each update is an average of many samples. A run of a batch Q-learning is divided into phases, at the end of each phase an update is made. The update after each phase is reliable since it averages many samples.

The convergence of Q-learning with linear learning rate was studied by Szepesvari (1998) for special MDPs, where the next state distribution is the same for each state. (This setting is much closer to the PAC model, since there is no influence between the action performed and the states reached, and the states are i.i.d distributed). For this model Szepesvari (1998) shows a convergence rate, which is exponential in $1/(1-\gamma)$. Beleznay et al. (1999) give an exponential lower bound in the number of the states for undiscounted return.

2. The Model

We define a Markov Decision process (MDP) as follows

Definition 1 *A Markov Decision process (MDP) M is a 4-tuple (S,U,P,R), where S is a set of the states, U is a set of actions ($U(i)$ is the set of actions available at state i), $P_{i,j}^{M}(a)$ is the transition*

probability from state i to state j when performing action $a \in U(i)$ in state i, and $R_M(s,a)$ is the reward received when performing action a in state s.

We assume that $R_M(s,a)$ is non-negative and bounded by R_{max}, i.e., $\forall s,a: \ 0 \leq R_M(s,a) \leq R_{max}$. For simplicity we assume that the reward $R_M(s,a)$ is deterministic, however all our results apply when $R_M(s,a)$ is stochastic.

A strategy for an MDP assigns, at each time t, for each state s a probability for performing action $a \in U(s)$, given a history $F_{t-1} = \{s_1, a_1, r_1, ..., s_{t-1}, a_{t-1}, r_{t-1}\}$ which includes the states, actions and rewards observed until time $t-1$. A policy is memory-less strategy, i.e., it depends only on the current state and not on the history. A deterministic policy assigns each state a unique action.

While following a policy π we perform at time t action a_t at state s_t and observe a reward r_t (distributed according to $R_M(s,a)$), and the next state s_{t+1} (distributed according to $P^M_{s_t,s_{t+1}}(a_t)$). We combine the sequence of rewards to a single value called the return, and our goal is to maximize the return. In this work we focus on *discounted return*, which has a parameter $\gamma \in (0,1)$, and the discounted return of policy π is $V^\pi_M = \sum_{t=0}^{\infty} \gamma^t r_t$, where r_t is the reward observed at time t. Since all the rewards are bounded by R_{max} the discounted return is bounded by $V_{max} = \frac{R_{max}}{1-\gamma}$.

We define a value function for each state s, under policy π, as $V^\pi_M(s) = E[\sum_{i=0}^{\infty} r_i \gamma^i]$, where the expectation is over a run of policy π starting at state s. We define a state-action value function $Q^\pi_M(s,a) = R_M(s,a) + \gamma \sum_{\bar{s}} P^M_{s,\bar{s}}(a) V^\pi_M(\bar{s})$, whose value is the return of initially performing action a at state s and then following policy π. Since $\gamma < 1$ we can define another parameter $\beta = (1-\gamma)/2$, which will be useful for stating our results. (Note that as β decreases V_{max} increases.)

Let π^* be an optimal policy, which maximizes the return from any start state. (It is well known that there exists an optimal strategy, which is a deterministic policy (Puterman., 1994).) This implies that for any policy π and any state s we have $V^{\pi^*}_M(s) \geq V^\pi_M(s)$, and $\pi^*(s) = argmax_a(R_M(s,a) + \gamma(\sum_{s'} P^M_{s,s'}(a) \max_b Q(s',b))$. The optimal policy is also the only fixed point of the operator, $(TQ)(s,a) = R_M(s,a) + \gamma \sum_{s'} P_{s,s'}(a) \max_b Q(s',b)$. We use V^*_M and Q^*_M for $V^{\pi^*}_M$ and $Q^{\pi^*}_M$, respectively. We say that a policy π is an ε-approximation of the optimal policy if $\|V^*_M - V^\pi_M\|_\infty \leq \varepsilon$.

For a sequence of state-action pairs let the *covering time*, denoted by L, be an upper bound on the number of state-action pairs starting from any pair, until all state-action appear in the sequence. Note that the covering time can be a function of both the MDP and the sequence or just of the sequence. Initially we assume that from any start state, within L steps all state-action pairs appear in the sequence. Later, we relax the assumption and assume that with probability at least $\frac{1}{2}$, from any start state in L steps all state-action appear in the sequence. In this paper, the underlying policy generates the sequence of state action pairs.

The Parallel Sampling Model, $PS(M)$, as was introduced by Kearns and Singh (1999). The $PS(M)$ is an ideal exploration policy. A single call to $PS(M)$ returns for every pair (s,a) the next state s', distributed according to $P^M_{s,s'}(a)$ and a reward r distributed according to $R_M(s,a)$. The advantage of this model is that it allows to ignore the exploration and to focus on the learning. In some sense $PS(M)$ can be viewed as a perfect exploration policy.

Notations: The notation $g = \tilde{\Omega}(f)$ implies that there are constants c_1 and c_2 such that $g \geq c_1 f \ln^{c_2}(f)$. All the norms $\|\cdot\|$, unless otherwise specified, are L_∞ norms, i.e., $\|(x_1,\ldots,x_n)\| = \max_i x_i$.

3. Q-learning

The Q-learning algorithm (Watkins, 1989) estimates the state-action value function (for discounted return) as follows:

$$Q_{t+1}(s,a) = (1-\alpha_t(s,a))Q_t(s,a) + \alpha_t(s,a)(R_M(s,a) + \gamma \max_{b \in U(s')} Q_t(s',b)), \tag{1}$$

where s' is the state reached from state s when performing action a at time t. Let $T^{s,a}$ be the set of times, where action a was performed at state s, then $\alpha_t(s,a) = 0$ for $t \notin T^{s,a}$. It is known that Q-learning converges to Q^* if each state action pair is performed infinitely often and $\alpha_t(s,a)$ satisfies for each (s,a) pair: $\sum_{t=1}^{\infty} \alpha_t(s,a) = \infty$ and $\sum_{t=1}^{\infty} \alpha_t^2(s,a) < \infty$ (Bertsekas and Tsitsiklis, 1996, Tsitsiklis, 1994, Watkins and Dyan, 1992, Littman and Szepesvári, 1996, Jaakkola et al., 1994).

Q-learning is an asynchronous process in the sense that it updates a single entry each step. Next we describe two variants of Q-learning, which are used in the proofs. The first algorithm is *synchronous Q-learning*, which performs the updates by using the $PS(M)$. Specifically:

$$\begin{aligned} \forall s,a \quad &: \quad Q_0(s,a) = C \\ \forall s,a \quad &: \quad Q_{t+1}(s,a) = (1-\alpha_t^{\omega})Q_t(s,a) + \alpha_t^{\omega}(R_M(s,a) + \gamma \max_{b \in U(\bar{s})} Q_t(\bar{s},b)), \end{aligned}$$

where $\bar{s}$ is the state reached from state s when performing action a and C is some constant. The learning rate is $\alpha_t^{\omega} = \frac{1}{(t+1)^{\omega}}$, for $\omega \in (1/2, 1]$. We distinguish between a *linear learning rate*, which is $\omega = 1$, and a *polynomial learning rate*, which is $\omega \in (\frac{1}{2}, 1)$.

The *asynchronous Q-learning algorithm*, is simply regular Q-learning as define in (1), and we add the assumption that the underlying strategy has a covering time of L. The updates are as follows:

$$\begin{aligned} \forall s,a \quad &: \quad Q_0(s,a) = C \\ \forall s,a \quad &: \quad Q_{t+1}(s,a) = (1-\alpha_t^{\omega}(s,a))Q_t(s,a) + \alpha_t^{\omega}(s,a)(R_M(s,a) + \gamma \max_{b \in U(\bar{s})} Q_t(\bar{s},b)) \end{aligned}$$

where $\bar{s}$ is the state reached from state s when performing action a and C is some constant. Let $\#(s,a,t)$ be one plus the number of times, until time t, that we visited state s and performed action a. The learning rate $\alpha_t^{\omega}(s,a) = \frac{1}{[\#(s,a,t)]^{\omega}}$, if $t \in T^{s,a}$ and $\alpha_t^{\omega}(s,a) = 0$ otherwise. Again, $\omega = 1$ is a linear learning rate, and $\omega \in (\frac{1}{2}, 1)$ is a polynomial learning rate.

4. Our Main Results

Our main results are upper bounds on the convergence rates of Q-learning algorithms and showing their dependence on the learning rate. The basic case is the synchronous Q-learning. We show that for a polynomial learning rate we have a complexity, which is polynomial in $1/(1-\gamma) = 1/(2\beta)$. In contrast, we show that linear learning rate has an exponential dependence on $1/\beta$. Our results exhibit a sharp difference between the two learning rates, although they both converge with probability one. This distinction, which is highly important, can be observed only when we study the convergence rate, rather than convergence in the limit.

The bounds for asynchronous Q-learning are similar. The main difference is the introduction of a covering time L. For polynomial learning rate we derive a bound polynomial in $1/\beta$, and for linear learning rate our bound is exponential in $\frac{1}{\beta}$. We also show a lower bound for linear learning rate,

which is exponential in $\frac{1}{\beta}$. This implies that our upper bounds are tight, and that the gap between the two bounds is real.

We first prove the results for the synchronous Q-learning algorithm, where we update all the entries of the Q function at each time step, i.e., the updates are synchronous. The following theorem derives the bound for polynomial learning rate.

Theorem 2 *Let Q_T be the value of the synchronous Q-learning algorithm using polynomial learning rate at time T. Then with probability at least $1-\delta$, we have that $||Q_T - Q^*|| \leq \varepsilon$, given that*

$$T = \Omega\left(\left(\frac{V_{max}^2 \ln(\frac{|S|\,|A|V_{max}}{\delta\beta\varepsilon})}{\beta^2\varepsilon^2}\right)^{\frac{1}{\omega}} + \left(\frac{1}{\beta}\ln\frac{V_{max}}{\varepsilon}\right)^{\frac{1}{1-\omega}}\right)$$

The above bound is somewhat complicated. To simplify, assume that ω is a constant and consider first only its dependence on ε. This gives us $\Omega((\ln(1/\varepsilon)/\varepsilon^2)^{1/\omega} + (\ln(1/\varepsilon))^{1/(1-\omega)})$, which is optimized when ω approaches one. Considering the dependence only on β, recall that $V_{max} = R_{max}/(2\beta)$, therefore the complexity is $\tilde{\Omega}(1/\beta^{4/\omega} + 1/\beta^{1/(1-\omega)})$ which is optimized for $\omega = 4/5$. The following theorem bounds the time for linear learning rate.

Theorem 3 *Let Q_T be the value of the synchronous Q-learning algorithm using linear learning rate at time T. Then for any positive constant ψ with probability at least $1-\delta$, we have $||Q_T - Q^*|| \leq \varepsilon$, given that*

$$T = \Omega\left((2+\psi)^{\frac{1}{\beta}\ln(\frac{V_{max}}{\varepsilon})}\frac{V_{max}^2 \ln(\frac{|S|\,|A|V_{max}}{\delta\beta\psi\varepsilon})}{(\psi\beta\varepsilon)^2}\right).$$

Next we state our results to asynchronous Q-learning. The bounds are similar to those of synchronous Q-learning, but have the extra dependency on the covering time L.

Theorem 4 *Let Q_T be the value of the asynchronous Q-learning algorithm using polynomial learning rate at time T. Then with probability at least $1-\delta$, we have $||Q_T - Q^*|| \leq \varepsilon$, given that*

$$T = \Omega\left(\left(\frac{L^{1+3\omega}V_{max}^2 \ln(\frac{|S|\,|A|V_{max}}{\delta\beta\varepsilon})}{\beta^2\varepsilon^2}\right)^{\frac{1}{\omega}} + \left(\frac{L}{\beta}\ln\frac{V_{max}}{\varepsilon}\right)^{\frac{1}{1-\omega}}\right)$$

The dependence on the covering time, in the above theorem, is $\Omega(L^{2+1/\omega} + L^{1/(1-\omega)})$, which is optimized for $\omega \approx 0.77$. For the linear learning rate the dependence is much worse, since it has to be that $L \geq |S| \cdot |A|$, as is stated in the following theorem.

Theorem 5 *Let Q_T be the value of the asynchronous Q-learning algorithm using linear learning rate at time T. Then with probability at least $1-\delta$, for any positive constant ψ we have $||Q_T - Q^*|| \leq \varepsilon$, given that*

$$T = \Omega\left((L+\psi L+1)^{\frac{1}{\beta}\ln\frac{V_{max}}{\varepsilon}}\frac{V_{max}^2 \ln(\frac{|S|\,|A|V_{max}}{\delta\beta\varepsilon\psi})}{(\psi\beta\varepsilon)^2}\right)$$

The following theorem shows that a linear learning rate may require an exponential dependence on $1/(2\beta) = 1/(1-\gamma)$, thus showing that the gap between linear learning rate and polynomial learning rate is real and does exist for some MDPs.

Theorem 6 *There exists a deterministic MDP, M, such that Q-learning with linear learning rate after* $T = \Omega((\frac{1}{\varepsilon})^{\frac{1}{1-\gamma}})$ *steps has* $\|Q_T - Q_M^*\| > \varepsilon$.

5. Experiments

In this section we present experiments using two types of MDPs as well as one we call M_0, which is used in the lower bound example from Section 10. The two MDP types are the "random MDP" and "line MDP". Each type contains n states and two actions for each state.

We generate the "random MDP" as follows: For every state i, action a and state j, we assign a random number $n_{i,j}(a)$ uniformly from $[0,1]$. The probability of a transition from state i to state j while performing action a is $p_{i,j}(a) = \frac{n_{i,j}(a)}{\sum_k n_{i,k}(a)}$. The reward $R(s,a)$ is deterministic and chosen at random uniformly in the interval $[0,10]$.

For the line MDP, all the states are integers and the transition probability from state i to state j is proportional to $\frac{1}{|i-j|}$, where $i \neq j$. The reward distribution is identical to that of the random MDP. (We implemented the random function using the function rand() in C.)

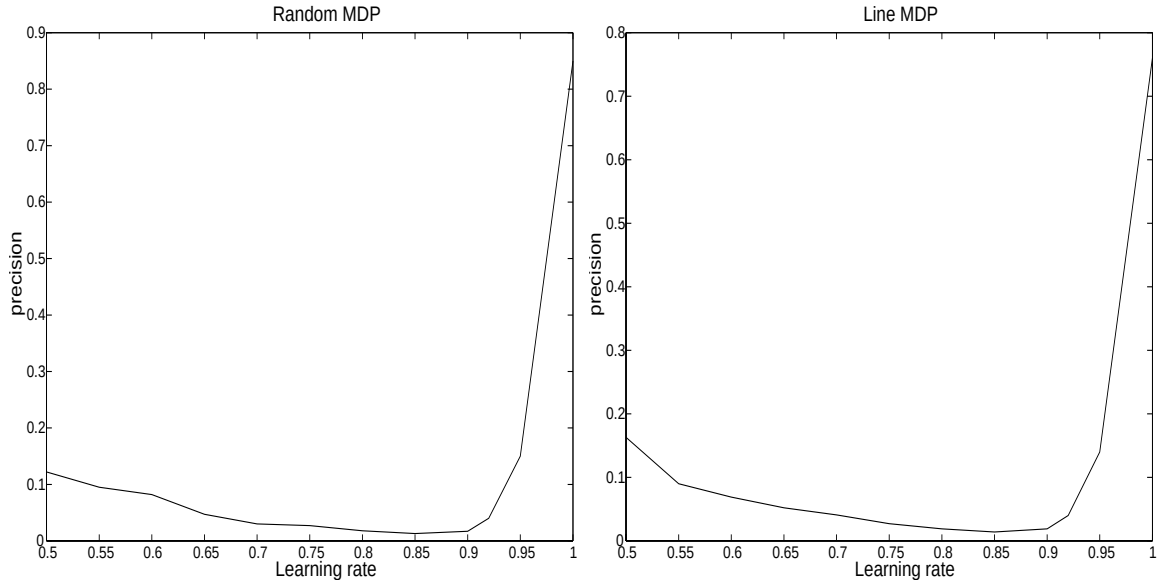


Figure 1: Example of 100 states MDP (both line and random), where the discount factor is $\gamma = 0.7$. We ran asynchronous Q-learning using a random exploration policy for 10^8 steps.

Figure 1 demonstrates the relation between the exponent of the learning rate ω and the accuracy of the model. The best experimental value for ω is about 0.85. Note that when ω approaches one (a linear learning rate), the precision deteriorates. This behavior coincides with our theoretical results on two points. First, our theoretical results predict bad behavior when the learning rate approaches

one (an exponential lower and upper bound). Second, the experiments suggest an optimal value for ω of approximately 0.85. Our theoretical results derive optimal values of optimal ω for different settings of the parameters but most give a similar range. Furthermore, the two types of MDP have similar behavior, which implies that the difference between linear and polynomial learning rates is inherent to many MDPs and not only special cases (as in the lower bound example).

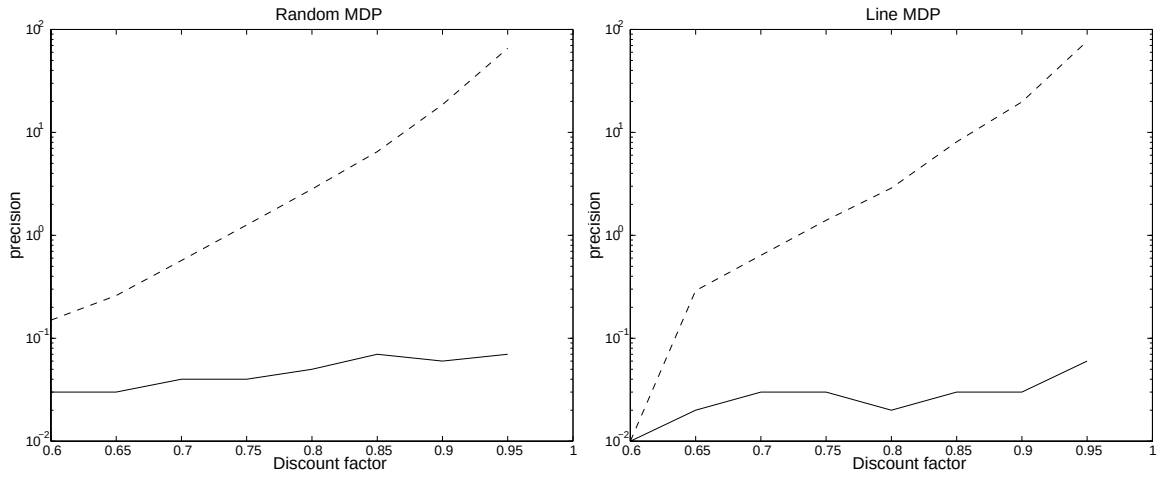


Figure 2: Example of 10 state MDPs (both random and line) using two different learning rates for Q-learning. Both use random exploration policy for 10^7 steps. The solid line is asynchronous Q-learning using $\omega = 0.7$; the dashed line is asynchronous Q-learning using a linear learning rate ($\omega = 1.0$).

Figure 2 demonstrates the strong relationship between discount factor, γ, and convergence rate. In this experiment, we again see similar behavior in both MDPs. When the discount factor approaches one, Q-learning using linear learning rate estimation of the Q value becomes unreliable, while Q-learning using learning rate of $\omega = 0.7$ remains stable (the error is below 0.1).

Figure 3 compares two different learning rates $\omega = 0.6$ and $\omega = 0.9$ for ten state MDPs (both random and line) and finds an interesting tradeoff. For low precision levels, the learning rate of $\omega = 0.6$ was superior, while for high precision levels the learning rate of $\omega = 0.9$ was superior. An explanation for this behavior is that the dependence in terms of ε is $\Omega((\ln(1/\varepsilon)/\varepsilon^2)^{1/\omega} + (\ln(1/\varepsilon))^{1/(1-\omega)})$, which is optimized as the learning rate approaches one.

Our last experimental result is M_0, the lower bound example from Section 10. Here the difference between the learning rates is the most significant, as shown in Figure 4.

6. Background from Stochastic Algorithms

Before we derive our proofs, we first introduce the proof given by Bertsekas and Tsitsiklis (1996) for the convergence of stochastic iterative algorithms; in Section 7 we show that Q-learning algorithms fall in this category. In this section we review the proof for convergence in the limit, and in the next sections we will analyze the rate at which different Q-learning algorithms converge. (We will try to keep the background as close as possible to the needs for this paper rather than giving the most general results.)

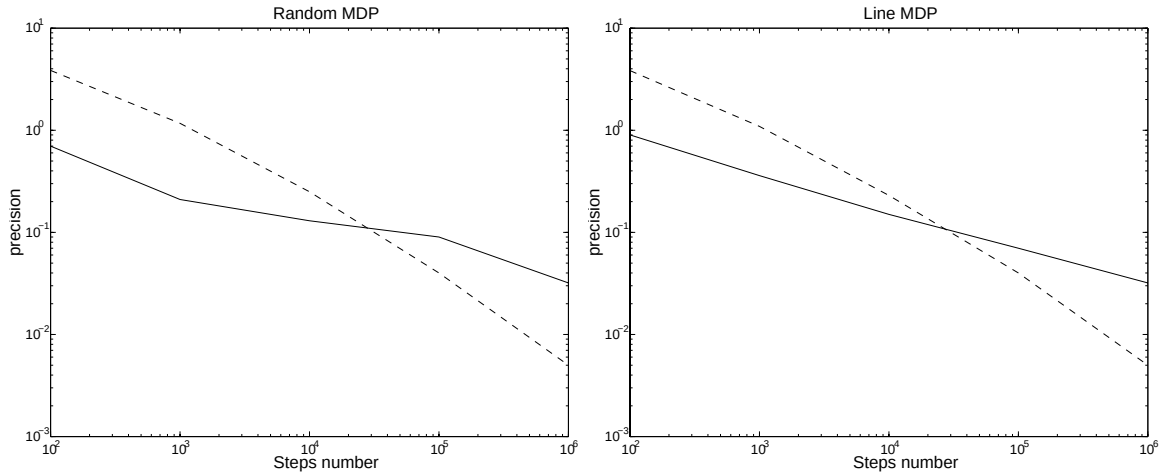


Figure 3: Random and Line MPDs (10 states each), where the discount factor is $\gamma = 0.9$. The dashed line is synchronous Q-learning using $\omega = 0.9$ and the the solid line is synchronous Q-learning using $\omega = 0.6$.

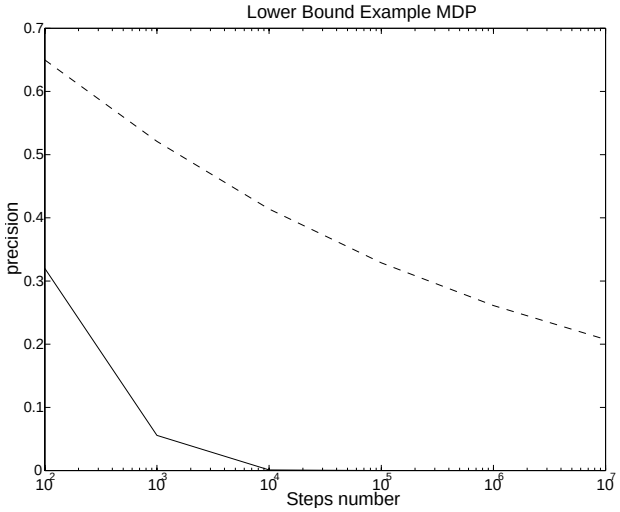


Figure 4: Lower bound example M_0, with discount factor $\gamma = 0.9$. Q-learning ran with two different learning rates, linear (dashed line) and $\omega = 0.65$ (solid line).

This section considers a general type of *iterative stochastic algorithms*, which is computed as follows:

$$X_{t+1}(i) = (1-\alpha_t(i))X_t(i) + \alpha_t(i)((H_tX_t)(i) + w_t(i)), \tag{2}$$

where w_t is a bounded random variable with zero expectation, and each H_t is assumed to belong to a family H of pseudo contraction mappings (See Bertsekas and Tsitsiklis (1996) for details).

Definition 7 *An iterative stochastic algorithm is well-behaved if:*

1. *The step size* $\alpha_t(i)$ *satisfies (1)* $\sum_{t=0}^{\infty}\alpha_t(i)=\infty$*, (2)* $\sum_{t=0}^{\infty}\alpha_t^2(i)<\infty$ *and (3)* $\alpha_t(i)\in(0,1)$.

2. *There exists a constant A that bounds* $w_t(i)$ *for any history* F_t*, i.e.,* $\forall t,i:\ |w_t(i)|\leq A$.

3. *There exists* $\gamma\in[0,1)$ *and a vector* X^* *such that for any X we have* $||H_tX-X^*||\leq\gamma||X-X^*||$.

The main theorem states that a well-behaved stochastic iterative algorithm converges in the limit.

Theorem 8 *[Bertsekas and Tsitsiklis (1996)] Let* X_t *be the sequence generated by a well-behaved stochastic iterative algorithm. Then* X_t *converges to* X^* *with probability* 1.

The following is an outline of the proof given by Bertsekas and Tsitsiklis (1996). Without loss of generality, assume that $X^*=0$ and $\|X_0\|\leq A$. The value of X_t is bounded since $\|X_0\|\leq A$ and for any history F_t we have $\|w_t\|\leq A$; hence, for any t we have $\|X_t\|\leq A$.

Recall that $\beta=\frac{1-\gamma}{2}$. Let $D_1=A$ and $D_{k+1}=(1-\beta)D_k$ for $k\geq 1$. Clearly the sequence D_k converges to zero. We prove by induction that for every k there exists some time τ_k such that for any $t\geq\tau_k$ we have $\|X_t\|\leq D_k$. Note that this will guarantee that at time $t\geq\tau_k$ for any i the value $\|X_t(i)\|$ is in the interval $[-D_k,D_k]$.

The proof is by induction. Assume that there is such a time τ_k and we show that there exists a time τ_{k+1} such that for $t\geq\tau_{k+1}$ we have $\|X_t\|\leq D_{k+1}$. Since D_k converges to zero this proves that X_t converges to zero, which equals X^*. For the proof we define for $t\geq\tau$ the quantity

$$W_{t+1;\tau}(i)=(1-\alpha_t(i))W_{t;\tau}(i)+\alpha_t(i)w_t(i),$$

where $W_{\tau;\tau}(i)=0$. The value of $W_{t;\tau}$ bounds the contributions of $w_j(i)$, $j\in[\tau,t]$, to the value of X_t (starting from time τ). We also define for $t\geq\tau_k$,

$$Y_{t+1;\tau}(i)=(1-\alpha_t(i))Y_{t;\tau}(i)+\alpha_t(i)\gamma D_k$$

where $Y_{\tau_k;\tau_k}=D_k$. Notice that $Y_{t;\tau_k}$ is a deterministic process. The following lemma gives the motivation for the definition of $Y_{t;\tau_k}$.

Lemma 9 *[Bertsekas and Tsitsiklis (1996)] For every i, we have*

$$-Y_{t;\tau_k}(i)+W_{t;\tau_k}(i)\leq X_t(i)\leq Y_{t;\tau_k}(i)+W_{t;\tau_k}(i)$$

Next we use Lemma 9 to complete the proof of Theorem 8. From the definition of $Y_{t;\tau}$ and the assumption that $\sum_{t=0}^{\infty}\alpha_t=\infty$, it follows that $Y_{t;\tau}$ converges to γD_k as t goes to infinity. In addition $W_{t;\tau_k}$ converges to zero as t goes to infinity. Therefore there exists a time τ_{k+1} such that $Y_{t;\tau}\leq(\gamma+\frac{\beta}{2})D_k$, and $|W_{t;\tau_k}|\leq\beta D_k/2$. This fact, together with Lemma. 9, yields that for $t\geq\tau_{k+1}$,

$$||X_t||\leq(\gamma+\beta)D_k=D_{k+1},$$

which completes the proof of Theorem 8.

7. Applying the Stochastic Theorem to Q-learning

In this section we show that both synchronous and asynchronous Q-learning are well-behaved iterative stochastic algorithms. The proof is similar in spirit to the proof given by Bertsekas and Tsitsiklis (1996) At the beginning we deal with synchronous Q-learning. First define operator H as

$$(HQ)(i,a) = \sum_{j=0}^{n} P_{ij}(a)(R(i,a) + \gamma \max_{b \in U(j)} Q(j,b))$$

Rewriting Q-learning with H, we get

$$Q_{t+1}(i,a) = (1-\alpha_t(i,a))Q_t(i,a) + \alpha_t(i,a)((HQ_t)(i,a) + w_t(i,a)).$$

Let $\bar{i}$ is the state reached by performing at time t action a in state i and $r(i,s)$ is the reward observed at time i; then

$$w_t(i,s) = r(i,s) + \gamma \max_{b \in U(\bar{i})} Q_t(\bar{i},b) - \sum_{j=0}^{n} P_{ij}(a) \left(R(i,a) + \gamma \max_{b \in U(j)} Q_t(j,b) \right)$$

In synchronous Q-learning, H is computed simultaneously on all states actions pairs.

Lemma 10 *Synchronous Q-learning is a well-behaved iterative stochastic algorithm.*

Proof We know that for any history F_t $E[w_t(i,a)|F_t] = 0$ and $|w_t(i,a)| \leq V_{max}$. We also know that for $\frac{1}{2} < \omega \leq 1$ we have that $\sum \alpha_t(s,a) = \infty$, $\sum \alpha_t^2(s,a) < \infty$ and $\alpha_t(s,a) \in (0,1)$.
We need only show that H satisfies the contraction property.

$$\begin{aligned}
|(HQ)(i,a) - (H\bar{Q})(i,a)| &\leq \sum_{j=0}^{n} P_{ij}(a) |\gamma \max_{b \in U(j)} Q(j,b) - \gamma \max_{b \in U(j)} \bar{Q}(j,b)| \\
&= \sum_{j=0}^{n} P_{ij}(a)\gamma |\max_{b \in U(j)} Q(j,b) - \max_{b \in U(j)} \bar{Q}(j,b)| \\
&\leq \sum_{j=0}^{n} P_{ij}(a)\gamma \max_{b \in U(j)} |Q(j,b) - \bar{Q}(j,b)| \\
&\leq \gamma \sum_{j=0}^{n} P_{ij}(a) \|Q - \bar{Q}\| \leq \gamma \|Q - \bar{Q}\|
\end{aligned}$$

Since we update all (i,a) pairs simultaneously, synchronous Q-learning is well-behaved stochastic iterative algorithm. ■

We next show that Theorem 8 can be applied also to asynchronous Q-learning.

Lemma 11 *Asynchronous Q-learning, where the input sequence has a finite covering time L, is a well-behaved iterative stochastic algorithm.*

Proof We define HQ for every start state i and start time t_1 of a phase (beginning of the covering time) until the end of the phase (completing the covering time) at time t_2, during which all state-action pairs are updated. Since a state action can be performed more than once, $HQ(i,a)$ can be

performed more than once. We consider time t, in which the policy performs action a at state i and Q_t is the vector. We have that

$$\begin{aligned}
|(HQ_t)(i,a)-(HQ^*)(i,a)| &\leq \sum_{j=0}^{n} P_{ij}(a)|\gamma \max_{b\in U(j)} Q_t(j,b)-\gamma \max_{b\in U(j)} Q^*(j,b)| \\
&= \sum_{j=0}^{n} P_{ij}(a)\gamma|\max_{b\in U(j)} Q_t(j,b)-\max_{b\in U(j)} Q^*(j,b)| \\
&\leq \sum_{j=0}^{n} P_{ij}(a)\gamma \max_{b\in U(j)} |Q_t(j,b)-Q^*(j,b)| \\
&\leq \sum_{j\in A} P_{ij}(a)\gamma \max_{b\in U(j)} |Q_t(j,b)-Q^*(j,b)| \\
&\quad + \sum_{j\in B} P_{ij}(a)\gamma \max_{b\in U(j)} |Q_t(j,b)-Q^*(j,b)| \\
&\leq \gamma\|Q_t-Q^*\|,
\end{aligned}$$

where A includes the states for which during (t_1,t) all the actions in $U(i)$ were performed, and $B=S-A$. We conclude that $\|Q_{t'}-Q^*\|\leq\|Q_{t'-1}-Q^*\|$, since we only change at each time a single state-action pair, which satisfies $|HQ_t(i,a)-Q^*(i,a)|\leq\gamma|Q_t-Q^*|$. We look at the operator H after performing all state-action pairs, $\|HQ-Q^*\|\leq max_{i,a}|HQ_t(i,a)-Q^*(i,a)|\leq\gamma\|Q_t-Q^*\|\leq\gamma\|Q-Q^*\|$. ■

8. Synchronous Q-learning

In this section we give the proof of Theorems 2 and 3. Our main focus will be the value of $r_t=\|Q_t-Q^*\|$, and our aim is to bound the time until $r_t\leq\varepsilon$. We use a sequence of values D_i, such that $D_{k+1}=(1-\beta)D_k$ and $D_1=V_{max}$. As in Section 6, we will consider times τ_k such that for any $t\geq\tau_k$ we have $r_t\leq D_k$. We call the time between τ_k and τ_{k+1} the kth iteration. (Note the distinction between a step of the algorithm and an iteration, which is a sequence of many steps.)

Our proof has two parts. The first (and simple) part is bounding the number of iterations until $D_i\leq\varepsilon$. The bound is derived in the following Lemma.

Lemma 12 *For $m\geq\frac{1}{\beta}\ln(V_{max}/\varepsilon)$ we have $D_m\leq\varepsilon$.*

Proof We have that $D_1=V_{max}$ and $D_i=(1-\beta)D_{i-1}$. We want to find the m that satisfies $D_m=V_{max}(1-\beta)^m\leq\varepsilon$. By taking a logarithm over both sides of the inequality we get $m\geq\frac{1}{\beta}\ln(V_{max}/\varepsilon)$. ■

The second (and much more involved) part is to bound the number of steps in an iteration. We use the following quantities introduced in Section 6. Let $W_{t+1,\tau}(s,a)=(1-\alpha_t^\omega(s,a))W_{t,\tau}(s,a)+\alpha_t^\omega(s,a)w_t(s,a)$, where $W_{\tau,\tau}(s,a)=0$ and

$$w_t(s,a)=R(s,a)+\gamma\max_{b\in U(s')}Q_t(s',b)-\sum_{j=1}^{|S|}P_{s,j}(a)\left(R(s,a)+\gamma\max_{b\in U(j)}Q_t(j,b)\right),$$

where s' is the state reached after performing action a at state s. Let

$$Y_{t+1;\tau_k}(s,a) = (1-\alpha_t^\omega(s,a))Y_{t;\tau_k}(s,a) + \alpha_t^\omega(s,a)\gamma D_k,$$

where $Y_{\tau_k;\tau_k}(s,a) = D_k$. Our first step is to rephrase Lemma 9 for our setting.

Lemma 13 *For every state s action a and time τ_k, we have*

$$-Y_{t;\tau_k}(s,a) + W_{t;\tau_k}(s,a) \le Q^*(s,a) - Q_t(s,a) \le Y_{t;\tau_k}(s,a) + W_{t;\tau_k}(s,a)$$

The above lemma suggests (once again) that in order to bound the error r_t one can bound $Y_{t;\tau_k}$ and $W_{t;\tau_k}$ separately, and the two bounds imply a bound on r_t. We first bound the Y_t term, which is deterministic process, and then we bound the term, $W_{t;\tau}$, which is stochastic.

8.1 Synchronous Q-learning using a Polynomial Learning Rate

We start with Q-learning using a polynomial learning rate and show that the duration of iteration k, which starts at time τ_k and ends at time τ_{k+1}, is bounded by τ_k^ω. For synchronous Q-learning with a polynomial learning rate we define $\tau_{k+1} = \tau_k + \tau_k^\omega$, where τ_1 will be specified latter.

Lemma 14 *Consider synchronous Q-learning with a polynomial learning rate and assume that for any $t \ge \tau_k$ we have $Y_{t;\tau_k}(s,a) \le D_k$. Then for any $t \ge \tau_k + \tau_k^\omega = \tau_{k+1}$ we have $Y_{t;\tau_k}(s,a) \le D_k(\gamma + \frac{2}{e}\beta)$.*

Proof Let $Y_{\tau_k;\tau_k}(s,a) = \gamma D_k + \rho_{\tau_k}$, where $\rho_{\tau_k} = (1-\gamma)D_k$. We can now write

$$Y_{t+1;\tau_k}(s,a) = (1-\alpha_t^\omega)Y_{t;\tau_k}(s,a) + \alpha_t^\omega\gamma D_k = \gamma D_k + (1-\alpha_t^\omega)\rho_t,$$

where $\rho_{t+1} = \rho_t(1-\alpha_t^\omega)$. We would like to show that after time $\tau_{k+1} = \tau_k + \tau_k^\omega$ for any $t \ge \tau_{k+1}$ we have $\rho_t \le \frac{2}{e}\beta D_k$. By definition we can rewrite ρ_t as

$$\rho_t = (1-\gamma)D_k \prod_{l=1}^{t-\tau_k}(1-\alpha_{l+\tau_k}^\omega) = 2\beta D_k \prod_{l=1}^{t-\tau_k}(1-\alpha_{l+\tau_k}^\omega) = 2\beta D_k \prod_{l=1}^{t-\tau_k}(1-\frac{1}{(l+\tau_k)^\omega}),$$

where the last identity follows from the fact that $\alpha_t^\omega = 1/t^\omega$. Since the α_t^ω's are monotonically decreasing

$$\rho_t \le 2\beta D_k(1-\frac{1}{\tau_k^\omega})^{t-\tau_k}.$$

For $t \ge \tau_k + \tau_k^\omega$ we have

$$\rho_t \le 2\beta D_k(1-\frac{1}{\tau_k^\omega})^{\tau_k^\omega} \le \frac{2}{e}\beta D_k.$$

Hence, $Y_{t;\tau_k}(s,a) \le (\gamma + \frac{2}{e}\beta)D_k$. ■

Next we bound the term $W_{t;\tau_k}$ by $(1-\frac{2}{e})\beta D_k$. The sum of the bounds for $W_{t;\tau_k}(s,a)$ and $Y_{t;\tau_k}(s,a)$ would be $(\gamma+\beta)D_k = (1-\beta)D_k = D_{k+1}$, as desired.

Definition 15 *Let $W_{t;\tau_k}(s,a) = (1-\alpha_t^\omega(s,a))W_{t-1;\tau_k}(s,a) + \alpha_t^\omega(s,a)w_t(s,a)$*
$= \sum_{i=\tau_k+1}^{t} \eta_i^{k,t}(s,a)w_i(s,a)$, where $\eta_i^{k,t}(s,a) = \alpha_{i+\tau_k}^\omega(s,a)\prod_{j=\tau_k+i+1}^{t}(1-\alpha_j^\omega(s,a))$ and let $W_{t;\tau_k}^l(s,a) = \sum_{i=\tau_k+1}^{\tau_k+l} \eta_i^{k,t}(s,a)w_i(s,a)$.

Note that in the synchronous model $\alpha_t^\omega(s,a)$ and $\eta_i^{k,t}(s,a)$ are identical for every state action pair. We also note that $W_{t;\tau_k}^{t-\tau_k+1}(s,a) = W_{t;\tau_k}(s,a)$. We have bounded the term $Y_{t;\tau_k}$, for $t = \tau_{k+1}$. This bound holds for any $t \geq \tau_{k+1}$, since the sequence $Y_{t;\tau_k}$ is monotonically decreasing. In contrast, the term $W_{t;\tau_k}$ is stochastic. Therefore it is not sufficient to bound $W_{\tau_{k+1};\tau_k}$, but we need to bound $W_{t;\tau_k}$ for $t \geq \tau_{k+1}$. However, it is sufficient to consider $t \in [\tau_{k+1}, \tau_{k+2}]$. The following lemma bounds the coefficients in that interval.

Lemma 16 *For any $t \in [\tau_{k+1}, \tau_{k+2}]$ and $i \in [\tau_k, t]$, we have $\eta_i^{k,t} = \Theta(\frac{1}{\tau_k^\omega})$,*

Proof Since $\eta_i^{k,t} = \alpha_{i+\tau_k}^\omega \prod_{j=\tau_k+i+1}^t (1-\alpha_j^\omega)$,we can divide $\eta_t^{k,t}$ into two parts, the first one $\alpha_{i+\tau_k}^\omega$ and the second one $\mu = \prod_{j=\tau_k+i+1}^t (1-\alpha_j^\omega)$. We show that the first one is $\Theta(\frac{1}{\tau_k^\omega})$ and the second is constant.
Since $\alpha_{i+\tau_k}^\omega$ are monotonically decreasing we have for every $i \in [\tau_k, \tau_{k+2}]$ we have $\alpha_{\tau_k}^\omega \leq \alpha_i^\omega \leq \alpha_{\tau_{k+2}}^\omega$, thus $\frac{1}{\tau_k^\omega} \leq \alpha_t^\omega \leq \frac{1}{(\tau_{k+1}+\tau_{k+1}^\omega)^\omega} \leq \frac{1}{(3\tau_k^\omega+\tau_k)^\omega} < \frac{1}{4\tau_k^\omega}$. Next we bound μ. Clearly μ is bounded from above by 1. Also $\mu \geq \prod_{j=\tau_k}^{\tau_{k+2}} (1-\alpha_j) \geq (1-\frac{1}{\tau_k^\omega})^{3\tau_k^\omega} \geq \frac{1}{e^3}$. Therefore, we have that for every $t \in [\tau_k, \tau_{k+2}]$, $\eta_i^{k,t} = \Theta(\frac{1}{\tau_k^\omega})$. ■

We introduce Azuma's inequality, which bounds the deviations of a martingale. The use of Azuma's inequality is mainly needed for the asynchronous case.

Lemma 17 (Azuma 1967) *Let $X_0, X_1, ..., X_n$ be a martingale sequence such that for each $1 \leq k \leq n$,*

$$|X_k - X_{k-1}| \leq c_k,$$

where the constant c_k may depend on k. Then for all $n \geq 1$ and any $\varepsilon > 0$

$$Pr\left[|X_n - X_0| > \varepsilon\right] \leq 2e^{-\frac{\varepsilon}{2\sum_{k=1}^n c_k^2}}$$

Next we show that Azuma's inequality can be applied to $W_{t;\tau_k}^l$.

Lemma 18 *For any $t \in [\tau_{k+1}, \tau_{k+2}]$ and $1 \leq l \leq t$ we have that $W_{t;\tau_k}^l(s,a)$ is a martingale sequence, which satisfies*

$$|W_{t;\tau_k}^l(s,a) - W_{t;\tau_k}^{l-1}(s,a)| \leq \Theta(\frac{V_{max}}{\tau_k^\omega})$$

Proof We first note that $W_{t;\tau_k}^l(s,a)$ is a martingale sequence, since

$$\begin{aligned} E[W_{t;\tau_k}^l(s,a) - W_{t;\tau_k}^{l-1}(s,a)|F_{\tau_k+l-1}] &= E[\eta_{\tau_k+l}^{k,t}(s,a) w_{\tau_k+l}(s,a)|F_{\tau_k+l-1}] \\ &= \eta_{\tau_k+l}^{k,t} E[w_{\tau_k+l}(s,a)|F_{\tau_k+l-1}] = 0. \end{aligned}$$

By Lemma 16 we have that $\eta_l^{k,t}(s,a) = \Theta(1/\tau_k^\omega)$, thus

$$|W_{t;\tau_k}^l(s,a) - W_{t;\tau_k}^{l-1}(s,a)| = \eta_{\tau_k+l}^{k,t}(s,a)|w_{\tau_k+l}(s,a)| \leq \Theta(\frac{V_{max}}{\tau_k^\omega}).$$

■

The following lemma provides a bound for the stochastic error caused by the term $W_{t;\tau_k}$ by using Azuma's inequality.

Lemma 19 *Consider synchronous Q-learning with a polynomial learning rate. With probability at least* $1-\frac{\delta}{m}$ *we have* $|W_{t;\tau_k}| \leq (1-\frac{2}{e})\beta D_k$ *for any* $t \in [\tau_{k+1},\tau_{k+2}]$*, i.e.,*

$$Pr\left[\forall t \in [\tau_{k+1},\tau_{k+2}]:\ |W_{t;\tau_k}| \leq (1-\frac{2}{e})\beta D_k\right] \geq 1-\frac{\delta}{m}$$

given that $\tau_k = \Theta((\frac{V_{max}^2 \ln(V_{max}|S|\,|A|m/(\delta\beta D_k))}{\beta^2 D_k^2})^{1/\omega})$.

Proof By Lemma 18 we can apply Azuma's inequality to $W_{t;\tau_k}^{t-\tau_k+1}$ with $c_i = \Theta(\frac{V_{max}}{\tau_k^{\omega}})$ (note that $W_{t;\tau_k}^{t-\tau_k+1} = W_{t;\tau_k}$) . Therefore, we derive that

$$Pr\left[|W_{t;\tau_k}(s,a)| \geq \tilde{\varepsilon} \mid t \in [\tau_{k+1},\tau_{k+2}]\right] \quad \leq \quad 2e^{\frac{-2\tilde{\varepsilon}^2}{\Sigma_{i=\tau_k}^{t} c_i^2}} \leq 2e^{-c\tau_k^{\omega}\tilde{\varepsilon}^2/V_{max}^2},$$

for some constant $c>0$. Set $\tilde{\delta}_k = 2e^{-c\tau_k^{\omega}\tilde{\varepsilon}^2/V_{max}^2}$, which holds for $\tau_k^{\omega} = \Theta(\ln(1/\tilde{\delta})V_{max}^2/\tilde{\varepsilon^2})$. Using the union bound we have,

$$Pr\left[\forall t \in [\tau_{k+1},\tau_{k+2}]:\ W_{t;\tau_k}(s,a) \leq \tilde{\varepsilon}\right] \leq \sum_{t=\tau_{k+1}}^{\tau_{k+2}} Pr\left[W_{t;\tau_k}(s,a) \leq \tilde{\varepsilon}\right],$$

thus taking $\tilde{\delta}_k = \frac{\delta}{m(\tau_{k+2}-\tau_{k+1})|S||A|}$ assures that with probability at least $1-\frac{\delta}{m}$ the statement hold at every state-action pair and time $t \in [\tau_{k+1},\tau_{k+2}]$. As a result we have,

$$\tau_k = \Theta\left((\frac{V_{max}^2 \ln(|S|\,|A|\,m\tau_k^{\omega}/\delta)}{\tilde{\varepsilon}^2})^{1/\omega}\right) = \Theta\left((\frac{V_{max}^2 \ln(|S|\,|A|\,mV_{max}/\delta\tilde{\varepsilon})}{\tilde{\varepsilon}^2})^{1/\omega}\right)$$

Setting $\tilde{\varepsilon} = (1-2/e)\beta D_k$ gives the desire bound. ■

We have bounded for each iteration the time needed to achieve the desired precision level with probability $1-\frac{\delta}{m}$. The following lemma provides a bound for the error in all the iterations.

Lemma 20 *Consider synchronous Q-learning using a polynomial learning rate. With probability at least* $1-\delta$*, for every iteration* $k \in [1,m]$ *and time* $t \in [\tau_{k+1},\tau_{k+2}]$ *we have* $W_{t;\tau_k} \leq (1-\frac{2}{e})\beta D_k$*, i.e.,*

$$Pr\left[\forall k \in [1,m],\ \forall t \in [\tau_{k+1},\tau_{k+2}]:\ |W_{t;\tau_k}| \leq (1-\frac{2}{e})\beta D_k\right] \geq 1-\delta,$$

given that $\tau_0 = \Theta((\frac{V_{max}^2 \ln(V_{max}|S|\,|A|/(\delta\beta\varepsilon))}{\beta^2\varepsilon^2})^{1/\omega})$.

Proof From Lemma 19 we know that

$$\begin{aligned} &Pr\left[\forall t \in [\tau_{k+1},\tau_{k+2}]:\ |W_{t;\tau_k}| \geq (1-\frac{2}{e})\beta D_k\right] \\ &\quad = Pr\left[\forall s,a\ \forall t \in [\tau_{k+1},\tau_{k+2}]:\ \sum_{i=\tau_k}^{t} w_i(s,a)\eta_i^{k,t} \geq \tilde{\varepsilon}\right] \leq \frac{\delta}{m} \end{aligned}$$

Using the union bound we have,

$$Pr\left[\forall s,a\ \forall k \le m, \forall t \in [\tau_{k+1},\tau_{k+2}]\ \sum_{i=\tau_k}^{t} w_i(s,a)\eta_i^{k,t} \ge \tilde{\varepsilon}\right]$$
$$\le \sum_{k=1}^{m} Pr\left[\forall s,a\ \forall t \in [\tau_{k+1},\tau_{k+2}]\ \sum_{i=\tau_k}^{t} w_i(s,a)\eta_i^{k,t} \ge \tilde{\varepsilon}\right] \le \delta$$

where $\tilde{\varepsilon} = (1-\frac{2}{e})\beta D_k$. ■

We have bounded both the size of each iteration, as a function of its starting time, and the number of the iterations needed. The following lemma solves the recurrence $\tau_{k+1} = \tau_k + \tau_k^{\omega}$ and bounds the total time required (which is a special case of Lemma 32).

Lemma 21 *Let*

$$a_{k+1} = a_k + a_k^{\omega} = a_0 + \sum_{i=0}^{k} a_i^{\omega}.$$

For any constant $\omega \in (0,1)$, $a_k = O((a_0^{1-\omega}+k)^{\frac{1}{1-\omega}}) = O(a_o + k^{\frac{1}{1-\omega}})$

The proof of Theorem 2 follows from Lemma 21, Lemma 20, Lemma 12 and Lemma 14.

8.2 Synchronous Q-learning using a Linear Learning Rate

In this subsection, we derive results for Q-learning with a linear learning rate. The proof is very similar in spirit to the proof of Theorem 2 and we give here analogous lemmas to the ones in Subsection 8.1. First, the number of iterations required for synchronous Q-learning with a linear learning rate is the same as that for a polynomial learning rate. Therefore, we only need to analyze the number of steps in an iteration.

Lemma 22 *Consider synchronous Q-learning with a linear learning rate and assume that for any* $t \ge \tau_k$ *we have* $Y_{t;\tau_k}(s,a) \le D_k$. *Then for any* $t \ge (2+\psi)\tau_k = \tau_{k+1}$ *we have* $Y_{t;\tau_k}(s,a) \le D_k(\gamma + \frac{2}{2+\psi}\beta)$

Proof Let $Y_{\tau_k;\tau_k}(s,a) = \gamma D_k + \rho_{\tau_k}$, where $\rho_{\tau_k} = (1-\gamma)D_k$. We can now write

$$Y_{t+1;\tau_k}(s,a) = (1-\alpha_t)Y_{t;\tau_k}(s,a) + \alpha_t\gamma D_k = \gamma D_k + (1-\alpha_t)\rho_t,$$

where $\rho_{t+1} = \rho_t(1-\alpha_t)$. We would like show that after time $(2+\psi)\tau_k = \tau_{k+1}$ for any $t \ge \tau_{k+1}$ we have $\rho_t \le \beta D_k$. By definition we can rewrite ρ_t as,

$$\rho_t = (1-\gamma)D_k \prod_{l=1}^{t-\tau_k}(1-\alpha_{l+\tau_k}) = 2\beta D_k \prod_{l=1}^{t-\tau_k}(1-\alpha_{l+\tau_k}) = 2\beta D_k \prod_{l=1}^{t-\tau_k}(1-\frac{1}{l+\tau_k}),$$

where the last identity follows from the fact that $\alpha_t = 1/t$. Simplifying the expression, and setting $t = (2+\psi)\tau_k$, we have

$$\rho_t \le 2D_k\beta\frac{\tau_k}{t} = \frac{2D_k\beta}{2+\psi}$$

Hence, $Y_{t;\tau_k}(s,a) \leq (\gamma + \frac{2}{2+\psi}\beta)D_k$. ■

The following lemma enables the use of Azuma's inequality.

Lemma 23 *For any $t \geq \tau_k$ and $1 \leq l \leq t$ we have that $W^l_{t;\tau_k}(s,a)$ is a martingale sequence, which satisfies*

$$|W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)| \leq \frac{V_{max}}{t}$$

Proof We first note that $W^l_{t;\tau_k}(s,a)$ is a martingale sequence, since

$$\begin{aligned} E[W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)|F_{\tau_k+l-1}] &= E[\eta^{k,t}_{\tau_k+l}(s,a)w_{\tau_k+l}(s,a)|F_{\tau_k+l-1}] \\ &= \eta^{k,t}_{\tau_k+l}E[w_{\tau_k+l}(s,a)|F_{\tau_k+l-1}] = 0. \end{aligned}$$

For linear learning rate we have that $\eta^{k,t}_{l+\tau_k}(s,a) = 1/t$, thus

$$|W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)| = \frac{w_{\tau_k+l}(s,a)}{t} \leq \frac{V_{max}}{t}.$$

■

The following Lemma provides a bound for the stochastic term $W_{t;\tau_k}$.

Lemma 24 *Consider synchronous Q-learning with a linear learning rate. With probability at least $1 - \frac{\delta}{m}$, we have $|W_{t;\tau_k}| \leq \frac{\psi}{2+\psi}\beta D_k$ for any $t \in [\tau_{k+1}, \tau_{k+2}]$ and any positive constant ψ, i.e.*

$$Pr\left[\forall t \in [\tau_{k+1}, \tau_{k+2}]: \ W_{t;\tau_k} \leq \frac{\psi}{2+\psi}\beta D_k\right] \geq 1 - \frac{\delta}{m}$$

given that $\tau_k = \Theta(\frac{V^2_{max}\ln(V_{max}|S|\,|A|m/(\psi\delta\beta D_k))}{\psi^2\beta^2 D_k^2})$

Proof By Lemma 23 for any $t \geq \tau_{k+1}$ we can apply Azuma's inequality to $W^{t-\tau_k+1}_{t;\tau_k}$ with $c_i = \frac{V_{max}}{i+\tau_k}$ (note that $W^{t-\tau_k+1}_{t;\tau_k} = W_{t;\tau_k}$). Therefore, we derive that

$$Pr[|W_{t;\tau_k}| \geq \tilde{\varepsilon} \mid t \geq \tau_{k+1}] \leq 2e^{\frac{-2\tilde{\varepsilon}^2}{\sum_{i=\tau_k}^{t} c_i^2}} = 2e^{-c\frac{t^2\tilde{\varepsilon}^2}{(t-\tau_k)V^2_{max}}} \leq 2e^{-c\frac{t\tilde{\varepsilon}^2}{V^2_{max}}}$$

for some positive constant c. Using the union bound we get

$$\begin{aligned} Pr[\forall t \in [\tau_{k+1}, \tau_{k+2}]: \ |W_{t;\tau_k}| \geq \tilde{\varepsilon}] &\leq Pr[\forall t \geq (2+\psi)\tau_k: \ |W_{t;\tau_k}| \geq \tilde{\varepsilon}] \\ &\leq \sum_{t=(2+\psi)\tau_k}^{\infty} Pr[|W_{t;\tau_k}| \geq \tilde{\varepsilon}] \\ &\leq \sum_{t=(2+\psi)\tau_k}^{\infty} 2e^{-c\frac{t\tilde{\varepsilon}^2}{V^2_{max}}} = 2e^{-c\frac{((2+\psi)\tau_k)\tilde{\varepsilon}^2}{V^2_{max}}} \sum_{t=0}^{\infty} e^{-\frac{t\tilde{\varepsilon}}{V^2_{max}}} \\ &= \frac{2e^{-c\frac{(2+\psi)\tau_k\tilde{\varepsilon}^2}{V^2_{max}}}}{1 - e^{\frac{-\tilde{\varepsilon}^2}{V^2_{max}}}} = \Theta(\frac{e^{-\frac{c'\tau_k\tilde{\varepsilon}^2}{V^2_{max}}}V^2_{max}}{\tilde{\varepsilon}^2}), \end{aligned}$$

where the last equality is due to Taylor expansion and c' is some positive constant. By setting $\frac{\delta}{m|S||A|} = \Theta(\frac{e^{-\frac{c'\tau_k\tilde{\varepsilon}^2}{V_{max}^2}}V_{max}^2}{\tilde{\varepsilon}^2})$, which holds for $\tau_k = \Theta(\frac{V_{max}^2 \ln(V_{max}|S||A|m/(\delta\tilde{\varepsilon}))}{\tilde{\varepsilon}^2})$, and $\tilde{\varepsilon} = \frac{\psi}{2+\psi}\beta D_k$ assures us that for every $t \geq \tau_{k+1}$ (and as a result for any $t \in [\tau_{k+1}, \tau_{k+2}]$) with probability at least $1-\frac{\delta}{m}$ the statement holds at every state-action pair. ■

We have bounded for each iteration the time needed to achieve the desired precision level with probability $1-\frac{\delta}{m}$. The following lemma provides a bound for the error in all the iterations.

Lemma 25 *Consider synchronous Q-learning using a linear learning rate. With probability* $1-\delta$*, for every iteration* $k \in [1,m]$*, time* $t \in [\tau_{k+1}, \tau_{k+2}]$ *and any constant* $\psi > 0$ *we have* $|W_{t;\tau_k}| \leq \frac{\psi\beta D_k}{2+\psi}$*), i.e.,*

$$Pr\left[\forall k \in [1,m],\ \forall t \in [\tau_{k+1}, \tau_{k+2}]:\ \ |W_{t;\tau_k}| \leq \frac{\psi\beta D_k}{2+\psi}\right] \geq 1-\delta,$$

given that $\tau_0 = \Theta(\frac{V_{max}^2 \ln(V_{max}|S||A|m/(\psi\delta\beta\varepsilon))}{\psi^2\beta^2\varepsilon^2})$.

Proof From Lemma 19 we know that

$$Pr\left[\forall t \in [\tau_{k+1}, \tau_{k+2}]:\ \ |W_{t;\tau_k}| \geq \frac{\psi\beta D_k}{2+\psi}\right] \leq \frac{\delta}{m}$$

Using the union bound we have that,

$$\begin{aligned} &Pr\left[\forall k \leq m, \forall t \in [\tau_{k+1}, \tau_{k+2}]:\ \ |W_{t;\tau_k}| \geq \frac{\psi\beta D_k}{2+\psi}\right] \\ &\quad \leq \sum_{k=1}^{m} Pr\left[\forall t \in [\tau_{k+1}, \tau_{k+2}]:\ \ |W_{t;\tau_k}| \geq \frac{\psi\beta D_k}{2+\psi}\right] \leq \delta \end{aligned}$$

■

The proof of Theorem 5 follows from Lemmas 25 and 22, 12 and the fact that $a_{k+1} = (2+\psi)a_k = (2+\psi)^k a_1$.

9. Asynchronous Q-learning

The major difference between synchronous and asynchronous Q-learning is that asynchronous Q-learning updates only one state action pair at each time while synchronous Q-learning updates all state-action pairs at each time unit. This causes two difficulties: the first is that different updates use different values of the Q function in their update. This problem is fairly easy to handle given the machinery introduced. The second, and more basic problem is that each state-action pair should occur enough times for the update to progress. To ensure this, we introduce the notion of covering time, denoted by L. We first extend the analysis of synchronous Q-learning to asynchronous Q-learning, in which each run always has covering time L, which implies that from any start state, in L steps all state-action pairs are performed. Later we relax the requirement such that the condition holds only with probability $1/2$, and show that with high probability we have a covering time of $L\log T$ for a run of length T. Note that our notion of covering time does not assume a stationary

distribution of the exploration strategy; it may be the case that at some periods of time certain state-action pairs are more frequent while in other periods different state-action pairs are more frequent. In fact, we do not even assume that the sequence of state-action pairs is generated by a strategy—it can be an arbitrary sequence of state-action pairs, along with their reward and next state.

Definition 26 *Let $n(s,a,t_1,t_2)$ be the number of times that the state action pair (s,a) was performed in the time interval $[t_1,t_2]$.*

In this section, we use the same notations as in Section 8 for D_k, τ_k, $Y_{t;\tau_k}$ and $W_{t;\tau_k}$, with a different set of values for τ_k. We first give the results for asynchronous Q-learning using polynomial learning rate (Subsection 9.1); we give a similar proof for linear learning rates in Subsection 9.2.

9.1 Asynchronous Q-learning using a Polynomial Learning Rate

Our main goal is to show that the size of the kth iteration is $L\tau_k^\omega$. The covering time property guarantees that in $L\tau_k^\omega$ steps each pair of state action is performed at least τ_k^ω times. For this reason we define for asynchronous Q-learning with polynomial learning rate the sequence $\tau_{k+1} = \tau_k + L\tau_k^\omega$, where τ_1 will be specified later. As in Subsection 8.1 we first bound the value of $Y_{t;\tau_k}$

Lemma 27 *Consider asynchronous Q-learning with a polynomial learning rate and assume that for any $t \geq \tau_k$ we have $Y_{t;\tau_k}(s,a) \leq D_k$. Then for any $t \geq \tau_k + L\tau_k^\omega = \tau_{k+1}$ we have $Y_{t;\tau_k}(s,a) \leq D(\gamma + \frac{2}{e}\beta)$*

Proof For each state-action pair (s,a) we are assured that $n(s,a,\tau_k,\tau_{k+1}) \geq \tau_k^\omega$, since the covering time is L and the underlying policy has made $L\tau_k^\omega$ steps. Using the fact that the $Y_{t;\tau_k}(s,a)$ are monotonically decreasing and deterministic, we can apply the same argument as in the proof of Lemma 14. ■

The next Lemma bounds the influence of each sample $w_t(s,a)$ on $W_{t;\tau_k}(s,a)$.

Lemma 28 *Let $\tilde{w}^t_{i+\tau_k}(s,a) = \eta_i^{k,t}(s,a) w_{i+\tau_k}(s,a)$ then for any $t \in [\tau_{k+1}, \tau_{k+2}]$ the random variable $\tilde{w}^t_{i+\tau_k}(s,a)$ has zero mean and bounded by $(L/\tau_k)^\omega V_{max}$.*

Proof Note that by definition $w_{\tau_k+i}(s,a)$ has zero mean and is bounded by V_{max} for any history and state-action pair. In a time interval of length τ, by definition of the covering time, each state-action pair is performed at least τ/L times; therefore, $\eta_i^{k,t}(s,a) \leq (L/\tau_k)^\omega$. Looking at the expectation of $\tilde{w}_{i+\tau_k}(s,a)$ we observe that

$$E[\tilde{w}^t_{i+\tau_k}(s,a)] = E[\eta_i^{k,t}(s,a) w_{i+\tau_k}(s,a)] = \eta_i^{k,t}(s,a) E[w_{i+\tau_k}(s,a)] = 0$$

Next we prove that it is bounded as well:

$$\begin{aligned} |\tilde{w}^t_{i+\tau_k}(s,a)| &= |\eta_i^{k,t}(s,a) w_{i+\tau_k}(s,a)| \\ &\leq |\eta_i^{k,t}(s,a)| V_{max} \\ &\leq (L/\tau_k)^\omega V_{max} \end{aligned}$$

■

Next we define $W^l_{t;\tau_k}(s,a) = \sum_{i=1}^{l} \tilde{w}^t_{i+\tau_k}(s,a)$ and prove that it is martingale sequence with bounded differences.

Lemma 29 *For any $t \in [\tau_{k+1}, \tau_{k+2}]$ and $1 \le l \le t$ we have that $W^l_{t;\tau_k}(s,a)$ is a martingale sequence, which satisfies*

$$|W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)| \le (L/\tau_k)^\omega V_{max}$$

Proof We first note that $W^l_{t;\tau_k}(s,a)$ is a martingale sequence, since

$$E[W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)|F_{\tau_k+l-1}] \quad = \quad E[\tilde{w}^t_{l+\tau_k}(s,a)|F_{\tau_k+l-1}] = 0.$$

By Lemma 28 we have that $\tilde{w}^t_{l+\tau_k}(s,a)$ is bounded by $(L/\tau_k)^\omega V_{max}$, thus

$$|W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)| = \tilde{w}^t_{l+\tau_k}(s,a) \le (L/\tau_k)^\omega V_{max}.$$

■

The following Lemma bounds the value of the term $W_{t;\tau_k}$.

Lemma 30 *Consider asynchronous Q-learning with a polynomial learning rate. With probability at least $1-\frac{\delta}{m}$ we have for every state-action pair $|W_{t;\tau_k}(s,a)| \le (1-\frac{2}{e})\beta D_k$ for any $t \in [\tau_{k+1}, \tau_{k+2}]$, i.e.*

$$Pr\left[\forall s,a\ \forall t \in [\tau_{k+1}, \tau_{k+2}]:\ |W_{t;\tau_k}(s,a)| \le (1-\frac{2}{e})\beta D_k\right] \ge 1-\frac{\delta}{m}$$

given that $\tau_k = \Theta((\frac{L^{1+3\omega}V^2_{max}\ln(V_{max}|S|\,|A|m/(\delta\beta D_k))}{\beta^2 D_k^2})^{1/\omega})$.

Proof For each state-action pair we look on $W^l_{t;\tau_k}(s,a)$ and note that $W_{t;\tau_k}(s,a) = W^{t-\tau_k+1}_{t;\tau_k}(s,a)$. Let $\ell = n(s,a,\tau_k,t)$, then for any $t \in [\tau_{k+1}, \tau_{k+2}]$ we have that $\ell \le \tau_{k+2} - \tau_k \le \Theta(L^{1+\omega}\tau_k^\omega)$. By Lemma 29 we can apply Azuma's inequality to $W^{t-\tau_k+1}_{t;\tau_k}(s,a)$ with $c_i = (L/\tau_k)^\omega V_{max}$. Therefore, we derive that

$$\begin{aligned} Pr[|W_{t;\tau_k}(s,a)| \ge \tilde{\varepsilon} \mid t \in [\tau_{k+1}, \tau_{k+2}]] &\le 2e^{\frac{-\tilde{\varepsilon}^2}{2\sum_{i=\tau_k+1, i\in T^{s,a}}^{t} c_i^2}} \le 2e^{-c\frac{\tilde{\varepsilon}^2\tau_k^{2\omega}}{\ell V^2_{max}L^{2\omega}}} \\ &\le 2e^{-c\frac{\tilde{\varepsilon}^2\tau_k^\omega}{L^{1+3\omega}V^2_{max}}}, \end{aligned}$$

for some constant $c>0$. We can set $\tilde{\delta}_k = 2e^{-c\tau_k^\omega\tilde{\varepsilon}^2/(L^{1+3\omega}V^2_{max})}$, which holds for $\tau_k^\omega = \Theta(\ln(1/\tilde{\delta}_k)L^{1+3\omega}V^2_{max}/\tilde{\varepsilon}^2)$. Using the union bound we have

$$Pr[\forall t \in [\tau_{k+1}, \tau_{k+2}]:\ W_{t;\tau_k}(s,a) \le \tilde{\varepsilon}] \le \sum_{t=\tau_{k+1}}^{\tau_{k+2}} Pr[W_{t;\tau_k}(s,a) \le \tilde{\varepsilon}],$$

thus taking $\tilde{\delta}_k = \frac{\delta}{m(\tau_{k+2}-\tau_{k+1})|S||A|}$ assures a certainty level of $1-\frac{\delta}{m}$ for each state-action pair. As a result we have

$$\tau_k^\omega = \Theta(\frac{L^{1+3\omega}V^2_{max}\ln(|S|\,|A|\,m\tau_k^\omega/\delta)}{\tilde{\varepsilon}^2}) = \Theta(\frac{L^{1+3\omega}V^2_{max}\ln(|S|\,|A|\,mV_{max}/(\delta\tilde{\varepsilon}))}{\tilde{\varepsilon}^2})$$

Setting $\tilde{\varepsilon} = (1-2/e)\beta D_k$ give the desired bound. ■

We have bounded for each iteration the time needed to achieve the desired precision level with probability $1-\frac{\delta}{m}$. The following lemma provides a bound for the error in all the iterations.

Lemma 31 *Consider asynchronous Q-learning using a polynomial learning rate. With probability* $1-\delta$*, for every iteration* $k \in [1,m]$ *and time* $t \in [\tau_{k+1}, \tau_{k+2}]$ *we have* $|W_{t;\tau_k}(s,a)| \leq (1-\frac{2}{e})\beta D_k$*, i.e.,*

$$Pr\left[\forall k \in [1,m],\ \forall t \in [\tau_{k+1},\tau_{k+2}],\ \forall s,a:\ |W_{t;\tau_k}(s,a)| \leq (1-\frac{2}{e})\beta D_k\right] \geq 1-\delta,$$

given that $\tau_0 = \Theta((\frac{L^{1+3\omega}V_{max}^2 \ln(V_{max}|S|\,|A|m/(\delta\beta\varepsilon))}{\beta^2\varepsilon^2})^{1/\omega})$.

Proof From Lemma 30 we know that

$$Pr\left[\forall t \in [\tau_{k+1},\tau_{k+2}]:\ |W_{t;\tau_k}| \geq (1-\frac{2}{e})\beta D_k\right] \leq \frac{\delta}{m}$$

Using the union bound we have that

$$Pr\left[\forall k \leq m, \forall t \in [\tau_{k+1},\tau_{k+2}]\ \ |W_{t;\tau_k}| \geq \tilde{\varepsilon}\right] \quad \leq \quad \sum_{k=1}^{m} Pr\left[\forall t \in [\tau_{k+1},\tau_{k+2}]\ \ |W_{t;\tau_k}| \geq \tilde{\varepsilon}\right] \leq \delta,$$

where $\tilde{\varepsilon} = (1-\frac{2}{e})\beta D_k$ ■

The following lemma solves the recurrence $\sum_{i=0}^{m-1} L\tau_i^{\omega} + \tau_0$ and derives the time complexity.

Lemma 32 *Let*

$$a_{k+1} = a_k + La_k^{\omega} = a_0 + \sum_{i=0}^{k} La_i^{\omega}$$

Then for any constant $\omega \in (0,1)$, $a_k = O((a_0^{1-\omega} + Lk)^{\frac{1}{1-\omega}}) = O(a_0 + (LK)^{\frac{1}{1-\omega}}))$.

Proof We define the following series

$$b_{k+1} = \sum_{i=0}^{k} Lb_i^{\omega} + b_0$$

with an initial condition

$$b_0 = L^{\frac{1}{1-\omega}}.$$

We show by induction that $b_k \leq (L(k+1))^{\frac{1}{1-\omega}}$ for $k \geq 1$. For $k = 0$

$$b_0 = L^{\frac{1}{1-\omega}}(0+1)^{\frac{1}{1-\omega}} \leq L^{\frac{1}{1-\omega}}$$

We assume that the induction hypothesis holds for $k-1$ and prove it for k,

$$b_k = b_{k-1} + Lb_{k-1}^{\omega} \leq (Lk)^{\frac{1}{1-\omega}} + L(Lk)^{\frac{\omega}{1-\omega}} \leq L^{\frac{1}{1-\omega}} k^{\frac{\omega}{1-\omega}}(k+1) \leq (L(k+1))^{\frac{1}{1-\omega}}$$

and the claim is proved.

Now we lower bound b_k by $(L(k+1)/2)^{1/(1-\omega)}$. For $k = 0$

$$b_0 = L^{\frac{1}{1-\omega}} \geq (\frac{L}{2})^{\frac{1}{1-\omega}}$$

Assume that the induction hypothesis holds for $k-1$ and prove for k,

$$\begin{aligned} b_k &= b_{k-1} + L b_{k-1}^{\omega} = (Lk/2)^{\frac{1}{1-\omega}} + L(Lk/2)^{\frac{\omega}{1-\omega}} = L^{\frac{1}{1-\omega}}((k/2)^{\frac{1}{1-\omega}} + (k/2)^{\frac{\omega}{1-\omega}}) \\ &\geq L^{\frac{1}{1-\omega}}((k+1)/2)^{\frac{1}{1-\omega}}. \end{aligned}$$

For $a_0 > L^{\frac{1}{1-\omega}}$ we can view the series as starting at $b_k = a_0$. From the lower bound we know that the start point has moved $\Theta(a_0^{1-\omega}/L)$. Therefore we have a total complexity of $O((a_0^{1-\omega} + Lk)^{\frac{1}{1-\omega}}) = O(a_0 + (LK)^{\frac{1}{1-\omega}}))$.

■

The proof of Theorem 4 follows from Lemmas 27, 31,12 and 32. In the following lemma we relax the condition of the covering time.

Lemma 33 *Assume that from any start state with probability* $1/2$ *in L steps we perform all state action pairs. Then with probability* $1-\delta$*, from any start state we perform all state action pairs in* $L\log_2(1/\delta)$ *steps, for a run of length* $[L\log_2(1/\delta)]$.

Proof The proof follows from the fact that after k intervals of length L (where k is a natural number), the probability of not visiting all state action pairs is 2^{-k}. Since we have $k = [\log_2(1/\delta)]$ we get that the probability of failing is δ. ■

Corollary 34 *Assume that from any start state with probability* $1/2$ *in L steps we perform all state action pairs. Then with probability* $1-\delta$*, from any start state we perform all state action pairs in* $L\log(T/\delta)$ *steps, for a run of length* T.

9.2 Asynchronous Q-learning using a Linear Learning Rate

In this section we consider asynchronous Q-learning with a linear learning rate. Our main goal is to show that the size of the kth iteration is $L(1+\psi)\tau_k$, for any constant $\psi > 0$. The covering time property guarantees that in $(1+\psi)L\tau_k$ steps each pair of state action is performed at least $(1+\psi)\tau_k$ times. The sequence of times in this case is $\tau_{k+1} = \tau_k + (1+\psi)L\tau_k$, where the τ_0 will be defined latter. We first bound $Y_{t;\tau_k}$ and then bound the stochastic term $W_{t;\tau_k}$.

Lemma 35 *Consider asynchronous Q-learning with a polynomial learning rate and assume that for any* $t \geq \tau_k$ *we have* $Y_{t;\tau_k}(s,a) \leq D_k$*. Then for any* $t \geq \tau_k + (1+\psi)L\tau_k = \tau_{k+1}$ *we have* $Y_{t;\tau_k}(s,a) \leq (\gamma + \frac{2}{2+\psi}\beta)D_k$

Proof For each state-action pair (s,a) we are assured that $n(s,a,\tau_k,\tau_{k+1}) \geq (1+\psi)\tau_k$, since in an interval of $(1+\psi)L\tau_k$ steps each state-action pair is visited at least $(1+\psi)\tau_k$ times by the definition of the covering time. Using the fact that the $Y_{t;\tau_k}(s,a)$ are monotonically decreasing and deterministic (thus independent), we can apply the same argument as in the proof of Lemma 22. ■

The following Lemma enables the use of Azuma's inequality.

Lemma 36 *For any $t \geq \tau_k$ and $1 \leq l \leq t$ we have that $W^l_{t;\tau_k}(s,a)$ is a martingale sequence, which satisfies*

$$|W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)| \leq \frac{V_{max}}{n(s,a,0,t)}$$

Proof We first note that $W^l_{t;\tau_k}(s,a)$ is a martingale sequence, since

$$\begin{aligned} E[W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)|F_{\tau_k+l-1}] &= E[\eta^{k,t}_{\tau_k+l}(s,a)w_{\tau_k+l}(s,a)|F_{\tau_k+l-1}] \\ &= \eta^{k,t}_{\tau_k+l}E[w_{\tau_k+l}(s,a)|F_{\tau_k+l-1}] = 0. \end{aligned}$$

For a linear learning rate we have that $\eta^{k,t}_{\tau_k+l}(s,a) = 1/n(s,a,0,t)$, thus

$$|W^l_{t;\tau_k}(s,a) - W^{l-1}_{t;\tau_k}(s,a)| = \eta^{k,t}_{\tau_k+l}(s,a)|w_{\tau_k+l}(s,a)| \leq \frac{V_{max}}{n(s,a,0,t)}.$$

■

The following lemma bounds the value of the term $W_{t;\tau_k}$.

Lemma 37 *Consider asynchronous Q-learning with a linear learning rate. With probability at least $1-\frac{\delta}{m}$ we have for every state-action pair $|W_{t;\tau_k}(s,a)| \leq \frac{\psi}{2+\psi}\beta D_k$ for any $t \geq \tau_{k+1}$ and any positive constant ψ, i.e.*

$$Pr\left[\forall t \in [\tau_{k+1},\tau_{k+1}]: \ W_{t;\tau_k}(s,a) \leq \frac{\psi}{2+\psi}\beta D_k\right] \geq 1-\frac{\delta}{m}$$

given that $\tau_k \geq \Theta((\frac{V^2_{max}\ln(V_{max}|S|\,|A|m/(\delta\beta D_k\psi))}{\psi^2\beta^2 D_k^2}))$.

Proof By Lemma 36 we can apply Azuma's inequality on $W^{t-\tau_k+1}_{t;\tau_k}$ (We note that $W^{t-\tau_k+1}_{t;\tau_k} = W_{t;\tau_k}$) with $c_i = \Theta(\frac{V_{max}}{n(s,a,0,t}$) for any $t \geq \tau_{k+1}$. Therefore, we derive that

$$Pr[|W_{t;\tau_k}| \geq \tilde{\varepsilon}] \leq 2e^{\frac{-2\tilde{\varepsilon}^2}{\Sigma^t_{i=\tau_k, i\in T^{s,a}} c_i^2}} \leq 2e^{-c\frac{n(s,a,\tau_k,t)\tilde{\varepsilon}^2}{V^2_{max}}},$$

for some positive constant c. Let us define the following variable

$$\zeta_t(s,a) = \begin{cases} 1, & \alpha_t(s,a) \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

Using the union bound and the fact in an interval of length $(1+\psi)L\tau_k$ each state-action pair is visited at least $(1+\psi)\tau_k$ times, we get

$$\begin{aligned} Pr[\forall t \in [\tau_{k+1},\tau_{k+2}]: \ |W_{t;\tau_k}(s,a)| \geq \tilde{\varepsilon}] &\leq Pr[\forall t \geq ((1+\psi)L+1)\tau_k: \ |W_{t;\tau_k}(s,a)| \geq \tilde{\varepsilon}] \\ &\leq \sum_{t=((1+\psi)L+1)\tau_k}^{\infty} Pr[|W_{t;\tau_k}(s,a)| \geq \tilde{\varepsilon}] \end{aligned}$$

$$\begin{aligned}
&\leq \sum_{t=((1+\psi)L+1)\tau_k}^{\infty} \zeta_t(s,a) 2e^{-c\frac{n(s,a,0,t)\tilde{\varepsilon}^2}{V_{max}^2}} \\
&\leq 2e^{-c\frac{((1+\psi)\tau_k)\tilde{\varepsilon}^2}{V_{max}^2}} \sum_{t=0}^{\infty} e^{-\frac{t\tilde{\varepsilon}^2}{2V_{max}^2}} \\
&= \frac{2e^{-c\frac{(1+\psi)\tau_k\tilde{\varepsilon}^2}{V_{max}^2}}}{1-e^{\frac{-\tilde{\varepsilon}^2}{V_{max}^2}}} = \Theta\left(\frac{e^{-\frac{c'\tau_k\tilde{\varepsilon}^2}{V_{max}^2}} V_{max}^2}{\tilde{\varepsilon}^2}\right),
\end{aligned}$$

for some positive constant c'. Setting $\frac{\delta}{m|S||A|} = \Theta\left(\frac{e^{-\frac{c'\tau_k\tilde{\varepsilon}^2}{V_{max}^2}} V_{max}^2}{\tilde{\varepsilon}^2}\right)$, which hold for $\tau_k = \Theta\left(\frac{V_{max}^2 \ln(V_{max}|S||A|m/(\delta\tilde{\varepsilon}))}{\tilde{\varepsilon}^2}\right)$, and $\tilde{\varepsilon} = \frac{\psi}{2+\psi}\beta D_k$ assures us that for every $t \geq \tau_{k+1}$ (and as a result for any $t \in [\tau_{k+1}, \tau_{k+2}]$) with probability at least $1-\frac{\delta}{m}$ the statement holds at every state-action pair.

■

We have bounded for each iteration the time needed to achieve the desired precision level with probability $1-\frac{\delta}{m}$. The following lemma provides a bound for the error in all the iterations.

Lemma 38 *Consider synchronous Q-learning using a linear learning rate. With probability* $1-\delta$, *for every iteration* $k \in [1,m]$, *time* $t \in [\tau_{k+1}, \tau_{k+2}]$ *and any constant* $\psi > 0$ *we have* $|W_{t;\tau_k}| \leq \frac{\psi\beta D_k}{2+\psi}$, *i.e.,*

$$Pr\left[\forall k \in [1,m],\ \forall t \in [\tau_{k+1}, \tau_{k+2}] :\ |W_{t;\tau_k}| \leq \frac{\psi\beta D_k}{2+\psi}\right] \geq 1-\delta,$$

given that $\tau_0 = \Theta\left(\frac{V_{max}^2 \ln(V_{max}|S||A|m/(\delta\beta\varepsilon\psi))}{\psi^2\beta^2\varepsilon^2}\right)$.

Proof From Lemma 37 we know that

$$Pr\left[\forall t \in [\tau_{k+1}, \tau_{k+2}] :\ |W_{t;\tau_k}| \geq \frac{\psi\beta D_k}{2+\psi}\right] \leq \frac{\delta}{m}$$

Using the union bound we have that,

$$\begin{aligned}
&Pr\left[\forall k \leq m, \forall t \in [\tau_{k+1}, \tau_{k+2}]\ \ |W_{t;\tau_k}| \geq \frac{\psi\beta D_k}{2+\psi}\right] \\
&\quad \leq \sum_{k=1}^{m} Pr\left[\forall t \in [\tau_{k+1}, \tau_{k+2}]\ \ |W_{t;\tau_k}| \geq \frac{\psi\beta D_k}{2+\psi}\right] \leq \delta
\end{aligned}$$

■

Theorem 5 follows from Lemmas 35, 38, 12 and the fact that $a_{k+1} = a_k + (1+\psi)La_k = a_0((1+\psi)L+1)^k$.

10. Lower Bound for Q-learning using a Linear Learning Rate

In this section we show a lower bound for Q-learning with a linear learning rate, which is $O((\frac{1}{\varepsilon})^{\frac{1}{1-\gamma}})$. We consider the following MDP, denoted M_0, that has a single state s, a single action a, and a deterministic reward $R_{M_0}(s,a) = 0$. Since there is only one action in the MDP we denote $Q_t(s,a)$ as $Q_t(s)$. We initialize $Q_0(s) = 1$ and observe the time until $Q_t(s) \leq \varepsilon$.

Lemma 39 *Consider running synchronous Q-learning with linear learning rate on MDP M_0, when initializing $Q_0(s) = 1$. Then there is a time $t = c(\frac{1}{\varepsilon})^{\frac{1}{1-\gamma}}$ for some constant $c > 0$, such that $Q_t \geq \varepsilon$.*

Proof First we prove by induction on t that

$$Q_t(s) = \prod_{i=1}^{t-1} \frac{i+\gamma}{i+1}.$$

For $t = 1$ we have $Q_1(s) = (1-1/2)Q_0(s) + (1/2)\gamma Q_0(s) = (1+\gamma)/2$. Assume the hypothesis holds for $t-1$ and prove it for t. By definition,

$$Q_t(s) = (1 - \frac{1}{t})Q_{t-1}(s) + \frac{1}{t}\gamma Q_{t-1}(s) = \frac{t-1+\gamma}{t} Q_{t-1}(s).$$

In order to help us estimate this quantity we use the Γ function. Let

$$\Gamma(x+1,k) = \frac{1 \cdot 2 \cdots k}{(x+1)\cdot(x+2)\cdot\cdots\cdot(x+k)} k^x$$

The limit of $\Gamma(1+x,k)$, as k goes to infinity, is constant for any x. We can rewrite $Q_t(s)$ as

$$Q_t(s) = \frac{1}{\Gamma(\gamma+1,t)} \frac{t^\gamma}{t+1} = \Theta(t^{\gamma-1})$$

Therefore, there is a time $t = c(\frac{1}{\varepsilon})^{\frac{1}{1-\gamma}}$, for some constant $c > 0$, such that $Q_t(s) \geq \varepsilon$. ■

Acknowledgments

This research was supported in part by a grant from the Israel Science Foundation. Eyal Even-Dar was partially supported by the Deutsch Institute.

References

K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 68:357–367, 1967.

F. Beleznay, T. Grobler, and C. Szepesvari. Comparing value-function estimation algorithms in undiscounted problems. Technical Report TR-99-02, Mindmaker Ltd, 1999.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

V.S. Borkar and S.P. Meyn. The O.D.E method for convergence of stochstic approximation and reinforcement learning. *Siam J. Control*, 38 (2):447–69, 2000.

Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, *6*, 1994.

Michael Kearns and Satinder P. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In M.J. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 996–1002, 1999.

Michael L. Littman and Gaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning (ICML-96)*, pages 310–318, Bari, Italy, 1996. Morgan Kaufmann. URL `citeseer.nj.nec.com/littman96generalized.html`.

Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, NY, 1994.

R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press., Cambridge, MA., 1998.

C. Szepesvari. The asymptotic convergence-rate of Q-learning. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 1064–1070, 1998.

J. Tsitsiklis. Asynchronous stochastic approximation and Q-learning, 1994.

C. Watkins and P. Dyan. Q-learning. *Machine Learning*, 8(3/4):279–292, 1992.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge, England, 1989.

Journal of Machine Learning Research 5 (2004) 27-72 Submitted 10/02; Revised 8/03; Published 1/04

Learning the Kernel Matrix with Semidefinite Programming

Gert R.G. Lanckriet GERT@EECS.BERKELEY.EDU
Department of Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720, USA

Nello Cristianini NELLO@SUPPORT-VECTOR.NET
Department of Statistics
University of California
Davis, CA 95616, USA

Peter Bartlett BARTLETT@STAT.BERKELEY.EDU
Department of Electrical Engineering and Computer Science and Department of Statistics
Berkeley, CA 94720, USA

Laurent El Ghaoui ELGHAOUI@EECS.BERKELEY.EDU
Department of Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720, USA

Michael I. Jordan JORDAN@STAT.BERKELEY.EDU
Department of Electrical Engineering and Computer Science and Department of Statistics
University of California
Berkeley, CA 94720, USA

Editor: Bernhard Schölkopf

Abstract

Kernel-based learning algorithms work by embedding the data into a Euclidean space, and then searching for linear relations among the embedded data points. The embedding is performed implicitly, by specifying the inner products between each pair of points in the embedding space. This information is contained in the so-called kernel matrix, a symmetric and positive semidefinite matrix that encodes the relative positions of all points. Specifying this matrix amounts to specifying the geometry of the embedding space and inducing a notion of similarity in the input space—classical model selection problems in machine learning. In this paper we show how the kernel matrix can be learned from data via semidefinite programming (SDP) techniques. When applied to a kernel matrix associated with both training and test data this gives a powerful transductive algorithm—using the labeled part of the data one can learn an embedding also for the unlabeled part. The similarity between test points is inferred from training points and their labels. Importantly, these learning problems are convex, so we obtain a method for learning both the model class and the function without local minima. Furthermore, this approach leads directly to a convex method for learning the 2-norm soft margin parameter in support vector machines, solving an important open problem.

Keywords: kernel methods, learning kernels, transduction, model selection, support vector machines, convex optimization, semidefinite programming

1. Introduction

Recent advances in kernel-based learning algorithms have brought the field of machine learning closer to the desirable goal of autonomy—the goal of providing learning systems that require as little intervention as possible on the part of a human user. In particular, kernel-based algorithms are generally formulated in terms of convex optimization problems, which have a single global optimum and thus do not require heuristic choices of learning rates, starting configurations or other free parameters. There are, of course, statistical model selection problems to be faced within the kernel approach; in particular, the choice of the kernel and the corresponding feature space are central choices that must generally be made by a human user. While this provides opportunities for prior knowledge to be brought to bear, it can also be difficult in practice to find prior justification for the use of one kernel instead of another. It would be desirable to explore model selection methods that allow kernels to be chosen in a more automatic way based on data.

It is important to observe that we do not necessarily need to choose a kernel *function*, specifying the inner product between the images of all possible data points when mapped from an input space $\mathcal{X}$ to an appropriate feature space $\mathcal{F}$. Since kernel-based learning methods extract all information needed from inner products of training data points in $\mathcal{F}$, the values of the kernel function at pairs which are not present are irrelevant. So, there is no need to learn a kernel function over the entire sample space to specify the embedding of a finite training data set via a kernel function mapping. Instead, it is sufficient to specify a finite-dimensional *kernel matrix* (also known as a *Gram matrix*) that contains as its entries the inner products in $\mathcal{F}$ between all pairs of data points. Note also that it is possible to show that any symmetric positive semidefinite matrix is a valid Gram matrix, based on an inner product in some Hilbert space. This suggests viewing the model selection problem in terms of Gram matrices rather than kernel functions.

In this paper our main focus is *transduction*—the problem of completing the labeling of a partially labeled dataset. In other words, we are required to make predictions only at a finite set of points, which are specified a priori. Thus, instead of learning a function, we only need to learn a set of labels. There are many practical problems in which this formulation is natural—an example is the prediction of gene function, where the genes of interest are specified a priori, but the function of many of these genes is unknown.

We will address this problem by learning a kernel matrix corresponding to the entire dataset, a matrix that optimizes a certain cost function that depends on the available labels. In other words, we use the available labels to learn a good embedding, and we apply it to both the labeled and the unlabeled data. The resulting kernel matrix can then be used in combination with any of a number of existing learning algorithms that use kernels. One example that we discuss in detail is the support vector machine (SVM), where our methods yield a new transduction method for SVMs that scales polynomially with the number of test points. Furthermore, this approach will offer us a method to optimize the 2-norm soft margin parameter for these SVM learning algorithms, solving an important open problem.

All this can be done in full generality by using techniques from semidefinite programming (SDP), a branch of convex optimization that deals with the optimization of convex functions over the convex cone of positive semidefinite matrices, or convex subsets thereof. Any convex set of kernel matrices is a set of this kind. Furthermore, it turns out that many natural cost functions, motivated by error bounds, are convex in the kernel matrix.

A second application of the ideas that we present here is to the problem of combining data from multiple sources. Specifically, assume that each source is associated with a kernel function, such that a training set yields a set of kernel matrices. The tools that we develop in this paper make

it possible to optimize over the coefficients in a linear combination of such kernel matrices. These coefficients can then be used to form linear combinations of kernel functions in the overall classifier. Thus this approach allows us to combine possibly heterogeneous data sources, making use of the reduction of heterogeneous data types to the common framework of kernel matrices, and choosing coefficients that emphasize those sources most useful in the classification decision.

In Section 2, we recall the main ideas from kernel-based learning algorithms, and introduce a variety of criteria that can be used to assess the suitability of a kernel matrix: the hard margin, the 1-norm and 2-norm soft margin, and the kernel alignment. Section 3 reviews the basic concepts of semidefinite programming. In Section 4 we put these ideas together and consider the optimization of the various criteria over sets of kernel matrices. For a set of linear combinations of fixed kernel matrices, these optimization problems reduce to SDP. If the linear coefficients are constrained to be positive, they can be simplified even further, yielding a quadratically-constrained quadratic program, a special case of the SDP framework. If the linear combination contains the identity matrix, we obtain a convex method for optimizing the 2-norm soft margin parameter in support vector machines. Section 5 presents statistical error bounds that motivate one of our cost functions. Empirical results are reported in Section 6.

Notation

Vectors are represented in bold notation, e.g., $\mathbf{v} \in \mathbb{R}^n$, and their scalar components in italic script, e.g., $v_1, v_2, \ldots, v_n$. Matrices are represented in italic script, e.g., $X \in \mathbb{R}^{m \times n}$. For a square, symmetric matrix X, $X \succeq 0$ means that X is positive semidefinite, while $X \succ 0$ means that X is positive definite. For a vector $\mathbf{v}$, the notations $\mathbf{v} \geq 0$ and $\mathbf{v} > 0$ are understood componentwise.

2. Kernel Methods

Kernel-based learning algorithms (see, for example, Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) work by embedding the data into a Hilbert space, and searching for linear relations in such a space. The embedding is performed implicitly, by specifying the inner product between each pair of points rather than by giving their coordinates explicitly. This approach has several advantages, the most important deriving from the fact that the inner product in the embedding space can often be computed much more easily than the coordinates of the points themselves.

Given an input set $\mathcal{X}$, and an embedding space $\mathcal{F}$, we consider a map $\Phi : \mathcal{X} \to \mathcal{F}$. Given two points $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{x}_j \in \mathcal{X}$, the function that returns the inner product between their images in the space $\mathcal{F}$ is known as the *kernel function*.

Definition 1 *A* kernel *is a function k, such that $k(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$ for all $\mathbf{x}, \mathbf{z} \in \mathcal{X}$, where Φ is a mapping from $\mathcal{X}$ to an (inner product) feature space $\mathcal{F}$. A* kernel matrix *is a square matrix $K \in \mathbb{R}^{n \times n}$ such that $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for some $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and some kernel function k.*

The kernel matrix is also known as the Gram matrix. It is a symmetric, positive semidefinite matrix, and since it specifies the inner products between all pairs of points $\{\mathbf{x}_i\}_{i=1}^n$, it completely determines the relative positions of those points in the embedding space.

Since in this paper we will consider a *finite* input set $\mathcal{X}$, we can characterize kernel functions and matrices in the following simple way.

Proposition 2 *Every positive semidefinite and symmetric matrix is a kernel matrix. Conversely, every kernel matrix is symmetric and positive semidefinite.*

Notice that, if we have a kernel matrix, we do not need to know the kernel function, nor the implicitly defined map Φ, nor the coordinates of the points $\Phi(\mathbf{x}_i)$. We do not even need $\mathcal{X}$ to be a vector space; in fact in this paper it will be a generic finite set. We are guaranteed that the data are implicitly mapped to some Hilbert space by simply checking that the kernel matrix is symmetric and positive semidefinite.

The solutions sought by kernel-based algorithms such as the support vector machine (SVM) are affine functions in the feature space:

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b,$$

for some weight vector $\mathbf{w} \in \mathcal{F}$. The kernel can be exploited whenever the weight vector can be expressed as a linear combination of the training points, $\mathbf{w} = \sum_{i=1}^{n} \alpha_i \Phi(\mathbf{x}_i)$, implying that we can express f as

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b.$$

For example, for binary classification, we can use a thresholded version of $f(\mathbf{x})$, i.e., $\text{sign}\,(f(\mathbf{x}))$, as a decision function to classify unlabeled data. If $f(\mathbf{x})$ is positive, then we classify $\mathbf{x}$ as belonging to class $+1$; otherwise, we classify $\mathbf{x}$ as belonging to class -1. An important issue in applications is that of choosing a kernel k for a given learning task; intuitively, we wish to choose a kernel that induces the "right" metric in the input space.

2.1 Criteria Used in Kernel Methods

Kernel methods choose a function that is linear in the feature space by optimizing some criterion over the sample. This section describes several such criteria (see, for example, Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004). All of these criteria can be considered as measures of separation of the labeled data. We first consider the *hard margin* optimization problem.

Definition 3 Hard Margin *Given a labeled sample* $S_l = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$*, the hyperplane* $(\mathbf{w}_*, b_*)$ *that solves the optimization problem*

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \langle \mathbf{w}, \mathbf{w} \rangle \\ \textit{subject to} \quad & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1, \quad i = 1, \ldots, n, \end{aligned} \tag{1}$$

realizes the maximal margin classifier *with geometric margin* $\gamma = 1/\|\mathbf{w}_*\|_2$*, assuming it exists.*

Geometrically, γ corresponds to the distance between the convex hulls (the smallest convex sets that contain the data in each class) of the two classes (Bennett and Bredensteiner, 2000).

By transforming (1) into its corresponding Lagrangian dual problem, the solution is given by

$$\begin{aligned} \omega(K) &= 1/\gamma^2 \\ &= \langle \mathbf{w}_*, \mathbf{w}_* \rangle \\ &= \max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T \mathbf{e} - \boldsymbol{\alpha}^T G(K) \boldsymbol{\alpha} \;:\; \boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}^T \mathbf{y} = 0, \end{aligned} \tag{2}$$

where $\mathbf{e}$ is the n-vector of ones, $\boldsymbol{\alpha} \in \mathbb{R}^n$, $G(K)$ is defined by $G_{ij}(K) = [K]_{ij} y_i y_j = k(\mathbf{x}_i, \mathbf{x}_j) y_i y_j$, and $\boldsymbol{\alpha} \geq 0$ means $\alpha_i \geq 0, \; i = 1, \ldots, n$.

The hard margin solution exists only when the labeled sample is linearly separable in feature space. For a non-linearly-separable labeled sample S_l, we can define the *soft margin*. We consider the 1-norm and 2-norm soft margins.

Definition 4 1-norm Soft Margin *Given a labeled sample* $S_l = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$*, the hyperplane* $(\mathbf{w}_*, b_*)$ *that solves the optimization problem*

$$\begin{array}{ll} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \langle \mathbf{w}, \mathbf{w} \rangle + C \sum\limits_{i=1}^{n} \xi_i \\ \textit{subject to} & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{array} \tag{3}$$

realizes the 1-norm soft margin classifier *with geometric margin* $\gamma = 1/\|\mathbf{w}_*\|_2$*. This margin is also called the 1-norm soft margin.*

As for the hard margin, we can express the solution of (3) in a revealing way by considering the corresponding Lagrangian dual problem:

$$\begin{aligned} \omega_{S1}(K) &= \langle \mathbf{w}_*, \mathbf{w}_* \rangle + C \sum_{i=1}^{n} \xi_{i,*} \\ &= \max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T \mathbf{e} - \boldsymbol{\alpha}^T G(K) \boldsymbol{\alpha} \;:\; C \geq \boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}^T \mathbf{y} = 0. \end{aligned} \tag{4}$$

Definition 5 2-norm Soft Margin *Given a labeled sample* $S_l = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$*, the hyperplane* $(\mathbf{w}_*, b_*)$ *that solves the optimization problem*

$$\begin{array}{ll} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \langle \mathbf{w}, \mathbf{w} \rangle + C \sum\limits_{i=1}^{n} \xi_i^2 \\ \textit{subject to} & y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{array} \tag{5}$$

realizes the 2-norm soft margin classifier *with geometric margin* $\gamma = 1/\|\mathbf{w}_*\|_2$*. This margin is also called the 2-norm soft margin.*

Again, by considering the corresponding dual problem, the solution of (5) can be expressed as

$$\begin{aligned} \omega_{S2}(K) &= \langle \mathbf{w}_*, \mathbf{w}_* \rangle + C \sum_{i=1}^{n} \xi_{i,*}^2 \\ &= \max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T \mathbf{e} - \boldsymbol{\alpha}^T \left(G(K) + \frac{1}{C} I_n \right) \boldsymbol{\alpha} \;:\; \boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}^T \mathbf{y} = 0. \end{aligned} \tag{6}$$

With a fixed kernel, all of these criteria give upper bounds on misclassification probability (see, for example, Chapter 4 of Cristianini and Shawe-Taylor, 2000). Solving these optimization problems for a single kernel matrix is therefore a way of optimizing an upper bound on error probability.

In this paper, we allow the kernel matrix to be chosen from a class of kernel matrices. Previous error bounds are not applicable in this case. However, as we will see in Section 5, the margin γ can be used to bound the performance of support vector machines for transduction, with a linearly parameterized class of kernels.

We do not discuss further the merit of these different cost functions, deferring to the current literature on classification, where these cost functions are widely used with fixed kernels. Our goal is to show that these cost functions can be optimized—with respect to the kernel matrix—in an SDP setting.

Finally, we define the *alignment* of two kernel matrices (Cristianini et al., 2001, 2002). Given an (unlabeled) sample $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we use the following (Frobenius) inner product between Gram matrices, $\langle K_1, K_2 \rangle_F = \text{trace}(K_1^T K_2) = \sum_{i,j=1}^n k_1(\mathbf{x}_i, \mathbf{x}_j) k_2(\mathbf{x}_i, \mathbf{x}_j)$.

Definition 6 Alignment *The* (empirical) alignment *of a kernel k_1 with a kernel k_2 with respect to the sample S is the quantity*

$$\hat{A}(S, k_1, k_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}},$$

where K_i is the kernel matrix for the sample S using kernel k_i.

This can also be viewed as the cosine of the angle between two bi-dimensional vectors K_1 and K_2, representing the Gram matrices. Notice that we do not need to know the labels for the sample S in order to define the alignment of two kernels with respect to S. However, when the vector $\mathbf{y}$ of $\{\pm 1\}$ labels for the sample is known, we can consider $K_2 = \mathbf{y}\mathbf{y}^T$—the optimal kernel since $k_2(\mathbf{x}_i, \mathbf{x}_j) = 1$ if $y_i = y_j$ and $k_2(\mathbf{x}_i, \mathbf{x}_j) = -1$ if $y_i \neq y_j$. The alignment of a kernel k with k_2 with respect to S can be considered as a quality measure for k:

$$\hat{A}(S, K, \mathbf{y}\mathbf{y}^T) = \frac{\langle K, \mathbf{y}\mathbf{y}^T \rangle_F}{\sqrt{\langle K, K \rangle_F \langle \mathbf{y}\mathbf{y}^T, \mathbf{y}\mathbf{y}^T \rangle_F}} = \frac{\langle K, \mathbf{y}\mathbf{y}^T \rangle_F}{n\sqrt{\langle K, K \rangle_F}}, \tag{7}$$

since $\langle \mathbf{y}\mathbf{y}^T, \mathbf{y}\mathbf{y}^T \rangle_F = n^2$.

3. Semidefinite Programming (SDP)

In this section we review the basic definition of semidefinite programming as well as some important concepts and key results. Details and proofs can be found in Boyd and Vandenberghe (2003).

Semidefinite programming (Nesterov and Nemirovsky, 1994; Vandenberghe and Boyd, 1996; Boyd and Vandenberghe, 2003) deals with the optimization of convex functions over the convex cone[1] of symmetric, positive semidefinite matrices

$$\mathcal{P} = \left\{ X \in \mathbb{R}^{p \times p} \mid X = X^T, X \succeq 0 \right\},$$

or affine subsets of this cone. Given Proposition 2, $\mathcal{P}$ can be viewed as a search space for possible kernel matrices. This consideration leads to the key problem addressed in this paper—we wish to specify a convex cost function that will enable us to learn the optimal kernel matrix within $\mathcal{P}$ using semidefinite programming.

3.1 Definition of Semidefinite Programming

A *linear matrix inequality,* abbreviated LMI, is a constraint of the form

$$F(\mathbf{u}) := F_0 + u_1 F_1 + \ldots + u_q F_q \preceq 0.$$

Here, $\mathbf{u}$ is the vector of decision variables, and $F_0, \ldots, F_q$ are given symmetric $p \times p$ matrices. The notation $F(\mathbf{u}) \preceq 0$ means that the symmetric matrix F is negative semidefinite. Note that such a constraint is in general a *nonlinear* constraint; the term "linear" in the name LMI merely

1. $S \subseteq \mathbb{R}^d$ is a convex cone if and only if $\forall \mathbf{x}, \mathbf{y} \in S$ and $\forall \lambda, \mu \geq 0$, we have $\lambda \mathbf{x} + \mu \mathbf{y} \in S$.

emphasizes that F is affine in $\mathbf{u}$. Perhaps the most important feature of an LMI constraint is its convexity: the set of $\mathbf{u}$ that satisfy the LMI is a convex set.

An LMI constraint can be seen as an *infinite* set of scalar, affine constraints. Indeed, for a given $\mathbf{u}$, $F(\mathbf{u}) \preceq 0$ if and only if $\mathbf{z}^T F(\mathbf{u})\mathbf{z} \leq 0$ for every $\mathbf{z}$; every constraint indexed by $\mathbf{z}$ is an affine inequality, in the ordinary sense, i.e., the left-hand side of the inequality is a scalar, composed of a linear term in $\mathbf{u}$ and a constant term. Alternatively, using a standard result from linear algebra, we may state the constraint as

$$\forall Z \in \mathcal{P} \ : \ \mathrm{trace}(F(\mathbf{u})Z) \leq 0. \tag{8}$$

This can be seen by writing down the spectral decomposition of Z and using the fact that $\mathbf{z}^T F(\mathbf{u})\mathbf{z} \leq 0$ for every $\mathbf{z}$.

A semidefinite program (SDP) is an optimization problem with a linear objective, and linear matrix inequality and affine equality constraints.

Definition 7 *A semidefinite program is a problem of the form*

$$\begin{array}{rl} \min\limits_{\mathbf{u}} & \mathbf{c}^T\mathbf{u} \\ \textit{subject to} & F^j(\mathbf{u}) = F_0^j + u_1 F_1^j + \ldots + u_q F_q^j \preceq 0, \quad j = 1, \ldots, L \\ & A\mathbf{u} = \mathbf{b}, \end{array} \tag{9}$$

where $\mathbf{u} \in \mathbb{R}^q$ is the vector of decision variables, $\mathbf{c} \in \mathbb{R}^q$ is the objective vector, and matrices $F_i^j = (F_i^j)^T \in \mathbb{R}^{p\times p}$ are given.

Given the convexity of its LMI constraints, SDPs are convex optimization problems. The usefulness of the SDP formalism stems from two important facts. First, despite the seemingly very specialized form of SDPs, they arise in a host of applications; second, there exist interior-point algorithms to solve SDPs that have good theoretical and practical computational efficiency (Vandenberghe and Boyd, 1996).

One very useful tool to reduce a problem to an SDP is the so-called Schur complement lemma; it will be invoked repeatedly.

Lemma 8 (Schur Complement Lemma) *Consider the partitioned symmetric matrix*

$$X = X^T = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix},$$

where A, C are square and symmetric. If $\det(A) \neq 0$, we define the Schur complement of A in X by the matrix $S = C - B^T A^{-1} B$. The Schur Complement Lemma states that if $A \succ 0$, then $X \succeq 0$ if and only if $S \succeq 0$.

To illustrate how this lemma can be used to cast a nonlinear convex optimization problem as an SDP, consider the following result:

Lemma 9 *The quadratically constrained quadratic program (QCQP)*

$$\begin{array}{rl} \min\limits_{\mathbf{u}} & f_0(\mathbf{u}) \\ \textit{subject to} & f_i(\mathbf{u}) \leq 0, \quad i = 1, \ldots, M, \end{array} \tag{10}$$

with $f_i(\mathbf{u}) \triangleq (A_i\mathbf{u}+\mathbf{b}_i)^T(A_i\mathbf{u}+\mathbf{b}_i) - \mathbf{c}_i^T\mathbf{u} - d_i$*, is equivalent to the semidefinite programming problem*

$$\begin{array}{ll} \min\limits_{\mathbf{u},t} & t \\ \textit{subject to} & \begin{pmatrix} I & A_0\mathbf{u}+\mathbf{b_0} \\ (A_0\mathbf{u}+\mathbf{b_0})^T & \mathbf{c_0}^T\mathbf{u}+d_0+t \end{pmatrix} \succeq 0, \\ & \begin{pmatrix} I & A_i\mathbf{u}+\mathbf{b}_i \\ (A_i\mathbf{u}+\mathbf{b}_i)^T & \mathbf{c}_i^T\mathbf{u}+d_i \end{pmatrix} \succeq 0, \quad i=1,\ldots,M. \end{array} \tag{11}$$

This can be seen by rewriting the QCQP (10) as

$$\begin{array}{ll} \min\limits_{\mathbf{u},t} & t \\ \text{subject to} & t - f_0(\mathbf{u}) \geq 0, \\ & -f_i(\mathbf{u}) \geq 0, \quad i=1,\ldots,M. \end{array}$$

Note that for a fixed and feasible $\mathbf{u}$, $t = f_0(\mathbf{u})$ is the optimal solution. The convex quadratic inequality $t - f_0(\mathbf{u}) = (t + \mathbf{c_0}^T\mathbf{u} + d_0) - (A_0\mathbf{u}+\mathbf{b_0})^T I^{-1}(A_0\mathbf{u}+\mathbf{b_0}) \geq 0$ is now equivalent to the following LMI, using the Schur Complement Lemma 8:

$$\begin{pmatrix} I & A_0\mathbf{u}+\mathbf{b_0} \\ (A_0\mathbf{u}+\mathbf{b_0})^T & \mathbf{c_0}^T\mathbf{u}+d_0+t \end{pmatrix} \succeq 0.$$

Similar steps for the other quadratic inequality constraints finally yield (11), an SDP in standard form (9), equivalent to (10). This shows that a QCQP can be cast as an SDP. Of course, in practice a QCQP should not be solved using general-purpose SDP solvers, since the particular structure of the problem at hand can be efficiently exploited. The above show that QCQPs, and in particular linear programming problems, belong to the SDP family.

3.2 Duality

An important principle in optimization—perhaps even the most important principle—is that of *duality*. To illustrate duality in the case of an SDP, we will first review basic concepts in duality theory and then show how they can be extended to semidefinite programming. In particular, duality will give insights into optimality conditions for the semidefinite program.

Consider an optimization problem with n variables and m scalar constraints:

$$\begin{array}{ll} \min\limits_{\mathbf{u}} & f_0(\mathbf{u}) \\ \text{subject to} & f_i(\mathbf{u}) \leq 0, \quad i=1,\ldots,m, \end{array} \tag{12}$$

where $\mathbf{u} \in \mathbb{R}^n$. In the context of duality, problem (12) is called the *primal problem*; we denote its optimal value p^*. For now, we do not assume convexity.

Definition 10 Lagrangian *The* Lagrangian $\mathcal{L} : \mathbb{R}^{n+m} \to \mathbb{R}$ *corresponding to the minimization problem (12) is defined as*

$$\mathcal{L}(\mathbf{u},\boldsymbol{\lambda}) = f_0(\mathbf{u}) + \lambda_1 f_1(\mathbf{u}) + \ldots + \lambda_m f_m(\mathbf{u}).$$

The $\lambda_i \in \mathbb{R},\ i=1,\ldots,m$ *are called* Lagrange multipliers *or* dual variables.

One can now notice that

$$h(\mathbf{u}) = \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = \begin{cases} f_0(\mathbf{u}) & \text{if } f_i(\mathbf{u}) \leq 0,\ i = 1, \ldots, m \\ +\infty & \text{otherwise.} \end{cases}$$

So, the function $h(\mathbf{u})$ coincides with the objective $f_0(\mathbf{u})$ in regions where the constraints $f_i(\mathbf{u}) \leq 0$, $i = 1, \ldots, m$, are satisfied and $h(\mathbf{u}) = +\infty$ in infeasible regions. In other words, h acts as a "barrier" of the feasible set of the primal problem. Thus we can as well use $h(\mathbf{u})$ as objective function and rewrite the original primal problem (12) as an *unconstrained* optimization problem:

$$p^* = \min_{\mathbf{u}} \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}). \tag{13}$$

The notion of weak duality amounts to exchanging the "min" and "max" operators in the above formulation, resulting in a lower bound on the optimal value of the primal problem. Strong duality refers to the case when this exchange can be done without altering the value of the result: the lower bound is actually equal to the optimal value p^*. While weak duality always hold, even if the primal problem (13) is not convex, strong duality may not hold. However, for a large class of generic convex problems, strong duality holds.

Lemma 11 Weak duality *For all functions $f_0, f_1, \ldots, f_m$ in (12), not necessarily convex, we can exchange the* max *and the* min *and get a lower bound on p^*:*

$$d^* = \max_{\boldsymbol{\lambda} \geq 0} \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) \leq \min_{\mathbf{u}} \max_{\boldsymbol{\lambda} \geq 0} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = p^*.$$

The objective function of the maximization problem is now called the (Lagrange) dual function.

Definition 12 (Lagrange) dual function *The* (Lagrange) dual function $g : \mathbb{R}^m \to \mathbb{R}$ *is defined as*

$$\begin{aligned} g(\boldsymbol{\lambda}) &= \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) \\ &= \min_{\mathbf{u}}\ f_0(\mathbf{u}) + \lambda_1 f_1(\mathbf{u}) + \ldots + \lambda_m f_m(\mathbf{u}). \end{aligned} \tag{14}$$

Furthermore $g(\boldsymbol{\lambda})$ is concave, even if the $f_i(\mathbf{u})$ are not convex.

The concavity can easily be seen by considering first that for a given $\mathbf{u}$, $\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda})$ is an affine function of $\boldsymbol{\lambda}$ and hence is a concave function. Since $g(\boldsymbol{\lambda})$ is the pointwise minimum of such concave functions, it is concave.

Definition 13 Lagrange dual problem *The* Lagrange dual problem *is defined as*

$$d^* = \max_{\boldsymbol{\lambda} \geq 0} g(\boldsymbol{\lambda}).$$

Since $g(\boldsymbol{\lambda})$ is concave, this will always be a convex optimization problem, even if the primal is not. By *weak duality*, we always have $d^* \leq p^*$, even for non-convex problems. The value $p^* - d^*$ is called the duality gap. For **convex** problems, we usually (although not always) have *strong duality* at the optimum, i.e.,

$$d^* = p^*,$$

which is also referred to as a *zero duality gap*. For convex problems, a sufficient condition for zero duality gap is provided by *Slater's condition*:

Lemma 14 Slater's condition *If the primal problem (12) is convex and is* **strictly feasible**, *i.e.,* $\exists\ \mathbf{u}_0\ :\ f_i(\mathbf{u}_0) < 0,\ i = 1, \ldots, m$, *then*

$$p^* = d^*.$$

3.3 SDP Duality and Optimality Conditions

Consider for simplicity the case of an SDP with a single LMI constraint, and no affine equalities:

$$p^* = \min_{\mathbf{u}} \mathbf{c}^T\mathbf{u} \text{ subject to } F(\mathbf{u}) = F_0 + u_1F_1 + \ldots u_qF_q \preceq 0. \tag{15}$$

The general case of multiple LMI constraints and affine equalities can be handled by elimination of the latter and using block-diagonal matrices to represent the former as a single LMI.

The classical Lagrange duality theory outlined in the previous section does not directly apply here, since we are not dealing with finitely many constraints in scalar form; as noted earlier, the LMI constraint involves an infinite number of such constraints, of the form (8). One way to handle such constraints is to introduce a Lagrangian of the form

$$\mathcal{L}(\mathbf{u}, Z) = \mathbf{c}^T\mathbf{u} + \text{trace}(ZF(\mathbf{u})),$$

where the dual variable Z is now a symmetric matrix, of the same size as $F(\mathbf{u})$. We can check that such a Lagrange function fulfills the same role assigned to the function defined in Definition 10 for the case with scalar constraints. Indeed, if we define $h(\mathbf{u}) = \max_{Z\succeq 0} \mathcal{L}(\mathbf{u}, Z)$ then

$$h(\mathbf{u}) = \max_{Z\succeq 0} \mathcal{L}(\mathbf{u}, Z) = \begin{cases} \mathbf{c}^T\mathbf{u} & \text{if } F(\mathbf{u}) \preceq 0, \\ +\infty & \text{otherwise.} \end{cases}$$

Thus, $h(\mathbf{u})$ is a barrier for the primal SDP (15), that is, it coincides with the objective of (15) on its feasible set, and is infinite otherwise. Notice that to the LMI constraint we now associate a multiplier *matrix*, which will be constrained to the positive semidefinite cone.

In the above, we made use of the fact that, for a given symmetric matrix F,

$$\phi(F) := \sup_{Z\succeq 0} \text{trace}(ZF)$$

is $+\infty$ if F has a positive eigenvalue, and zero if F is negative semidefinite. This property is obvious for diagonal matrices, since in that case the variable Z can be constrained to be diagonal without loss of generality. The general case follows from the fact that if F has the eigenvalue decomposition $F = U\Lambda U^T$, where Λ is a diagonal matrix containing the eigenvalues of F, and U is orthogonal, then $\text{trace}(ZF) = \text{trace}(Z'\Lambda)$, where $Z' = U^TZU$ spans the positive semidefinite cone whenever Z does.

Using the above Lagrangian, one can cast the original problem (15) as an unconstrained optimization problem:

$$p^* = \min_{\mathbf{u}} \max_{Z\succeq 0} \mathcal{L}(\mathbf{u}, Z).$$

By weak duality, we obtain a lower bound on p^* by exchanging the min and max:

$$d^* = \max_{Z\succeq 0} \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, Z) \leq \min_{\mathbf{u}} \max_{Z\succeq 0} \mathcal{L}(\mathbf{u}, Z) = p^*.$$

The inner minimization problem is easily solved analytically, due to the special structure of the SDP. We obtain a closed form for the (Lagrange) dual function:

$$\begin{aligned} g(Z) = \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, Z) &= \min_{\mathbf{u}} \mathbf{c}^T\mathbf{u} + \text{trace}(ZF_0) + \sum_{i=1}^{q} u_i \text{ trace}(ZF_i) \\ &= \begin{cases} \text{trace}(ZF_0) & \text{if } c_i = -\text{trace}(ZF_i),\ i = 1, \ldots, q \\ -\infty & \text{otherwise.} \end{cases} \end{aligned}$$

The dual problem can be explicitly stated as follows:

$$d^* = \max_{Z \succeq 0} \min_{\mathbf{u}} \mathcal{L}(\mathbf{u}, Z) = \max_{Z} \operatorname{trace}(ZF_0) \;\text{ subject to } Z \succeq 0,\; c_i = -\operatorname{trace}(ZF_i),\; i = 1, \ldots, q. \quad (16)$$

We observe that the above problem is an SDP, with a single LMI constraint and q affine equalities in the matrix dual variable Z.

While weak duality always holds, strong duality may not, even for SDPs. Not surprisingly, a Slater-type condition ensures strong duality. Precisely, if the primal SDP (15) is strictly feasible, that is, there exists a $\mathbf{u}_0$ such that $F(\mathbf{u}_0) \prec 0$, then $p^* = d^*$. If, in addition, the dual problem is also strictly feasible, meaning that there exists a $Z \succ 0$ such that $c_i = \operatorname{trace}(ZF_i)$, $i = 1, \ldots, q$, then both primal and dual optimal values are attained by some optimal pair $(\mathbf{u}^*, Z^*)$. In that case, we can characterize such optimal pairs as follows. In view of the equality constraints of the dual problem, the duality gap can be expressed as

$$\begin{aligned} p^* - d^* &= \mathbf{c}^T \mathbf{u}^* - \operatorname{trace}(Z^* F_0) \\ &= -\operatorname{trace}(Z^* F(\mathbf{u}^*)). \end{aligned}$$

A zero duality gap is equivalent to $\operatorname{trace}(Z^* F(\mathbf{u}^*)) = 0$, which in turn is equivalent to $Z^* F(\mathbf{u}^*) = O$, where O denotes the zero matrix, since the product of a positive semidefinite and a negative semidefinite matrix has zero trace if and only if it is zero.

To summarize, consider the SDP (15) and its Lagrange dual (16). If either problem is strictly feasible, then they share the same optimal value. If both problems are strictly feasible, then the optimal values of both problems are attained and coincide. In this case, a primal-dual pair $(\mathbf{u}^*, Z^*)$ is optimal if and only if

$$\begin{aligned} &F(\mathbf{u}^*) \preceq 0, \\ &Z^* \succeq 0, \\ &c_i = -\operatorname{trace}(Z^* F_i), \;\; i = 1, \ldots, q, \\ &Z^* F(\mathbf{u}^*) = O. \end{aligned}$$

The above conditions represent the expression of the general Karush-Kuhn-Tucker (KKT) conditions in the semidefinite programming setting. The first three sets of conditions express that $\mathbf{u}^*$ and Z^* are feasible for their respective problems; the last condition expresses a complementarity condition.

For a pair of strictly feasible primal-dual SDPs, solving the primal minimization problem is equivalent to maximizing the dual problem and both can thus be considered simultaneously. Algorithms indeed make use of this relationship and use the duality gap as a stopping criterion. A general-purpose program such as SeDuMi (Sturm, 1999) handles those problems efficiently. This code uses interior-point methods for SDP (Nesterov and Nemirovsky, 1994); these methods have a worst-case complexity of $O(q^2 p^{2.5})$ for the general problem (15). In practice, problem structure can be exploited for great computational savings: e.g., when $F(\mathbf{u}) \in \mathbb{R}^{p \times p}$ consists of L diagonal blocks of size p_i, $i = 1, \ldots, L$, these methods have a worst-case complexity of $O(q^2(\sum_{i=1}^{L} p_i^2)p^{0.5})$ (Vandenberghe and Boyd, 1996).

4. Algorithms for Learning Kernels

We work in a transduction setting, where some of the data (the training set $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$) are labeled, and the remainder (the test set $T_{n_t} = \{\mathbf{x}_{n_{tr}+1}, \ldots, \mathbf{x}_{n_{tr}+n_t}\}$) are unlabeled, and the

aim is to predict the labels of the test data. In this setting, optimizing the kernel corresponds to choosing a kernel matrix. This matrix has the form

$$K = \begin{pmatrix} K_{tr} & K_{tr,t} \\ K_{tr,t}^T & K_t \end{pmatrix}, \tag{17}$$

where $K_{ij} = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$, $i, j = 1, \ldots, n_{tr}, n_{tr}+1, \ldots, n_{tr}+n_t$. By optimizing a cost function over the "training-data block" K_{tr}, we want to learn the optimal mixed block $K_{tr,t}$ and the optimal "test-data block" K_t.

This implies that training and test-data blocks must somehow be entangled: tuning training-data entries in K (to optimize their embedding) should imply that test-data entries are automatically tuned in some way as well. This can be achieved by constraining the search space of possible kernel matrices: we control the capacity of the search space of possible kernel matrices in order to prevent overfitting and achieve good generalization on test data.

We first consider a general optimization problem in which the kernel matrix K is restricted to a convex subset $\mathcal{K}$ of $\mathcal{P}$, the positive semidefinite cone. We then consider two specific examples. The first is the set of positive semidefinite matrices with bounded trace that can be expressed as a linear combination of kernel matrices from the set $\{K_1, \ldots, K_m\}$. That is, $\mathcal{K}$ is the set of matrices K satisfying

$$\begin{aligned} K &= \sum_{i=1}^m \mu_i K_i, \\ K &\succeq 0, \\ \operatorname{trace}(K) &\leq c. \end{aligned} \tag{18}$$

In this case, the set $\mathcal{K}$ lies in the intersection of a low-dimensional linear subspace with the positive semidefinite cone $\mathcal{P}$. Geometrically this can be viewed as computing all embeddings (for every K_i), in disjoint feature spaces, and then weighting these. The set $\{K_1, \ldots, K_m\}$ could be a set of initial "guesses" of the kernel matrix, e.g., linear, Gaussian or polynomial kernels with different kernel parameter values. Instead of fine-tuning the kernel parameter for a given kernel using cross-validation, one can now evaluate the given kernel for a range of kernel parameters and then optimize the weights in the linear combination of the obtained kernel matrices. Alternatively, the K_i could be chosen as the rank-one matrices $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ a subset of the eigenvectors of K_0, an initial kernel matrix, or with $\mathbf{v}_i$ some other set of orthogonal vectors. A practically important form is the case in which a diverse set of possibly good Gram matrices K_i (similarity measures/representations) has been constructed, e.g., using heterogeneous data sources. The challenge is to combine these measures into one optimal similarity measure (embedding), to be used for learning.

The second example of a restricted set $\mathcal{K}$ of kernels is the set of positive semidefinite matrices with bounded trace that can be expressed as a linear combination of kernel matrices from the set $\{K_1, \ldots, K_m\}$, but with the parameters μ_i constrained to be non-negative. That is, $\mathcal{K}$ is the set of matrices K satisfying

$$\begin{aligned} K &= \sum_{i=1}^m \mu_i K_i, \\ \mu_i &\geq 0 \qquad i \in \{1, \ldots, m\} \\ K &\succeq 0, \\ \operatorname{trace}(K) &\leq c. \end{aligned}$$

This further constrains the class of functions that can be represented. It has two advantages: we shall see that the corresponding optimization problem has significantly reduced computational complexity, and it is more convenient for studying the statistical properties of a class of kernel matrices.

As we will see in Section 5, we can estimate the performance of support vector machines for transduction using properties of the class $\mathcal{K}$. As explained in Section 2, we can use a thresholded version of $f(\mathbf{x})$, i.e., $\text{sign}\,(f(\mathbf{x}))$, as a binary classification decision. Using this decision function, we will prove that the proportion of errors on the test data T_n (where, for convenience, we suppose that training and test data have the same size $n_{tr} = n_t = n$) is, with probability $1 - \delta$ (over the random draw of the training set S_n and test set T_n), bounded by

$$\frac{1}{n}\sum_{i=1}^{n} \max\{1 - y_i f(\mathbf{x}_i), 0\} + \frac{1}{\sqrt{n}}\left(4 + \sqrt{2\log(1/\delta)} + \sqrt{\frac{\mathcal{C}(\mathcal{K})}{n\gamma^2}}\right), \tag{19}$$

where γ is the 1-norm soft margin on the data and $\mathcal{C}(\mathcal{K})$ is a certain measure of the complexity of the kernel class $\mathcal{K}$. For instance, for the class $\mathcal{K}$ of positive linear combinations defined above, $\mathcal{C}(\mathcal{K}) \leq mc$, where m is the number of kernel matrices in the combination and c is the bound on the trace. So, the proportion of errors on the test data is bounded by the average error on the training set and a complexity term, determined by the richness of the class $\mathcal{K}$ and the margin γ. Good generalization can thus be expected if the error on the training set is small, while having a large margin and a class $\mathcal{K}$ that is not too rich.

The next section presents the main optimization result of the paper: minimizing a generalized performance measure $\omega_{C,\tau}(K)$ with respect to the kernel matrix K can be realized in a semidefinite programming framework. Afterwards, we prove a second general result showing that minimizing $\omega_{C,\tau}(K)$ with respect to a kernel matrix K, constrained to the linear subspace $K = \sum_{i=1}^{m} \mu_i K_i$ with $\boldsymbol{\mu} \geq 0$, leads to a quadratically constrained quadratic programming (QCQP) problem. Maximizing the margin of a hard margin SVM with respect to K, as well as both soft margin cases can then be treated as specific instances of this general result and will be discussed in later sections.

4.1 General Optimization Result

In this section, we first of all show that minimizing the generalized performance measure

$$\omega_{C,\tau}(K) = \max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T(G(K) + \tau I)\boldsymbol{\alpha} \;:\; C \geq \boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}^T\mathbf{y} = 0, \tag{20}$$

with $\tau \geq 0$, on the training data with respect to the kernel matrix K, in some convex subset $\mathcal{K}$ of positive semidefinite matrices with trace equal to c,

$$\min_{K \in \mathcal{K}} \omega_{C,\tau}(K_{tr}) \qquad \text{s.t. trace}(K) = c, \tag{21}$$

can be realized in a semidefinite programming framework.

We first note a fundamental property of the generalized performance measure, a property that is crucial for the remainder of the paper.

Proposition 15 *The quantity*

$$\omega_{C,\tau}(K) = \max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T(G(K) + \tau I)\boldsymbol{\alpha} \;:\; C \geq \boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}^T\mathbf{y} = 0,$$

is convex in K.

This is easily seen by considering first that $2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T(G(K)+\tau I)\boldsymbol{\alpha}$ is an affine function of K, and hence is a convex function as well. Secondly, we notice that $\omega_{C,\tau}(K)$ is the pointwise maximum of such convex functions and is thus convex. The constraints $C \geq \boldsymbol{\alpha} \geq 0, \ \ \boldsymbol{\alpha}^T\mathbf{y} = 0$ are obviously convex.

Problem (21) is now a convex optimization problem. The following theorem shows that, for a suitable choice of the set $\mathcal{K}$, this problem can be cast as an SDP.

Theorem 16 *Given a labeled sample $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted $\mathbf{y} \in \mathbb{R}^{n_{tr}}$, the kernel matrix $K \in \mathcal{K}$ that optimizes (21), with $\tau \geq 0$, can be found by solving the following convex optimization problem:*

$$\begin{array}{ll} \min\limits_{K,t,\lambda,\boldsymbol{\nu},\boldsymbol{\delta}} & t \\ \textit{subject to} & \textit{trace}(K) = c, \\ & K \in \mathcal{K}, \\ & \begin{pmatrix} G(K_{tr}) + \tau I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T & t - 2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0, \\ & \boldsymbol{\delta} \geq 0. \end{array} \tag{22}$$

Proof We begin by substituting $\omega_{C,\tau}(K_{tr})$, as defined in (20), into (21), which yields

$$\min_{K \in \mathcal{K}} \max_{\boldsymbol{\alpha}} \ 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T(G(K_{tr}) + \tau I_{n_{tr}})\boldsymbol{\alpha} \ : \ C \geq \boldsymbol{\alpha} \geq 0, \ \ \boldsymbol{\alpha}^T\mathbf{y} = 0, \ \ \text{trace}(K) = c, \tag{23}$$

with c a constant. Assume that $K_{tr} \succ 0$, hence $G(K_{tr}) \succ 0$ and $G(K_{tr}) + \tau I_{n_{tr}} \succ 0$ since $\tau \geq 0$ (the following can be extended to the general semidefinite case). From Proposition 15, we know that $\omega_{C,\tau}(K_{tr})$ is convex in K_{tr} and thus in K. Given the convex constraints in (23), the optimization problem is thus certainly convex in K. We write this as

$$\begin{array}{ll} \min\limits_{K \in \mathcal{K}, t} t \ : & t \geq \max\limits_{\boldsymbol{\alpha}} \ 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T(G(K_{tr}) + \tau I_{n_{tr}})\boldsymbol{\alpha}, \\ & C \geq \boldsymbol{\alpha} \geq 0, \ \ \boldsymbol{\alpha}^T\mathbf{y} = 0, \ \ \text{trace}(K) = c. \end{array} \tag{24}$$

We now express the constraint $t \geq \max_{\boldsymbol{\alpha}} \ 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T(G(K_{tr}) + \tau I_{n_{tr}})\boldsymbol{\alpha}$ as an LMI using duality. In particular, duality will allow us to drop the minimization and the Schur complement lemma then yields an LMI.

Define the Lagrangian of the maximization problem (20) by

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\nu}, \lambda, \boldsymbol{\delta}) = 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T(G(K_{tr}) + \tau I_{n_{tr}})\boldsymbol{\alpha} + 2\boldsymbol{\nu}^T\boldsymbol{\alpha} + 2\lambda\mathbf{y}^T\boldsymbol{\alpha} + 2\boldsymbol{\delta}^T(C\mathbf{e} - \boldsymbol{\alpha}),$$

where $\lambda \in \mathbb{R}$ and $\boldsymbol{\nu}, \boldsymbol{\delta} \in \mathbb{R}^{n_{tr}}$. By duality, we have

$$\omega_{C,\tau}(K_{tr}) = \max_{\boldsymbol{\alpha}} \min_{\boldsymbol{\nu} \geq 0, \boldsymbol{\delta} \geq 0, \lambda} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\nu}, \lambda, \boldsymbol{\delta}) = \min_{\boldsymbol{\nu} \geq 0, \boldsymbol{\delta} \geq 0, \lambda} \max_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\nu}, \lambda, \boldsymbol{\delta}).$$

Since $G(K_{tr}) + \tau I_{n_{tr}} \succ 0$, at the optimum we have

$$\boldsymbol{\alpha} = (G(K_{tr}) + \tau I_{n_{tr}})^{-1}(\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y}),$$

and we can form the dual problem

$$\omega_{C,\tau}(K_{tr}) = \min_{\boldsymbol{\nu}, \boldsymbol{\delta}, \lambda} \ (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T(G(K_{tr}) + \tau I_{n_{tr}})^{-1}(\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y}) + 2C\boldsymbol{\delta}^T\mathbf{e} \ : \ \boldsymbol{\nu} \geq 0, \boldsymbol{\delta} \geq 0.$$

This implies that for any $t > 0$, the constraint $\omega_{C,\tau}(K_{tr}) \leq t$ holds if and only if there exist $\boldsymbol{\nu} \geq 0, \boldsymbol{\delta} \geq 0$ and λ such that

$$(\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T (G(K_{tr}) + \tau I_{n_{tr}})^{-1} (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y}) + 2C\boldsymbol{\delta}^T\mathbf{e} \leq t,$$

or, equivalently (using the Schur complement lemma), such that

$$\begin{pmatrix} G(K_{tr}) + \tau I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T & t - 2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0$$

holds. Taking this into account, (24) can be expressed as

$$\begin{aligned} \min_{K,t,\lambda,\boldsymbol{\nu},\boldsymbol{\delta}} \quad & t \\ \text{subject to} \quad & \operatorname{trace}(K) = c, \\ & K \in \mathcal{K}, \\ & \begin{pmatrix} G(K_{tr}) + \tau I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T & t - 2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0, \\ & \boldsymbol{\delta} \geq 0, \end{aligned}$$

which yields (22). Notice that $\boldsymbol{\nu} \geq 0 \Leftrightarrow \operatorname{diag}(\boldsymbol{\nu}) \succeq 0$, and is thus an LMI; similarly for $\boldsymbol{\delta} \geq 0$. ■

Notice that if $\mathcal{K} = \{K \succeq 0\}$, this optimization problem is an SDP in the standard form (9). Of course, in that case there is no constraint to ensure entanglement of training and test-data blocks. Indeed, it is easy to see that the criterion would be optimized with a test matrix $K_t = O$.

Consider the constraint $\mathcal{K} = \operatorname{span}\{K_1, \ldots, K_m\} \cap \{K \succeq 0\}$. We obtain the following convex optimization problem:

$$\begin{aligned} \min_{K} \quad & \omega_{C,\tau}(K_{tr}) \\ \text{subject to} \quad & \operatorname{trace}(K) = c, \\ & K \succeq 0, \\ & K = \sum_{i=1}^{m} \mu_i K_i, \end{aligned} \tag{25}$$

which can be written in the standard form of a semidefinite program, in a manner analogous to (22):

$$\begin{aligned} \min_{\boldsymbol{\mu},t,\lambda,\boldsymbol{\nu},\boldsymbol{\delta}} \quad & t \\ \text{subject to} \quad & \operatorname{trace}\left(\sum_{i=1}^{m} \mu_i K_i\right) = c, \\ & \sum_{i=1}^{m} \mu_i K_i \succeq 0, \\ & \begin{pmatrix} G(\sum_{i=1}^{m} \mu_i K_{i,tr}) + \tau I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} - \boldsymbol{\delta} + \lambda\mathbf{y})^T & t - 2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0, \\ & \boldsymbol{\delta} \geq 0. \end{aligned} \tag{26}$$

To solve this general optimization problem, one has to solve a semidefinite programming problem. General-purpose programs such as SeDuMi (Sturm, 1999) use interior-point methods to solve SDP problems (Nesterov and Nemirovsky, 1994). These methods are polynomial time. However, applying the complexity results mentioned in Section 3.3 leads to a worst-case complexity $O\big((m+n_{tr})^2(n^2+n_{tr}^2)(n+n_{tr})^{0.5}\big)$, or roughly $O\left((m+n_{tr})^2 n^{2.5}\right)$, in this particular case.

Consider a further restriction on the set of kernel matrices, where the matrices are restricted to positive linear combinations of kernel matrices $\{K_1,\ldots,K_m\}\cap\{K\succeq 0\}$:

$$K=\sum_{i=1}^{m}\mu_i K_i, \qquad \boldsymbol{\mu}\geq 0.$$

For this restricted linear subspace of the positive semidefinite cone $\mathcal{P}$, we can prove the following theorem:

Theorem 17 *Given a labeled sample $S_{n_{tr}}=\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_{n_{tr}},y_{n_{tr}})\}$ with the set of labels denoted $\mathbf{y}\in\mathbb{R}^{n_{tr}}$, the kernel matrix $K=\sum_{i=1}^{m}\mu_i K_i$ that optimizes (21), with $\tau\geq 0$, under the additional constraint $\boldsymbol{\mu}\geq 0$ can be found by solving the following convex optimization problem, and considering its dual solution:*

$$\begin{aligned} \max_{\boldsymbol{\alpha},t} \quad & 2\boldsymbol{\alpha}^T\mathbf{e}-\tau\boldsymbol{\alpha}^T\boldsymbol{\alpha}-ct \\ \textit{subject to} \quad & t\geq\frac{1}{r_i}\boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha}, \quad i=1,\ldots,m \\ & \boldsymbol{\alpha}^T\mathbf{y}=0, \\ & C\geq\boldsymbol{\alpha}\geq 0, \end{aligned} \tag{27}$$

where $\mathbf{r}\in\mathbb{R}^m$ with trace$(K_i)=r_i$.

Proof Solving problem (21) subject to $K=\sum_{i=1}^{m}\mu_i K_i$, with $K_i\succeq 0$, and the extra constraint $\boldsymbol{\mu}\geq 0$ yields

$$\begin{aligned} \min_{K} \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0,\boldsymbol{\alpha}^T\mathbf{y}=0} \quad & 2\boldsymbol{\alpha}^T\mathbf{e}-\boldsymbol{\alpha}^T(G(K_{tr})+\tau I_{n_{tr}})\boldsymbol{\alpha} \\ \text{subject to} \quad & \text{trace}(K)=c, \\ & K\succeq 0, \\ & K=\sum_{i=1}^{m}\mu_i K_i, \\ & \boldsymbol{\mu}\geq 0, \end{aligned}$$

when $\omega_{C,\tau}(K_{tr})$ is expressed using (20). We can omit the second constraint, because this is implied by the last two constraints, if $K_i\succeq 0$. The problem then reduces to

$$\begin{aligned} \min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha}\,:\,C\geq\boldsymbol{\alpha}\geq 0,\boldsymbol{\alpha}^T\mathbf{y}=0} \quad & 2\boldsymbol{\alpha}^T\mathbf{e}-\boldsymbol{\alpha}^T(G(\sum_{i=1}^{m}\mu_i K_{i,tr})+\tau I_{n_{tr}})\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\mu}^T\mathbf{r}=c, \\ & \boldsymbol{\mu}\geq 0, \end{aligned}$$

where $K_{i,tr} = K_i(1:n_{tr}, 1:n_{tr})$. We can write this as

$$\begin{aligned}
&\min_{\boldsymbol{\mu} \,:\, \boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} \quad \max_{\boldsymbol{\alpha} \,:\, C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} 2\boldsymbol{\alpha}^T\mathbf{e} - \boldsymbol{\alpha}^T\left(\operatorname{diag}(\mathbf{y})(\sum_{i=1}^m \mu_i K_{i,tr})\operatorname{diag}(\mathbf{y}) + \tau I_{n_{tr}}\right)\boldsymbol{\alpha} \\
&= \min_{\boldsymbol{\mu} \,:\, \boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} \quad \max_{\boldsymbol{\alpha} \,:\, C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} 2\boldsymbol{\alpha}^T\mathbf{e} - \sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T\operatorname{diag}(\mathbf{y})K_{i,tr}\operatorname{diag}(\mathbf{y})\boldsymbol{\alpha} - \tau\boldsymbol{\alpha}^T\boldsymbol{\alpha} \\
&= \min_{\boldsymbol{\mu} \,:\, \boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} \quad \max_{\boldsymbol{\alpha} \,:\, C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} 2\boldsymbol{\alpha}^T\mathbf{e} - \sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha} - \tau\boldsymbol{\alpha}^T\boldsymbol{\alpha} \\
&= \max_{\boldsymbol{\alpha} \,:\, C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} \quad \min_{\boldsymbol{\mu} \,:\, \boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} 2\boldsymbol{\alpha}^T\mathbf{e} - \sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha} - \tau\boldsymbol{\alpha}^T\boldsymbol{\alpha},
\end{aligned}$$

with $G(K_{i,tr}) = \operatorname{diag}(\mathbf{y})K_{i,tr}\operatorname{diag}(\mathbf{y})$. The interchange of the order of the minimization and the maximization is justified (see, e.g., Boyd and Vandenberghe, 2003) because the objective is convex in $\boldsymbol{\mu}$ (it is linear in $\boldsymbol{\mu}$) and concave in $\boldsymbol{\alpha}$, because the minimization problem is strictly feasible in $\boldsymbol{\mu}$, and the maximization problem is strictly feasible in $\boldsymbol{\alpha}$ (we can skip the case for all elements of $\mathbf{y}$ having the same sign, because we cannot even define a margin in such a case). We thus obtain

$$\begin{aligned}
&\max_{\boldsymbol{\alpha} \,:\, C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} \quad \min_{\boldsymbol{\mu} \,:\, \boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} 2\boldsymbol{\alpha}^T\mathbf{e} - \sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha} - \tau\boldsymbol{\alpha}^T\boldsymbol{\alpha} \\
&= \max_{\boldsymbol{\alpha} \,:\, C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} \left[2\boldsymbol{\alpha}^T\mathbf{e} - \tau\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \max_{\boldsymbol{\mu} \,:\, \boldsymbol{\mu}\geq 0, \boldsymbol{\mu}^T\mathbf{r}=c} \left(\sum_{i=1}^m \mu_i \boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha}\right)\right] \\
&= \max_{\boldsymbol{\alpha} \,:\, C\geq\boldsymbol{\alpha}\geq 0, \boldsymbol{\alpha}^T\mathbf{y}=0} \left[2\boldsymbol{\alpha}^T\mathbf{e} - \tau\boldsymbol{\alpha}^T\boldsymbol{\alpha} - \max_i \left(\frac{c}{r_i}\boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha}\right)\right].
\end{aligned}$$

Finally, this can be reformulated as

$$\begin{aligned}
\max_{\boldsymbol{\alpha},t} \quad & 2\boldsymbol{\alpha}^T\mathbf{e} - \tau\boldsymbol{\alpha}^T\boldsymbol{\alpha} - ct \\
\text{subject to} \quad & t \geq \frac{1}{r_i}\boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha}, \quad i = 1, \ldots, m \\
& \boldsymbol{\alpha}^T\mathbf{y} = 0, \\
& C \geq \boldsymbol{\alpha} \geq 0,
\end{aligned}$$

which proves the theorem. ■

This convex optimization problem, a QCQP more precisely, is a special instance of an SOCP (second-order cone programming problem), which is in turn a special form of SDP (Boyd and Vandenberghe, 2003). SOCPs can be solved efficiently with programs such as SeDuMi (Sturm, 1999) or Mosek (Andersen and Andersen, 2000). These codes use interior-point methods (Nesterov and Nemirovsky, 1994) which yield a worst-case complexity of $O(mn_{tr}^3)$. This implies a major improvement compared to the worst-case complexity of a general SDP. Furthermore, the codes simultaneously solve the above problem and its dual form. They thus return optimal values for the dual variables as well—this allows us to obtain the optimal weights μ_i, for $i = 1, \ldots, m$.

4.2 Hard Margin

In this section, we show how maximizing the margin of a hard margin SVM with respect to the kernel matrix can be realized in the semidefinite programming framework derived in Theorem 16.

Inspired by (19), let us try to find the kernel matrix K in some convex subset $\mathcal{K}$ of positive semidefinite matrices for which the corresponding embedding shows maximal margin on the training data, keeping the trace of K constant:

$$\min_{K \in \mathcal{K}} \omega(K_{tr}) \qquad \text{s.t. } \operatorname{trace}(K) = c. \tag{28}$$

Note that $\omega(K_{tr}) = \omega_{\infty,0}(K_{tr})$. From Proposition 15, we then obtain the following important result:

Corollary 18 *The quantity*

$$\omega(K) = \max_{\boldsymbol{\alpha}} \; 2\boldsymbol{\alpha}^T \mathbf{e} - \boldsymbol{\alpha}^T G(K) \boldsymbol{\alpha} \;:\; \boldsymbol{\alpha} \geq 0, \;\; \boldsymbol{\alpha}^T \mathbf{y} = 0,$$

is convex in K.

So, a fundamental property of the inverse margin is that it is convex in K. This is essential, since it allows us to optimize this quantity in a convex framework. The following theorem shows that, for a suitable choice of the set $\mathcal{K}$, this convex optimization problem can be cast as an SDP.

Theorem 19 *Given a linearly separable labeled sample $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted $\mathbf{y} \in \mathbb{R}^{n_{tr}}$, the kernel matrix $K \in \mathcal{K}$ that optimizes (28) can be found by solving the following problem:*

$$\begin{array}{ll} \min\limits_{K,t,\lambda,\boldsymbol{\nu}} & t \\ \textit{subject to} & \textit{trace}(K) = c, \\ & K \in \mathcal{K}, \\ & \begin{pmatrix} G(K_{tr}) & \mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y})^T & t \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0. \end{array} \tag{29}$$

Proof Observe $\omega(K_{tr}) = \omega_{\infty,0}(K_{tr})$. Apply Theorem 16 for $C = \infty$ and $\tau = 0$. ■

If $\mathcal{K} = \{K \succeq 0\}$, there is no constraint to ensure that a large margin on the training data will give a large margin on the test data: a test matrix $K_t = O$ would optimize the criterion.

If we restrict the kernel matrix to a linear subspace $\mathcal{K} = \operatorname{span}\{K_1, \ldots, K_m\} \cap \{K \succeq 0\}$, we obtain

$$\begin{array}{rl} \min\limits_{K} & \omega(K_{tr}) \\ \text{subject to} & \operatorname{trace}(K) = c, \\ & K \succeq 0, \\ & K = \sum\limits_{i=1}^{m} \mu_i K_i, \end{array} \tag{30}$$

which can be written in the standard form of a semidefinite program, in a manner analogous to (29):

$$\begin{aligned} \min_{\mu_i, t, \lambda, \boldsymbol{\nu}} \quad & t \\ \text{subject to} \quad & \operatorname{trace}\left(\sum_{i=1}^{m} \mu_i K_i\right) = c, \\ & \sum_{i=1}^{m} \mu_i K_i \succeq 0, \\ & \begin{pmatrix} G(\sum_{i=1}^{m} \mu_i K_{i,tr}) & \mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y})^T & t \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0. \end{aligned} \tag{31}$$

Notice that the SDP approach is consistent with the bound in (19). The margin is optimized over the labeled data (via the use of $K_{i,tr}$), while the positive semidefiniteness and the trace constraint are imposed for the entire kernel matrix K (via the use of K_i). This leads to a general method for learning the kernel matrix with semidefinite programming, when using a margin criterion for hard margin SVMs. Applying the complexity results mentioned in Section 3.3 leads to a worst-case complexity $O\left((m + n_{tr})^2 n^{2.5}\right)$ when using general-purpose interior-point methods to solve this particular SDP.

Furthermore, this gives a new transduction method for hard margin SVMs. Whereas Vapnik's original method for transduction scales exponentially in the number of test samples, the new SDP method has polynomial time complexity.

Remark. For the specific case in which the K_i are rank-one matrices $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal (e.g., the normalized eigenvectors of an initial kernel matrix K_0), the semidefinite program reduces to a QCQP:

$$\begin{aligned} \max_{\boldsymbol{\alpha}, t} \quad & 2\boldsymbol{\alpha}^T \mathbf{e} - ct \\ \text{subject to} \quad & t \geq (\breve{\mathbf{v}}_i^T \boldsymbol{\alpha})^2, i = 1, \ldots, m \\ & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{32}$$

with $\breve{\mathbf{v}}_i = \operatorname{diag}(\mathbf{y})\ \mathbf{v}_i(1 : n_{tr})$.

This can be seen by observing that, for $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$, we have that $\sum_{i=1}^{m} \mu_i K_i \succeq 0$ is equivalent to $\boldsymbol{\mu} \geq 0$. So, we can apply Theorem 17, with $\tau = 0$ and $C = \infty$, where $\frac{1}{r_i}\ \boldsymbol{\alpha}^T\ G(K_{i,tr})\ \boldsymbol{\alpha} = \boldsymbol{\alpha}^T \operatorname{diag}(\mathbf{y})\ \mathbf{v}_i(1 : n_{tr})\ \mathbf{v}_i(1 : n_{tr})^T\ \operatorname{diag}(\mathbf{y})\ \boldsymbol{\alpha} = (\breve{\mathbf{v}}_i^T \boldsymbol{\alpha})^2$.

4.3 Hard Margin with Kernel Matrices that are Positive Linear Combinations

To learn a kernel matrix from this linear class $\mathcal{K}$, one has to solve a semidefinite programming problem: interior-point methods (Nesterov and Nemirovsky, 1994) are polynomial time, but have a worst-case complexity $O\left((m + n_{tr})^2 n^{2.5}\right)$ in this particular case.

We now restrict $\mathcal{K}$ to the positive linear combinations of kernel matrices:

$$K = \sum_{i=1}^{m} \mu_i K_i, \qquad \boldsymbol{\mu} \geq 0.$$

Assuming positive weights yields a smaller set of kernel matrices, because the weights need not be positive for K to be positive semidefinite, even if the components K_i are positive semidefinite. Moreover, the restriction has beneficial computational effects: (1) the general SDP reduces to a QCQP, which can be solved with significantly lower complexity $O(mn_{tr}^3)$; (2) the constraint can result in improved numerical stability—it prevents the algorithm from using large weights with opposite sign that cancel. Finally, we shall see in Section 5 that the constraint also yields better estimates of the generalization performance of these algorithms.

Theorem 20 *Given a labeled sample $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted $\mathbf{y} \in \mathbb{R}^{n_{tr}}$, the kernel matrix $K = \sum_{i=1}^m \mu_i K_i$ that optimizes (21), with $\tau \geq 0$, under the additional constraint $\boldsymbol{\mu} \geq 0$ can be found by solving the following convex optimization problem, and considering its dual solution:*

$$\begin{aligned} \max_{\boldsymbol{\alpha}, t} \quad & 2\boldsymbol{\alpha}^T \mathbf{e} - ct \\ \textit{subject to} \quad & t \geq \frac{1}{r_i} \boldsymbol{\alpha}^T G(K_{i,tr}) \boldsymbol{\alpha}, \quad i = 1, \ldots, m \\ & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0. \end{aligned} \tag{33}$$

where $\mathbf{r} \in \mathbb{R}^m$ with trace$(K_i) = r_i$.

Proof Apply Theorem 17 for $C = \infty$ and $\tau = 0$. ■

Note once again that the optimal weights $\mu_i, \quad i = 1, \ldots, m$, can be recovered from the primal-dual solution found by standard software such as SeDuMi (Sturm, 1999) or Mosek (Andersen and Andersen, 2000).

4.4 1-Norm Soft Margin

For the case of non-linearly separable data, we can consider the 1-norm soft margin cost function in (3). Training the SVM for a given kernel involves minimizing this quantity with respect to $\mathbf{w}, b$, and $\boldsymbol{\xi}$, which yields the optimal value (4): obviously this minimum is a function of the particular choice of K, which is expressed explicitly in (4) as a dual problem. Let us now optimize this quantity with respect to the kernel matrix K, i.e., let us try to find the kernel matrix $K \in \mathcal{K}$ for which the corresponding embedding yields minimal $\omega_{S1}(K_{tr})$, keeping the trace of K constant:

$$\min_{K \in \mathcal{K}} \omega_{S1}(K_{tr}) \qquad \text{s.t. trace}(K) = c. \tag{34}$$

This is again a convex optimization problem.

Theorem 21 *Given a labeled sample $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted $\mathbf{y} \in \mathbb{R}^{n_{tr}}$, the kernel matrix $K \in \mathcal{K}$ that optimizes (34), can be found by solving the following convex*

optimization problem:

$$\begin{array}{ll} \min\limits_{K,t,\lambda,\boldsymbol{\nu},\boldsymbol{\delta}} & t \\ \textit{subject to} & \textit{trace}(K) = c, \\ & K \in \mathcal{K}, \\ & \begin{pmatrix} G(K_{tr}) & \mathbf{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\mathbf{y} \\ (\mathbf{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\mathbf{y})^T & t-2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0, \\ & \boldsymbol{\delta} \geq 0. \end{array} \tag{35}$$

Proof Observe $\omega_{S1}(K_{tr}) = \omega_{C,0}(K_{tr})$. Apply Theorem 16 for $\tau = 0$. ■

Again, if $\mathcal{K} = \{K \succeq 0\}$, this is an SDP. Adding the additional constraint (18) that K is a linear combination of fixed kernel matrices leads to the following SDP:

$$\begin{array}{ll} \min\limits_{\mu_i,t,\lambda,\boldsymbol{\nu},\boldsymbol{\delta}} & t \\ \text{subject to} & \text{trace}\left(\sum_{i=1}^m \mu_i K_i\right) = c, \\ & \sum_{i=1}^m \mu_i K_i \succeq 0, \\ & \begin{pmatrix} G(\sum_{i=1}^m \mu_i K_{i,tr}) & \mathbf{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\mathbf{y} \\ (\mathbf{e}+\boldsymbol{\nu}-\boldsymbol{\delta}+\lambda\mathbf{y})^T & t-2C\boldsymbol{\delta}^T\mathbf{e} \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu}, \boldsymbol{\delta} \geq 0. \end{array} \tag{36}$$

Remark. For the specific case in which the K_i are rank-one matrices $K_i = \mathbf{v}_i\mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal (e.g., the normalized eigenvectors of an initial kernel matrix K_0), the SDP reduces to a QCQP using Theorem 17, with $\tau = 0$, in a manner analogous to the hard margin case:

$$\begin{array}{ll} \max\limits_{\boldsymbol{\alpha},t} & 2\boldsymbol{\alpha}^T\mathbf{e} - ct \\ \text{subject to} & t \geq (\check{\mathbf{v}}_i^T\boldsymbol{\alpha})^2, i = 1,\ldots,m \\ & \boldsymbol{\alpha}^T\mathbf{y} = 0, \\ & C \geq \boldsymbol{\alpha} \geq 0, \end{array} \tag{37}$$

with $\check{\mathbf{v}}_i = \text{diag}(\mathbf{y})\ \mathbf{v}_i(1:n_{tr})$.

Solving the original learning problem subject to the extra constraint $\boldsymbol{\mu} \geq 0$ yields, after applying Theorem 17, with $\tau = 0$:

$$\begin{array}{ll} \max\limits_{\boldsymbol{\alpha},t} & 2\boldsymbol{\alpha}^T\mathbf{e} - ct \\ \text{subject to} & t \geq \dfrac{1}{r_i}\boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha}, \quad i = 1,\ldots,m \\ & \boldsymbol{\alpha}^T\mathbf{y} = 0, \\ & C \geq \boldsymbol{\alpha} \geq 0. \end{array} \tag{38}$$

4.5 2-Norm Soft Margin

For the case of non-linearly separable data, we can also consider the 2-norm soft margin cost function (5). Again, training for a given kernel will minimize this quantity with respect to $\mathbf{w}, b$, and $\boldsymbol{\xi}$ and the minimum is a function of the particular choice of K, as expressed in (6) in dual form. Let us now optimize this quantity with respect to the kernel matrix K:

$$\min_{K \in \mathcal{K}} \omega_{S2}(K_{tr}) \qquad \text{s.t. } \operatorname{trace}(K) = c. \tag{39}$$

This is again a convex optimization problem, and can be restated as follows.

Theorem 22 *Given a labeled sample $S_{n_{tr}} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n_{tr}}, y_{n_{tr}})\}$ with the set of labels denoted $\mathbf{y} \in \mathbb{R}^{n_{tr}}$, the kernel matrix $K \in \mathcal{K}$ that optimizes (39) can be found by solving the following optimization problem:*

$$\begin{aligned} \min_{K,t,\lambda,\boldsymbol{\nu}} \quad & t \\ \textit{subject to} \quad & \textit{trace}(K) = c, \\ & K \in \mathcal{K}, \\ & \begin{pmatrix} G(K_{tr}) + \frac{1}{C} I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y})^T & t \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0. \end{aligned} \tag{40}$$

Proof Observe $\omega_{S2}(K_{tr}) = \omega_{\infty,\tau}(K_{tr})$. Apply Theorem 16 for $C = \infty$. ■

Again, if $\mathcal{K} = \{K \succeq 0\}$, this is an SDP. Moreover, constraining K to be a linear combination of fixed kernel matrices, we obtain

$$\begin{aligned} \min_{\mu_i,t,\lambda,\boldsymbol{\nu}} \quad & t \\ \text{subject to} \quad & \operatorname{trace}\left(\sum_{i=1}^{m} \mu_i K_i\right) = c, \\ & \sum_{i=1}^{m} \mu_i K_i \succeq 0, \\ & \begin{pmatrix} G(\sum_{i=1}^{m} \mu_i K_{i,tr}) + \frac{1}{C} I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y})^T & t \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu} \geq 0. \end{aligned} \tag{41}$$

Also, when the K_i are rank-one matrices, $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal, we obtain a QCQP:

$$\begin{aligned} \max_{\boldsymbol{\alpha},t} \quad & 2\boldsymbol{\alpha}^T \mathbf{e} - \frac{1}{C} \boldsymbol{\alpha}^T \boldsymbol{\alpha} - ct \\ \text{subject to} \quad & t \geq (\check{\mathbf{v}}_i^T \boldsymbol{\alpha})^2, i = 1, \ldots, m \\ & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{42}$$

and, finally, imposing the constraint $\boldsymbol{\mu} \geq 0$ yields

$$\begin{aligned} \max_{\boldsymbol{\alpha},t} \quad & 2\boldsymbol{\alpha}^T\mathbf{e} - \frac{1}{C}\boldsymbol{\alpha}^T\boldsymbol{\alpha} - ct \\ \text{subject to} \quad & t \geq \frac{1}{r_i}\boldsymbol{\alpha}^T G(K_{i,tr})\boldsymbol{\alpha}, \quad i = 1, \ldots, m \\ & \boldsymbol{\alpha}^T\mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{43}$$

following a similar derivation as before: apply Theorem 17 with $C = \infty$, and, for (42), observe that $\boldsymbol{\mu} \geq 0$ is equivalent to $\sum_{i=1}^m \mu_i K_i \succeq 0$ if $K_i = \mathbf{v}_i\mathbf{v}_i^T$ and $\mathbf{v}_i^T\mathbf{v}_j = \delta_{ij}$.

4.6 Learning the 2-Norm Soft Margin Parameter $\tau = 1/C$

This section shows how the 2-norm soft margin parameter of SVMs can be learned using SDP or QCQP. More details can be found in De Bie et al. (2003).

In the previous section, we tried to find the kernel matrix $K \in \mathcal{K}$ for which the corresponding embedding yields minimal $\omega_{S2}(K_{tr})$, keeping the trace of K constant. Since in the dual formulation (6) the identity matrix induced by the 2-norm formulation appears in exactly the same way as the other matrices K_i, we can treat it on the same basis and optimize its weight to obtain the optimal dual formulation, i.e., to minimize $\omega_{S2}(K_{tr})$. Since this weight now happens to correspond to the parameter $\tau = 1/C$, optimizing it corresponds to learning the 2-norm soft margin parameter and thus has a significant meaning.

Since the parameter $\tau = 1/C$ can be treated in the same way as the weights μ_i, tuning it such that the quantity $\omega_{S2}(K_{tr}, \tau)$ is minimized can be viewed as a method for choosing τ. First of all, consider the dual formulation (6) and notice that $\omega_{S2}(K_{tr}, \tau)$ is convex in $\tau = 1/C$ (being the pointwise maximum of affine and thus convex functions in τ). Secondly, since $\tau \to \infty$ leads to $\omega_{S2}(K_{tr}, \tau) \to 0$, we impose the constraint $\text{trace}\,(K + \tau I_n) = c$. This results in the following convex optimization problem:

$$\min_{K \in \mathcal{K}, \tau \geq 0} \omega_{S2}(K_{tr}, \tau) \qquad \text{s.t. } \text{trace}\,(K + \tau I_n) = c.$$

According to Theorem 22, this can be restated as follows:

$$\begin{aligned} \min_{K,t,\lambda,\boldsymbol{\nu},\tau} \quad & t \\ \text{subject to} \quad & \text{trace}\,(K + \tau I_n) = c, \\ & K \in \mathcal{K}, \\ & \begin{pmatrix} G(K_{tr}) + \tau I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} + \lambda\mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} + \lambda\mathbf{y})^T & t \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu}, \tau \geq 0. \end{aligned} \tag{44}$$

Again, if $\mathcal{K} = \{K \succeq 0\}$, this is an SDP. Imposing the additional constraint that K is a linear function of fixed kernel matrices, we obtain the SDP

$$\begin{aligned} \min_{\mu_i, t, \lambda, \boldsymbol{\nu}, \tau} \quad & t \\ \text{subject to} \quad & \operatorname{trace}\left(\sum_{i=1}^{m} \mu_i K_i + \tau I_n\right) = c, \\ & \sum_{i=1}^{m} \mu_i K_i \succeq 0, \\ & \begin{pmatrix} G(\sum_{i=1}^{m} \mu_i K_{i,tr}) + \tau I_{n_{tr}} & \mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y} \\ (\mathbf{e} + \boldsymbol{\nu} + \lambda \mathbf{y})^T & t \end{pmatrix} \succeq 0, \\ & \boldsymbol{\nu}, \tau \geq 0, \end{aligned} \tag{45}$$

and imposing the additional constraint that the K_i are rank-one matrices, we obtain a QCQP:

$$\begin{aligned} \max_{\boldsymbol{\alpha}, t} \quad & 2\boldsymbol{\alpha}^T \mathbf{e} - ct \\ \text{subject to} \quad & t \geq (\check{\mathbf{v}}_i^T \boldsymbol{\alpha})^2, i = 1, \ldots, m \\ & t \geq \frac{1}{n} \boldsymbol{\alpha}^T \boldsymbol{\alpha} \\ & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{46}$$

with $\check{\mathbf{v}}_i = \operatorname{diag}(\mathbf{y})\ \bar{\mathbf{v}}_i = \operatorname{diag}(\mathbf{y})\ \mathbf{v}_i(1 : n_{tr})$. Finally, imposing the constraint that $\boldsymbol{\mu} \geq 0$ yields the following:

$$\begin{aligned} \max_{\boldsymbol{\alpha}, t} \quad & 2\boldsymbol{\alpha}^T \mathbf{e} - ct \\ \text{subject to} \quad & t \geq \frac{1}{r_i} \boldsymbol{\alpha}^T G(K_{i,tr}) \boldsymbol{\alpha}, \quad i = 1, \ldots, m \\ & t \geq \frac{1}{n} \boldsymbol{\alpha}^T \boldsymbol{\alpha} \\ & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0, \end{aligned} \tag{47}$$

(48)

which, as before, is a QCQP.

Solving (47) corresponds to learning the kernel matrix as a positive linear combination of kernel matrices according to a 2-norm soft margin criterion and simultaneously learning the 2-norm soft margin parameter $\tau = 1/C$. Comparing (47) with (33), we can see that this reduces to learning an augmented kernel matrix K' as a positive linear combination of kernel matrices and the identity matrix, $K' = K + \tau I_n = \sum_{i=1}^{m} \mu_i K_i + \tau I_n$, using a hard margin criterion. However, there is an important difference: when evaluating the resulting classifier, the actual kernel matrix K is used, instead of the augmented K' (see, for example, Shawe-Taylor and Cristianini, 1999).

For $m = 1$, we notice that (45) directly reduces to (47) if $K_1 \succeq 0$. This corresponds to automatically tuning the parameter $\tau = 1/C$ for a 2-norm soft margin SVM with kernel matrix K_1. So, even when not learning the kernel matrix, this approach can be used to tune the 2-norm soft margin parameter $\tau = 1/C$ automatically.

4.7 Alignment

In this section, we consider the problem of optimizing the alignment between a set of labels and a kernel matrix from some class $\mathcal{K}$ of positive semidefinite kernel matrices. We show that, if $\mathcal{K}$ is a class of linear combinations of fixed kernel matrices, this problem can be cast as an SDP. This result generalizes the approach presented in Cristianini et al. (2001, 2002).

Theorem 23 *The kernel matrix $K \in \mathcal{K}$ which is maximally aligned with the set of labels $\mathbf{y} \in \mathbb{R}^{n_{tr}}$ can be found by solving the following optimization problem:*

$$\begin{aligned} \max_{A,K} \quad & \langle K_{tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \textit{subject to} \quad & \textit{trace}(A) \leq 1 \\ & \begin{pmatrix} A & K^T \\ K & I_n \end{pmatrix} \succeq 0 \\ & K \in \mathcal{K}, \end{aligned} \tag{49}$$

where I_n is the identity matrix of dimension n.

Proof We want to find the kernel matrix K which is maximally aligned with the set of labels $\mathbf{y}$:

$$\begin{aligned} \max_{K} \quad & \hat{A}(S, K_{tr}, \mathbf{y}\mathbf{y}^T) \\ \text{subject to} \quad & K \in \mathcal{K},\ \text{trace}(K) = 1. \end{aligned}$$

This is equivalent to the following optimization problem:

$$\begin{aligned} \max_{K} \quad & \langle K_{tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \text{subject to} \quad & \langle K, K \rangle_F = 1 \\ & K \in \mathcal{K},\ \text{trace}(K) = 1. \end{aligned} \tag{50}$$

To express this in the standard form (9) of a semidefinite program, we need to express the quadratic equality constraint $\langle K, K \rangle_F = 1$ as an LMI. First, notice that (50) is equivalent to

$$\begin{aligned} \max_{K} \quad & \langle K_{tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \text{subject to} \quad & \langle K, K \rangle_F \leq 1 \\ & K \in \mathcal{K}. \end{aligned} \tag{51}$$

Indeed, we are maximizing an objective which is linear in the entries of K, so at the optimum $K = K^*$, the constraint $\langle K, K \rangle_F = \text{trace}(K^T K) \leq 1$ is achieved: $\langle K^*, K^* \rangle_F = 1$. The quadratic inequality constraint in (51) is now equivalent to

$$\exists A \ : \ K^T K \preceq A \quad \text{and} \quad \text{trace}(A) \leq 1.$$

Indeed, $A - K^T K \succeq 0$ implies $\text{trace}(A - K^T K) = \text{trace}(A) - \text{trace}(K^T K) \geq 0$ because of linearity of the trace. Using the Schur complement lemma, we can express $A - K^T K \succeq 0$ as an LMI:

$$A - K^T K \succeq 0 \Leftrightarrow \begin{pmatrix} A & K^T \\ K & I_n \end{pmatrix} \succeq 0.$$

We can thus rewrite the optimization problem (50) as

$$\begin{aligned} \max_{A,K} \quad & \langle K_{tr}, \mathbf{y}\mathbf{y}^T \rangle_F \\ \text{subject to} \quad & \text{trace}(A) \leq 1 \\ & \begin{pmatrix} A & K^T \\ K & I_n \end{pmatrix} \succeq 0 \\ & K \in \mathcal{K}, \end{aligned}$$

which corresponds to (49). ■

Notice that, when $\mathcal{K}$ is the set of all positive semidefinite matrices, this is an SDP (an inequality constraint corresponds to a one-dimensional LMI; consider the entries of the matrices A and K as the unknowns, corresponding to the u_i in (9)). In that case, one solution of (49) is found by simply selecting $K_{tr} = \frac{c}{n}\mathbf{y}\mathbf{y}^T$, for which the alignment (7) is equal to one and thus maximized.

Adding the additional constraint (18) that K is a linear combination of fixed kernel matrices leads to

$$\begin{aligned} \max_{K} \quad & \left\langle K_{tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ \text{subject to} \quad & \langle K, K \rangle_F \leq 1, \\ & K \succeq 0, \\ & K = \sum_{i=1}^{m} \mu_i K_i, \end{aligned} \tag{52}$$

which can be written in the standard form of a semidefinite program, in a similar way as for (49):

$$\begin{aligned} \max_{A,\mu_i} \quad & \left\langle \sum_{i=1}^{m} \mu_i K_{i,tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ \text{subject to} \quad & \text{trace}(A) \leq 1, \\ & \begin{pmatrix} A & \sum_{i=1}^{m} \mu_i K_i^T \\ \sum_{i=1}^{m} \mu_i K_i & I_n \end{pmatrix} \succeq 0, \\ & \sum_{i=1}^{m} \mu_i K_i \succeq 0. \end{aligned} \tag{53}$$

Remark. For the specific case where the K_i are rank-one matrices $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal (e.g., the normalized eigenvectors of an initial kernel matrix K_0), the semidefinite program reduces to a QCQP (see Appendix A):

$$\begin{aligned} \max_{\mu_i} \quad & \sum_{i=1}^{m} \mu_i (\bar{\mathbf{v}}_i^T \mathbf{y})^2 \\ \text{subject to} \quad & \sum_{i=1}^{m} \mu_i^2 \leq 1 \\ & \mu_i \geq 0, i = 1, \ldots, m \end{aligned} \tag{54}$$

with $\bar{\mathbf{v}}_i = \mathbf{v}_i(1 : n_{tr})$. This corresponds exactly to the QCQP obtained as an illustration in Cristianini et al. (2002), which is thus entirely captured by the general SDP result obtained in this section.

Solving the original learning problem (52) subject to the extra constraint $\boldsymbol{\mu} \geq 0$ yields

$$\begin{aligned} \max_{K} \quad & \left\langle K_{tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ \text{subject to} \quad & \langle K, K \rangle_F \leq 1, \\ & K \succeq 0, \\ & K = \sum_{i=1}^{m} \mu_i K_i, \\ & \boldsymbol{\mu} \geq 0. \end{aligned}$$

We can omit the second constraint, because this is implied by the last two constraints, if $K_i \succeq 0$. This reduces to

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & \left\langle \sum_{i=1}^{m} \mu_i K_{i,tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ \text{subject to} \quad & \left\langle \sum_{i=1}^{m} \mu_i K_i, \sum_{j=1}^{m} \mu_j K_j \right\rangle_F \leq 1, \\ & \boldsymbol{\mu} \geq 0, \end{aligned}$$

where $K_{i,tr} = K_i(1 : n_{tr}, 1 : n_{tr})$. Expanding this further yields

$$\begin{aligned} \left\langle \sum_{i=1}^{m} \mu_i K_{i,tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F &= \sum_{i=1}^{m} \mu_i \left\langle K_{i,tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ &= \boldsymbol{\mu}^T \mathbf{q}, \qquad (55) \\ \left\langle \sum_{i=1}^{m} \mu_i K_i, \sum_{j=1}^{m} \mu_j K_j \right\rangle_F &= \sum_{i,j=1}^{m} \mu_i \mu_j \langle K_i, K_j \rangle_F \\ &= \boldsymbol{\mu}^T S \boldsymbol{\mu} \qquad (56) \end{aligned}$$

with $q_i = \left\langle K_{i,tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F = \text{trace}(K_{i,tr}\mathbf{y}\mathbf{y}^T) = \text{trace}(\mathbf{y}^T K_{i,tr}\mathbf{y}) = \mathbf{y}^T K_{i,tr}\mathbf{y}$ and $S_{ij} = \langle K_i, K_j \rangle_F$, where $\mathbf{q} \in \mathbb{R}^m, S \in \mathbb{R}^{m \times m}$. We used the fact that $\text{trace}(ABC) = \text{trace}(BCA)$ (if the products are well-defined). We obtain the following learning problem:

$$\begin{aligned} \max_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T \mathbf{q} \\ \text{subject to} \quad & \boldsymbol{\mu}^T S \boldsymbol{\mu} \leq 1, \\ & \boldsymbol{\mu} \geq 0, \end{aligned}$$

which is a QCQP.

4.8 Induction

In previous sections we have considered the transduction setting, where it is assumed that the covariate vectors for both training (labeled) and test (unlabeled) data are known beforehand. While

this setting captures many realistic learning problems, it is also of interest to consider possible extensions of our approach to the more general setting of induction, in which the covariates are known beforehand only for the training data.

Consider the following situation. We learn the kernel matrix as a positive linear combination of normalized kernel matrices K_i. Those K_i are obtained through the evaluation of a kernel function or through a known procedure (e.g., a string-matching kernel), yielding $K_i \succeq 0$. So, $K = \sum_{i=1}^{m} \mu_i K_i \succeq 0$. Normalization is done by replacing $K_i(k,l)$ by $K_i(k,l)/\sqrt{K_i(k,k) \cdot K_i(l,l)}$. In this case, the extension to an induction setting is elegant and simple.

Let n_{tr} be the number of training data points (all labeled). Consider the transduction problem for those n_{tr} data points and one unknown test point, e.g., for a hard margin SVM. The optimal weights $\mu_i^*, \quad i = 1, \ldots, m$ are learned by solving (33):

$$\begin{aligned} \max_{\boldsymbol{\alpha}, t} \quad & 2\boldsymbol{\alpha}^T \mathbf{e} - ct \\ \text{subject to} \quad & t \geq \frac{1}{n_{tr}+1} \boldsymbol{\alpha}^T G(K_{i,tr}) \boldsymbol{\alpha}, \quad i = 1, \ldots, m \\ & \boldsymbol{\alpha}^T \mathbf{y} = 0, \\ & \boldsymbol{\alpha} \geq 0. \end{aligned} \tag{57}$$

Even without knowing the test point and the entries of the K_i's related to it (column and row $n_{tr}+1$), we know that $K(n_{tr}+1, n_{tr}+1) = 1$ because of the normalization. So, $\text{trace}(K_i) = n_{tr}+1$. This allows solving for the optimal weights $\mu_i^*, \quad i = 1, \ldots, m$ and the optimal SVM parameters $\alpha_j^*, \quad j = 1, \ldots, n_{tr}$ and b^*, without knowing the test point. When a test point becomes available, we complete the K_i's by computing their $(n_{tr}+1)$-th column and row (evaluate the kernel function or follow the procedure and normalize). Combining those K_i with weights μ_i^* yields the final kernel matrix K, which can then be used to label the test point:

$$y = \text{sign}\left(\sum_{i=1}^{m} \sum_{j=1}^{n_{tr}} \mu_i^* \alpha_j K_i(x_j, x)\right).$$

Remark: The optimal weights are independent of the number of unknown test points that are considered in this setting. Consider the transduction problem (57) for l unknown test points instead of one unknown test point:

$$\begin{aligned} \max_{\tilde{\boldsymbol{\alpha}}, \tilde{t}} \quad & 2\tilde{\boldsymbol{\alpha}}^T \mathbf{e} - c\tilde{t} \\ \text{subject to} \quad & \tilde{t} \geq \frac{1}{n_{tr}+l} \tilde{\boldsymbol{\alpha}}^T G(K_{i,tr}) \tilde{\boldsymbol{\alpha}}, \quad i = 1, \ldots, m \\ & \tilde{\boldsymbol{\alpha}}^T \mathbf{y} = 0, \\ & \tilde{\boldsymbol{\alpha}} \geq 0. \end{aligned} \tag{58}$$

One can see that solving (58) is equivalent to solving (57) where the optimal values relate as $\tilde{\boldsymbol{\alpha}}^* = \frac{n_{tr}+l}{n_{tr}+1} \boldsymbol{\alpha}^*$ and $\tilde{t}^* = \frac{n_{tr}+l}{n_{tr}+1} t^*$ and where the optimal weights $\mu_i^*, \quad i = 1, \ldots, m$ are the same.

Tackling the induction problem in full generality remains a challenge for future work. Obviously, one could consider the transduction case with zero test points, yielding the induction case. If the weights μ_i are constrained to be nonnegative and furthermore the matrices K_i are guaranteed to be positive semidefinite, the weights can be reused at new test points. To deal with induction in a general SDP setting, one could solve a transduction problem for each new test point. For every

test point, this leads to solving an SDP of dimension $n_{tr}+1$, which is computationally expensive. Clearly there is a need to explore recursive solutions to the SDP problem that allow the solution of the SDP of dimension n_{tr} to be used in the solution of an SDP of dimension $n_{tr}+1$. Such solutions would of course also have immediate applications to on-line learning problems.

5. Error Bounds for Transduction

In the problem of transduction, we have access to the unlabeled test data, as well as the labeled training data, and the aim is to optimize accuracy in predicting the test data. We assume that the data are fixed, and that the order is chosen randomly, yielding a random partition into a labeled training set and an unlabeled test set. For convenience, we suppose here that the training and test sets have the same size. Of course, if we can show a performance guarantee that holds with high probability over uniformly chosen training/test partitions of this kind, it also holds with high probability over an i.i.d. choice of the training and test data, since permuting an i.i.d. sample leaves the distribution unchanged.

The following theorem gives an upper bound on the error of a kernel classifier on the test data in terms of the average over the training data of a certain margin cost function, together with properties of the kernel matrix. We focus on the 1-norm soft margin classifier, although our results extend in a straightforward way to other cases, including the 2-norm soft margin classifier. The 1-norm soft margin classifier chooses a kernel classifier f to minimize a weighted combination of a regularization term, $\|\mathbf{w}\|^2$, and the average over the training sample of the slack variables,

$$\xi_i = \max\left(1 - y_i f(x_i), 0\right).$$

We can view this regularized empirical criterion as the Lagrangian for the constrained minimization of

$$\frac{1}{n}\sum_{i=1}^{n}\xi_i = \frac{1}{n}\sum_{i=1}^{n}\max(1 - y_i f(x_i), 0)$$

subject to the upper bound $\|\mathbf{w}\|^2 \leq 1/\gamma^2$.

Fix a sequence of $2n$ pairs $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{2n}, Y_{2n})$ from $\mathcal{X}\times\mathcal{Y}$. Let $\pi : \{1, \ldots, 2n\} \to \{1, \ldots, 2n\}$ be a random permutation, chosen uniformly, and let $(\mathbf{x}_i, y_i) = (\mathbf{X}_{\pi(i)}, Y_{\pi(i)})$. The first half of this randomly ordered sequence is the training data $T_n = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$, and the second half is the test data $S_n = ((\mathbf{x}_{n+1}, y_{n+1}), \ldots, (\mathbf{x}_{2n}, y_{2n}))$. For a function $f : \mathcal{X} \to \mathbb{R}$, the proportion of errors on the test data of a thresholded version of f can be written as

$$\mathrm{er}(f) = \frac{1}{n}|\{n+1 \leq i \leq 2n : y_i f(\mathbf{x}_i) \leq 0\}|.$$

We consider kernel classifiers obtained by thresholding kernel expansions of the form

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x})\rangle = \sum_{i=1}^{2n}\alpha_i k(\mathbf{x}_i, \mathbf{x}), \tag{59}$$

where $\mathbf{w} = \sum_{i=1}^{2n}\alpha_i\Phi(\mathbf{x}_i)$ is chosen with bounded norm,

$$\|\mathbf{w}\|^2 = \sum_{i,j=1}^{2n}\alpha_i\alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\alpha}^T K\boldsymbol{\alpha} \leq \frac{1}{\gamma^2}, \tag{60}$$

and K is the $2n \times 2n$ kernel matrix with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Let $F_{\mathcal{K}}$ denote the class of functions on S of the form (59) satisfying (60), for some $K \in \mathcal{K}$,

$$F_{\mathcal{K}} = \left\{ \mathbf{x}_j \mapsto \sum_{i=1}^{2n} \alpha_i K_{ij} : K \in \mathcal{K},\, \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \leq \frac{1}{\gamma^2} \right\},$$

where $\mathcal{K}$ is a set of positive semidefinite $2n \times 2n$ matrices. We also consider the class of kernel expansions obtained from certain linear combinations of a fixed set $\{K_1, \ldots, K_m\}$ of kernel matrices: Define the class $F_{\mathcal{K}_c}$ as

$$\mathcal{K}_c = \left\{ K = \sum_{j=1}^{m} \mu_j K_j : K \succeq 0,\, \mu_j \in \mathbb{R},\, \mathrm{trace}(K) \leq c \right\},$$

and the class $F_{\mathcal{K}_c^+}$ as

$$\mathcal{K}_c^+ = \left\{ \sum_{j=1}^{m} \mu_j K_j : K \succeq 0,\, \mu_j \geq 0,\, \mathrm{trace}(K) \leq c \right\}.$$

Theorem 24 *For every $\gamma > 0$, with probability at least $1-\delta$ over the data $(\mathbf{x}_i, y_i)$ chosen as above, every function $f \in F_{\mathcal{K}}$ has $\mathrm{er}(f)$ no more than*

$$\frac{1}{n} \sum_{i=1}^{n} \max\{1 - y_i f(\mathbf{x}_i), 0\} + \frac{1}{\sqrt{n}} \left(4 + \sqrt{2 \log(1/\delta)} + \sqrt{\frac{\mathcal{C}(\mathcal{K})}{n\gamma^2}} \right),$$

where

$$\mathcal{C}(\mathcal{K}) = \mathbf{E} \max_{K \in \mathcal{K}} \boldsymbol{\sigma}^T K \boldsymbol{\sigma},$$

with the expectation over $\boldsymbol{\sigma}$ chosen uniformly from $\{\pm 1\}^{2n}$.

Furthermore,

$$\mathcal{C}(\mathcal{K}_c) = c\mathbf{E} \max_{K \in \mathcal{K}} \boldsymbol{\sigma}^T \frac{K}{trace(K)} \boldsymbol{\sigma},$$

and this is always no more than cn, and

$$\mathcal{C}(\mathcal{K}_c^+) \leq c \min\left(m, n \max_j \frac{\lambda_j}{trace(K_j)} \right),$$

where λ_j is the largest eigenvalue of K_j.

Notice that the test error is bounded by a sum of the average over the training data of a margin cost function plus a complexity penalty term that depends on the ratio between the trace of the kernel matrix and the squared margin parameter, γ^2. The kernel matrix here is the full matrix, combining both test and training data.

The proof of the theorem is in Appendix B. The proof technique for the first part of the theorem was introduced by Koltchinskii and Panchenko (2002), who used it to give error bounds for boosting algorithms.

Although the theorem requires the margin parameter γ to be specified in advance, it is straightforward to extend the result to give an error bound that holds with high probability over all values of γ. In this case, the $\log(1/\delta)$ in the bound would be replaced by $\log(1/\delta) + |\log(1/\gamma)|$ and the

constants would increase slightly. See, for example, Proposition 8 and its applications in the work of Bartlett (1998).

The result is presented for the 1-norm soft margin classifier, but the proof uses only two properties of the cost function $a \mapsto \max\{1-a, 0\}$: that it is an upper bound on the indicator function for $a \leq 0$, and that it satisfies a Lipschitz constraint on $[0, \infty)$. These conditions are also satisfied by the cost function associated with the 2-norm soft margin classifier, $a \mapsto (\max\{1-a, 0\})^2$, for example.

The bound on the complexity $\mathcal{C}(\mathcal{K}_B^+)$ of the kernel class $\mathcal{K}_B^+$ is easier to check than the bound on $\mathcal{C}(\mathcal{K}_B)$. The first term in the minimum shows that the set of positive linear combinations of a small set of kernel matrices is not very complex. The second term shows that if, for each matrix in the set, the largest eigenvalue does not dominate the sum of the eigenvalues (the trace), then the set of positive linear combinations is not too complex, even if the set is large. In either case, the upper bound is linear in c, the upper bound on the trace of the combined kernel matrix.

6. Empirical Results

We first present results on benchmark data sets, using kernels K_i that are derived from the same input vector. The goal here is to explore different possible representations of the same data source, and to choose a representation or combinations of representations that yield the best performance. We compare to the soft margin SVM with an RBF kernel, in which the hyperparameter is tuned via cross-validation. Note that in our framework there is no need for cross-validation to tune the corresponding kernel hyperparameters. Moreover, when using the 2-norm soft margin SVM, the methods are directly comparable, because the hyperparameter C is present in both cases.

In the second section we explore the use of our framework to combine kernels that are built using data from heterogeneous sources. Here our main interest is in comparing the combined classifier to the best *individual* classifier. To the extent that the heterogeneous data sources provide complementary information, we might expect that the performance of the combined classifier can dominate that of the best individual classifier.

6.1 Benchmark Data Sets

We present results for hard margin and soft margin support vector machines. We use a kernel matrix $K = \sum_{i=1}^3 \mu_i K_i$, where the K_i's are initial "guesses" of the kernel matrix. We use a polynomial kernel function $k_1(\mathbf{x}_1, \mathbf{x}_2) = (1+\mathbf{x}_1^T\mathbf{x}_2)^d$ for K_1, a Gaussian kernel function $k_2(\mathbf{x}_1, \mathbf{x}_2) = \exp(-0.5(\mathbf{x}_1-\mathbf{x}_2)^T(\mathbf{x}_1-\mathbf{x}_2)/\sigma)$ for K_2 and a linear kernel function $k_3(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^T\mathbf{x}_2$ for K_3. Afterwards, all K_i are normalized. After evaluating the initial kernel matrices $\{K_i\}_{i=1}^3$, the weights $\{\mu_i\}_{i=1}^3$ are optimized in a transduction setting according to a hard margin, a 1-norm soft margin and a 2-norm soft margin criterion, respectively; the semidefinite programs (31), (36) and (41) are solved using the general-purpose optimization software SeDuMi (Sturm, 1999), leading to optimal weights $\{\mu_i^*\}_{i=1}^3$. Next, the weights $\{\mu_i\}_{i=1}^3$ are constrained to be non-negative and optimized according to the same criteria, again in a transduction setting: the second order cone programs (33), (38) and (43) are solved using the general-purpose optimization software Mosek (Andersen and Andersen, 2000), leading to optimal weights $\{\mu_{i,+}^*\}_{i=1}^3$. For positive weights, we also report results in which the 2-norm soft margin hyperparameter C is tuned according to (47).

Empirical results on standard benchmark datasets are summarized in Tables 1, 2 and 3.[2] The Wisconsin breast cancer dataset contained 16 incomplete examples which were not used. The breast

2. It is worth noting that the first three columns of these columns are based on an inductive algorithm whereas the last two columns are based on a transductive algorithm. This may favor the kernel combinations in the last

cancer, ionosphere and sonar data were obtained from the UCI repository. The heart data were obtained from STATLOG and normalized. Data for the 2-norm problem data were generated as specified by Breiman (1998). Each dataset was randomly partitioned into 80% training and 20% test sets. The reported results are the averages over 30 random partitions. The kernel parameters for K_1 and K_2 are given in Tables 1, 2 and 3 by d and σ respectively. For each of the kernel matrices, an SVM is trained using the training block K_{tr} and tested using the mixed block $K_{tr,t}$ as defined in (17). The margin γ (for a hard margin criterion) and the optimal soft margin cost functions ω^*_{S1} and ω^*_{S2} (for soft margin criteria) are reported for the initial kernel matrices K_i, as well as for the optimal $\sum_i \mu^*_i K_i$ and $\sum_i \mu^*_{i,+} K_i$. Furthermore, the average test set accuracy (TSA), the average value for C and the average weights over the 30 partitions are listed. For comparison, the performance of the best soft margin SVM with an RBF (Gaussian) kernel is reported—the soft margin hyperparameter C and the kernel parameter σ for the Gaussian kernel were tuned using cross-validation over 30 random partitions of the training set.

Note that not every K_i gives rise to a linearly separable embedding of the training data, in which case no hard margin classifier can be found (indicated with a dash). The matrices $\sum_i \mu^*_i K_i$ and $\sum_i \mu^*_{i,+} K_i$ however, always allow the training of a hard margin SVM and its margin is indeed larger than the margin for each of the different components K_i—this is consistent with the SDP/QCQP optimization. For the soft margin criteria, the optimal value of the cost function for $\sum_i \mu^*_i K_i$ and $\sum_i \mu^*_{i,+} K_i$ is smaller than its value for the individual K_i—again consistent with the SDP/QCQP optimizations. Notice that constraining the weights μ_i to be positive results in slightly smaller margins and larger cost functions, as expected.

Furthermore, the number of test set errors for $\sum_i \mu^*_i K_i$ and $\sum_i \mu^*_{i,+} K_i$ is in general comparable in magnitude to the best value achieved among the different components K_i. Also notice that $\sum_i \mu^*_{i,+} K_i$ does often almost as well as $\sum_i \mu^*_i K_i$, and sometimes even better: we can thus achieve a substantial reduction in computational complexity without a significant loss of performance. Moreover, the performance of $\sum_i \mu^*_i K_i$ and $\sum_i \mu^*_{i,+} K_i$ is comparable with the best soft margin SVM with an RBF kernel. In making this comparison note that the RBF SVM requires tuning of the kernel parameter using cross-validation, while the kernel learning approach achieves a similar effect without cross-validation.[3] Moreover, when using the 2-norm soft margin SVM with tuned hyperparameter C, we no longer need to do cross-validation for C. This leads to a smaller value of the optimal cost function ω^*_{S2} (compared to the case SM2, with $C = 1$) and performs well on the test set, while offering the advantage of automatically adjusting C.

One might wonder why there is a difference between the SDP and the QCQP approach for the 2-norm data, since both seem to find positive weights μ_i. However, it must be recalled that

two columns and thus the results should be interpreted with caution. However, it is also worth noting that the transduction is a weak form of transduction that is based only on the norm of the test data point.

3. The experiments were run on a 2GHz Windows XP machine. We used the programs SeDuMi to solve the SDP for kernel learning and Mosek to solve multiple QP's for cross-validated SVM and the QCQP for kernel learning with positive weights. The run time for the SDP is on the order of minutes (approximately 10 minutes for 300 data points and 5 kernels), while the run time for the QP and QCQP is on the order of seconds (approximately 1 second for 300 data points and 1 kernel, and approximately 3 seconds for 300 data points and 5 kernels). Thus, we see that kernel learning with positive weights, which requires only a QCQP solution, achieves an accuracy which is comparable to the full SDP approach at a fraction of the computational cost, and our tentative recommendation is that the QCQP approach is to be preferred. It is worth noting, however, that special-purpose implementations of SDPs that take advantage of the structure of the kernel learning problem may well yield significant speed-ups, and the recommendation should be taken with caution. Finally, the QCQP approach also compares favorably in terms of run time to the multiple runs of a QP that are required for cross-validation, and should be considered a viable alternative to cross-validation, particularly given the high variance associated with cross-validation in small data sets.

		K_1	K_2	K_3	$\sum_i \mu_i^* K_i$	$\sum_i \mu_{i,+}^* K_i$	best c/v RBF
Heart		$d=2$	$\sigma=0.5$				
HM	γ	0.0369	0.1221	-	0.1531	0.1528	
	TSA	**72.9 %**	**59.5 %**	-	**84.8 %**	**84.6 %**	**77.7 %**
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**-0.09/2.68/0.41**	**0.01/2.60/0.39**	
SM1	ω_{S1}^*	58.169	33.536	74.302	21.361	21.446	
	TSA	**79.3 %**	**59.5 %**	**84.3 %**	**84.8 %**	**84.6 %**	**83.9 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**-0.09/2.68/0.41**	**0.01/2.60/0.39**	
SM2	ω_{S2}^*	32.726	25.386	45.891	15.988	16.034	
	TSA	**78.1 %**	**59.0 %**	**84.3 %**	**84.8 %**	**84.6 %**	**83.2 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**-0.08/2.54/0.54**	**0.01/2.47/0.53**	
SM2,C	ω_{S2}^*	19.643	25.153	16.004		15.985	
	TSA	**81.3 %**	**59.6 %**	**84.7 %**		**84.6 %**	**83.2 %**
	C	0.3378	1.18e+7	0.2880		0.4365	
	$\mu_1/\mu_2/\mu_3$	1.04/0/0	0/3.99/0	0/0/0.53		**0.01/0.80/0.53**	
Sonar		$d=2$	$\sigma=0.1$				
HM	γ	0.0246	0.1460	0.0021	0.1517	0.1459	
	TSA	**80.9 %**	**85.8 %**	**74.2 %**	**84.6 %**	**85.8 %**	**84.2 %**
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**-2.23/3.52/1.71**	**0/3/0**	
SM1	ω_{S1}^*	87.657	23.288	102.68	21.637	23.289	
	TSA	**78.1 %**	**85.6 %**	**73.3 %**	**84.6 %**	**85.6 %**	**84.2 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**-2.20/3.52/1.69**	**0/3/0**	
SM2	ω_{S2}^*	45.048	15.893	53.292	15.219	15.893	
	TSA	**79.1 %**	**85.2 %**	**76.7 %**	**84.5 %**	**85.2 %**	**84.2 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**-1.78/3.46/1.32**	**0/3/0**	
SM2,C	ω_{S2}^*	20.520	15.640	20.620		15.640	
	TSA	**60.9 %**	**84.6 %**	**51.0 %**		**84.6 %**	**84.2 %**
	C	0.2591	0.6087	0.2510		0.6087	
	$\mu_1/\mu_2/\mu_3$	0.14/0/0	0/2.36/0	0/0/0.02		**0/2.34/0**	

Table 1: SVMs trained and tested with the initial kernel matrices K_1, K_2, K_3 and with the optimal kernel matrices $\sum_i \mu_i^* K_i$ and $\sum_i \mu_{i,+}^* K_i$. For hard margin SVMs (HM), the resulting margin γ is given—a dash meaning that no hard margin classifier could be found; for soft margin SVMs (SM1 = 1-norm soft margin with $C = 1$, SM2 = 2-norm soft margin with $C = 1$ and SM2,C = 2-norm soft margin with auto tuning of C) the optimal value of the cost function ω_{S1}^* or ω_{S2}^* is given. Furthermore, the test-set accuracy (TSA), the average weights and the average C-values are given. For c we used $c = \sum_i \mathrm{trace}(K_i)$ for HM, SM1 and SM2. The initial kernel matrices are evaluated after being multiplied by 3. This assures we can compare the different γ for HM, ω_{S1}^* for SM1 and ω_{S2}^* for SM2, since the resulting kernel matrix has a constant trace (thus, everything is on the same scale). For SM2,C we use $c = \sum_i \mathrm{trace}(K_i) + \mathrm{trace}(I_n)$. This not only allows comparing the different ω_{S2}^* for SM2,C but also it allows comparing ω_{S2}^* between SM2 and SM2,C (since we choose $C = 1$ for SM2, we have that trace $\left(\sum_{i=1}^m \mu_i K_i + \frac{1}{C} I_n\right)$ is constant in both cases, so again, we are on the same scale). Finally, the column 'best c/v RBF' reports the performance of the best soft margin SVM with RBF kernel, tuned using cross-validation.

the values in Table 3 are averages over 30 randomizations—for some randomizations the SDP has actually found negative weights, although the averages are positive.

As a further example illustrating the flexibility of the SDP framework, consider the following setup. Let $\{K_i\}_{i=1}^5$ be Gaussian kernels with $\sigma = 0.01, 0.1, 1, 10, 100$ respectively. Combining those optimally with $\mu_i \geq 0$ for a 2-norm soft margin SVM, with tuning of C, yields the results

		K_1	K_2	K_3	$\sum_i \mu_i^* K_i$	$\sum_i \mu_{i,+}^* K_i$	best c/v RBF
Breast cancer		$d=2$	$\sigma=0.5$				
HM	γ	0.0036	0.1055	-	0.1369	0.1219	
	TSA	**92.9 %**	**89.0 %**	-	**95.5 %**	**94.4 %**	**96.1 %**
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**1.90/2.35/-1.25**	**0.65/2.35/0**	
SM1	ω_{S1}^*	77.012	44.913	170.26	26.694	33.689	
	TSA	**96.4 %**	**89.0 %**	**87.7 %**	**95.5 %**	**94.4 %**	**96.7 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**1.90/2.35/-1.25**	**0.65/2.35/0**	
SM2	ω_{S2}^*	43.138	35.245	102.51	20.696	21.811	
	TSA	**96.4 %**	**88.5 %**	**87.4 %**	**95.4 %**	**94.3 %**	**96.8 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**2.32/2.13/-1.46**	**0.89/2.11/0**	
SM2,C	ω_{S2}^*	27.682	33.685	41.023		25.267	
	TSA	**94.5 %**	**89.0 %**	**87.3 %**		**94.4 %**	**96.8 %**
	C	0.3504	1.48e+8	0.3051		6.77e+7	
	$\mu_1/\mu_2/\mu_3$	1.15/0/0	0/3.99/0	0/0/0.72		**0.87/3.13/0**	
Ionosphere		$d=2$	$\sigma=0.5$				
HM	γ	0.0613	0.1452	-	0.1623	0.1616	
	TSA	**91.2 %**	**92.0 %**	-	**94.4 %**	**94.4 %**	**93.9 %**
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**1.08/2.18/-0.26**	**0.79/2.21/0**	
SM1	ω_{S1}^*	30.786	23.233	52.312	18.117	18.303	
	TSA	**94.5 %**	**92.1 %**	**83.1 %**	**94.8 %**	**94.5 %**	**94.0 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**1.23/2.07/-0.30**	**0.90/2.10/0**	
SM2	ω_{S2}^*	18.533	17.907	31.662	13.382	13.542	
	TSA	**94.7 %**	**92.0 %**	**91.6 %**	**94.5 %**	**94.4 %**	**94.2 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**1.68/1.73/-0.41**	**1.23/1.78/0**	
SM2,C	ω_{S2}^*	14.558	17.623	18.975		13.5015	
	TSA	**93.5 %**	**92.1 %**	**90.0 %**		**94.6 %**	**94.2 %**
	C	0.4144	5.8285	0.3442		0.8839	
	$\mu_1/\mu_2/\mu_3$	1.59/0/0	0/3.83/0	0/0/1.09		**1.24/1.61/0**	

Table 2: See the caption to Table 1 for explanation.

		K_1	K_2	K_3	$\sum_i \mu_i^* K_i$	$\sum_i \mu_{i,+}^* K_i$	best c/v RBF
2-norm		$d=2$	$\sigma=0.1$				
HM	γ	0.1436	0.1072	0.0509	0.2170	0.2169	
	TSA	**94.6 %**	**55.4 %**	**94.3 %**	**96.6 %**	**96.6 %**	**96.3 %**
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**0.03/1.91/1.06**	**0.06/1.88/1.06**	
SM1	ω_{S1}^*	23.835	43.509	22.262	10.636	10.641	
	TSA	**95.0 %**	**55.4 %**	**95.7 %**	**96.6 %**	**96.6 %**	**97.5 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**0.03/1.91/1.06**	**0.06/1.88/1.06**	
SM2	ω_{S2}^*	16.134	32.631	11.991	7.9780	7.9808	
	TSA	**95.9 %**	**55.4 %**	**95.6 %**	**96.6 %**	**96.6 %**	**97.2 %**
	C	1	1	1	1	1	
	$\mu_1/\mu_2/\mu_3$	3/0/0	0/3/0	0/0/3	**0.05/1.54/1.41**	**0.08/1.51/1.41**	
SM2,C	ω_{S2}^*	16.057	32.633	7.9880		7.9808	
	TSA	**96.2 %**	**55.4 %**	**96.6 %**		**96.6 %**	**97.2 %**
	C	0.8213	0.5000	0.3869		0.8015	
	$\mu_1/\mu_2/\mu_3$	2.78/0/0	0/2/0	0/0/1.42		**0.08/1.25/1.41**	

Table 3: See the caption to Table 1 for explanation.

in Table 4—averages over 30 randomizations in 80% training and 20% test sets. The test set accuracies obtained for $\sum_i \mu_{i,+}^* K_i$ are competitive with those for the best soft margin SVM with an RBF kernel, tuned using cross-validation. The average weights show that some kernels are selected and others are not. Effectively we obtain a data-based choice of smoothing parameter without recourse to cross-validation.

	$\mu_{1,+}$	$\mu_{2,+}$	$\mu_{3,+}$	$\mu_{4,+}$	$\mu_{5,+}$	C	TSA SM2,C	TSA best c/v RBF
Breast Cancer	0	0	3.24	0.94	0.82	3.6e+08	97.1 %	96.8 %
Ionosphere	0.85	0.85	2.63	0.68	0	4.0e+06	94.5 %	94.2 %
Heart	0	3.89	0.06	1.05	0	2.5e+05	84.1 %	83.2 %
Sonar	0	3.93	1.07	0	0	3.2e+07	84.8 %	84.2 %
2-norm	0.49	0.49	0	3.51	0	2.0386	96.5 %	97.2 %

Table 4: The initial kernel matrices $\{K_i\}_{i=1}^5$ are Gaussian kernels with $\sigma = 0.01, 0.1, 1, 10, 100$ respectively. For c we used $c = \sum_i \text{trace}(K_i) + \text{trace}(I_n)$. $\{\mu_{i,+}\}_{i=1}^5$ are the average weights of the optimal kernel matrix $\sum_i \mu_{i,+}^* K_i$ for a 2-norm soft margin SVM with $\mu_i \geq 0$ and tuning of C. The average C-value is given as well. The test set accuracies (TSA) of the optimal 2-norm soft margin SVM with tuning of C (SM2,C) and the best crossvalidation soft margin SVM with RBF kernel (best c/v RBF) are reported.

In Cristianini et al. (2002) empirical results are given for optimization of the alignment using a kernel matrix $K = \sum_{i=1}^N \mu_i \mathbf{v}_i \mathbf{v}_i^T$. The results show that optimizing the alignment indeed improves the generalization power of Parzen window classifiers. As explained in Section 4.7, it turns out that in this particular case, the SDP in (53) reduces to exactly the quadratic program that is obtained in Cristianini et al. (2002) and thus those results also provide support for the general framework presented in the current paper.

6.2 Combining Heterogeneous Data

6.2.1 Reuters-21578 Data Set

To explore the value of this approach for combining data from heterogeneous sources, we run experiments on the Reuters-21578 data set, using two different kernels. The first kernel K_1 is derived as a linear kernel from the "bag-of-words" representation of the different documents, capturing information about the frequency of terms in the different documents (Salton and McGill, 1983). K_1 is centered and normalized. The second kernel K_2 is constructed by extracting 500 concepts from documents via probabilistic latent semantic analysis (Cai and Hofmann, 2003). This kernel can be viewed as arising from a document-concept-term graphical model, with the concepts as hidden nodes. After inferring the conditional probabilities of the concepts, given a document, a linear kernel is applied to the vector of these probabilistic "concept memberships," representing each document. Also K_2 is then centered and normalized. The concept-based document information contained in K_2 is likely to be partly overlapping and partly complementary to the term-frequency information in K_1. Although the "bag-of-words" and graphical model representation are clearly heterogeneous, they can both be cast into a homogeneous framework of kernel matrices, allowing the information that they convey to be combined according to $K = \mu_1 K_1 + \mu_2 K_2$.

The Reuters-21578 dataset consists of Reuters newswire stories from 1987 (`www.davidlewis.com/resources/testcollections/reuters21578/`). After a preprocessing stage that includes tokenization and stop word removal, 37926 word types remained. We used the modified Apte ("ModApte") split to split the collection into 12902 used and 8676 unused documents. The 12902 used documents consist of 9603 training documents and 3299 test documents. From the 9603 training documents, we randomly select a 1000-document subset as training set for a soft margin support vector machine with $C = 1$. We train the SVM for the binary classification tasks of

distinguishing documents about a certain topic versus those not about that topic. We restrict our attention to the topics that appear in the most documents (cf. Cai and Hofmann (2003); Huang (2003); Eyheramendy et al. (2003)); in particular, we focused on the top five Reuters-21578 topics.

After training the SVM on the randomly selected documents using either K_1 or K_2, the accuracy is tested on the 3299 test documents from the ModApte split. This is done 20 times, i.e., for 20 randomly chosen 1000-document training sets. The average accuracies and standard errors are reported in Figure 1. After evaluating the performance of K_1 and K_2, the weights μ_1 and μ_2 are constrained to be non-zero and optimized (using only the training data) according to (38). The test set performance of the optimal combination is then evaluated and the average accuracy reported in Figure 1. The optimal weights, μ_1^* and μ_2^*, do not vary greatly over the different topics, with averages of 1.37 for μ_1^* and 0.63 for μ_2^*.

We see that in four cases out of five the optimal combination of kernels performs better than either of the individual kernels. This suggests that these kernels indeed provide complementary information for the classification decision, and that the SDP approach is able to find a combination that exploits this complementarity.

6.2.2 Protein Function Prediction

Here we illustrate the SDP approach for fusing heterogeneous genomic data in order to predict protein function in yeast; see Lanckriet et al. (2004) for more details. The task is to predict functional classifications associated with yeast proteins. We use as a gold standard the functional catalogue provided by the MIPS Comprehensive Yeast Genome Database (CYGD—`mips.gsf.de/proj/yeast`). The top-level categories in the functional hierarchy produce 13 classes, which contain 3588 proteins; the remaining yeast proteins have uncertain function and are therefore not used in evaluating the classifier. Because a given protein can belong to several functional classes, we cast the prediction problem as 13 binary classification tasks, one for each functional class. Using this setup, we follow the experimental paradigm of Deng et al. (2003).

The primary input to the classification algorithm is a collection of kernel matrices representing different types of data:

1. Amino acid sequences: this kernel incorporates information about the domain structure of each protein, by looking at the presence or absence in the protein of Pfam domains (`pfam.wustl.edu`). The corresponding kernel is simply the inner product between binary vectors describing the presence or absence of one Pfam domain. Afterwards, we also construct a richer kernel by replacing the binary scoring with log E-values using the HMMER software toolkit (`hmmer.wustl.edu`). Moreover, an additional kernel matrix is constructed by applying the Smith-Waterman (SW) pairwise sequence comparison algorithm (Smith and Waterman, 1981) to the yeast protein sequences and applying the empirical kernel map (Tsuda, 1999).

2. Protein-protein interactions: this type of data can be represented as a graph, with proteins as nodes and interactions as edges. Such interaction graph allows to establish similarities among proteins through the construction of a corresponding diffusion kernel (Kondor and Lafferty, 2002).

3. Genetic interactions: in a similar way, these interactions give rise to a diffusion kernel.

4. Protein complex data: co-participation in a protein complex can be seen as a weak sort of interaction, giving rise to a third diffusion kernel.

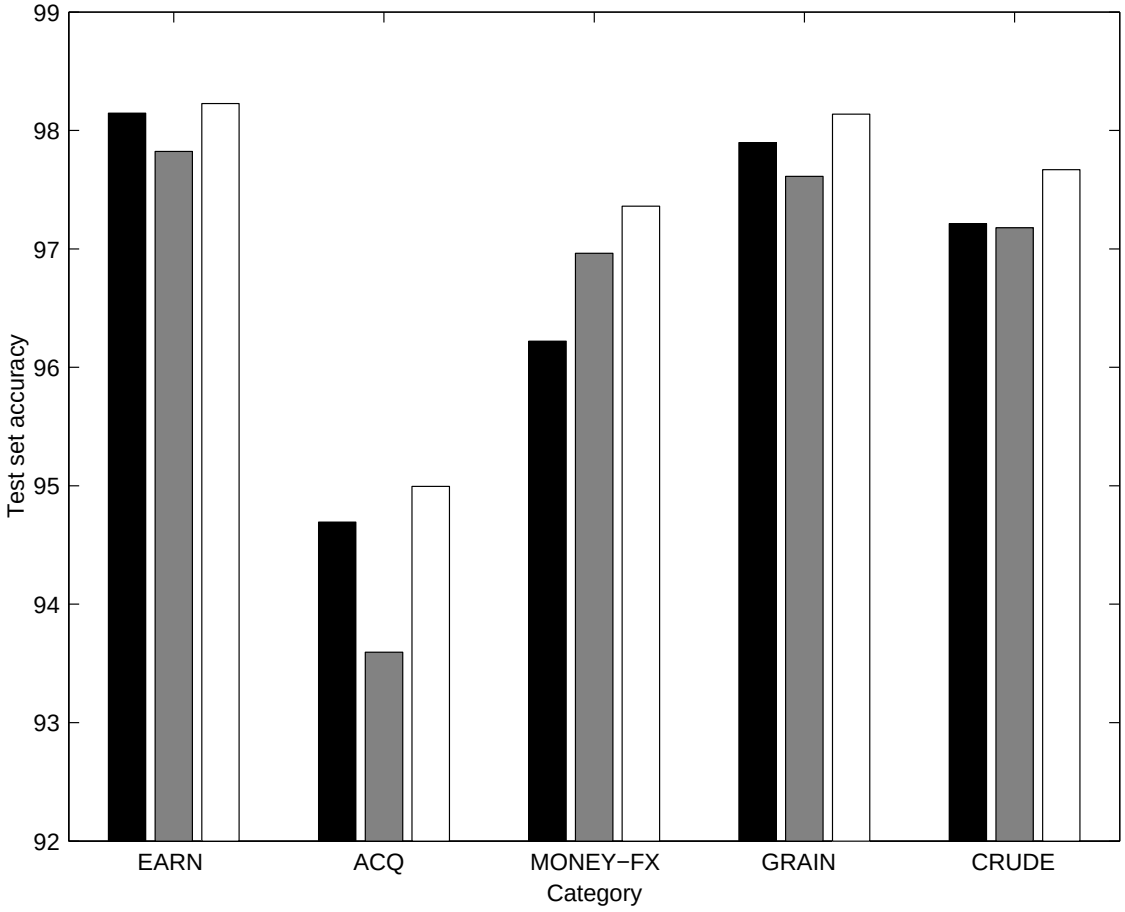


Figure 1: **Classification performance for the top five Reuters-21578 topics.** The height of each bar is proportional to the average test set accuracy for a 1-norm soft margin SVM with $C = 1$. Black bars correspond to using only kernel matrix K_1; grey bars correspond to using only kernel matrix K_2, and white bars correspond to the optimal combination $\mu_1^* K_1 + \mu_2^* K_2$. The kernel matrices K_1 and K_2 are derived from different types of data, i.e., from the "bag-of-words" representation of documents and the concept-based graphical model representation (with 500 concepts) of documents respectively. For c we used $c = \text{trace}(K_1) + \text{trace}(K_2) = 4000$. The standard errors across the 20 experiments are approximately 0.1 or smaller; indeed, all of the depicted differences between the optimal combination and the individual kernels are statistically significant except for EARN.

5. Expression data: two genes with similar expression profiles are likely to have similar functions; accordingly, Deng et al. (2003) convert the expression matrix to a square binary interaction matrix in which a 1 indicates that the corresponding pair of expression profiles exhibits a Pearson correlation greater than 0.8. This can be used to define a diffusion kernel. Also, a richer Gaussian kernel is defined directly on the expression profiles.

In order to compare the SDP/SVM approach to the Markov random field (MRF) method of Deng et al. (2003), Lanckriet et al. (2004) perform two variants of the experiment: one in which the five kernels are restricted to contain precisely the same binary information as used by the MRF method, and a second experiment in which the richer Pfam and expression kernels are used and the SW kernel is added. They show that a combined SVM classifier trained with the SDP approach performs better than an SVM trained on any single type of data. Moreover it outperforms the MRF method designed for this data set. To illustrate the latter, Figure 2 presents the average ROC scores on the test set when performing five-fold cross-validation three times.

The figure shows that, for each of the 13 classifications, the ROC score of the SDP/SVM method is better than that of the MRF method. Overall, the mean ROC improves from 0.715 to 0.854. The improvement of the SDP/SVM method over the MRF method is consistent and statistically significant across all 13 classes. An additional improvement, though not as large and only statistically significant for nine of the 13 classes, is gained by using richer kernels and adding the SW kernel.

7. Discussion

In this paper we have presented a new method for learning a kernel matrix from data. Our approach makes use of semidefinite programming (SDP) ideas. It is motivated by the fact that every symmetric, positive semidefinite matrix can be viewed as a kernel matrix (corresponding to a certain embedding of a finite set of data), and the fact that SDP deals with the optimization of convex cost functions over the convex cone of positive semidefinite matrices (or convex subsets of this cone). Thus convex optimization and machine learning concerns merge to provide a powerful methodology for learning the kernel matrix with SDP.

We have focused on the transductive setting, where the labeled data are used to learn an embedding, which is then applied to the unlabeled part of the data. Based on a new generalization bound for transduction, we have shown how to impose convex constraints that effectively control the capacity of the search space of possible kernels and yield an efficient learning procedure that can be implemented by SDP. Furthermore, this approach leads to a convex method to learn the 2-norm soft margin parameter in support vector machines, solving an important open problem. Promising empirical results are reported on standard benchmark datasets; these results show that the new approach provides a principled way to combine multiple kernels to yield a classifier that is comparable with the best individual classifier, and can perform better than any individual kernel. Performance is also comparable with a classifier in which the kernel hyperparameter is tuned with cross-validation; our approach achieves the effect of this tuning without cross-validation.

We have also shown how optimizing a linear combination of kernel matrices provides a novel method for fusing heterogeneous data sources. In this case, the empirical results show a significant improvement of the classification performance for the optimal combination of kernels when compared to individual kernels.

There are several challenges that need to be met in future research on SDP-based learning algorithms. First, it is clearly of interest to explore other convex quality measures for a kernel matrix, which may be appropriate for other learning algorithms. For example, in the setting of Gaussian

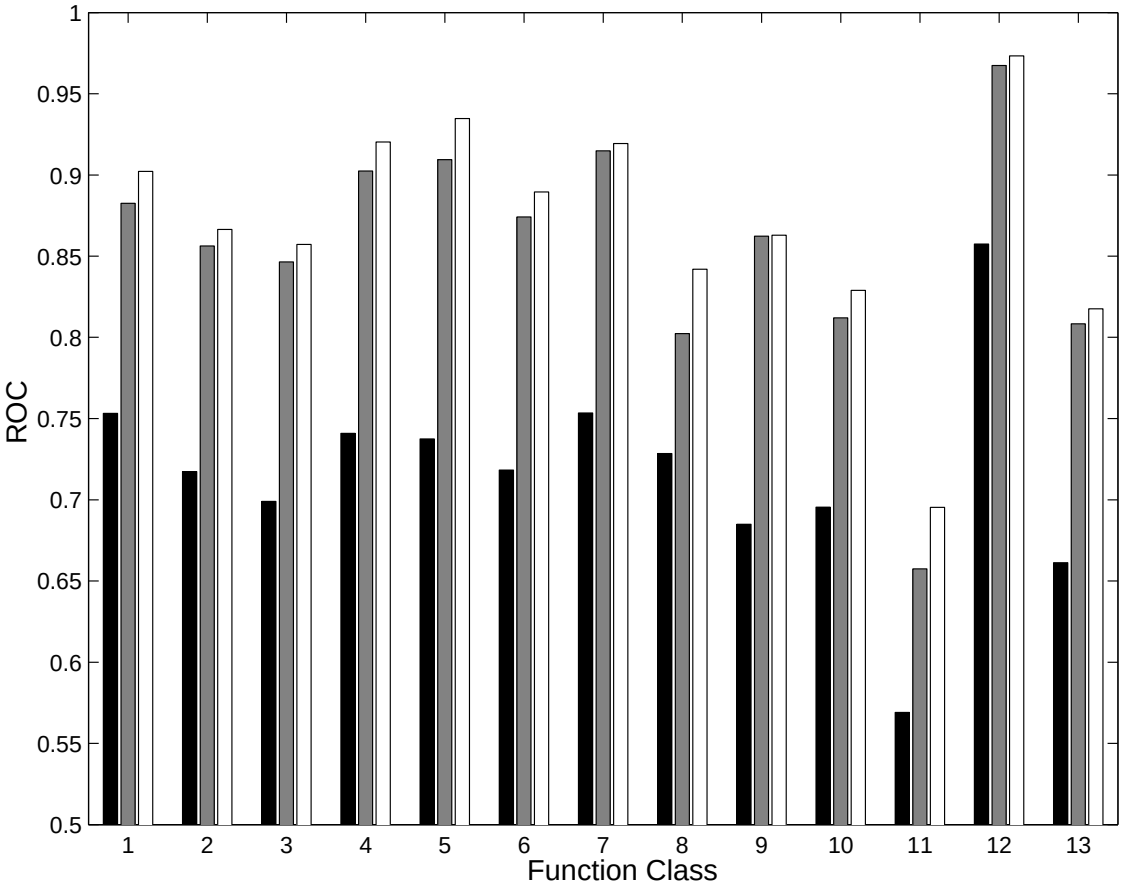


Figure 2: **Classification performance for the 13 functional protein classes.** The height of each bar is proportional to the ROC score. The standard error across the 13 experiments is usually 0.01 or smaller, so most of the depicted differences are statistically significant: between black and grey bars, all depicted differences are statistically significant, while nine of the 13 differences between grey and white bars are statistically significant. Black bars correspond to the MRF method of Deng *et al.*; grey bars correspond to the SDP/SVM method using five kernels computed on binary data, and white bars correspond to the SDP/SVM using the enriched Pfam kernel and replacing the expression kernel with the SW kernel. See Lanckriet et al. (2004) for more details.

processes, the relative entropy between the zero-mean Gaussian process prior P with covariance kernel K and the corresponding Gaussian process approximation Q to the true intractable posterior process depends on K as

$$D[P||Q] = \frac{1}{2}\log\det K + \frac{1}{2}\text{trace}\left(\mathbf{y}^T K\mathbf{y}\right) + d,$$

where the constant d is independent of K. One can verify that $D[P||Q]$ is convex with respect to $R = K^{-1}$ (see, e.g., Vandenberghe et al., 1998). Minimizing this measure with respect to R, and thus K, is motivated from PAC-Bayesian generalization error bounds for Gaussian processes (see, e.g., Seeger, 2002) and can be achieved by solving a so-called *maximum-determinant problem* (Vandenberghe et al., 1998)—an even more general framework that contains semidefinite programming as a special case.

Second, the investigation of other parameterizations of the kernel matrix is an important topic for further study. While the linear combination of kernels that we have studied here is likely to be useful in many practical problems—capturing a notion of combining Gram matrix "experts"—it is also worth considering other parameterizations as well. Any such parameterizations have to respect the constraint that the quality measure for the kernel matrix is convex with respect to the parameters of the proposed parameterization. One class of examples arises via the positive definite matrix completion problem (Vandenberghe et al., 1998). Here we are given a symmetric kernel matrix K that has some entries which are fixed. The remaining entries—the parameters in this case—are to be chosen such that the resulting matrix is positive definite, while simultaneously a certain cost function is optimized, e.g., $\text{trace}(SK) + \log\det K^{-1}$, where S is a given matrix. This specific case reduces to solving a maximum-determinant problem which is convex in the unknown entries of K, the parameters of the proposed parameterization.

A third important area for future research consists in finding faster implementations of semidefinite programming. As in the case of quadratic programming (Platt, 1999), it seems likely that special purpose methods can be developed to exploit the exchangeable nature of the learning problem in classification and result in more efficient algorithms.

Finally, by providing a general approach for combining heterogeneous data sources in the setting of kernel-based statistical learning algorithms, this line of research suggests an important role for kernel matrices as general building blocks of statistical models. Much as in the case of finite-dimensional sufficient statistics, kernel matrices generally involve a significant reduction of the data and represent the only aspects of the data that are used by subsequent algorithms. Moreover, given the panoply of methods that are available to accommodate not only the vectorial and matrix data that are familiar in classical statistical analysis, but also more exotic data types such as strings, trees and graphs, kernel matrices have an appealing universality. It is natural to envision libraries of kernel matrices in fields such as bioinformatics, computational vision, and information retrieval, in which multiple data sources abound. Such libraries would summarize the statistically-relevant features of primary data, and encapsulate domain specific knowledge. Tools such as the semidefinite programming methods that we have presented here can be used to bring these multiple data sources together in novel ways to make predictions and decisions.

Acknowledgements

We acknowledge support from ONR MURI N00014-00-1-0637 and NSF grant IIS-9988642. Sincere thanks to Tijl De Bie for helpful conversations and suggestions, as well as to Lijuan Cai and Thomas Hofmann for providing the data for the Reuters-21578 experiments.

Appendix A. Proof of Result (54)

For the case $K_i = \mathbf{v}_i \mathbf{v}_i^T$, with $\mathbf{v}_i$ orthonormal, the original learning problem (52) becomes

$$\begin{array}{rl} \max\limits_{K} & \left\langle K_{tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F \\ \text{subject to} & \langle K, K \rangle_F \leq 1, \\ & K \succeq 0, \\ & K = \sum\limits_{i=1}^{m} \mu_i \mathbf{v}_i \mathbf{v}_i^T. \end{array} \tag{61}$$

Expanding this further gives

$$\begin{aligned} \left\langle K_{tr}, \mathbf{y}\mathbf{y}^T \right\rangle_F &= \text{trace}(K(1:n_{tr}, 1:n_{tr})\mathbf{y}\mathbf{y}^T) \\ &= \text{trace}((\sum_{i=1}^{m} \mu_i \mathbf{v}_i(1:n_{tr})\mathbf{v}_i(1:n_{tr})^T)\mathbf{y}\mathbf{y}^T) \\ &= \sum_{i=1}^{m} \mu_i \text{trace}(\bar{\mathbf{v}}_i \bar{\mathbf{v}}_i^T \mathbf{y}\mathbf{y}^T) \\ &= \sum_{i=1}^{m} \mu_i (\bar{\mathbf{v}}_i^T \mathbf{y})^2, \end{aligned} \tag{62}$$

$$\begin{aligned} \langle K, K \rangle_F &= \text{trace}(K^T K) \\ &= \text{trace}(KK) \\ &= \text{trace}((\sum_{i=1}^{m} \mu_i \mathbf{v}_i \mathbf{v}_i^T)(\sum_{j=1}^{m} \mu_j \mathbf{v}_j \mathbf{v}_j^T)) \\ &= \text{trace}(\sum_{i,j=1}^{m} \mu_i \mu_j \mathbf{v}_i \mathbf{v}_i^T \mathbf{v}_j \mathbf{v}_j^T) \\ &= \text{trace}(\sum_{i=1}^{m} \mu_i^2 \mathbf{v}_i \mathbf{v}_i^T) \\ &= \sum_{i=1}^{m} \mu_i^2 \text{trace}(\mathbf{v}_i \mathbf{v}_i^T) \\ &= \sum_{i=1}^{m} \mu_i^2 \text{trace}(\mathbf{v}_i^T \mathbf{v}_i) \\ &= \sum_{i=1}^{m} \mu_i^2 \end{aligned} \tag{63}$$

with $\bar{\mathbf{v}}_i = \mathbf{v}_i(1:n_{tr})$. We used the fact that $\text{trace}(ABC) = \text{trace}(BCA)$ (if the products are well-defined) and that the vectors $\mathbf{v}_i, i = 1, \ldots, n$ are orthonormal: $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$. Furthermore, because the $\mathbf{v}_i$ are orthogonal, the μ_i in $K = \sum_{i=1}^{m} \mu_i \mathbf{v}_i \mathbf{v}_i^T$ are the eigenvalues of K. This implies

$$K \succeq 0 \Leftrightarrow \boldsymbol{\mu} \geq 0 \Leftrightarrow \mu_i \geq 0, \quad i = 1, \ldots, m. \tag{64}$$

Using (62), (63) and (64) in (61), we obtain the following optimization problem:

$$\begin{aligned} \max_{\mu_i} \quad & \sum_{i=1}^{m} \mu_i (\bar{\mathbf{v}}_i^T \mathbf{y})^2 \\ \text{subject to} \quad & \sum_{i=1}^{m} \mu_i^2 \leq 1 \\ & \mu_i \geq 0, \quad i = 1, \ldots, m, \end{aligned}$$

which yields the result (54).

Appendix B. Proof of Theorem 24

For a function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, define

$$\begin{aligned} \hat{\mathbf{E}}_1 g(\mathbf{X}, \mathbf{Y}) &= \frac{1}{n} \sum_{i=1}^{n} g(\mathbf{x}_i, y_i), \\ \hat{\mathbf{E}}_2 g(\mathbf{X}, \mathbf{Y}) &= \frac{1}{n} \sum_{i=1}^{n} g(\mathbf{x}_{n+i}, y_{n+i}). \end{aligned}$$

Define a margin cost function $\phi : \mathbb{R} \to \mathbb{R}^+$ as

$$\phi(a) = \begin{cases} 1 & \text{if } a \leq 0, \\ 1 - a & 0 < a \leq 1, \\ 0 & a > 1. \end{cases}$$

Notice that in the 1-norm soft margin cost function, the slack variable ξ_i is a convex upper bound on $\phi(y_i f(x_i))$ for the kernel classifier f, that is,

$$\max\{1 - a, 0\} \geq \phi(a) \geq \mathbf{1}[a \leq 0],$$

where the last expression is the indicator function of $a \leq 0$.

The proof of the first part is due to Koltchinskii and Panchenko Koltchinskii and Panchenko (2002), and involves the following five steps:

Step 1. For any class F of real functions defined on $\mathcal{X}$,

$$\sup_{f \in F} \mathrm{er}(f) - \hat{\mathbf{E}}_1 \phi(Y f(\mathbf{X})) \leq \sup_{f \in F} \hat{\mathbf{E}}_2 \phi(Y f(\mathbf{X})) - \hat{\mathbf{E}}_1 \phi(Y f(\mathbf{X})).$$

To see this, notice that $\mathrm{er}(f)$ is the average over the test set of the indicator function of $Y f(\mathbf{X}) \leq 0$, and that $\phi(Y f(\mathbf{X}))$ bounds this function.

Step 2. For any class G of $[0, 1]$-valued functions,

$$\Pr\left(\sup_{g \in G} \hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g \geq \mathbf{E}\left(\sup_{g \in G} \hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g \right) + \epsilon \right) \leq \exp\left(\frac{-\epsilon^2 n}{4} \right),$$

where the expectation is over the random permutation. This follows from McDiarmid's inequality. To see this, we need to define the random permutation π using a set of $2n$ independent random variables. To this end, choose $\pi_1, \ldots, \pi_{2n}$ uniformly at random from the interval $[0, 1]$. These

are almost surely distinct. For $j = 1, \ldots, 2n$, define $\pi(j) = |\{i : \pi_i \leq \pi_j\}|$, that is, $\pi(j)$ is the position of π_j when the random variables are ordered by size. It is easy to see that, for any g, $\hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g$ changes by no more than $2/n$ when one of the π_i changes. McDiarmid's bounded difference inequality (McDiarmid, 1989) implies the result.

Step 3. For any class G of $[0,1]$-valued functions,

$$\mathbf{E}\left(\sup_{g\in G} \hat{\mathbf{E}}_2 g - \hat{\mathbf{E}}_1 g\right) \leq \hat{R}_{2n}(G) + \frac{4}{\sqrt{n}},$$

where $\hat{R}_{2n}(G) = \mathbf{E}\sup_{g\in G} \frac{1}{n}\sum_{i=1}^{2n} \sigma_i g(\mathbf{X}_i, Y_i)$, and the expectation is over the independent, uniform, $\{\pm 1\}$-valued random variables $\sigma_1, \ldots, \sigma_{2n}$. This result is essentially Lemma 3 of (Bartlett and Mendelson, 2002); that lemma contained a similar bound for i.i.d. data, but the same argument holds for fixed data, randomly permuted.

Step 4. If the class F of real-valued functions defined on $\mathcal{X}$ is closed under negations, $\hat{R}_{2n}(\phi \circ F) \leq \hat{R}_{2n}(F)$, where each $f \in F$ defines a $g \in \phi \circ F$ by $g(\mathbf{x}, y) = \phi(y f(\mathbf{x}))$. This bound is the contraction lemma of Ledoux and Talagrand (1991).

Step 5. For the class $F_{\mathcal{K}}$ of kernel expansions, notice (as in the proof of Lemma 26 of Bartlett and Mendelson (2002)) that

$$\begin{aligned}
\hat{R}_{2n}(F_{\mathcal{K}}) &= \frac{1}{n}\mathbf{E}\max_{f\in F_{\mathcal{K}}} \sum_{i=1}^{2n} \sigma_i f(\mathbf{X}_i) \\
&= \frac{1}{n}\mathbf{E}\max_{K\in\mathcal{K}} \max_{\|\mathbf{w}\|\leq 1/\gamma} \langle \mathbf{w}, \sum_{i=1}^{2n} \sigma_i \Phi(\mathbf{X}_i)\rangle \\
&= \frac{1}{n\gamma}\mathbf{E}\max_{K\in\mathcal{K}} \left\| \sum_{i=1}^{2n} \sigma_i \Phi(\mathbf{X}_i) \right\| \\
&\leq \frac{1}{n\gamma}\sqrt{\mathbf{E}\max_{K\in\mathcal{K}} \boldsymbol{\sigma}^T K \boldsymbol{\sigma}} \\
&= \frac{1}{n\gamma}\sqrt{\mathcal{C}(\mathcal{K})},
\end{aligned}$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{2n})$ is the vector of Rademacher random variables.

Combining gives the first part of the theorem. For the second part, consider

$$\mathcal{C}(\mathcal{K}_c) = \mathbf{E}\max_{K\in\mathcal{K}_c} \boldsymbol{\sigma}^T K \boldsymbol{\sigma} = \mathbf{E}\max_{\boldsymbol{\mu}} \sum_{j=1}^{m} \mu_j \boldsymbol{\sigma}^T K_j \boldsymbol{\sigma},$$

where the max is over $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m)$ for which the matrix $K = \sum_{j=1}^m \mu_j K_j$ satisfies the conditions $K \succeq 0$ and $\mathrm{trace}(K) \leq c$. Now,

$$\mathrm{trace}(K) = \sum_{j=1}^{m} \mu_j \mathrm{trace}(K_j),$$

and each trace in the sum is positive, so the supremum must be achieved for $\mathrm{trace}(K) = c$. So we can write

$$\mathcal{C}(\mathcal{K}_c) = c\mathbf{E}\max_{K\in\mathcal{K}_c} \sum_{j=1}^{m} \boldsymbol{\sigma}^T \frac{K}{\mathrm{trace}(K)} \boldsymbol{\sigma}.$$

Notice that $\boldsymbol{\sigma}^T K \boldsymbol{\sigma}$ is no more than $\lambda \|\boldsymbol{\sigma}\|^2 = n\lambda$, where λ is the maximum eigenvalue of K. Using $\lambda \leq \mathrm{trace}(K) = c$ shows that $\mathcal{C}(\mathcal{K}_c) \leq cn$.

Finally, for $\mathcal{K}_c^+$ we have

$$\begin{aligned}
\mathcal{C}(\mathcal{K}_c^+) &= \mathbf{E} \max_{K \in \mathcal{K}_c^+} \boldsymbol{\sigma}^T K \boldsymbol{\sigma} \\
&= \mathbf{E} \max_{\mu_j} \sum_{j=1}^{m} \mu_j \boldsymbol{\sigma}^T K_j \boldsymbol{\sigma} \\
&= \mathbf{E} \max_{j} \frac{c}{\mathrm{trace}(K_j)} \boldsymbol{\sigma}^T K_j \boldsymbol{\sigma}.
\end{aligned}$$

Since each term in the maximum is non-negative, we can replace it with a sum to show that

$$\begin{aligned}
\mathcal{C}(\mathcal{K}_c^+) &\leq c\mathbf{E}\boldsymbol{\sigma}^T \left(\sum_j \frac{K_j}{\mathrm{trace}(K_j)} \right) \boldsymbol{\sigma} \\
&= cm.
\end{aligned}$$

Alternatively, we can write $\boldsymbol{\sigma}^T K_j \boldsymbol{\sigma} \leq \lambda_j \|\boldsymbol{\sigma}\| = \lambda_j n$, where λ_j is the maximum eigenvalue of K_j. This shows that

$$\mathcal{C}(\mathcal{K}_c^+) \leq cn \max_j \frac{\lambda_j}{\mathrm{trace}(K_j)}.$$

References

Andersen, E. D. and Andersen, A. D. (2000). The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In Frenk, H., Roos, C., Terlaky, T., and Zhang, S., editors, *High Performance Optimization*, pages 197–232. Kluwer Academic Publishers.

Bartlett, P. L. (1998). 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.

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

Bennett, K. P. and Bredensteiner, E. J. (2000). Duality and geometry in SVM classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 57–64. Morgan Kaufmann.

Boyd, S. and Vandenberghe, L. (2003). Convex optimization. Course notes for EE364, Stanford University. Available at `http://www.stanford.edu/class/ee364`.

Breiman, L. (1998). Arcing classifiers. *Annals of Statistics*, 26(3):801–849.

Cai, L. and Hofmann, T. (2003). Text categorization by boosting automatically extracted concepts. In *Proceedings of the 26th ACM-SIGIR International Conference on Research and Development in Information Retrieval*. ACM Press.

Cristianini, N., Kandola, J., Elisseeff, A., and Shawe-Taylor, J. (2001). On kernel target alignment. Technical Report NeuroColt 2001-099, Royal Holloway University London.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. (2002). On kernel-target alignment. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*. MIT Press.

De Bie, T., Lanckriet, G., and Cristianini, N. (2003). Convex tuning of the soft margin parameter. Technical Report CSD-03-1289, University of California, Berkeley.

Deng, M., Chen, T., and Sun, F. (2003). An integrated probabilistic model for functional prediction of proteins. In *RECOMB*, pages 95–103.

Eyheramendy, S., Genkin, A., Ju, W., Lewis, D. D., and Madigan, D. (2003). Sparse bayesian classifiers for text categorization. Technical report, Department of Statistics, Rutgers University.

Huang, Y. (2003). Support vector machines for text categorization based on latent semantic indexing. Technical report, Electrical and Computer Engineering Department, The Johns Hopkins University.

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

Kondor, R. I. and Lafferty, J. (2002). Diffusion kernels on graphs and other discrete input spaces. In Sammut, C. and Hoffmann, A., editors, *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann.

Lanckriet, G. R. G., Deng, M., Cristianini, N., Jordan, M. I., and Noble, W. S. (2004). Kernel-based data fusion and its application to protein function prediction in yeast. In *Pacific Symposium on Biocomputing*.

Ledoux, M. and Talagrand, M. (1991). *Probability in Banach Spaces: Isoperimetry and Processes*. Springer-Verlag.

McDiarmid, C. (1989). On the method of bounded differences. In *Surveys in Combinatorics 1989*, pages 148–188. Cambridge University Press.

Nesterov, Y. and Nemirovsky, A. (1994). *Interior Point Polynomial Methods in Convex Programming: Theory and Applications*. SIAM.

Platt, J. (1999). Using sparseness and analytic QP to speed training of support vector machines. In M. S. Kearns, S. A. Solla, D. A. C., editor, *Advances in Neural Information Processing Systems 11*. MIT Press.

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

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

Seeger, M. (2002). PAC-Bayesian generalization error bounds for Gaussian process classification. Technical Report EDI-INF-RR-0094, University of Edinburgh, Division of Informatics.

Shawe-Taylor, J. and Cristianini, N. (1999). Soft margin and margin distribution. In Smola, A., Schölkopf, B., Bartlett, P., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*. MIT Press.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653. Special issue on Interior Point Methods (CD supplement with software).

Tsuda, K. (1999). Support vector classification with asymmetric kernel function. In Verleysen, M., editor, *Proceedings of the European Symposium on Artificial Neural Networks*, pages 183–188.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite programming. *SIAM Review*, 38(1):49–95.

Vandenberghe, L., Boyd, S., and Wu, S.-P. (1998). Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19(2):499–533.

Journal of Machine Learning Research 5 (2004) 73-99 Submitted 5/03; Revised 10/03; Published 1/04

Dimensionality Reduction for Supervised Learning with Reproducing Kernel Hilbert Spaces

Kenji Fukumizu FUKUMIZU@ISM.AC.JP
Institute of Statistical Mathematics
4-6-7 Minami-Azabu, Minato-ku, Tokyo 106-8569, Japan

Francis R. Bach FBACH@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA

Editor: Chris Williams

Abstract

We propose a novel method of dimensionality reduction for supervised learning problems. Given a regression or classification problem in which we wish to predict a response variable Y from an explanatory variable X, we treat the problem of dimensionality reduction as that of finding a low-dimensional "effective subspace" for X which retains the statistical relationship between X and Y. We show that this problem can be formulated in terms of conditional independence. To turn this formulation into an optimization problem we establish a general nonparametric characterization of conditional independence using covariance operators on reproducing kernel Hilbert spaces. This characterization allows us to derive a contrast function for estimation of the effective subspace. Unlike many conventional methods for dimensionality reduction in supervised learning, the proposed method requires neither assumptions on the marginal distribution of X, nor a parametric model of the conditional distribution of Y. We present experiments that compare the performance of the method with conventional methods.

Keywords: regression, dimensionality reduction, variable selection, feature selection, kernel methods, conditional independence.

1. Introduction

Many statistical learning problems involve some form of dimensionality reduction, either explicitly or implicitly. The goal may be one of *feature selection*, in which we aim to find linear or nonlinear combinations of the original set of variables, or one of *variable selection*, in which we wish to select a subset of variables from the original set. The setting may be unsupervised learning, in which a set of observations of a random vector X are available, or supervised learning, in which desired responses or labels Y are also available. Developing methods for dimensionality reduction requires being clear on the goal and the setting, as methods developed for one combination of goal and setting are not generally appropriate for

another. There are additional motivations for dimensionality reduction that it is also helpful to specify, including: providing a simplified explanation of a phenomenon for a human (possibly as part of a visualization algorithm), suppressing noise so as to make a better prediction or decision, or reducing the computational burden. These various motivations are often complementary.

In this paper we study dimensionality reduction in the setting of supervised learning. Thus, we consider problems in which our data consist of observations of (X, Y) pairs, where X is an m-dimensional explanatory variable and where Y is an ℓ-dimensional response. The variable Y may be either continuous or discrete. We refer to these problems generically as "regression" problems, which indicates our focus on the conditional probability density function $p_{Y|X}(y \mid x)$. In particular, our framework includes discriminative approaches to classification problems, where Y is a discrete label.

We wish to solve a problem of feature selection in which the features are linear combinations of the components of X. In particular, our modeling framework posits that there is an r-dimensional subspace $S \subset \mathbb{R}^m$ such that

$$p_{Y|X}(y \mid x) = p_{Y|\Pi_S X}(y \mid \Pi_S x),$$

for all x and y, where Π_S is the orthogonal projection of $\mathbb{R}^m$ onto S. The subspace S is called the *effective subspace for regression*. Based on a set of observations of (X, Y) pairs, we wish to recover a matrix whose columns span the effective subspace. The effective subspace can help to provide explanation of the statistical relation between X and Y, by isolating the feature vectors that capture that relation. Also, finding such a space can suppress noise, to the extent that the orthogonal direction to the effective subspace is noisy vis-a-vis the prediction of Y.

We approach the problem as a *semiparametric* statistical problem; in particular, we make no assumptions regarding the conditional distribution $p_{Y|\Pi_S X}(y \mid \Pi_S x)$, nor do we make any assumptions regarding the marginal distribution $p_X(x)$. That is, we wish to estimate a finite-dimensional parameter (a matrix whose columns span the effective subspace), while treating the distributions $p_{Y|\Pi_S X}(y \mid \Pi_S x)$ and $p_X(x)$ nonparametrically.

Having found an effective subspace, we may then proceed to build a parametric or nonparametric regression model on that subspace. Thus our approach is an explicit dimensionality reduction method for supervised learning that does not require any particular form of regression model, and can be used as a preprocessor for any supervised learner. This can be compared to the use of methods such as principal components analysis (PCA) in regression, which also make no assumption regarding the subsequent regression model, but fail to make use of the response variable Y.

There are a variety of related approaches in the literature, but most of them involve making specific assumptions regarding the conditional distribution $p_{Y|\Pi_S X}(y \mid \Pi_S x)$, the marginal distribution $p_X(x)$, or both. For example, classical two-layer neural networks involve a linear transformation in the first "layer," followed by a specific nonlinear function and a second layer (Bishop, 1995). Thus, neural networks can be seen as attempting to estimate an effective subspace based on specific assumptions about the regressor $p_{Y|\Pi_S X}(y \mid \Pi_S x)$. Similar comments apply to projection pursuit regression (Friedman and Stuetzle, 1981), ACE (Breiman and Friedman, 1985) and additive models (Hastie and Tibshirani,

1986), all of which provide a methodology for dimensionality reduction in which an additive model $E[Y \mid X] = g_1(\beta_1^T X) + \cdots + g_K(\beta_K^T X)$ is assumed for the regressor.

Bayesian approaches to variable selection include the Automatic Relevance Determination (ARD) method proposed by Neal (1996) for neural networks. Vivarelli and Williams (1999) have adapted this approach to feature selection in the setting of Gaussian process regression. These methods also depend on a specific parametric model, on which a Bayesian prior distribution is assumed.

Canonical correlation analysis (CCA) and partial least squares (PLS, Höskuldsson, 1988, Helland, 1988) are classical multivariate statistical methods that can be used for dimensionality reduction in regression (Fung et al., 2002, Nguyen and Rocke, 2002). These methods are based on a linearity assumption for the regressor, however, and thus are quite strongly parametric.

The line of research that is closest to our work has its origin in a technique known as sliced inverse regression (SIR, Li, 1991). SIR is a semiparametric method for finding effective subspaces in regression. The basic idea is that the range of the response variable Y is partitioned into a set of "slices," and the sample means of the observations X are computed within each slice. This can be viewed as a rough approximation to the inverse regression of X on Y. For univariate Y the method is particularly easy to implement. Noting that the inverse regression must lie in the effective subspace if the forward regression lies in such a subspace, principal component analysis is then used on the sample means to find the effective subspace. Li (1991) has shown that this approach can find effective subspaces, but only under strong assumptions on the marginal distribution $p_X(x)$—in particular, the marginal distribution must be elliptically symmetric.

Further developments in the wake of SIR include principal Hessian directions (pHd, Li, 1992), and sliced average variance estimation (SAVE, Cook and Weisberg, 1991, Cook and Yin, 2001). These are all semiparametric methods in that they make no assumptions about the regressor (see also Cook, 1998). However, they again place strong restrictions on the probability distribution of the explanatory variables. If these assumptions do not hold, there is no guarantee of finding the effective subspace.

There are also related nonparametric approaches that estimate the derivative of the regressor to achieve dimensionality reduction, based on the fact that the derivative of the conditional expectation $E[y \mid B^T x]$ with respect to x belongs to the effective subspace (Samarov, 1993, Hristache et al., 2001). However, nonparametric estimation of derivatives is quite challenging in high-dimensional spaces.

There are also dimensionality reduction methods with a semiparametric flavor in the area of classification, notably the work of Torkkola (2003), who has proposed using nonparametric estimation of the mutual information between X and Y, and subsequent maximization of this estimate of mutual information with respect to a matrix representing the effective subspace.

In this paper we present a novel semiparametric method for dimensionality reduction that we refer to as *Kernel Dimensionality Reduction (KDR)*. KDR is based on the estimation and optimization of a particular class of operators on reproducing kernel Hilbert spaces (Aronszajn, 1950). Although our use of reproducing kernel Hilbert spaces is related to their role in algorithms such as the support vector machine and kernel PCA (Boser et al., 1992, Vapnik et al., 1997, Schölkopf et al., 1998), where the kernel function allows linear op-

erations in function spaces to be performed in a computationally-efficient manner, our work differs in that it cannot be viewed as a "kernelization" of an underlying linear algorithm. Rather, we use reproducing kernel Hilbert spaces to provide characterizations of general notions of independence, and we use these characterizations to design objective functions to be optimized. We build on earlier work by Bach and Jordan (2002), who showed how to use reproducing kernel Hilbert spaces to characterize *marginal independence* between pairs of variables, and thereby design an objective function for independent component analysis. In the current paper, we extend this line of work, showing how to characterize *conditional independence* using reproducing kernel Hilbert spaces. We achieve this by expressing conditional independence in terms of covariance operators on reproducing kernel Hilbert spaces.

How does conditional independence relate to our dimensionality reduction problem? Recall that our problem is to find a projection Π_S of X onto a subspace S such that the conditional probability of Y given X is equal to the conditional probability of Y given $\Pi_S X$. This is equivalent to finding a projection Π_S which makes Y and $(I - \Pi_S)X$ conditionally independent given $\Pi_S X$. Thus we can turn the dimensionality reduction problem into an optimization problem by expressing it in terms of covariance operators.

In a presence of a finite sample, we need to estimate the covariance operator so as to obtain a sample-based objective function that we can optimize. We derive a natural plug-in estimate of the covariance operator, and find that the resulting estimate is identical to the *kernel generalized variance* that has been described earlier by Bach and Jordan (2002) in the setting of independent component analysis. In that setting, the goal is to measure departures from independence, and the minimization of the kernel generalized variance can be viewed as a surrogate for minimizing a certain mutual information. In the dimensionality reduction setting, on the other hand, the goal is to measure *conditional* independence, and minimizing the kernel generalized variance can be viewed as a surrogate for *maximizing* a certain mutual information. Not surprisingly, the derivation that leads to the kernel generalized variance that we present here is quite different from the one presented in the earlier work on kernel ICA. Moreover, the argument that we present here can be viewed as providing a rigorous foundation for other, more heuristic, ways in which the kernel generalized variance has been used, including the model selection algorithms for graphical models presented by Bach and Jordan (2003b).

The paper is organized as follows. In Section 2, we introduce the problem of dimensionality reduction for supervised learning, and describe its relation with conditional independence and mutual information. Section 3 derives the objective function for estimation of the effective subspace for regression, and describes the KDR method. All of the mathematical details needed for the results in Section 3 are presented in the Appendix, which also provides a general introduction to covariance operators in reproducing kernel Hilbert spaces. In Section 4, we present a series of experiments that test the effectiveness of our method, comparing it with several conventional methods. Section 5 describes an extension of KDR to the problem of variable selection. Section 6 presents our conclusions.

2. Dimensionality Reduction for Regression

We consider a regression problem, in which Y is an ℓ-dimensional random vector, and X is an m-dimensional explanatory variable. (Note again that we use "regression" in a generic sense that includes both continuous and discrete Y). The probability density function of Y given X is denoted by $p_{Y|X}(y \mid x)$. Assume that there is an r-dimensional subspace $S \subset \mathbb{R}^m$ such that

$$p_{Y|X}(y \mid x) = p_{Y|\Pi_S X}(y \mid \Pi_S x), \tag{1}$$

for all x and y, where Π_S is the orthogonal projection of $\mathbb{R}^m$ onto S. The subspace S is called the *effective subspace for regression*.

The problem that we treat here is that of finding the subspace S given an *i.i.d.* sample $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ from p_X and $p_{Y|X}$. The crux of the problem is that we assume no *a priori* knowledge of the regressor, and place no assumptions on the conditional probability $p_{Y|X}$.

As in the simpler setting of principal component analysis, we make the (generally unrealistic) assumption that the dimensionality r is known and fixed. While choosing the dimensionality is a very important problem, we separate it from the task of finding the best subspace of a fixed dimensionality, and mainly focus on the latter problem. In Section 6, we briefly discuss various approaches to the estimation of the dimensionality.

The notion of effective subspace can be formulated in terms of conditional independence. Let (B, C) be the m-dimensional orthogonal matrix such that the column vectors of B span the subspace S, and define $U = B^T X$ and $V = C^T X$. Because (B, C) is an orthogonal matrix, we have

$$p_X(x) = p_{U,V}(u, v), \qquad p_{X,Y}(x, y) = p_{U,V,Y}(u, v, y), \tag{2}$$

for the probability density functions. From Equation (2), Equation (1) is equivalent to

$$p_{Y|U,V}(y \mid u, v) = p_{Y|U}(y \mid u).$$

This shows that the effective subspace S is the one which makes Y and V conditionally independent given U (see Figure 1).

Mutual information provides another point of view on the equivalence between conditional independence and the existence of the effective subspace. From Equation (2), it is straightforward to see that

$$I(Y, X) = I(Y, U) + E_U\big[I(Y|U, V|U)\big], \tag{3}$$

where $I(Z, W)$ denotes the mutual information defined by

$$I(Z, W) := \int\int p_{Z,W}(z, w) \log \frac{p_{Z,W}(z, w)}{p_Z(z) p_W(w)} dz\, dw.$$

Because Equation (1) means $I(Y, X) = I(Y, U)$, the effective subspace S is characterized as the subspace which retains the mutual information of X and Y by the projection onto that subspace, or equivalently, which gives $I(Y|U, V|U) = 0$. This is again the conditional independence of Y and V given U.

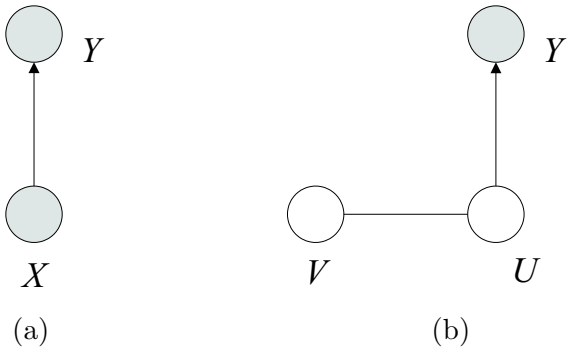


Figure 1: Graphical representation of dimensionality reduction for regression. The variables Y and V are conditionally independent given U, where $X = (U, V)$.

The expression in Equation (3) can be understood in terms of the decomposition of the mutual information according to a tree-structured graphical model—a quantity that has been termed the *T-mutual information* by Bach and Jordan (2003a). Considering the tree $Y - U - V$ in Figure 1(b), we have that the T-mutual information I^T is given by

$$I^T = I(Y, U, V) - I(Y, U) - I(U, V).$$

This is equal to the KL-divergence between a probability distribution on (Y, U, V) and its projection onto the family of distributions that factor according to the tree; that is, the set of distributions that verify $Y \perp\!\!\!\perp V \mid U$. Using Equation (2), we can easily see that $I(Y, U, V) = I(Y, X) + I(U, V)$, and thus we obtain

$$I^T = I(Y, X) - I(Y, U) = E_U[I(Y|U, V|U)]. \tag{4}$$

Then, dimensionality reduction for regression can be viewed as the problem of minimizing the T-mutual information for the fixed tree structure in Figure 1(b).

From Equation (3) and Equation (4), we see that there are two approaches to solve the problem of dimensionality reduction for regression; one is to maximize the mutual information $I(Y, U)$, and the other is to make Y and V conditionally independent given U. We choose the latter in deriving our method. Direct evaluation of mutual information is not a straightforward task in general, because it requires an explicit form for the probability density functions. Although assuming a specific probability model leads readily to a solution, it implies a restriction of the range of problems to which the method can be applied, and does not satisfy our goal of developing a general semiparametric method. Nonparametric estimation of the probabilities also provides an approach to evaluation of mutual information. However, nonparametric estimation and numerical integration do not give accurate estimates for high-dimensional variables, and it is a challenge to make a nonparametric method viable (Torkkola, 2003). An alternative semiparametric approach was presented by Bach and Jordan (2002), who showed that the kernel generalized variance could serve as a surrogate for the mutual information in the setting of independent component analysis.

Our approach to dimensionality reduction also makes use of the kernel generalized variance. However, we do not take the mutual information as our point of departure, because we expect to be far from the setting of mutual independence in which Bach and Jordan (2002) showed that the mutual information and kernel generalized variance are closely related. Instead, we take an entirely different path, presenting a rigorous characterization of conditional independence using reproducing kernels, and showing that this characterization leads once again to the kernel generalized variance.

3. Kernel Method for Dimensionality Reduction in Regression

In this section we present our kernel-based method for dimensionality reduction. We discuss the basic definition and properties of cross-covariance operators on reproducing kernel Hilbert spaces, derive an objective function for characterizing conditional independence using cross-covariance operators, and finally present a sample-based objective function based on this characterization.

3.1 Cross-Covariance Operators on Reproducing Kernel Hilbert Spaces

We use cross-covariance operators on reproducing kernel Hilbert spaces to derive an objective function for dimensionality reduction. While cross-covariance operators are generally defined for random variables in Banach spaces (Vakhania et al., 1987, Baker, 1973), the theory is much simpler for reproducing kernel Hilbert spaces. We summarize only basic mathematical facts in this subsection, and defer the details to the Appendix. Let $(\mathcal{H}, k)$ be a reproducing kernel Hilbert space of functions on a set Ω with a positive definite kernel $k : \Omega \times \Omega \to \mathbb{R}$. The inner product of $\mathcal{H}$ is denoted by $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. We consider only real Hilbert spaces for simplicity. The most important aspect of reproducing kernel Hilbert spaces is the reproducing property:

$$\langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x) \quad \text{for all } x \in \Omega \text{ and } f \in \mathcal{H}.$$

Throughout this paper we use the Gaussian kernel

$$k(x_1, x_2) = \exp\bigl(-\|x_1 - x_2\|^2 / \sigma^2\bigr),$$

which corresponds to a Hilbert space of smooth functions.

Let $(\mathcal{H}_1, k_1)$ and $(\mathcal{H}_2, k_2)$ be reproducing kernel Hilbert spaces over measurable spaces $(\Omega_1, \mathcal{B}_1)$ and $(\Omega_2, \mathcal{B}_2)$, respectively, with k_1 and k_2 measurable. For a random vector (X, Y) on $\Omega_1 \times \Omega_2$, the cross-covariance operator from $\mathcal{H}_1$ to $\mathcal{H}_2$ is defined by the relation

$$\langle g, \Sigma_{YX} f \rangle_{\mathcal{H}_2} = E_{XY}[f(X)g(Y)] - E_X[f(X)]E_Y[g(Y)] \tag{5}$$

for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$. Equation (5) implies that the covariance of $f(X)$ and $g(Y)$ is given by the action of the linear operator Σ_{YX} and the inner product. (See the Appendix for a basic exposition of cross-covariance operators.)

Covariance operators provide a useful framework for discussing conditional probability and conditional independence. As we show in Corollary 3 of the Appendix, the following

relation holds between the conditional expectation and the cross-covariance operator, given that Σ_{XX} is invertible:[1]

$$E_{Y|X}[g(Y) \mid X] = \Sigma_{XX}^{-1}\Sigma_{XY}g \quad \text{for all } g \in \mathcal{H}_2, \tag{6}$$

Equation (6) can be understood by analogy to the conditional expectation of Gaussian random variables. If X and Y are Gaussian random variables, it is well known that the conditional expectation is given by

$$E_{Y|X}[a^T Y \mid X = x] = x^T \Sigma_{XX}^{-1}\Sigma_{XY}a, \tag{7}$$

for an arbitrary vector a, where Σ_{XX} and Σ_{XY} are the variance-covariance matrices in the ordinary sense.

3.2 Conditional Covariance Operators and Conditional Independence

We derive an objective function for characterizing conditional independence using cross-covariance operators. Suppose we have random variables X and Y on $\mathbb{R}^m$ and $\mathbb{R}^\ell$, respectively. The variable X is decomposed into $U \in \mathbb{R}^r$ and $V \in \mathbb{R}^{m-r}$ so that $X = (U, V)$. For the function spaces corresponding to Y, U and V, we consider the reproducing kernel Hilbert spaces $(\mathcal{H}_1, k_1)$, $(\mathcal{H}_2, k_2)$, and $(\mathcal{H}_3, k_3)$ on $\mathbb{R}^\ell$, $\mathbb{R}^r$, and $\mathbb{R}^{m-r}$, respectively, each endowed with Gaussian kernels. We define the conditional covariance operator $\Sigma_{YY|U}$ on $\mathcal{H}_1$ by

$$\Sigma_{YY|U} := \Sigma_{YY} - \Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY}, \tag{8}$$

where Σ_{YY}, Σ_{UU}, Σ_{YU} are the corresponding covariance operators. As shown by Proposition 5 in the Appendix, the operator $\Sigma_{YY|U}$ captures the conditional variance of a random variable in the following way:

$$\langle g, \Sigma_{YY|U} g \rangle_{\mathcal{H}_1} = E_U\big[\mathrm{Var}_{Y|U}[g(Y) \mid U]\big], \tag{9}$$

where g is an arbitrary function in $\mathcal{H}_1$. As in the case of Equation (7), we can make an analogy to Gaussian variables. In particular, Equations (8) and (9) can be viewed as the analogs of the following well-known equality for the conditional variance of Gaussian variables:

$$\mathrm{Var}[a^T Y \mid U] = a^T(\Sigma_{YY} - \Sigma_{YU}\Sigma_{UU}^{-1}\Sigma_{UY})a.$$

It is natural to use minimization of $\Sigma_{YY|U}$ as the basis of a method for finding the most informative direction U. This intuition is justified theoretically by Theorem 7 in the Appendix. That theorem shows that

$$\Sigma_{YY|U} \geq \Sigma_{YY|X} \qquad \text{for any} \quad U, \tag{10}$$

and

$$\Sigma_{YY|U} - \Sigma_{YY|X} = 0 \qquad \Longleftrightarrow \qquad Y \perp\!\!\!\perp V \,|\, U,$$

1. Even if Σ_{XX} is not invertible, a similar fact holds. See Corollary 3.

where, in Equation (10), the inequality should be understood as the partial order of self-adjoint operators. From these relations, the effective subspace S can be characterized in terms of the solution to the following minimization problem:

$$\min_S \Sigma_{YY|U}, \quad \text{subject to } U = \Pi_S X. \tag{11}$$

In the following section we show how to turn this population-based criterion into a sample-based criterion that can be optimized in the presence of a finite sample.

3.3 Kernel Generalized Variance for Dimensionality Reduction

To derive a sample-based objective function from Equation (11), we have to estimate the conditional covariance operator from data, and choose a specific way to evaluate the size of self-adjoint operators. While there are many possibilities for approaching these two problems, we make a specific choice, adopting an approach which has been used successfully for independent component analysis (Bach and Jordan, 2002).

We describe a regularized empirical estimate of the cross-covariance operator. Define $\tilde{k}_1^{(i)} \in \mathcal{H}_1$ by $\tilde{k}_1^{(i)} = k_1(\cdot, Y_i) - \frac{1}{n}\sum_{j=1}^n k_1(\cdot, Y_j)$, and $\tilde{k}_2^{(i)} \in \mathcal{H}_2$ similarly using U_i. By replacing the expectation with the empirical average, the covariance is estimated by

$$\big\langle f, \Sigma_{YU} g \big\rangle_{\mathcal{H}_1} \approx \frac{1}{n}\sum_{i=1}^n \big\langle \tilde{k}_1^{(i)}, f \big\rangle \big\langle \tilde{k}_2^{(i)}, g \big\rangle.$$

Let $\hat{K}_Y$ be the centralized Gram matrix (Bach and Jordan, 2002, Schölkopf et al., 1998), defined by

$$\hat{K}_Y = \big(I_n - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\big) G_Y \big(I_n - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^T\big), \tag{12}$$

where $(G_Y)_{ij} = k_1(Y_i, Y_j)$ is the Gram matrix and $\mathbf{1}_n = (1, \ldots, 1)^T$ is the vector with all elements equal to 1. The matrix $\hat{K}_U$ is defined similarly using $\{U_i\}_{i=1}^n$. Then, it is easy to see that $\big\langle \tilde{k}_1^{(i)}, \tilde{k}_1^{(j)} \big\rangle = \big(\hat{K}_Y\big)_{ij}$ and $\big\langle \tilde{k}_2^{(i)}, \tilde{k}_2^{(j)} \big\rangle = \big(\hat{K}_U\big)_{ij}$. If we decompose f and g as $f = \sum_{\ell=1}^n a_i \tilde{k}_1^{(\ell)} + f^\perp$ and $g = \sum_{m=1}^n b_m \tilde{k}_2^{(m)} + g^\perp$, where $f^\perp$ and $g^\perp$ are orthogonal to the linear hull of $\{\tilde{k}_1^{(i)}\}_{i=1}^n$ and $\{\tilde{k}_2^{(i)}\}_{i=1}^n$, respectively, we see that the covariance is approximated by

$$\big\langle f, \Sigma_{YU} g \big\rangle_{\mathcal{H}_1} \approx \sum_{\ell=1}^n \sum_{m=1}^n a_\ell b_m \big(\hat{K}_Y \hat{K}_U\big)_{lm}.$$

Thus, by restricting the covariance operator Σ_{YU} to the n-dimensional subspaces spanned by $\{k_1^{(i)}\}_{i=1}^n$ and $\{k_2^{(i)}\}_{i=1}^n$, we can estimate the operator by

$$\hat{\Sigma}_{YU} = \hat{K}_Y \hat{K}_U.$$

In estimating Σ_{YY} and Σ_{UU}, we use the same regularization technique as Bach and Jordan (2002). The empirical conditional covariance matrix $\hat{\Sigma}_{YY|U}$ is then defined by

$$\hat{\Sigma}_{YY|U} := \hat{\Sigma}_{YY} - \hat{\Sigma}_{YU}\hat{\Sigma}_{UU}^{-1}\hat{\Sigma}_{UY} = (\hat{K}_Y + \varepsilon I_n)^2 - \hat{K}_Y \hat{K}_U (\hat{K}_U + \varepsilon I_n)^{-2} \hat{K}_U \hat{K}_Y, \tag{13}$$

where $\varepsilon > 0$ is a regularization constant.

The size of $\hat{\Sigma}_{YY|U}$ in the ordered set of positive definite matrices can be evaluated by its determinant. Although there are other choices for measuring the size of $\hat{\Sigma}_{YY|U}$, such as the trace and the largest eigenvalue, we focus on the determinant in this paper. Using the Schur decomposition $\det(A - BC^{-1}B^T) = \det\left(\begin{smallmatrix} A & B \\ B^T & C \end{smallmatrix}\right)/\det C$, the determinant of $\hat{\Sigma}_{YY|U}$ can be written as follows:

$$\det \hat{\Sigma}_{YY|U} = \frac{\det \hat{\Sigma}_{[YU][YU]}}{\det \hat{\Sigma}_{UU}},$$

where $\hat{\Sigma}_{[YU][YU]}$ is defined by

$$\hat{\Sigma}_{[YU][YU]} = \begin{pmatrix} \hat{\Sigma}_{YY} & \hat{\Sigma}_{YU} \\ \hat{\Sigma}_{UY} & \hat{\Sigma}_{UU} \end{pmatrix} = \begin{pmatrix} (\hat{K}_Y + \varepsilon I_n)^2 & \hat{K}_Y \hat{K}_U \\ \hat{K}_U \hat{K}_Y & (\hat{K}_U + \varepsilon I_n)^2 \end{pmatrix}.$$

We symmetrize the objective function by dividing by the constant $\det \hat{\Sigma}_{YY}$, which yields the objective function

$$\frac{\det \hat{\Sigma}_{[YU][YU]}}{\det \hat{\Sigma}_{YY} \det \hat{\Sigma}_{UU}}. \tag{14}$$

We refer to the problem of minimizing this function with respect to the choice of subspace S as *Kernel Dimensionality Reduction (KDR)*.

Equation (14) has been termed the "kernel generalized variance" by Bach and Jordan (2002), who used it as a contrast function for independent component analysis. In that setting, the goal is to *minimize* a mutual information (among a set of recovered "source" variables), in the attempt to obtain independent components. Bach and Jordan (2002) showed that the kernel generalized variance is in fact an approximation of the mutual information of the recovered sources, when this mutual information is expanded around the manifold of factorized distributions. In the current setting, on the other hand, our goal is to *maximize* the mutual information $I(Y, U)$, and we certainly do not expect to be near a manifold in which Y and U are independent. Thus the argument for the kernel generalized variance as an objective function in the ICA setting does not apply here. What we have provided in the previous section is an entirely distinct argument that shows that the kernel generalized variance is in fact an appropriate objective function for the dimensionality reduction problem, and that minimizing the kernel generalized variance in Equation (14) can be viewed as a surrogate for maximizing the mutual information $I(Y, U)$, while the value of $I(Y, U)$ may not be explicitly related to the value of the kernel generalized variance.

For the optimization of Equation (14), we use gradient descent with line search in our experiments. In a straightforward implementation, the matrix B is updated iteratively according to

$$\begin{aligned} B(t+1) &= B(t) - \eta \frac{\partial \log \det \hat{\Sigma}_{YY|U}}{\partial B} \\ &= B(t) - \eta \mathrm{Tr}\Big[\hat{\Sigma}_{YY|U}^{-1} \frac{\partial \hat{\Sigma}_{YY|U}}{\partial B}\Big], \end{aligned} \tag{15}$$

where η is optimized through golden section search. The trace term in Equation (15) is rewritten by

$$\mathrm{Tr}\Big[\hat{\Sigma}_{YY|U}^{-1} \frac{\partial \hat{\Sigma}_{YY|U}}{\partial B}\Big] = 2\varepsilon \mathrm{Tr}\Big[\hat{\Sigma}_{YY|U}^{-1} \hat{K}_Y \big(\hat{K}_U + \varepsilon I_n\big)^{-1} \frac{\partial \hat{K}_U}{\partial B} \big(\hat{K}_U + \varepsilon I_n\big)^{-2} \hat{K}_U \hat{K}_Y\Big].$$

All of these matrices are calculated directly from the definitions in Equations (12) and (13).

Given that the numerical task that must be solved in KDR is the same as the numerical task that must be solved in kernel ICA, we can import all of the computational techniques developed by Bach and Jordan (2002) for minimizing kernel generalized variance in the KDR setting. In particular, we can exploit incomplete Cholesky decomposition, which approximates an $n \times n$ positive semidefinite matrix K by $K \approx AA^T$, where A is an $n \times \ell$ matrix for $\ell < n$. Application of this decomposition reduces the $n \times n$ matrices $\hat{K}_Y$ and $\hat{K}_U$ required in Equation (15) to low-rank approximations. This diminishes the computational cost associated with multiplying and inverting large matrices, especially for a large n. For an exposition of incomplete Cholesky decomposition, see Bach and Jordan (2002). Another computational issue, which may arise in minimizing Equation (14), is the problem of local minima, because the objective function is not a convex function. To cope with this problem, we make use of an annealing technique, in which the scale parameter σ for the Gaussian kernel is decreased gradually during the iterations of optimization. For a larger σ, the contrast function is smoother with fewer local optima, which makes optimization easier. The search becomes more accurate as σ is decreased.

4. Experimental Results

We study the effectiveness of the new method through experiments, comparing it with several conventional methods: SIR, pHd, CCA, and PLS. For the experiments with SIR and pHd, we use an implementation in R due to Weisberg (2002).

In all of our experiments, we use a fixed value 0.1 for the regularization coefficient ε; empirically, the performance of the algorithm is robust to small variations in this coefficient. This coefficient could also be chosen using cross-validation.

4.1 Synthetic Data

The data sets Data I and Data II comprise one-dimensional Y and two-dimensional $X = (X_1, X_2)$. One hundred *i.i.d.* data points are generated by

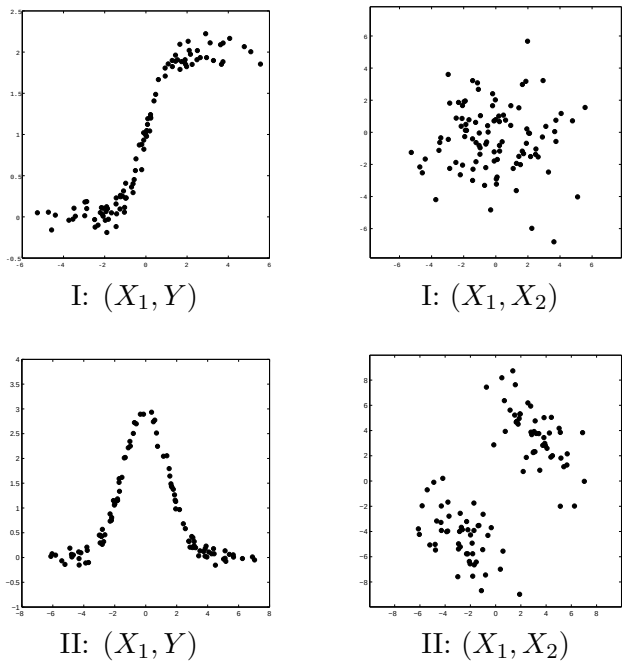
$$\begin{aligned} \text{I}: &\quad Y \sim 1/(1+\exp(-X_1)) + Z, \\ \text{II}: &\quad Y \sim 2\exp(-X_1^2) + Z, \end{aligned}$$

where $Z \sim N(0, 0.1^2)$, and $X = (X_1, X_2)$ follows a normal distribution and a normal mixture with two components for Data I and Data II, respectively. The effective subspace is spanned by $B_0 = (1, 0)^T$ in both cases. The data sets are depicted in Figure 2.

Table 1 shows the angles between B_0 and the estimated direction. For Data I, all the methods except PLS yield a good estimate of B_0. Data II is surprisingly difficult for the conventional methods, presumably because the distribution of X is not spherical and the regressor has a strong nonlinearity. This indicates the weakness of model-based methods; if data do not fit the model, the obtained result may not be meaningful. The KDR method succeeds in finding the correct direction for both data sets.

Data III has 300 samples of 17 dimensional X and one dimensional Y, which are generated by

$$\text{III}: \quad Y \sim 0.9X_1 + 0.2\frac{1}{1+X_{17}} + Z,$$



I: (X_1, Y) I: (X_1, X_2)

II: (X_1, Y) II: (X_1, X_2)

Figure 2: Data I and II. One-dimensional Y depends only on X_1 in $X = (X_1, X_2)$.

	SIR	pHd	CCA	PLS	Kernel
I: angle (rad.)	0.0087	-0.1971	0.0099	0.2736	-0.0014
II: angle (rad.)	-1.5101	-0.9951	-0.1818	0.4554	0.0052

Table 1: Angles between the true and the estimated subspaces for Data I and II.

where $Z \sim N(0, 0.01^2)$ and X follows a uniform distribution on $[0,1]^{17}$. The effective subspace is given by $\boldsymbol{b}_1 = (1, 0, \ldots, 0)$ and $\boldsymbol{b}_2 = (0, \ldots, 0, 1)$. We compare the KDR method with SIR and pHd only—CCA and PLS cannot find a two-dimensional subspace, because Y is one-dimensional. To evaluate the accuracy of the results, we use the multiple correlation coefficient

$$R(\boldsymbol{b}) = \max_{\boldsymbol{\beta} \in B} \frac{\boldsymbol{\beta}^T \Sigma_{XX} \boldsymbol{b}}{\sqrt{\boldsymbol{\beta}^T \Sigma_{XX} \boldsymbol{\beta} \cdot \boldsymbol{b}^T \Sigma_{XX} \boldsymbol{b}}}, \qquad (\boldsymbol{b} \in B_0),$$

which is used in Li (1991). As shown in Table 2, the KDR method outperforms the others in finding the weak contribution of the second direction.

	SIR(10)	SIR(15)	SIR(20)	SIR(25)	pHd	Kernel
$R(\boldsymbol{b}_1)$	0.987	0.993	0.988	0.990	0.110	0.999
$R(\boldsymbol{b}_2)$	0.421	0.705	0.480	0.526	0.859	0.984

Table 2: Correlation coefficients for Data III. SIR(m) indicates SIR with m slices.

Data set	dim. of X	training sample	test sample
Heart-disease	13	149	148
Ionosphere	34	151	200
Breast-cancer-Wisconsin	30	200	369

Table 3: Data description for the binary classification problem.

4.2 Real Data: Classification

In this section we apply the KDR method to classification problems. Many conventional methods of dimensionality reduction for regression are not suitable for classification. In particular, in the case of SIR, the dimensionality of the effective subspace must be less than the number of classes, because SIR uses the average of X in slices along the variable Y. Thus, in binary classification, only a one-dimensional subspace can be found, because at most two slices are available. The methods CCA and PLS have a similar limitation on the dimensionality of the effective subspace; they cannot find a subspace of larger dimensionality than that of Y. Thus our focus is the comparison between KDR and pHd, which is applicable to general binary classification problems. Note that Cook and Lee (1999) discuss dimensionality reduction methods for binary classification, and propose the *difference of covariance (DOC)* method. They compare pHd and DOC theoretically, and show that these methods are the same in binary classification if the population ratio of the classes is 1/2, which is almost the case in our experiments.

In the first experiment, we demonstrate the visualization capability of the dimensionality reduction methods. We use the *Wine* data set in the UCI machine learning repository (Murphy and Aha, 1994) to see how the projection onto a low-dimensional space realizes an effective description of data. The wine data consist of 178 samples with 13 variables and a label of three classes. We apply the KDR method, CCA, PLS, SIR, and pHd to these data. Figure 3 shows the projection onto the two-dimensional subspace estimated by each method. The KDR method separates the data into three classes most completely, while CCA also shows perfect separation. We can see that the data are nonlinearly separable in the two-dimensional space. The other methods do not separate the classes completely.

Next we investigate how much information on Y is preserved in the estimated subspace. After reducing the dimensionality, we use the support vector machine (SVM) method to build a classifier in the reduced space, and compare its accuracy with an SVM trained using the full dimensional vector X.[2] Although we should theoretically use the (unknown) optimum classifier to evaluate the extent of the information preserved in the subspace, we use an SVM classifier whose parameters were chosen by exhaustive grid search. We use the *Heart-disease* data set,[3] *Ionosphere*, and *Wisconsin-breast-cancer* from the UCI repository. A description of these data is presented in Table 3.

2. In our experiments with the SVM, we used the Matlab Support Vector Toolbox by S. Gunn; see http://www.isis.ecs.soton.ac.uk/resources/svminfo.
3. We use the *Cleveland* data set, created by Dr. Robert Detrano of V.A. Medical Center, Long Beach and Cleveland Clinic Foundation. Although the original data set has five classes, we use only "no presence" (0) and "presence" (1-4) for the binary class labels. Samples with missing values are removed in our experiments.

Figure 4 shows the classification rates for the test set in subspaces of various dimensionality. We can see that KDR yields good separation even in low-dimensional subspaces, while pHd is much worse in low dimensions. It is noteworthy that in the Ionosphere data set the classifier in dimensions 5, 10, and 20 outperforms the classifier in the full dimensional space. This is presumably due to the suppression of noise irrelevant to the prediction of Y. These results show that the kernel method successfully finds a subspace which preserves the class information even when the dimensionality is reduced significantly.

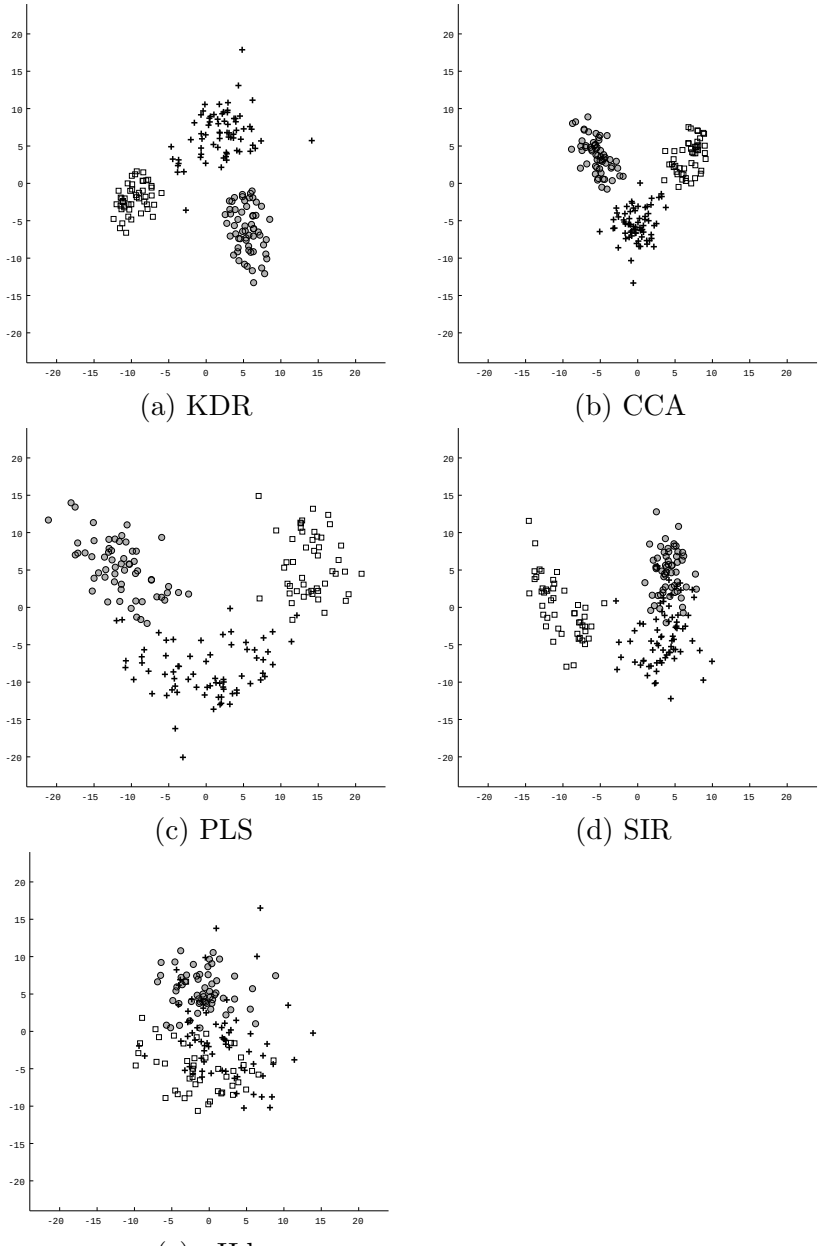
5. Extension to Variable Selection

In this section, we describe an extension of the KDR method to the problem of variable selection. Variable selection is different from dimensionality reduction; the former involves selecting a subset of the explanatory variables $\{X_1, \ldots, X_m\}$ in order to obtain a simplified prediction of Y from X, while the latter involves finding linear combinations of the variables. However, the objective function that we have presented for dimensionality reduction can be extended straightforwardly to variable selection. In particular, given a fixed number of variables to be selected, we can compare the KGV for subspaces spanned by combinations of this number of selected variables. This gives a reasonable way to select variables, because for a subset $W = \{X_{j_1}, \ldots, X_{j_r}\} \subset \{X_1, \ldots, X_m\}$, the variables Y and W^C are conditionally independent given W if and only if Y and $\Pi_{W^C} X$ are conditionally independent given $\Pi_W X$, where Π_W and Π_{W^C} are the orthogonal projections onto the subspaces spanned by W and W^C, respectively. If we try to select r variables from among m explanatory variables, the total number of evaluations is $\binom{m}{r}$.

When $\binom{m}{r}$ is large, we must address the computational cost that arises in comparing large numbers of subsets. As in most other approaches to variable selection (see, e.g., Guyon and Elisseeff, 2003), we propose the use of a greedy algorithm and random search for this combinatorial aspect of the problem. (In the experiments presented in the current paper, however, we confine ourselves to small problems in which all combinations are tractably evaluated).

We apply this kernel-based method of variable selection to the *Boston Housing* data (Harrison and Rubinfeld, 1978) and the *Ozone* data (Breiman and Friedman, 1985), which have been often used as testbed examples for variable selection. Tables 4 and 5 give the detailed description of the data sets. There are 506 samples in the Boston Housing data, for which the variable MV, the median value of house prices in a tract, is estimated by using the 13 other variables. We use the corrected version of the data set given by Gilley and Pace (1996). In the Ozone data in which there are 330 samples, the variable UPO3 (the ozone concentration) is to be predicted by 9 other variables.

Table 6 shows the best three sets of four variables that attain the smallest values of the kernel generalized variance. For the Boston Housing data, RM and LSTAT are included in all the three of the result sets in Table 6, and PTRATIO and TAX are included in two of them. This observation agrees well with the analysis using alternating conditional expectation (ACE) by Breiman and Friedman (1985), which gives RM, LSTAT, PTRATIO, and TAX as the four major contributors. The original motivation in the study was to investigate the influence of nitrogen oxide concentration (NOX) on the house price (Harrison and Rubinfeld, 1978). In accordance with the previous studies, our analysis shows a rel-



(a) KDR (b) CCA

(c) PLS (d) SIR

(e) pHd

Figure 3: Wine data. Projections onto the estimated two-dimensional space. The symbols '+', '□', and gray '◯' represent the three classes.

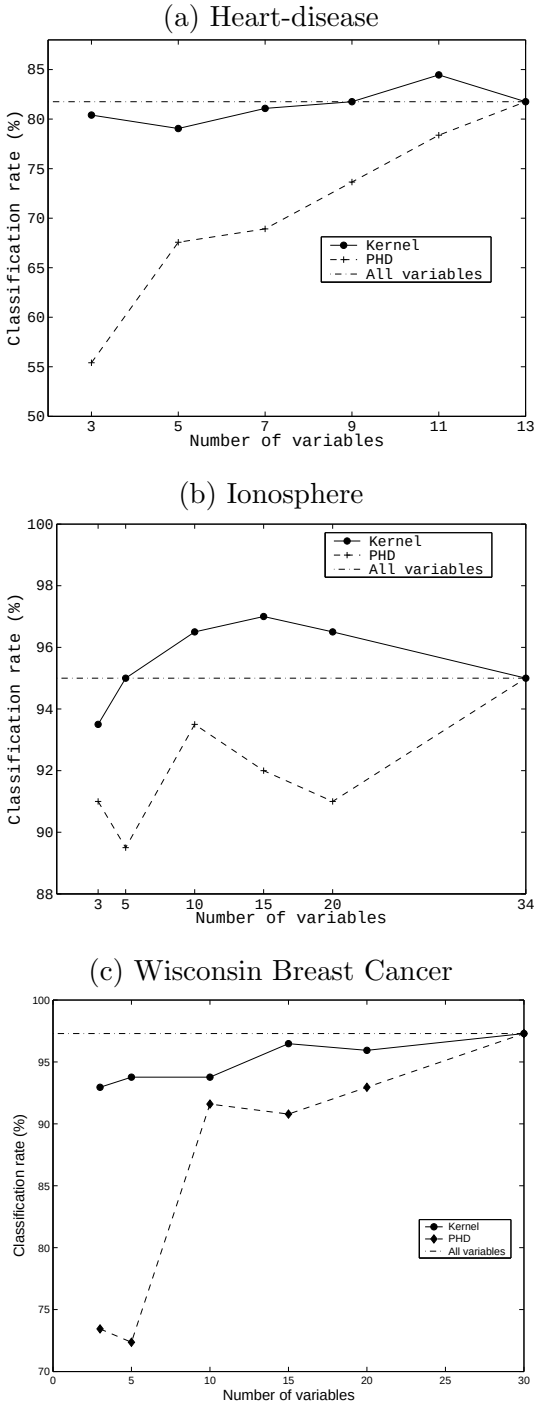


Figure 4: Classification accuracy of the SVM for test data after dimensionality reduction.

Variable		Description
MV	—	median value of owner-occupied home
CRIM	—	crime rate by town
ZN	—	proportion of town's residential land zoned for lots greater than 25,000 square feet
INDUS	—	proportion of nonretail business acres per town
CHAS	—	Charles River dummy (= 1 if tract bounds the Charles River, 0 otherwise)
NOX	—	nitrogen oxide concentration in pphm
RM	—	average number of rooms in owner units
AGE	—	proportion of owner units build prior to 1940
DIS	—	weighted distances to five employment centers in the Boston region
RAD	—	index of accessibility to radial highways
TAX	—	full property tax rate ($/$10,000)
PTRATIO	—	pupil-teacher ratio by town school district
B	—	black proportion of population
LSTAT	—	proportion of population that is lower status

Table 4: Boston Housing Data

Variable		Description
UPO3	—	upland ozone concentration (ppm)
VDHT	—	Vandenburg 500 millibar height (m)
HMDT	—	himidity (percent)
IBHT	—	inversion base height (ft.)
DGPG	—	Daggett pressure gradient (mmhg)
IBTP	—	inversion base temperature (°F)
SBTP	—	Sandburg Air Force Base temperature (°C)
VSTY	—	visibility (miles)
WDSP	—	wind speed (mph)
DAY	—	day of the year

Table 5: Ozone data

atively small contribution of NOX. For the Ozone data, all three of the result sets in the variable selection method include HMDT, SBTP, and IBHT. The variables IBTP, DGPG, and VDHT are chosen in one of the sets. This shows a fair accordance with earlier results by Breiman and Friedman (1985) and Li et al. (2000); the former concludes by ACE that SBTP, IBHT, DGPG, and VSTY are the most influential, and the latter selects HMDT, IBHT, and DGPG using a pHd-based method.

6. Conclusion

We have presented KDR, a new kernel-based approach to dimensionality reduction for regression and classification. One of the most notable aspects of this method is its generality—

Boston	1st	2nd	3rd
CRIM		X	
ZN			
INDUS			
CHAS			
NOX			
RM	X	X	X
AGE			
DIS			X
RAD			
TAX	X		X
PTRATIO	X	X	
B			
LSTAT	X	X	X
KGV	.1768	.1770	.1815

Ozone	1st	2nd	3rd
VDHT			X
HMDT	X	X	X
IBHT	X	X	X
DGPG		X	
IBTP	X		
SBTP	X	X	X
VSTY			
WDSP			
DAY			
KGV	.2727	.2736	.2758

Table 6: Variable selection using the proposed kernel method.

we do not impose any strong assumptions on either the conditional or the marginal distribution. This allows the method to be applicable to a wide range of problems, and gives it a significant practical advantage over existing methods such as CCA, PPR, SIR, and pHd. These methods all impose significant restrictions on the conditional probability, the marginal distribution, or the dimensionality of the effective subspaces.

Our experiments have shown that the KDR method can provide many of the desired effects of dimensionality reduction: it provides data visualization capabilities, it can successfully select important explanatory variables in regression, and it can yield classification performance that is better than the performance achieved with the full-dimensional covariate space. We have also discussed the extension of the KDR method to variable selection. Experiments with classical data sets has shown an accordance with the previous results on these data sets and suggest that further study of this application of KDR is warranted.

The theoretical basis of KDR lies in the nonparametric characterization of conditional independence that we have presented in this paper. Extending earlier work on the kernel-based characterization of independence in ICA (Bach and Jordan, 2002), we have shown that conditional independence can be characterized in terms of covariance operators on a reproducing kernel Hilbert space. While our focus has been on the problem of dimensionality reduction, it is also worth noting that there are many other possible applications of this characterization. In particular, conditional independence plays an important role in the structural definition of probabilistic graphical models, and our results may have applications to model selection and inference in graphical models.

There are several statistical problems which need to be addressed in further research on KDR. First, a basic analysis of the statistical consistency of the KDR-based estimator—the convergence of the estimator to the true subspace when such a space really exists—is needed. We expect that, to prove consistency, we will need to impose a condition on the rate of decrease of the regularization coefficient ε as the sample size n goes to infinity. Second, and most significantly, we need rigorous methods for choosing the dimensionality

of the effective subspace. If the goal is that of achieving high predictive performance after dimensionality reduction, we can use one of many existing methods (e.g., cross-validation, penalty-based methods) to assess the expected generalization as a function of dimensionality. Note in particular that by using KDR as a method to select an estimator given a fixed dimensionality, we have substantially reduced the number of hypotheses being considered, and expect to find ourselves in a regime in which methods such as cross-validation are likely to be effective. It is also worth noting, however, that the goals of dimensionality reduction are not always simply that of prediction; in particular, the search for small sets of explanatory variables will need to be guided by other principles. Finally, asymptotic analysis may provide useful guidance for selecting the dimensionality; an example of such an analysis that we believe can be adopted for KDR has been presented by Li (1991) for the SIR method.

Acknowledgments

This work was done while the first author was visiting the University of California, Berkeley. We thank the action editor and reviewers for their valuable comments. In particular, the proof of Theorem 6 was suggested by one of the reviewers. We also thank Dr. Noboru Murata of Waseda University and Dr. Motoaki Kawanabe of Fraunhofer, FIRST for their helpful comments on the early version of this work. We wish to acknowledge support from JSPS KAKENHI 15700241, ONR MURI N00014-00-1-0637, NSF grant IIS-9988642, and a grant from Intel Corporation.

Appendix A. Cross-Covariance Operators on Reproducing Kernel Hilbert Spaces and Independence of Random Variables

In this appendix, we present additional background and detailed proofs for results relating cross-covariance operators to marginal and conditional independence between random variables.

A.1 Cross-Covariance Operators

While cross-covariance operators are generally defined for random variables on Banach spaces Vakhania et al. (1987), Baker (1973), they are more easily defined on reproducing kernel Hilbert spaces (RKHS). In this subsection, we summarize some of the basic mathematical facts used in Sections 3.1 and 3.3. While we discuss only real Hilbert spaces, extension to the complex case is straightforward.

Theorem 1 *Let $(\Omega_1, \mathcal{B}_1)$ and $(\Omega_2, \mathcal{B}_2)$ be measurable spaces, and let $(\mathcal{H}_1, k_1)$ and $(\mathcal{H}_2, k_2)$ be reproducing kernel Hilbert spaces on Ω_1 and Ω_2, respectively, with k_1 and k_2 measurable. Suppose we have a random vector (X, Y) on $\Omega_1 \times \Omega_2$ such that $E_X[k_1(X, X)]$ and $E_Y[k_2(Y, Y)]$ are finite. Then, there exists a unique operator Σ_{YX} from $\mathcal{H}_1$ to $\mathcal{H}_2$ such that*

$$\langle g, \Sigma_{YX} f \rangle_{\mathcal{H}_2} = E_{XY}[f(X)g(Y)] - E_X[f(X)]E_Y[g(Y)] \tag{16}$$

holds for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$. This is called the cross-covariance operator.

Proof Obviously, the operator is unique, if it exists. From Riesz's representation theorem (see Reed and Simon, 1980, Theorem II.4, for example), the existence of $\Sigma_{YX} f \in \mathcal{H}_2$ for a fixed f can be proved by showing that the right hand side of Equation (16) is a bounded linear functional on $\mathcal{H}_2$. The linearity is obvious, and the boundedness is shown by

$$\begin{aligned}
&\big|E_{XY}[f(X)g(Y)] - E_X[f(X)]E_Y[g(Y)]\big| \\
&\leq E_{XY}\big|\langle k_1(\cdot, X), f\rangle_{\mathcal{H}_1}\langle k_2(\cdot, Y), g\rangle_{\mathcal{H}_2}\big| + E_X\big|\langle k_1(\cdot, X), f\rangle_{\mathcal{H}_1}\big| \cdot E_Y\big|\langle k_2(\cdot, Y), g\rangle_{\mathcal{H}_2}\big| \\
&\leq E_{XY}\big[\|k_1(\cdot, X)\|_{\mathcal{H}_1}\|f\|_{\mathcal{H}_1}\|k_2(\cdot, Y)\|_{\mathcal{H}_2}\|g\|_{\mathcal{H}_2}\big] \\
&\quad + E_X\big[\|k_1(\cdot, X)\|_{\mathcal{H}_1}\|f\|_{\mathcal{H}_1}\big]E_Y\big[\|k_2(\cdot, Y)\|_{\mathcal{H}_2}\|g\|_{\mathcal{H}_2}\big] \\
&\leq \big\{E_X[k_1(X, X)]^{1/2}E_Y[k_2(Y, Y)]^{1/2} + E_X[k_1(X, X)^{1/2}]E_Y[k_2(Y, Y)^{1/2}]\big\}\|f\|_{\mathcal{H}_1}\|g\|_{\mathcal{H}_2} \\
&\leq 2E_X\big[k_1(X, X)\big]^{1/2}E_Y\big[k_2(Y, Y)\big]^{1/2}\|f\|_{\mathcal{H}_1}\|g\|_{\mathcal{H}_2}. \qquad (17)
\end{aligned}$$

For the second last inequality, $\|k(\cdot, x)\|^2_{\mathcal{H}} = k(x, x)$ is used. The linearity of the map Σ_{YX} is given by the uniqueness part of Riesz's representation theorem. ■

From Equation (17), Σ_{YX} is bounded, and by definition, we see $\Sigma^*_{YX} = \Sigma_{XY}$, where A^* denotes the adjoint of A. If the two RKHS are the same, the operator Σ_{XX} is called the *covariance operator*. A covariance operator Σ_{XX} is bounded, self-adjoint, and trace-class.

In an RKHS, conditional expectations can be expressed by cross-covariance operators, in a manner analogous to finite-dimensional Gaussian random variables.

Theorem 2 *Let $(\mathcal{H}_1, k_1)$ and $(\mathcal{H}_2, k_2)$ be RKHS on measurable spaces Ω_1 and Ω_2, respectively, with k_1 and k_2 measurable, and (X, Y) be a random vector on $\Omega_1 \times \Omega_2$. Assume that $E_X[k_1(X, X)]$ and $E_Y[k_2(Y, Y)]$ are finite, and for all $g \in \mathcal{H}_2$ the conditional expectation $E_{Y|X}[g(Y) \mid X = \cdot]$ is an element of $\mathcal{H}_1$. Then, we have for all $g \in \mathcal{H}_2$*

$$\Sigma_{XX}E_{Y|X}[g(Y) \mid X = \cdot] = \Sigma_{XY}g,$$

where Σ_{XX} and Σ_{XY} are the covariance and cross-covariance operator.

Proof For any $f \in \mathcal{H}_1$, we have

$$\begin{aligned}
&\langle f, \Sigma_{XX}E_{Y|X}[g(Y) \mid X = \cdot]\rangle_{\mathcal{H}_1} \\
&\quad = E_X\big[f(X)E_{Y|X}[g(Y) \mid X]\big] - E_X[f(X)]E_X\big[E_{Y|X}[g(Y) \mid X]\big] \\
&\quad = E_{XY}[f(X)g(Y)] - E_X[f(X)]E_Y[g(Y)] \quad = \quad \langle f, \Sigma_{XY}g\rangle_{\mathcal{H}_1}.
\end{aligned}$$

This completes the proof. ■

Corollary 3 *Let $\tilde{\Sigma}^{-1}_{XX}$ be the right inverse of Σ_{XX} on $(Ker\Sigma_{XX})^{\perp}$. Under the same assumptions as Theorem 2, we have*

$$\langle f, \tilde{\Sigma}^{-1}_{XX}\Sigma_{XY}g\rangle = \langle f, E_{Y|X}[g(Y) \mid X = \cdot]\rangle$$

for all $f \in (\mathrm{Ker}\Sigma_{XX})^{\perp}$ and $g \in \mathcal{H}_2$. In particular, if $\mathrm{Ker}\Sigma_{XX} = 0$, we have

$$\Sigma^{-1}_{XX}\Sigma_{XY}g = E_{Y|X}[g(Y) \mid X = \cdot].$$

Proof Note that the product $\tilde{\Sigma}_{XX}^{-1}\Sigma_{XY}$ is well-defined, because $\overline{\mathrm{Range}\Sigma_{XY}} \subset \overline{\mathrm{Range}\Sigma_{XX}} = (\mathrm{Ker}\Sigma_{XX})^{\perp}$. The first inclusion is shown from the expression $\Sigma_{XY} = \Sigma_{XX}^{1/2} V \Sigma_{YY}^{1/2}$ with a bounded operator V (Baker, 1973, Theorem 1), and the second equation holds for any self-adjoint operator. Take $f = \Sigma_{XX} h \in \mathrm{Range}\Sigma_{XX}$. Then, Theorem 2 yields

$$\begin{aligned}\langle f, \tilde{\Sigma}_{XX}^{-1}\Sigma_{XY} g\rangle &= \langle h, \Sigma_{XX}\tilde{\Sigma}_{XX}^{-1}\Sigma_{XX} E_{Y|X}[g(Y) \mid X = \cdot]\rangle \\ &= \langle h, \Sigma_{XX} E_{Y|X}[g(Y) \mid X = \cdot]\rangle = \langle f, E_{Y|X}[g(Y) \mid X = \cdot]\rangle.\end{aligned}$$

This completes the proof. ■

The assumption $E_{Y|X}[g(Y) \mid X = \cdot] \in \mathcal{H}_1$ in Theorem 2 can be simplified so that it can be checked without reference to a specific g.

Proposition 4 *Under the condition of Theorem 2, if there exists $C > 0$ such that*

$$E_{Y|X}[k_2(y_1, Y) \mid X = x_1] E_{Y|X}[k_2(y_2, Y) \mid X = x_2] \leq C k_1(x_1, x_2) k_2(y_1, y_2)$$

for all $x_1, x_2 \in \Omega_1$ and $y_1, y_2 \in \Omega_2$, then for all $g \in \mathcal{H}_2$ the conditional expectation $E_{Y|X}[g(Y) \mid X = \cdot]$ is an element of $\mathcal{H}_1$.

Proof See Theorem 2.3.13 in Alpay (2001). ■

From this proposition, it is obvious that $E_{Y|X}[g(Y) \mid X = \cdot] \in \mathcal{H}_1$ holds, if the range of X and Y are bounded.

For a function f in an RKHS, the expectation of $f(X)$ can be formulated as the inner product of f and a fixed element. Let $(\Omega, \mathcal{B})$ be a measurable space, and $(\mathcal{H}, k)$ be an RKHS on Ω with k measurable. Note that for a random variable X on Ω, the linear functional $f \mapsto E_X[f(X)]$ is bounded if $E_X[k(X, X)]$ exists. By Riesz's theorem, there is $u \in \mathcal{H}$ such that $\langle u, f\rangle_{\mathcal{H}} = E_X[f(X)]$ for all $f \in \mathcal{H}$. If we define $E_X[k(\cdot, X)] \in \mathcal{H}$ by this element u, we formally obtain the equality

$$\langle E_X[k(\cdot, X)], f\rangle_{\mathcal{H}} = E_X[\langle k(\cdot, X), f\rangle_{\mathcal{H}}],$$

which looks like the interchangeability of the expectation by X and the inner product. While the expectation $E_X[k(\cdot, X)]$ can be defined, in general, as an integral with respect to the distribution on $\mathcal{H}$ induced by $k(\cdot, X)$, the element $E_X[k(\cdot, X)]$ is formally obtained as above in a reproducing kernel Hilbert space.

A.2 Conditional Covariance Operator and Conditional Independence

We define the conditional (cross-)covariance operator, and derive its relation with the conditional covariance of random variables. Let $(\mathcal{H}_1, k_1)$, $(\mathcal{H}_2, k_2)$, let $(\mathcal{H}_3, k_3)$ be RKHS on measurable spaces Ω_1, Ω_2, and Ω_3, respectively, and let (X, Y, Z) be a random vector on $\Omega_1 \times \Omega_2 \times \Omega_3$. The *conditional cross-covariance operator of (X, Y) given Z* is defined by

$$\Sigma_{YX|Z} := \Sigma_{YX} - \Sigma_{YZ}\tilde{\Sigma}_{ZZ}^{-1}\Sigma_{ZX}.$$

Because $\mathrm{Ker}\Sigma_{ZZ} \subset \mathrm{Ker}\Sigma_{YZ}$ from the fact $\Sigma_{YZ} = \Sigma_{YY}^{1/2} V \Sigma_{ZZ}^{1/2}$ for some bounded operator V (Baker, 1973, Theorem 1), the operator $\Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{YX}$ can be uniquely defined, even if

Σ_{ZZ}^{-1} is not unique. By abuse of notation, we write $\Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZX}$, when cross-covariance operators are discussed.

The conditional cross-covariance operator is related to the conditional covariance of the random variables.

Proposition 5 *Let $(\mathcal{H}_1, k_1)$, $(\mathcal{H}_2, k_2)$, and $(\mathcal{H}_3, k_3)$ be reproducing kernel Hilbert spaces on measurable spaces Ω_1, Ω_2, and Ω_3, respectively, with k_i measurable, and let (X, Y, Z) be a measurable random vector on $\Omega_1 \times \Omega_2 \times \Omega_3$ such that $E_X[k_1(X,X)]$, $E_Y[k_2(Y,Y)]$, and $E_Z[k_3(Z,Z)]$ are finite. It is assumed that $E_{X|Z}[f(X) \mid Z = \cdot]$ and $E_{Y|Z}[g(Y) \mid Z = \cdot]$ are elements of $\mathcal{H}_3$ for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$. Then, for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$, we have*

$$\begin{aligned}\langle g, \Sigma_{YX|Z} f\rangle_{\mathcal{H}_2} &= E_{XY}[f(X)g(Y)] - E_Z\big[E_{X|Z}[f(X) \mid Z]E_{Y|Z}[g(Y) \mid Z]\big] \\ &= E_Z\big[\mathrm{Cov}_{XY|Z}\big(f(X), g(Y) \mid Z\big)\big]. \end{aligned} \tag{18}$$

Proof From the decomposition $\Sigma_{YZ} = \Sigma_{YY}^{1/2} V \Sigma_{ZZ}^{1/2}$, we have $\Sigma_{ZY} g \in (\mathrm{Ker}\Sigma_{ZZ})^{\perp}$. Then, by Corollary 3, we obtain

$$\begin{aligned}\langle g, \Sigma_{YZ}\tilde{\Sigma}_{ZZ}^{-1}\Sigma_{ZX} f\rangle = \langle \Sigma_{ZY} g, \tilde{\Sigma}_{ZZ}^{-1}\Sigma_{ZX} f\rangle &= \langle \Sigma_{ZY} g, E_{X|Z}[f(X) \mid Z]\rangle \\ &= E_{YZ}\big[g(Y)E_{X|Z}[f(X) \mid Z]\big] - E_X[f(X)]E_Y[g(Y)].\end{aligned}$$

From this equation, the theorem is proved by

$$\begin{aligned}\langle g, \Sigma_{YX|Z} f\rangle &= E_{XY}[f(X)g(Y)] - E_X[f(X)]E_Y[g(Y)] \\ &\quad - E_{YZ}\big[g(Y)E_{X|Z}[f(X) \mid Z]\big] + E_X[f(X)]E_Y[g(Y)] \\ &= E_{XY}[f(X)g(Y)] - E_Z\big[E_{X|Z}[f(X) \mid Z]E_{Y|Z}[g(Y) \mid Z]\big]. \end{aligned} \tag{19}$$

■

The following definition is needed to state our main theorem. Let $(\Omega, \mathcal{B})$ be a measurable space, let $(\mathcal{H}, k)$ be a RKHS over Ω with k measurable and bounded, and let $\mathcal{S}$ be the set of all the probability measures on $(\Omega, \mathcal{B})$. The RKHS $\mathcal{H}$ is called *probability-determining*, if the map

$$\mathcal{S} \ni P \quad \mapsto \quad (f \mapsto E_{X\sim P}[f(X)]) \in \mathcal{H}^*$$

is one-to-one, where $\mathcal{H}^*$ is the dual space of $\mathcal{H}$. From Riesz's theorem, $\mathcal{H}$ is probability-determining if and only if the map

$$\mathcal{S} \ni P \quad \mapsto \quad E_{X\sim P}[k(\cdot, X)] \in \mathcal{H}$$

is one-to-one. For Gaussian kernels, the following theorem can be proved by an argument similar to that used in the proof of Theorem 2 in Bach and Jordan (2002) and the uniqueness of the characteristic function. For completeness, we present another simple proof here.

Theorem 6 *For an arbitrary $\sigma > 0$, the reproducing kernel Hilbert space $\mathcal{H}$ with Gaussian kernel $k(x, y) = \exp(-\|x - y\|^2/\sigma^2)$ on $\mathbb{R}^m$ is probability-determining.*

Proof Suppose P and Q are different probabilities on $\mathbb{R}^m$ such that $E_{Z\sim P}[f(Z)] = E_{Z\sim Q}[f(Z)]$ for all $f \in \mathcal{H}$. Let y_1 and y_2 be two different vectors in $\mathbb{R}^m$, and Y be a random variable with probability $1/2$ for each of $Y = y_1$ and $Y = y_2$. Define a random variable X so that the probability of X given $Y = y_1$ and $Y = y_2$ are P and Q, respectively. Noting that the marginal distribution of X is $(P+Q)/2$, we have for all $f, g \in \mathcal{H}$,

$$\begin{aligned}
&E_{X,Y}[f(X)g(Y)] - E_X[f(X)]E_Y[g(Y)] = E_Y\big[E_{X|Y}[f(X)|Y]g(Y)\big] - E_X[f(X)]E_Y[g(Y)] \\
&= \frac{1}{2}g(y_1)E_{X|Y}[f(X)|Y = y_1] + \frac{1}{2}g(y_2)E_{X|Y}[f(X)|Y = y_2] \\
&\qquad - \frac{1}{2}\big(E_{Z\sim P}[f(Z)] + E_{Z\sim Q}[f(Z)]\big)\frac{g(y_1)+g(y_2)}{2} \\
&= 0.
\end{aligned}$$

From Theorem 2 in Bach and Jordan (2002), X and Y must be independent, which contradicts the construction of X and Y. ■

Recall that for two RKHS $\mathcal{H}_1$ and $\mathcal{H}_2$ on Ω_1 and Ω_2, respectively, the direct product $\mathcal{H}_1 \otimes \mathcal{H}_2$ is the RKHS on $\Omega_1 \times \Omega_2$ with the positive definite kernel $k_1 k_2$ (see Aronszajn, 1950). Note that if the two RKHS have Gaussian kernels, their direct product is also a RKHS with Gaussian kernel, and thus probability-determining. The relation between conditional independence and the conditional covariance operator is given by the following theorem:

Theorem 7 *Let $(\mathcal{H}_{11}, k_{11})$, $(\mathcal{H}_{12}, k_{12})$, and $(\mathcal{H}_2, k_2)$ be reproducing kernel Hilbert spaces on measurable spaces Ω_{11}, Ω_{12}, and Ω_2, respectively, with continuous and bounded kernels. Let $(X, Y) = (U, V, Y)$ be a random vector on $\Omega_{11} \times \Omega_{12} \times \Omega_2$, where $X = (U, V)$, and let $\mathcal{H}_1 = \mathcal{H}_{11} \otimes \mathcal{H}_{12}$ be the direct product. It is assumed that $E_{Y|U}[g(Y) \mid U = \cdot] \in \mathcal{H}_{11}$ and $E_{Y|X}[g(Y) \mid X = \cdot] \in \mathcal{H}_1$ for all $g \in \mathcal{H}_2$. Then, we have*

$$\Sigma_{YY|U} \geq \Sigma_{YY|X}, \tag{20}$$

where the inequality refers to the order of self-adjoint operators, and if further $\mathcal{H}_2$ is probability-determining, the following equivalence holds

$$\Sigma_{YY|X} = \Sigma_{YY|U} \qquad \Longleftrightarrow \qquad Y \perp\!\!\!\perp V \mid U. \tag{21}$$

Proof The right hand side of Equation (21) is equivalent to $P_{Y|X} = P_{Y|U}$, where $P_{Y|X}$ and $P_{Y|U}$ are the conditional probability of Y given X and given U, respectively. Taking the expectation of the well-known equality

$$\mathrm{Var}_{Y|U}[g(Y) \mid U] = E_{V|U}\big[\mathrm{Var}_{Y|U,V}[g(Y) \mid U, V]\big] + \mathrm{Var}_{V|U}\big[E_{Y|U,V}[g(Y) \mid U, V]\big]$$

with respect to U, we derive

$$E_U\big[\mathrm{Var}_{Y|U}[g(Y) \mid U]\big] = E_X\big[\mathrm{Var}_{Y|X}[g(Y) \mid X]\big] + E_U\big[\mathrm{Var}_{V|U}[E_{Y|X}[g(Y) \mid X]]\big]. \tag{22}$$

Since the last term of Equation (22) is nonnegative, we obtain Equation (20) from Proposition 5.

Equality holds if and only if $\mathrm{Var}_{V|U}[E_{Y|X}[g(Y) \mid X]] = 0$ for almost every U, which means $E_{Y|X}[g(Y) \mid X]$ does not depend on V almost surely. This is equivalent to

$$E_{Y|X}[g(Y) \mid X] = E_{Y|U}[g(Y) \mid U]$$

for almost every V and U. Because $\mathcal{H}_2$ is probability-determining, this means $P_{Y|X} = P_{Y|U}$. ■

A.3 Conditional Cross-Covariance Operator and Conditional Independence

Theorem 7 characterizes conditional independence using the conditional covariance operator. Another formulation is possible with a conditional cross-covariance operator.

Let $(\Omega_1, \mathcal{B}_1)$, $(\Omega_2, \mathcal{B}_2)$, and $(\Omega_3, \mathcal{B}_3)$ be measurable spaces, and let (X, Y, Z) be a random vector on $\Omega_1 \times \Omega_2 \times \Omega_3$ with law P_{XYZ}. We define a probability measure $E_Z[P_{X|Z} \otimes P_{Y|Z}]$ on $\Omega_1 \times \Omega_2$ by

$$E_Z[P_{X|Z} \otimes P_{Y|Z}](A \times B) = E_Z\big[E_{X|Z}[\chi_A|Z]\, E_{Y|Z}[\chi_B \mid Z]\big],$$

where χ_A is the characteristic function of a measurable set A. It is canonically extended to any product-measurable sets in $\Omega_1 \times \Omega_2$.

Theorem 8 *Let $(\Omega_i, \mathcal{B}_i)$ $(i = 1, 2, 3)$ be a measurable space, let $(\mathcal{H}_i, k_i)$ be a RKHS on Ω_i with kernel measurable and bounded, and let (X, Y, Z) be a random vector on $\Omega_1 \times \Omega_2 \times \Omega_3$. It is assumed that $E_{X|Z}[f(X) \mid Z = \cdot]$ and $E_{Y|Z}[g(Y) \mid Z = \cdot]$ belong to $\mathcal{H}_3$ for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$, and that $\mathcal{H}_1 \otimes \mathcal{H}_2$ is probability-determining. Then, we have*

$$\Sigma_{YX|Z} = O \qquad \Longleftrightarrow \qquad P_{XY} = E_Z[P_{X|Z} \otimes P_{Y|Z}]. \tag{23}$$

Proof The right-to-left direction is trivial from Theorem 5 and the definition of $E_Z[P_{X|Z} \otimes P_{Y|Z}]$. The left-hand side yields $E_Z[E_{X|Z}[f(X) \mid Z]E_{Y|Z}[g(Y) \mid Z]] = E_{XY}[f(X)g(Y)]$ for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$. Because $\mathcal{H}_1 \otimes \mathcal{H}_2$ is defined as the completion of all the linear combinations of $f_i(x)g_i(y)$ for $f_i \in \mathcal{H}_1$ and $g_i \in \mathcal{H}_2$, we have $E_{(X',Y')\sim Q}[h(X', Y')] = E_{XY}[h(X, Y)]$ for every such linear combination, and thus every $h \in \mathcal{H}_1 \otimes \mathcal{H}_2$ as a limit, where $Q = E_Z[P_{X|Z} \otimes P_{Y|Z}]$. This implies the right-hand side, because $\mathcal{H}_1 \otimes \mathcal{H}_2$ is probability-determining. ■

The right-hand side of Equation (23) is weaker than the conditional independence of X and Y given Z. However, if Z is a part of X, we obtain conditional independence.

Corollary 9 *Let $(\mathcal{H}_{11}, k_{11})$, $(\mathcal{H}_{12}, k_{12})$, and $(\mathcal{H}_2, k_2)$ be reproducing kernel Hilbert spaces on measurable spaces Ω_{11}, Ω_{12}, and Ω_2, respectively, with kernels measurable and bounded. Let $(X, Y) = (U, V, Y)$ be a random vector on $\Omega_{11} \times \Omega_{12} \times \Omega_2$, where $X = (U, V)$, and let $\mathcal{H}_1 = \mathcal{H}_{11} \otimes \mathcal{H}_{12}$ be the direct product. It is assumed that $E_{X|U}[f(X) \mid U = \cdot]$ and $E_{Y|U}[g(Y) \mid U = \cdot]$ belong to $\mathcal{H}_{11}$ for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$, and $\mathcal{H}_1 \otimes \mathcal{H}_2$ is probability-determining. Then, we have*

$$\Sigma_{YX|U} = O \qquad \Longleftrightarrow \qquad Y \perp\!\!\!\perp V \,|\, U. \tag{24}$$

Proof For any measurable sets $A \subset \Omega_{11}$, $B \subset \Omega_{12}$, and $C \subset \Omega_2$, we have, in general,

$$\begin{aligned}
&E_U\big[E_{X|U}[\chi_{A\times B}(U,V) \mid U]E_{Y|U}[\chi_C(Y) \mid U]\big] - E_{XY}[\chi_{A\times B}(U,V)\chi_C(Y)] \\
&= E_U\big[E_{V|U}[\chi_B(V) \mid U]\chi_A(U)E_{Y|U}[\chi_C(Y) \mid U]\big] - E_U\big[E_{VY|U}[\chi_B(V)\chi_C(Y) \mid U]\chi_A(U)\big] \\
&= \int_A \big\{P_{V|U}(B \mid u)P_{Y|U}(C \mid u) - P_{VY|U}(B \times C \mid u)\big\}dP_U(u). \qquad (25)
\end{aligned}$$

From Theorem 8, the left-hand side of Equation (24) is equivalent to $E_U[P_{X|U} \otimes P_{Y|U}] = P_{XY}$, which implies that the last integral in Equation (25) is zero for all A. This means $P_{V|U}(B \mid u)P_{Y|U}(C \mid u) - P_{VY|U}(B \times C \mid u) = 0$ for almost every u-P_U. Thus, Y and V are conditional independent given U. The converse is trivial. ■

Note that the left-hand side of Equation (24) is not $\Sigma_{YV|U}$ but $\Sigma_{YX|U}$, which is defined on the direct product $\mathcal{H}_{11} \otimes \mathcal{H}_{12}$.

References

Daniel Alpay. *The Schur Algorithm, Reproducing Kernel Spaces and System Theory.* American Mathematical Society, 2001.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 69(3):337–404, 1950.

Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

Francis R. Bach and Michael I. Jordan. Beyond independent components: trees and clusters. *Journal of Machine Learning Research*, 2003a. In press.

Francis R. Bach and Michael I. Jordan. Learning graphical models with Mercer kernels. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, 2003b.

Charles R. Baker. Joint measures and cross-covariance operators. *Transactions of the American Mathematical Society*, 186:273–289, 1973.

Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

Leo Breiman and Jerome H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American Statistical Association*, 80:580–598, 1985.

R. Dennis Cook. *Regression Graphics.* Wiley Inter-Science, 1998.

R. Dennis Cook and Hakbae Lee. Dimension reduction in regression with a binary response. *Journal of the American Statistical Association*, 94:1187–1200, 1999.

R. Dennis Cook and S. Weisberg. Discussion of Li (1991). *Journal of the American Statistical Association*, 86:328–332, 1991.

R. Dennis Cook and Xiangrong Yin. Dimension reduction and visualization in discriminant analysis (with discussion). *Australian & New Zealand Journal of Statistics*, 43(2):147–199, 2001.

Jerome H. Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

Wing Kam Fung, Xuming He, Li Liu, and Peide Shi. Dimension reduction based on canonical correlation. *Statistica Sinica*, 12(4):1093–1114, 2002.

Otis W. Gilley and R. Kelly Pace. On the Harrison and Rubingeld data. *Journal of Environmental Economics Management*, 31:403–405, 1996.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

David Harrison and Daniel L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics Management*, 5:81–102, 1978.

Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1: 297–318, 1986.

Inge S. Helland. On the structure of partial least squares. *Communications in Statistics - Simulations and Computation*, 17(2):581–607, 1988.

Agnar Höskuldsson. PLS regression methods. *Journal of Chemometrics*, 2:211–228, 1988.

Marian Hristache, Anatoli Juditsky, Jörg Polzehl, and Vladimir Spokoiny. Structure adaptive approach for dimension reduction. *The Annals of Statistics*, 29(6):1537–1566, 2001.

Ker-Chau Li. Sliced inverse regression for dimension reduction (with discussion). *Journal of the American Statistical Association*, 86:316–342, 1991.

Ker-Chau Li. On principal Hessian directions for data visualization and dimension reduction: Another application of Stein's lemma. *Journal of the American Statistical Association*, 87:1025–1039, 1992.

Ker-Chau Li, Heng-Hui Lue, and Chun-Houh Chen. Interactive tree-structured regression via principal Hessian directions. *Journal of the American Statistical Association*, 95(450): 547–560, 2000.

Patrick M. Murphy and David W. Aha. UCI repository of machine learning databases. Technical report, University of California, Irvine, Department of Information and Computer Science. http://www.ics.uci.edu/˜mlearn/MLRepository.html, 1994.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer Verlag, 1996.

Danh V. Nguyen and David M. Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18(1):39–50, 2002.

Michael Reed and Barry Simon. *Functional Analysis*. Academic Press, 1980.

Alexander M. Samarov. Exploring regression structure using nonparametric functional estimation. *Journal of the American Statistical Association*, 88(423):836–847, 1993.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

Kari Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.

Nikolai N. Vakhania, Vazha I. Tarieladze, and Sergei A. Chobanyan. *Probability Distributions on Banach Spaces*. D. Reidel Publishing Company, 1987.

Vladimir N. Vapnik, Steven E. Golowich, and Alexander J. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287. MIT Press, 1997.

Francesco Vivarelli and Christopher K.I. Williams. Discovering hidden features with Gaussian process regression. In Michael Kearns, Sara Solla, and David Cohn, editors, *Advances in Neural Processing Systems*, volume 11, pages 613–619. MIT Press, 1999.

Sanford Weisberg. Dimension reduction regression in R. *Journal of Statistical Software*, 7 (1), 2002.

Journal of Machine Learning Research 5 (2004) 101-141 Submitted 4/03; Revised 8/03; Published 1/04

In Defense of One-Vs-All Classification

Ryan Rifkin RIF@ALUM.MIT.EDU
Honda Research Institute USA
145 Tremont Street
Boston, MA 02111-1208, USA

Aldebaro Klautau A.KLAUTAU@IEEE.ORG
UC San Diego
La Jolla, CA 92093-0407, USA

Editor: John Shawe-Taylor

Abstract

We consider the problem of multiclass classification. Our main thesis is that a simple "one-vs-all" scheme is as accurate as any other approach, assuming that the underlying binary classifiers are well-tuned regularized classifiers such as support vector machines. This thesis is interesting in that it disagrees with a large body of recent published work on multiclass classification. We support our position by means of a critical review of the existing literature, a substantial collection of carefully controlled experimental work, and theoretical arguments.

Keywords: Multiclass Classification, Regularization

1. Introduction

We consider the problem of multiclass classification. A training set consisting of data points belonging to N different classes is given, and the goal is to construct a function which, given a new data point, will correctly predict the class to which the new point belongs.[1]

Over the last decade, there has been great interest in classifiers that use *regularization* to control the capacity of the function spaces they operate in. These classifiers—the best-known example of which is the support vector machine (SVM) (Boser et al., 1992)—have proved extremely successful at binary classification tasks (Vapnik, 1998, Evgeniou et al., 2000, Rifkin, 2002). It therefore seems interesting to consider whether the advantages of regularization approaches for binary classifiers carried over to the multiclass situation.

One of the simplest multiclass classification schemes built on top of real-valued binary classifiers is to train N different binary classifiers, each one trained to distinguish the examples in a single class from the examples in all remaining classes. When it is desired to classify a new example, the N classifiers are run, and the classifier which outputs the largest (most positive) value is chosen. This scheme will be referred to as the "one-vs-all" or OVA

1. In our framework, each data point is required to belong to a single class. We distinguish this from the case when there are more than two classes, but a given example can be a member of more than one class simultaneously. In the latter case, if the labels are independent, the problem very naturally decomposes into N *unlinked* binary problems, where the ith binary learner simply learns to distinguish whether or not an example is in class i. If the labels are dependent, then how best to perform multiclass classification is an interesting research problem, but is beyond the scope of this paper.

scheme throughout this paper. The one-vs-all scheme is conceptually simple, and has been independently discovered numerous times by different researchers.

One might argue that OVA is the first thing thought of when asked to come up with an approach for combining binary classifiers to solve multiclass problems. Although it is simple and obvious, the primary thesis of this paper is that the OVA scheme is extremely powerful, producing results that are often at least as accurate as other methods. This thesis seems quite innocuous and hardly worth writing a paper about, until one realizes that this idea is in opposition to a large body of recent literature on multiclass classification, in which a number of more complicated methods have been developed and their superiority over OVA claimed. These methods can be roughly divided between two different approaches—the "single machine" approaches, which attempt to construct a multiclass classifier by solving a *single* optimization problem (Weston and Watkins, 1998, Lee et al., 2001a,b, Crammer and Singer, 2001) and the "error correcting" approaches (Dietterich and Bakiri, 1995, Allwein et al., 2000, Crammer and Singer, 2002, Fürnkranz, 2002, Hsu and Lin, 2002), which use ideas from error correcting coding theory to choose a collection of binary classifiers to train and a method for combining the binary classifiers.

A substantial portion of this paper is devoted to a detailed review and discussion of this literature. What we find is that although a wide array of more sophisticated methods for multiclass classification exist, experimental evidence of the superiority of these methods over a simple OVA scheme is either lacking or improperly controlled or measured.

One scheme that is particularly worthy of attention is the "all-pairs", or AVA ("all-vs-all") scheme. In this approach, $\binom{N}{2}$ binary classifiers are trained; each classifier separates a pair of classes. This scheme, like the OVA scheme, has a simple conceptual justification, and can be implemented to train faster and test as quickly as the OVA scheme. Several authors have reported that the AVA scheme offers better performance than the OVA scheme (Allwein et al., 2000, Fürnkranz, 2002, Hsu and Lin, 2002). Our results disagree with the ones presented in all three of these papers, essentially because we feel their experiments were not as carefully controlled and reported as ours.

For an experiment to be carefully *controlled* means that a reasonable effort was made to find correct settings for the hyperparameters of the algorithm (in a way that does not involve mining the test set, obviously) and that the best available binary classifiers are used. This point is critical; it is easy to show (and many authors have) that if relatively weak binary learners are used, then a wide variety of clever methods for combining them will exploit the independence in the error rates of these weak classifiers to improve the overall result. In contrast, we demonstrate empirically in a substantial collection of well-controlled experiments that when well-tuned SVMs (or regularized least squares classifiers) are used as the binary learners, there is little to no independence in the errors of the binary classifiers, and therefore nothing to be gained from sophisticated methods of combination. The crucial question for the practitioner then becomes whether sophisticated methods of combining weak classifiers can achieve stronger results than a simple method of combining strong classifiers. The empirical evidence supports the notion that they cannot.

For an experiment to be carefully *reported* indicates that the results are presented in a way that is easy to understand, and that reasonable conclusions are drawn from them. This is obviously a subjective notion, but we wish to point out a specific area which we feel is problematic: the notion that a lower absolute error rate is strongly indicative of the

superiority of a classifier *when these absolute error rates are very close*. In other words, we feel it is not appropriate simply to present results where the best-performing classifier has its score given in bold on each experiment, and the classifier with the most bold scores is declared the strongest, because this ignores the very real possibility (see for example Hsu and Lin, 2002) that on nearly all the experiments, the actual experimental differences are tiny. Therefore, it is worthwhile to assess statistically the relative performance of classification systems.

One possible test for comparing two schemes is McNemar's test (McNemar, 1947, Everitt, 1977) (see Appendix A). A difficulty with McNemar's test is that it is insensitive to the number of points that the two schemes *agree* on; directly related to this, McNemar's test simply tells *whether* or not two scores are statistically significantly different (according to the assumptions inherent in the test), but gives no indication of *how* different they are. For this reason, we advocate and use a simple bootstrap method for computing confidence intervals of the difference in performance between a pair of classifiers. This method is described in Appendix A.[2]

Additionally, in order to allow for comparisons by different researchers, we feel that it is crucial to present the actual error rates of various classification schemes, rather than (or in addition to) the relative differences in error between two schemes (see, for example, Dietterich and Bakiri, 1995, Crammer and Singer, 2001, 2002); this is particularly important given that different researchers are often unable to produce identical results using identical methodologies (see Appendix B).

In general, we are *not* stating that an OVA scheme will perform substantially better than other approaches. Instead, we are stating that it will perform *just as well* as these approaches, and therefore it is often to be preferred due to its computational and conceptual simplicity.

In Section 2 we describe support vector machines and regularized least squares classifiers, which are the binary classifiers used throughout this paper. In Section 3, we describe previous work in multiclass classification using binary classifiers. In Section 4, we present a large, carefully controlled and carefully measured set of experiments, as well as analysis of these experiments. In Section 5, we present theoretical arguments that help to indicate why OVA approaches perform well. Finally, in Section 6, we discuss our results and outline open questions. Note that notation which is used in Sections 4 and 5 is introduced in Section 3.2, as this seemed the most natural approach to the presentation.

It is our hope that this paper will be of equal interest to machine learning researchers and general practitioners who are actually faced with multiclass classification problems in

2. Other statistical tests are of course possible. Dietterich (1998) compares a number of approaches to testing the statistical difference between classifiers. Dietterich derives and recommends a test known as a 5x2 CV test, suggesting that it is more powerful than McNemar's test while having essentially equivalent probability of incorrectly finding a distinction where none exists. We must confess that at the time our experiments were performed, we were not aware of this work. However, there are several advantages to our current protocol. The 5x2 CV test, like McNemar's test, only gives an estimate of whether two classifiers have different performance, whereas the bootstrap method we advocate naturally produces an estimate of the *size* of this difference. Additionally, in order to use the 5x2 CV test, it would have been necessary to use training sets whose size was exactly half the size of the total training set; this would have made comparisons to previous work (especially that of Fürnkranz, 2002) much more difficult.

engineering applications. In particular, we hope to demonstrate that for practical purposes, simple approaches such as one-vs-all classification work as well as more complicated schemes, and are therefore to be preferred.

2. Regularized Kernel Classifiers

A broad class of classification (and regression) algorithms can be derived from the general approach of Tikhonov regularization. In our case, we consider Tikhonov regularization in a reproducing kernel Hilbert space:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{\ell} V(f(\mathbf{x_i}), y_i) + \lambda ||f||_K^2.$$

Here, ℓ is the size of the training set $S \equiv \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$. $V(f(\mathbf{x}, y))$ represents the cost we incur when the algorithm sees $\mathbf{x}$, predicts $f(\mathbf{x})$, and the actual value is y. The parameter λ is the "regularization parameter" which controls the tradeoff between the two conflicting goals of minimizing the training error and ensuring that f is smooth. A detailed discussion of RKHSs is beyond the scope of this paper (for details, see Evgeniou et al., 2000, and the references therein). For our purposes, there are essentially only two key facts about RKHS. The first is the "Representer Theorem" (Wahba, 1990), which states that under very general conditions on the loss function V, the solution to a Tikhonov minimization problem can be written as a sum of kernel products on the training set:

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} c_i K(\mathbf{x_i}, \mathbf{x_j}). \tag{1}$$

The goal of a Tikhonov minimization procedure is to find the c_i. The other fact is that for functions represented in this form,

$$||f||_K^2 = \mathbf{c}^T K \mathbf{c}.$$

Given these facts, a range of different algorithms can be derived by choosing the function V.

If we choose $V(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$, the so-called "square loss", the Tikhonov minimization problem becomes regularized least squares classification (RLSC): [3]

$$(K + \lambda \ell I)\mathbf{c} = \mathbf{y}.$$

3. Regularized least-squares classification is not a new algorithm. The mathematics of finding a linear function that minimizes the square loss over a set of data was first derived by Gauss (1823). The idea of regularization is apparent in the work of Tikhonov and Arsenin (1977), who used least-squares regularization to restore well-posedness to ill-posed problems. Schönberg's seminal article on smoothing splines (Schönberg, 1964) also used regularization. These authors considered regression problems rather than classification, and did not use reproducing kernel Hilbert spaces as regularizers.

In 1971, Wahba and Kimeldorf (1971) considered square-loss regularization using the norm in a reproducing kernel Hilbert space as a stabilizer. Only regression problems were considered in this work.

In 1989, Girosi and Poggio considered regularized classification and regression problems with the square loss (Girosi and Poggio, 1989, Poggio and Girosi, 1990). They used pseudodifferential operators as their stabilizers; these are essentially equivalent to using the norm in an RKHS.

If we choose $V(f(\mathbf{x}), y) = \max(1 - y_i f(\mathbf{x_i}), 0) \equiv (1 - y_i f(\mathbf{x_i}))_+$, the so-called "hinge loss", we arrive at the standard support vector machine:[4]

$$\begin{aligned} \min_{\mathbf{c}\in\mathbb{R}^\ell, \xi\in\mathbb{R}^\ell} \quad & C\sum_{i=1}^{\ell}\xi_i + \tfrac{1}{2}\mathbf{c}^T K\mathbf{c} \\ \text{subject to :} \quad & y_i(\textstyle\sum_{j=1}^{\ell} c_j K(\mathbf{x_i}, \mathbf{x_j}) + b) \geq 1 - \xi_i \quad i = 1, \ldots, \ell, \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, \ldots, \ell. \end{aligned}$$

We note that RLSC and SVM are *both* instances of Tikhonov regularization, and will both have solutions in the form given by Equation 1. From the standpoint of theoretical bounds on the generalization of these algorithms using measures of the size of the function class such as covering numbers, the choice of the loss function is almost irrelevant and the two methods will provide very similar bounds (Vapnik, 1998, Bousquet and Elisseeff, 2002).

Intuitively, it seems that the square loss may be less well suited to classification than the hinge loss—if a point $\mathbf{x_i}$ is in the positive class ($y_i = 1$) and we observe $f(\mathbf{x_i}) = 5$, we pay nothing under the hinge loss but we pay $(5-1)^2 = 16$ under the square loss, the same penalty we would pay if $f(\mathbf{x_i})$ were -3. However, in practice, we have found that the *accuracy* of RLSC is essentially equivalent to that of SVMs (Rifkin, 2002), and substantial additional evidence of that claim will be presented in Section 4; while the performance differs substantially on a few data sets (in both directions), on most data sets the difference in the accuracy of the methods is very small.

For this reason, we believe that the choice between RLSC and SVMs should be made on the basis of computational tractability rather than accuracy. For linear kernels (or other cases where the kernel matrix K will be sparse or can be decomposed in a way that is known a priori), RLSC will be substantially faster than SVM both at training and test times. On the other hand, for a general nonlinear kernel, the first step in solving RLSC is computing the matrix K; for large problems, an SVM can be trained in less time than it takes to

These earlier works tended to belong to the statistical rather than the machine learning community. As such, the technique called RLSC in the present work was not given a name *per se* in these works.

More recently, the algorithm (or a minor variant) has been rediscovered independently by many authors who were not fully aware of the above literature. Saunders et al. (1998) rederives the algorithm as "kernel ridge regression"; he derives it by means of applying the "kernel trick" to ridge regression, rather than directly via regularization, and does not consider the use of this algorithm for classification. Mika et al. (1999) present a similar algorithm under the name kernel fisher discriminant, but in this work, the algorithm *without* regularization is presented as primary, with regularization added "to improve stability"; in our view, the regularization is central to both theory and practice. Fung and Mangasarian, under the name "proximal support vector machines" (Fung and Mangasarian, 2001b,a), and Suykens et al., under the name "least-squares support vector machines" (Suykens and Vandewalle, 1999a,b, Suykens et al., 1999), both derive essentially the same algorithm (we view the presence or absence of a bias term b in either the function or the cost function as a relatively minor detail) by modifying the cost function of SVMs. We strongly prefer to view *both* RLSC and SVM as instantiations of Tikhonov regularization, on an equal footing, rather than viewing RLSC as a "modified" SVM. Although it is regrettable to have to introduce yet another name for the same algorithm, we do not find any of the above names to be satisfactory. We believe that regularized least-squares classification is a highly appropriate name, as it draws attention to all the key features of the algorithm, and we hope (likely in vain) that future users of this algorithm will make use of this name.

4. In order to arrive at the standard SVM, we modify our notation slightly, defining $C = \frac{1}{2\lambda\ell}$, and also add an unregularized bias term b to the formulation. Details of this derivation, as well as the derivation of RLSC, can be found in Rifkin's PhD thesis (Rifkin, 2002).

compute K. This is done by solving the SVM *dual problem*:

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^\ell} \quad & \textstyle\sum_{i=1}^{\ell} \alpha_i - \frac{1}{(2\lambda)^2} \alpha^T Q \alpha \\ \text{subject to :} \quad & \textstyle\sum_{i=1}^{\ell} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq \frac{1}{\ell} \qquad i = 1, \ldots, \ell. \end{aligned}$$

Here, Q is the matrix defined by the relationship

$$Q = YKY \Longleftrightarrow Q_{ij} = y_i y_j K(\mathbf{x_i}, \mathbf{x_j}),$$

where Y is a diagonal matrix whose satisfying $Y_{i,i} = y_i$. The SVM dual has only simple box constraints and a single equality constraint. For this reason, a large SVM problem can be *decomposed* and solved as a sequence of smaller problems. (Osuna et al., 1997, Osuna, 1998) If a data point never has a nonzero coefficient over the course of this procedure (the point is not a support vector and the algorithm never conjectures that it might be), then the associated row of K (equivalently Q) need never be computed at all. Very often, this condition holds for a large majority of the data points, and the time required to train an SVM is substantially less than the time required to compute all of K; this is what makes it possible to solve large SVM problems (relatively) quickly. It is also important to note that in state-of-the-art implementations of SVMs (Rifkin, 2002, Collobert and Bengio, 2001, Joachims, 1998), the idea of *caching* kernel products which were needed previously and will probably be needed again is crucial; if the data is high-dimensional, the time required to obtain a kernel product from a cache is much less than the time required to compute it anew.

Furthermore, the SVM will exhibit *sparsity*—generally only a small percentage of the c_i will be non-zero, making it much faster at test time as well. Therefore, for large problems with nonlinear kernels, the SVM is preferred to RLSC for computational reasons. For further discussion of this point, see Rifkin's PhD thesis (Rifkin, 2002).

In our paper, we use only the Gaussian kernel

$$K(\mathbf{x_1}, \mathbf{x_2}) = \exp^{-\gamma ||\mathbf{x_1} - \mathbf{x_2}||^2},$$

making the SVM the preferred algorithm. However, we also perform a large number of experiments with RLSC, both in order to support our claim that the accuracy of RLSC and SVM are essentially the same, and to motivate the theoretical results in Section 5, which only apply to the RLSC algorithm.

It is worth noting that in many "classic" derivations of SVMs, the primal problem is derived for the case of a linear hyperplane and separable data, using the idea of "maximizing margin". Non-separability is handled by introducing slack variables, the dual is taken, and only then is it observed that the $\mathbf{x_i}$ appear only as dot products $\mathbf{x_i} \cdot \mathbf{x_j}$, which can be replaced by kernel products $K(\mathbf{x_i}, \mathbf{x_j})$ (the so-called "kernel trick"). Developing SVMs and RLSC in a unified framework from the perspective of Tikhonov regularization makes clear that we can use kernel products directly in the primal formulations, taking the dual only when it is useful for computational purposes. Many authors of the papers discussed in the next section instead take the more "classical" approach of deriving their algorithm for the linear case, taking the dual, and *then* nonlinearizing by means of kernel functions. This issue is discussed in more detail in Rifkin's PhD thesis (Rifkin, 2002).

3. Previous Work

The central thesis of this chapter is that one-vs-all classification using SVMs or RLSC is an excellent choice for multiclass classification. In the past few years, many papers have been presented that claim to represent an advance on this technique. We will review these papers in detail, directly considering the hypothesis that the new techniques outperform a simple OVA approach. These papers fall into two main categories. The first category attempts to solve a single optimization problem rather than combine the solutions to a collection of binary problems. The second category attempts to use the power of error-correcting codes to improve multiclass classification. We deal with these two approaches separately.

3.1 Single Machine Approaches

We now discuss the single-machine approaches that have been presented in the literature.

3.1.1 Vapnik and Blanz, Weston and Watkins

The single machine approach was introduced simultaneously by Vapnik (1998) and Weston and Watkins (1998). The formulations introduced in these two sources are essentially identical. The approach is a multiclass generalization of support vector machines. A standard SVM finds a function

$$f(x) = \sum_{j=1}^{\ell} c_j K(\mathbf{x}, \mathbf{x_j}) + b.$$

The multiclass SVM of Weston and Watkins finds N functions $f_1, \ldots, f_N$ simultaneously, where

$$f_i(x) = \sum_{j=1}^{\ell} c_{ij} K(\mathbf{x}, \mathbf{x_j}) + b_i.$$

The basic idea behind the multiclass SVM of Weston and Watkins (as well as all other single machine approaches, with slight modifications, as we shall see) is that instead of paying a penalty for each machine separately based on whether each machine satisfies its margin requirements for a given point, we pay a penalty based on the *relative* values output by the different machines. More concretely, given a single data point x belonging to class i, in the one-vs-all scheme we pay a penalty for machine i if $f_i(x) < 1$, and for all other classes j we pay a penalty if $f_j(x) > -1$. In the Weston and Watkins scheme, for each pair $i \neq j$, we pay a penalty if $f_i(x) < f_j(x) + 2$. If $f_i(x) < 1$, we may not pay a penalty, as long as $f_j(x)$ is sufficiently small for $i \neq j$; similarly, if $f_j(x) > 1$, we will not pay a penalty for x if $f_i(x)$ is sufficiently large. To facilitate this, we will use $\ell(N-1)$ slack variables ξ_{ij}, where $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, N\} \backslash y_i$. Using these slack variables, the optimization problem being solved can be expressed (using our notation) as

$$\begin{aligned} \min_{\mathbf{f_1},\ldots,\mathbf{f_N} \in \mathcal{H}, \xi \in \mathbb{R}^{\ell(\mathbf{N-1})}} \quad & \textstyle\sum_{i=1}^{N} ||f_i||_K^2 + C \sum_{i=1}^{\ell} \sum_{j \neq y_i} \xi_{ij} \\ \text{subject to :} \quad & f_{y_i}(\mathbf{x_i}) + b_{y_i} \geq f_j(\mathbf{x_i}) + b_j + 2 - \xi_{ij}, \\ & \xi_{ij} \geq 0. \end{aligned}$$

where the constraints all run over $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, N\} \backslash y_i$. As in Section 2, we can write for each f_i

$$||f_i||_K^2 = \mathbf{c_i}.^T K \mathbf{c_i}.,$$

where $\mathbf{c_i}.$ is the vector whose jth entry is c_{ij}. Doing so leads to a single quadratic programming problem with $N\ell$ function defining variables c_{ij}, $(N-1)\ell$ slack variables ξ_{ij}, and N bias terms b_i. The dual of this problem can be taken using the standard Lagrangian approach. Weston and Watkins define α_{ij} to be the dual variables associated with the first set of constraints (including "dummy" variables α_{i,y_i}), and β_{ij} to be the dual variables associated with the second set of constraints. Introducing the notation

$$A_i = \sum_{j=1}^{N} \alpha_{ij},$$

and skipping intermediate algebra, the dual problem derived by Weston and Watkins is

$$\begin{array}{ll} \max\limits_{\alpha \in \mathbb{R}^{\ell N}} & 2\sum_{ij} \alpha_{ij} + \sum_{i,j,k} \left[-\frac{1}{2} c_{j,y_i} A_i A_j + \alpha_{i,k} \alpha j, y_i - \frac{1}{2} \alpha_{i,k} \alpha_{j,k} \right] K(\mathbf{x_i}, \mathbf{x_j}) \\ \text{subject to :} & \sum_{i=1}^{\ell} \alpha_{ij} = \sum_{i=1}^{\ell} c_{ij} A_i, \\ & 0 \leq \alpha_{ij} \leq C, \\ & \alpha_{i,y_i} = 0. \end{array}$$

The first set of constraints holds for $j \in \{1, \ldots, N\}$, the second over $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, N\}$, and the third over $i \in \{1, \ldots, \ell\}$.

It is not clear whether this is useful or not, as it is unknown whether the resulting dual problem can be decomposed in any useful way. Weston and Watkins mention in passing that "decomposition techniques can be used, as in the usual SV case," but provide no mathematical derivation or implementation. Unlike the SVM, which has box constraints and a single equality constraint over all the variables, this system has N equality constraints, where the equality constraint for class j involves $\ell + JN$ terms, and J is the number of points in class j. The relative complexity of the constraints makes it likely that the decomposition algorithm would have to be substantially more complicated to maintain feasibility of the generated solutions at each iteration. Also, unlike the SVM scenario, the "dual" problem does not succeed in fully eliminating the primal variables c_{ij}. Weston and Watkins consider nonlinear kernels only in the dual formulation.

Weston and Watkins perform two different sorts of experiments. In the first set of experiments, they work with toy examples where several classes in the plane are classified using their algorithm. They show examples which are both separable and nonseparable, but they do not compare their algorithm to any other method, so these experiments only serve as a proof of concept that the algorithm works in some reasonable way.

In the second set of experiments, they compare their algorithm to a one-vs-all scheme on five data sets from the UCI repository (Merz and Murphy, 1998); the data sets used were `iris, wine, glass, soy,`[5] and `vowel`. The authors conclude that their method seems to be approximately equivalent in accuracy to a one-vs-all or an all-pairs scheme, and suggest

5. It is unclear to us what `soy` refers to. The UCI Repository contains a directory titled `soybean`, which contains two data sets, `soybean-large` and `soybean-small`. The `soy` data set considered by Weston

that the single-machine approach may have an advantage as regards the number of support vectors needed. However, the authors also state that "to enable comparison, for each algorithm $C = \infty$ was chosen (the training data must be classified without error)." Setting C to ∞ implies that the regularization is very weak; although there is some regularization, we are only able to select the smoothest function from among those functions having zero loss. This is rarely desirable in realistic applications, so it is hard to draw any conclusions about accuracy from these experiments. Furthermore, setting C to ∞ will tend to induce a much less smooth function and greatly increase the number of support vectors, making it difficult to use these experiments to draw conclusions about the number of support vectors required for different methods. There is an additional problem with the claim that the single-machine approach requires fewer support vectors, which is that in the OVA (or AVA) case, it is computationally easy to "reuse" support vectors that appear in multiple machines, leading to a large reduction in the total computational costs.

3.1.2 Lee, Lin and Wahba

Lee, Lin and Wahba present a substantially different single-machine approach to multiclass classification (Lee et al., 2001a,b). The work has its roots in an earlier paper by Lin (1999) on the asymptotic properties of support vector machine regularization for binary classification. If we define $p(\mathbf{x})$ to be the probability that a data point located at $\mathbf{x}$ is in class 1, Lin proved using elegant elementary arguments that the minimizer of $E[(1 - yf(\mathbf{x}))_+)]$ is $f(\mathbf{x}) = \text{sign}(p(\mathbf{x}) - \frac{1}{2})$. In other words, if we consider solving an SVM problem and let the number of data points tend to infinity, the minimizer of the loss functional (ignoring the regularization term $\lambda||f||_K^2$, and the fact that the functional f has to live in the RKHS $\mathcal{H}_\mathcal{K}$) tends to $\text{sign}(p(\mathbf{x}) - \frac{1}{2})$. Lin refers to this function as the Bayes-optimal solution.

Considering this to be a useful property, Lee et al. design a multiclass classification technique with similar behavior. They begin by noting that a standard one-vs-all SVM approach does not have this property. In particular, defining $p_i(\mathbf{x})$ to be the probability that a point located at $\mathbf{x}$ belongs to class i, the Lin's results show that $f_i(\mathbf{x}) \to \text{sign}(p_i(\mathbf{x}) - \frac{1}{2})$ as $\ell \to \infty$. For all points for which $\arg\max_i p_i(\mathbf{x}) \geq \frac{1}{2}$, we will recover the correct result: asymptotically, $f_i(\mathbf{x}) = 1$, and $f_j(\mathbf{x}) = -1$ for $j \neq i$. However, if $\arg\max_i p_i(\mathbf{x}) < \frac{1}{2}$, then asymptotically, $f_i(\mathbf{x}) = -1\ \forall i$, and we will be unable to recover the correct class. Lee et al. note that for other formulations such as the one of Weston and Watkins (1998), the asymptotic behavior is hard to analyze.

Lee, Lin and Wahba proceed to derive a multiclass formulation with the desired correct asymptotic behavior. For $1 \leq i \leq N$, they define v_i to be an N dimensional vector with a 1 in the ith coordinate and $\frac{1}{N-1}$ elsewhere.[6] The v_i vector plays the role of a "target" for points in class i—we try to get function outputs that are very close to the entries of v_i. However, for technical reasons, instead of worrying about all N functions, they only worry about $f_j(\mathbf{x_i})$ for $j \neq y_{\mathbf{x}_i}$, and ensure (approximate) correctness of $f_i(\mathbf{x_i})$ by requiring that

and Watkins does not match (in size or number of classes) either of these. There is also a note in this directory indicating the existence of other versions of the data set; from this note it seems that this may have been a version used separately by Mooney, Stepp and Reinke. Since this version does not seem to be publicly available, it is difficult to compare against it directly.

6. We have taken some liberties with the notation in order to shorten the presentation and keep notation consistent with the rest of the paper.

for all $\mathbf{x}$, $\sum_{i=1}^{N} f_i(\mathbf{x}) = 0$. This leads to the following optimization problem:

$$\begin{array}{ll} \min\limits_{f_1,\ldots,f_N \in \mathcal{H}_K} & \frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{j=1,j\neq y_i}^{N}(f_j(\mathbf{x}_i) + \frac{1}{N-1})_+ + \lambda\sum_{j=1}^{C}||f_j||_K^2 \\ \text{subject to :} & \sum_{j=1}^{C} f_j(\mathbf{x}) = 0, \qquad \forall \mathbf{x}. \end{array}$$

Using arguments along the same lines of those in Lin (1999), it is shown that the asymptotic solution to this regularization problem (again ignoring the λ term and the fact that the functions must live in the RKHS) is $f_i(\mathbf{x}) = 1$ if $i = \arg\max_{j=1,\ldots,N} p_j(\mathbf{x})$ and $f_i(\mathbf{x}) = -\frac{1}{N-1}$ otherwise; $f_i(\mathbf{x})$ is one if and only if i is the most likely class for a point located at $\mathbf{x}$. They point out that this is a natural generalization of binary SVMs, if we view binary SVMs as producing two functions, one for each class, constrained so that $f_1(\mathbf{x}) = f_{-1}(\mathbf{x})$ for all $\mathbf{x}$. A Lagrangian dual is derived, and it is noted that the approach retains some of the sparsity properties of binary SVMs. The resulting optimization problem is approximately $N-1$ times as large as a single SVM problem, and no decomposition method is provided.

Although this approach is interesting, there are a number of problems with it. The primary difficulty is that the analysis is entirely asymptotic, holding only as the number of data points goes to infinity and the regularization term is ignored. In this framework, any method which asymptotically estimates densities accurately will also perform optimally. However, such density estimation methods have been shown to be grossly inferior to discriminative methods such as SVMs in real-world classification tasks using limited amounts of data. Therefore, it is difficult to argue the superiority of a method based only on its asymptotic behavior. In the Lee, Lin and Wahba analysis, no information is provided about the rate of convergence to the Bayes-optimal solution. In order to arrive at this Bayes-optimal solution, we must also let $\lambda \to 0$ and $\ell \to \infty$; although this is of course the right thing to do, the result is not at all surprising viewed in this context, and no information about rates is provided. Additionally, comparing this method to one-vs-all SVMs, the only points $\mathbf{x}$ for which this approach (asymptotically) makes a difference are points for which $\arg\max_i p_i(\mathbf{x}) < \frac{1}{2}$. In other words, if a single class is more than 50% likely at a given $\mathbf{x}$, this approach and the computationally much simpler one-vs-all approach will make the same prediction (asymptotically). We expect this to be the case for the vast majority of the probability mass of many real-world problems, although this is an intuition rather than a known fact.

If the class densities are highly overlapping in some region (one class is only slightly more likely than all the others), there are two additional problems. The first is that classification accuracy is inherently limited to the likelihood of the most likely class, indicating that our problem is too difficult to be usefully solved or that we have not represented our problem in a manner that allows good classification. There may be exceptions to this, such as problems involving financial data (which are notoriously hard to achieve good performance on), but in general, we are most interested in problems which can be solved fairly accurately. The second, more important difficulty is that in high dimensions, if two class densities are similar over a region, we expect that we will need a large number of points to capture this distinction.

Another intimately related problem with this approach is that it ignores the fundamental characteristics of the regularization approach, which is the attempt to find *smooth* functions

that fit the data accurately. Although the optimization problem suggested does include a regularization term, the analysis of the technique is completely dependent on ignoring the regularization term. The Bayes-optimal asymptotic solution can be arbitrarily non-smooth, and convergence to it relies on the use of an RKHS that is dense in L_2 (such as the one obtained when the kernel K is Gaussian).

Two toy examples illustrating the method are presented. In one example, the method is illustrated graphically, and no comparisons to other methods are made. In the other example, a comparison to one-vs-all is made. The training data consists of 200 one-dimensional points (in the interval $[0, 1]$) from three overlapping classes, and the test data consists of $10,000$ independent test points from the distribution. The distributions are chosen so that class 2 never has a conditional probability of more than 50%. In the example, the method of Lee, Lin and Wahba is able to predict class 2 over the region where it is more likely than any other class, and a one-vs-all system is not. On the test set, the one-vs-all system has an error rate of .4243 and the Lee, Lin and Wahba method has an error of .389. However, it is difficult to understand how the parameter settings were chosen, possibly indicating that different parameter settings would help the one-vs-all system. Additionally, the example is only one dimensional, and involved a relatively large number of points for one dimension. Furthermore, the difference in test error rates was not especially large. Nevertheless, this experiment is somewhat interesting, and it would be good to see a number of better-controlled experiments on more realistic data sets.

Some additional insights can be gained from taking another look at the original Lin paper. The Lee, Lin and Wahba paper was based on Lin's results for the SVM hinge loss function: $V(f(\mathbf{x}), y) = (1 - yf(\mathbf{x}))_+$. The Lin paper also includes easily proved results stating that for any $q > 1$, if the loss function is either $(1 - yf(\mathbf{x})_+)^q$ or $|y - f(\mathbf{x})|^q$, then the asymptotic minimizer is given by (recalling that $p(\mathbf{x})$ is the conditional probability of a point at $\mathbf{x}$ being in class 1):

$$f(\mathbf{x}) = \frac{(p(\mathbf{x}))^{\frac{1}{q-1}} - (1 - p(\mathbf{x}))^{\frac{1}{q-1}}}{(p(\mathbf{x}))^{\frac{1}{q-1}} + (1 - p(\mathbf{x}))^{\frac{1}{q-1}}}.$$

In the specific case of regularized least squares classification discussed in Section 2, $V(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2$, so the asymptotic discrimination function is

$$f(\mathbf{x}) = \frac{(p(\mathbf{x}))^{\frac{1}{2}} - (1 - p(\mathbf{x}))^{\frac{1}{2}}}{(p(\mathbf{x}))^{\frac{1}{2}} + (1 - p(\mathbf{x}))^{\frac{1}{2}}}.$$

Now, instead of SVM, let's consider the use of RLSC in a one-vs-all framework. We will (asymptotically) arrive at N functions, where

$$f_i(\mathbf{x}) = \frac{(p_i(\mathbf{x}))^{\frac{1}{2}} - (1 - p_i(\mathbf{x}))^{\frac{1}{2}}}{(p_i(\mathbf{x}))^{\frac{1}{2}} + (1 - p_i(\mathbf{x}))^{\frac{1}{2}}}.$$

Now assume $p_i(\mathbf{x}) > p_j(\mathbf{x})$. We will show that this implies that $f_i(\mathbf{x}) > f_j(\mathbf{x})$. Specifically, we consider the notationally simplified quantity

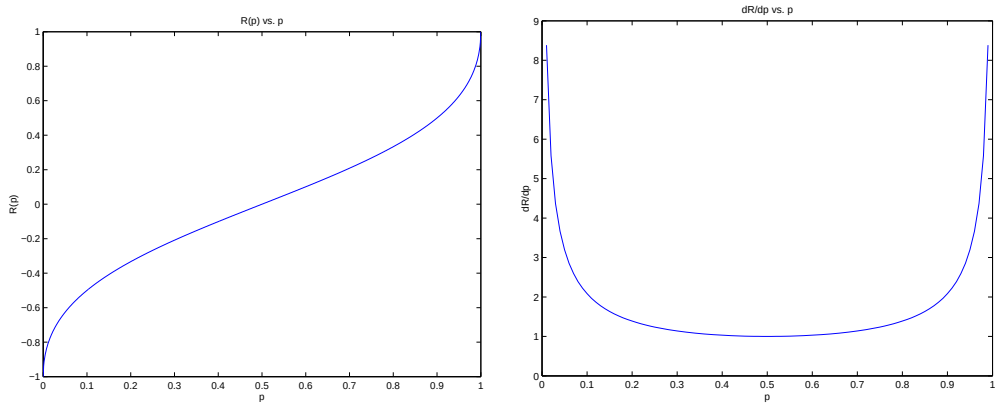
$$R(p) = \frac{p^{\frac{1}{2}} - (1 - p)^{\frac{1}{2}}}{p^{\frac{1}{2}} + (1 - p)^{\frac{1}{2}}},$$

and show that $R(p)$ is increasing as a function of p, for $p \in [0,1]$. We first note that $R(0) = -1$ and $R(1) = 1$. Next, for $p \in (0,1)$, we find that

$$\begin{aligned}
\frac{dR}{dp} &= \frac{\frac{d(p^{\frac{1}{2}}-(1-p)^{\frac{1}{2}})}{dp}(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}}) - (p^{\frac{1}{2}}-(1-p)^{\frac{1}{2}})\frac{d(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})}{dp}}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&= \frac{\frac{1}{2}\left[(p^{-\frac{1}{2}}+(1-p)^{-\frac{1}{2}})(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})\right]}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&+ \frac{\frac{1}{2}\left[(p^{\frac{1}{2}}-(1-p)^{\frac{1}{2}})(p^{-\frac{1}{2}}-(1-p)^{-\frac{1}{2}})\right]}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&= \frac{\frac{1}{2}\left[1+(\frac{1-p}{p})^{\frac{1}{2}}+(\frac{p}{1-p})^{\frac{1}{2}}+1-1+(\frac{1-p}{p})^{\frac{1}{2}}+(\frac{p}{1-p})^{\frac{1}{2}}-1\right]}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&= \frac{(\frac{1-p}{p})^{\frac{1}{2}}+(\frac{p}{1-p})^{\frac{1}{2}}}{(p^{\frac{1}{2}}+(1-p)^{\frac{1}{2}})^2} \\
&> 0.
\end{aligned}$$

In other words, $R(p)$ is a strictly increasing function of p (Figure 1 shows both $R(p)$ and $\frac{dR}{dP}$ as a function of p), which implies that if class i is more likely than class j at point $\mathbf{x}$, $f_i(\mathbf{x}) > f_j(\mathbf{x})$. This in turn implies that if we use a one-vs-all RLSC scheme, and classify test points using the function with the largest output value (which is of course the common procedure in one-vs-all classification), the error of our scheme will asymptotically converge to the Bayes error, just as the multiclass SVM of Lee, Lin and Wahba does. Put differently, *the need for a single-machine approach with a sum-to-zero constraint on the functions in order to asymptotically converge to the Bayes function was a specific technical requirement associated with the use of the SVM hinge loss.* When we change the loss function to the square loss, another commonly used loss function that gives equivalent accuracy, the one-vs-all approach has precisely the same asymptotic convergence properties.

We are not claiming that this analysis is a strong argument in favor of the one-vs-all RLSC scheme as opposed to the one-vs-all SVM. The argument is an asymptotic one, applying only in the limit of infinitely many data points. There are a large number of schemes that will work equivalently with infinite amounts of data, and it is something of a technical oddity that the one-vs-all SVM appears not to be one of them. However, we do not believe that this asymptotic analysis tells us anything especially useful about the performance of a multiclass scheme on finite, limited amounts of high-dimensional data. In this regime, both a one-vs-all SVM scheme and a one-vs-all RLSC scheme have been demonstrated to behave quite well empirically. However, the fact that the one-vs-all RLSC scheme has equivalent asymptotic behavior to the Lee, Lin and Wahba scheme casts further doubt on the idea that their scheme will prove superior to one-vs-all on real applications.



(a) (b)

Figure 1: An analysis of the quantity $R(p)$. (a): $R(p)$ vs. p. (b): $\frac{dR}{dp}$ vs. p. We see that $R(p)$ is a strictly increasing function of p, implying that if class i is more likely than class j at point $\mathbf{x}$, then, asymptotically, $f_i(\mathbf{x}) > f_j(\mathbf{x})$.

3.1.3 Bredensteiner and Bennett

Bredensteiner and Bennett (1999) also suggest a single-machine approach to multiclass classification.[7] Like Weston and Watkins, they begin by stating the invariant that they want the functions generated by their multiclass system to satisfy:

$$\mathbf{w_{y_i}}^T \cdot \mathbf{x_i} + b_i \geq \mathbf{w_j}^T \cdot \mathbf{x_i} + b_j + 1 - \xi_{ij},$$

where $\mathbf{x_i}$ is a member of class y_i and $j \neq y_i$. They rewrite this equation as

$$(\mathbf{w_{y_i}} - \mathbf{w_j})^T \cdot \mathbf{x_i} \geq (b_j - b_i) + 1 - \xi_{ij}.$$

They then argue that a good measure of the separation between class i and j is $\frac{2}{||w_i - w_j||}$, and suggest maximizing this quantity by minimizing $||w_i - w_j||$ over all pairs i and j. They also add the regularization term $\frac{1}{2}\sum_{i=1}^{N} ||w_i||^2$ to the objective function. The resulting optimization problem (where we have adjusted the notation substantially to fit with our development) is

$$\begin{aligned} \min_{(\mathbf{w_i}, b_i \in \mathbb{R}^{d+1})} \quad & \tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} ||w_i - w_j||^2 + \tfrac{1}{2}\sum_{i=1}^{N} ||w_i||^2 + C\sum_{i=1}^{\ell}\sum_{j \neq y_i} \xi_{ij} \\ \text{subject to :} \quad & \mathbf{w_{y_i}} - \mathbf{w_j}^T \cdot \mathbf{x_i} \geq (b_j - b_i) + 1 - \xi_{ij}, \\ & \xi_{ij} \geq 0. \end{aligned}$$

Using standard but rather involved techniques, they derive the Lagrangian dual problem, and observe that the dot products can be replaced with kernel products.

7. The Bredensteiner and Bennett formulation has been shown to be equivalent to the Weston and Watkins formulation (Guermeur, 2002, Hsu and Lin, 2002).

Two sets of experiments were performed. The first set involved two data sets from the UCI repository (Merz and Murphy, 1998), (`wine` and `glass`). Ten-fold cross validation was performed on each data set, and polynomials of degree one through five are used as models. On both data sets, the highest performance reported is for a one-vs-all SVM system rather than the multiclass system they derived (their multiclass SVM does perform better than an unregularized system which merely finds an arbitrary separating hyperplane).

In the second set of experiments, two separate subsets of the `USPS` data were constructed. Subsets of the training data were used, because training their multiclass method on the full data set was not computationally feasible.[8] On both data sets, a one-vs-all SVM system performs (slightly) better than their single-machine system.

3.1.4 Crammer and Singer

Crammer and Singer consider a similar but not identical single-machine approach to multiclass classification (Crammer and Singer, 2001). This work is a specific case of a general method for solving multiclass problems, presented in several papers (Crammer and Singer, 2000b,a, 2002) and discussed in Section 3.2 below. The method can be viewed as a simple modification of the approach of Weston and Watkins (1998). Weston and Watkins start from the idea that if a point $\mathbf{x}$ is in class i, we should try to make $f_i(\mathbf{x}) \geq f_j(\mathbf{x}) + 2$ for $i \neq j$, and arrive at the following formulation:

$$\begin{aligned} \min_{\mathbf{f_1},\ldots,\mathbf{f_N} \in \mathcal{H}, \xi \in \mathbb{R}^{\ell(\mathbf{N-1})}} \quad & \sum_{i=1}^{N} ||f_i||_K^2 + C \sum_{i=1}^{\ell} \sum_{j \neq y_i} \xi_{ij} \\ \text{subject to :} \quad & f_{y_i}(\mathbf{x_i}) + b_{y_i} \geq f_j(\mathbf{x_i}) + b_j + 2 - \xi_{ij}, \\ & \xi_{ij} \geq 0. \end{aligned}$$

Crammer and Singer begin with the same condition, but instead of paying for *each* class $j \neq i$ for which $f_i(\mathbf{x}) < f_j(\mathbf{x}) + 1$,[9] they pay only for the *largest* $f_j(\mathbf{x})$. This results in a *single* slack variable for each data point, rather than the $N-1$ slack variables per point in the Weston and Watkins formulation. The resulting mathematical programming problem is (as usual, placing the formulation into our own notations for consistency):

$$\begin{aligned} \min_{\mathbf{f_1},\ldots,\mathbf{f_N} \in \mathcal{H}, \xi \in \mathbb{R}^{\ell}} \quad & \sum_{i=1}^{N} ||f_i||_K^2 + C \sum_{i=1}^{\ell} \xi_i \\ \text{subject to :} \quad & f_{y_i}(\mathbf{x_i}) \geq f_j(\mathbf{x_i}) + 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

The majority of the paper is devoted to the development of an efficient algorithm for solving the above formulation. The Lagrangian dual is taken, and the standard observations that

8. For example, their method takes over 10,000 seconds to train on a data set of 1,756 examples in three classes. Modern, freely available SVM solvers such as SVMTorch (Collobert and Bengio, 2001) routinely solve problems on 5,000 or more points in 30 seconds or less.

9. The choice of 1 rather than 2 as a "required difference" is arbitrary. Crammer and Singer quite reasonably choose 1 for simplicity. The choice of 2 in Weston and Watkins seems to be motivated from a desire to make the system as similar to standard binary SVMs as possible, where we require a margin of 1 for points in the positive class and -1 for points in the negative class. This choice is arbitrary: if we required 1 for points in the positive class and 0 for points in the negative class, the details of the algorithm would change, but the function found by the algorithm would not.

the dot products can be replaced with kernels are made. An elegant dual decomposition algorithm is developed, in which a single data point is chosen at each step and an iterative algorithm is used to solve the reduced N-variable quadratic programming problem associated with the chosen data point. A number of additional implementation tricks are used, including using the KKT conditions for example selection, caching kernel values, maintaining an active set from which the example to be optimized in a given iteration is chosen, and cooling of the accuracy parameter.

In the experimental section of the paper, Crammer and Singer considered a number of data sets from the UCI repository. They produce a chart showing the difference in error rate between their one-machine system and an OVA system, but not the actual error rates. There are two data sets for which the difference between their system and OVA seems to be large: `satimage` with a difference of approximately 6.5% in performance, and `shuttle` with a difference of approximately 3% in performance. In personal communication, Crammer indicated that the actual error rates for his system on these two data sets were 8.1% and 0.1%, respectively. In our own one-vs-all experiments on the `satimage` data (see Section 4, we observed an error rate of 8.2%. Although we did not do experiments on the `shuttle` data set for this paper, we note that Fürnkranz (2002) achieved an error of 0.3% on this data set using a simple OVA system with Ripper as the binary learner. These numbers for an OVA system are in sharp contrast to the results implied by the paper; we have no explanation for the differences, and Crammer and Singer do not provide enough information to precisely reproduce their experiments.

3.1.5 Summary

When we apply the one-vs-all strategy, we solve a separate optimization problems for each of the N classes. The single machine approaches solve a single optimization problem to find N functions simultaneously. Of the papers considered here, only Crammer and Singer claimed that their single-machine approach outperformed OVA across realistic (non-toy) data sets, and as we show below in Section 4, performance equivalent to the best results they achieved can also be achieved by an OVA scheme when the underlying binary classifiers are properly tuned. Therefore, although these approaches may have theoretical interest, it does not appear that they offer any advantages over a simple OVA (or AVA) scheme in the solution of multiclass classification problems. Additionally, the methods are generally complicated to implement and slow to train, indicating that they would have to have some other compelling advantage, such as higher accuracy or a much sparser representation, to make them worth using in applications.

3.2 Error-Correcting Coding Approaches

We now turn to error-correcting code approaches, a second major approach to combining binary classifiers into a multiclass classification system.

3.2.1 Dietterich and Bakiri

Dietterich and Bakiri (1995) first popularized the idea of using error-correcting codes for multiclass classification. We will describe the method using notation introduced later by Allwein et al. (2000).

Dietterich and Bakiri suggested the use of a $\{-1,1\}$-valued matrix M of size N by F, that is $M \in \{-1,1\}^{N\times F}$, where N is the number of classes and F is the number of binary classifiers to be trained. We let M_{ij} refer to the entry in the ith row and the jth column of M. The ith column of the matrix induces a partition of the classes into two "metaclasses", where a point $\mathbf{x_i}$ is placed in the positive metaclass for the jth classifier if and only if $M_{y_i j} = 1$. In our framework, in which the binary classifiers implement Tikhonov regularization, the jth machine solves the following problem:

$$\min \sum_{i=1}^{\ell} V(f_j(\mathbf{x_i}), M_{y_i j}) + \lambda ||f_j||_K^2.$$

When faced with a new test point $\mathbf{x}$, we compute $f_1(\mathbf{x}), \ldots, f_F(\mathbf{x})$, take the signs of these values, and then compare the Hamming distance between the resulting vector and each row of the matrix, choosing the minimizer

$$f(\mathbf{x}) = \arg \min_{r\in 1,\ldots,N} \sum_{i=1}^{F} \left(\frac{1 - \text{sign}(M_{ri} f_i(\mathbf{x}))}{2} \right).$$

This representation had been previously used by Sejnowski and Rosenberg (1987), but in their case, the matrix M was chosen so that a column of M corresponded to the presence or absence of some specific feature across the given classes. For example (taken from Dietterich and Bakiri, 1995), in a digit recognizer, one might build a binary classifier that learned whether or not the digit contained a vertical line segment, placing the examples in classes 1, 4, and 5 in the positive metaclass for this classifier, and the remaining classes in the negative metaclass.

Dietterich and Bakiri take their cue from the theory of error-correcting codes (Bose and Ray-Chaudhuri, 1960), and suggest that the M matrix be constructed to have good error-correcting properties. The basic observation is that if the minimum Hamming distance between rows of M is d, then the resulting multiclass classification will be able to correct any $\lfloor \frac{d-1}{2} \rfloor$ errors. They also note that good *column* separation is important when using error-correcting codes for multiclass classification; if two columns of the matrix are identical (or are opposites of each other, assuming an algorithm that treats positive and negative examples equivalently), they will make identical errors.

After these initial observations, the majority of the paper is devoted to experimental results. A number of data sets from various sources are used, including several data sets from the UCI Machine Learning Repository (Merz and Murphy, 1998), and a subset of the NETtalk data set used by Sejnowski and Rosenberg (1987). Two learning algorithms were tested: decision trees using a modified version of the C4.5 algorithm (Quinlan, 1993), and feed-forward neural networks. The parameters were often tuned extensively to improve performance. In some cases, the algorithms were modified for individual data sets. They considered four different methods for constructing good error-correcting codes: an exhaustive method, a method that selects a subset of the columns generated by the exhaustive method, a method based on randomized hill climbing, and a method based on using BCH codes or a subset of BCH codes (sometimes selected using manual intervention). In summary, although the description of the experimental work is quite lengthy, it would be

essentially impossible to replicate the work exactly due to its complexity and the level of detail at which it was reported.

A large variety of experimental results are reported. It appears that in general, with the data sets and algorithms tried, the error-correcting code approach performs better than a one-vs-all approach. However, the difference is often small, and it is difficult to know how good the underlying binary classifiers are. In many instances, only relative performance results are given—the difference between the error-correcting and a one-vs-all method is given, but the actual performance numbers are not given, making comparison to alternate approaches (such as a one-vs-all SVM scheme) impossible.

3.2.2 Allwein, Schapire and Singer

In 2000, Allwein, Schapire, and Singer (2000) extended the earlier work of Dietterich and Bakiri in several directions. They were specifically interested in *margin-based* classifiers, where the underlying classifier is attempting to minimize an expression of the form

$$\frac{1}{\ell}\sum_{i=1}^{\ell} L(y_i f(\mathbf{x_i})),$$

where L is an arbitrary (chosen) function (while also possibly trying to minimize a regularization term). The quantity $y_i f(\mathbf{x_i})$ is referred to as the *margin*. In the case of the SVM, $L(yf(\mathbf{x})) = (1-yf(\mathbf{x_i}))_+$, and in the case of RLSC, $L(yf(\mathbf{x})) = (1-yf(\mathbf{x}))^2 = (y-f(\mathbf{x}))^2$; we see that both SVM and RLSC are margin-based classifiers, and that we can easily relate the margin loss function $L(yf(\mathbf{x}))$ to the loss function $V(f(\mathbf{x}), y)$ we considered in Section 2. Allwein et al. are also very interested in the AdaBoost algorithm (Freund and Schapire, 1997, Schapire and Singer, 1999), which builds a function $f(\mathbf{x})$ that is a weighted linear combination of base hypotheses h_t:

$$f(\mathbf{x}) = \sum_t \alpha_t h_t(\mathbf{x}),$$

where the h_t are selected by a (weak) base learning algorithm, and reference numerous papers indicating that AdaBoost is approximately greedily minimizing

$$\sum_{i=1}^{\ell} e^{-y_i f(\mathbf{x_i})},$$

demonstrating that AdaBoost is a margin-based classifier with $L(yf\mathbf{x}) = e^{-y_i f(\mathbf{x_i})}$.

Allwein et al. chose the matrix $M \in \{-1, 0, 1\}^{N\times F}$, rather than only allowing 1 and -1 as entries in the matrix as Dietterich and Bakiri did. If $M_{y_i j} = 0$, then example i is simply not used when the jth classifier is trained. With this extension, they were able to place one-vs-all classification, error-correcting code classification schemes, and all-pairs classification schemes (Hastie and Tibshirani, 1998) in a single theoretical framework.

If the classifiers are combined using Hamming decoding (taking the signs of the real values output by the classifiers, then finding the closest match among the rows of M), we again have

$$f(\mathbf{x}) = \arg\min_{r\in 1,\ldots,N} \sum_{i=1}^{F} \left(\frac{1-\text{sign}(M_{ri} f_i(\mathbf{x}))}{2}\right),$$

where it is now understood that if $M_{ri} = 0$ (class r was not used in the ith classifier), class r will contribute $\frac{1}{2}$ to the sum. Allwein et al. note that the major disadvantage of Hamming decoding is that it completely ignores the magnitude of the predictions, which can often be interpreted as a measure of "confidence" of a prediction. If the underlying classifiers are margin-based classifiers, they suggest using the loss function L instead of the Hamming distance. More specifically, they suggested that the prediction for a point $\mathbf{x}$ should be the class r that minimizes the total loss of the binary predictions under the assumptions that the label for point $\mathbf{x}$ for the ith machine is M_{ri}:

$$f(\mathbf{x}) = \arg \min_{r \in 1,\ldots,N} \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})).$$

This procedure is known as *loss-based decoding*. If the matrix M represents a one-vs-all coding scheme ($M_{ri} = 1$ if $r = i$, $M_{ri} = -1$ otherwise), the above equation simplifies to

$$\begin{aligned} f(\mathbf{x}) &= \arg \min_{r \in 1,\ldots,N} \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})) \\ &= \arg \min_{r \in 1,\ldots,N} L(f_r(\mathbf{x})) - \sum_{i \neq r}^{F} L(-f_i(\mathbf{x})). \end{aligned}$$

It is easy to check that for both SVM and RLSC, the prediction in the one-vs-all scheme will be chosen so that

$$f(\mathbf{x}) = \arg \max_{r} f_r(\mathbf{x_i}).$$

Allwein et al. provide an elegant analysis of the training error of multiclass error-correcting code based systems using both Hamming decoding and loss-based decoding. They also provide an analysis of the generalization performance of multiclass loss-based schemes in the particular case when the underlying binary classifier is AdaBoost. The arguments are extensions of those given by Schapire et al. (1998), and are beyond the scope of this paper.

The remainder of the paper is devoted to experiments on both toy and UCI Repository data sets, using AdaBoost and SVMs as the base learners. The two stated primary goals of the experiments are to compare Hamming and loss-based decoding and to compare the performance of different output codes.

The toy experiment considers $100k$ one-dimensional points generated from a single normal distribution, selecting the class boundaries so that each class contains 100 training points. AdaBoost is used as the weak learner, and comparisons are made between Hamming and loss-based decoding, and between a one-vs-all code and a complete code. The authors find that the loss-based decoding substantially outperforms the Hamming decoding, and that the one-vs-all and complete codes perform essentially identically.

Allwein et al. next consider experiments on a number of data sets from the machine learning repository. For SVMs, eight data sets are used: `dermatology, satimage, glass, ecoli, pendigits, yeast, vowel,` and `soybean`. Five different codes are considered: the one-vs-all code (OVA), the all-pairs code (omitted when there were too many classes) (AVA), the complete code (omitted when there were too many classes) (COM), and two types of

random codes. The first type had $\lceil 10 log_2(N) \rceil$ columns, and each entry was chosen to be 1 or -1 with equal probabilities. The codes were picked by considering $10,000$ random matrices, and picking the one with the highest value of ρ which did not have any identical columns. These codes were refereed to as *dense* (DEN) codes. They also considered *sparse* (SPA) codes, which had $\lceil 15 log_2(N) \rceil$ columns, and each entry was 0 with probability $\frac{1}{2}$, and 1 or -1 with probability $\frac{1}{4}$ each. Again, 10,000 random matrices were considered, and the one with the best ρ with no identical columns and no columns or rows containing only zeros was chosen.

Direct numerical results of the experiments are presented, as well as bar graphs showing the relative performance of the various codes. The authors conclude that "For SVM, it is clear that the widely used one-against-all code is inferior to all the other codes we tested." However, this conclusion is somewhat premature. All the SVM experiments were performed using a polynomial kernel of degree 4, and no justification for this choice of kernel was given. Additionally, the regularization parameter used (λ) was not specified by the authors. Looking at the bar graphs comparing relative performance, we see that there are two data sets on which the one-vs-all SVMs seem to be doing particularly badly compared to the other codes: `satimage` and `yeast`. We performed our own SVM experiments on this data, using a Gaussian kernel with γ and C tuned separately for each scheme (for details and actual parameter values, see Section 4).[10]

The results are summarized in Table 1 and Table 2. We find that while other codes do sometimes perform (slightly) better than one-vs-all, that none of the differences are large. This is in stark contrast to the gross differences reported by Allwein et al. Although we did not test the other data sets from the UCI repository (on which Allwein and Schapire found that all the schemes performed very similarly), this experiment strongly supported the hypothesis that the differences observed by Allwein et al. result from a poor choice of kernel parameters, which makes the SVM a much weaker classifier than it would be with a good choice of kernel. In this regime, it is plausible that the errors from the different classifiers will be somewhat decorrelated, and that a scheme with better error-correcting properties than the one-vs-all scheme will be superior. However, given that our goal is to solve the problem as accurately as possible, it appears that choosing the kernel parameters to maximize the strength of the individual binary classifiers, and then using a one-vs-all multiclass scheme, performs as well as the other coding schemes, and, in the case of the

10. We replicated the dense and sparse random codes as accurately as possible, but the information in Allwein et al. is incomplete. For both codes, we added the additional constraint that each column had to contain at least one $+1$ and at least one -1; one assumes that Allwein et al. had this constraint but did not report it, as without it, the individual binary classifiers could not be trained. For the sparse random code, the probability that a random column of length six (the number of classes in the `satimage` data set) generated according to the probabilities given fails to contain both a $+1$ and a -1 is more than 35%, and the procedure as defined in the paper fails to generate a single usable matrix. Personal communication with Allwein et al. indicate that it is likely that columns not satisfying this constraint were thrown out immediately upon generation. Additionally, there are only 601 possible length six columns containing at least one $+1$ and one -1 entry, and if these columns were chosen at random, only 28% of the matrices generated (the matrices have $\lceil 15 log_2(6) \rceil = 39$ columns) would not contain duplicate columns. Because there was no mention in either case of avoiding columns which were opposites of each other (which is equivalent to duplication if the learning algorithms are symmetric), we elected to allow duplicate columns in our sparse codes, in the belief that this would have little effect on the quality of the outcome.

	OVA	AVA	COM	DEN	SPA
Allwein et al.	40.9	27.8	13.9	14.3	13.3
Rifkin & Klautau	8.2	7.8	7.8	7.7	8.9

Table 1: Multiclass classification error rates for the `satimage` data set. Allwein et al. used a polynomial kernel of degree four and an unknown value of C. Rifkin and Klautau used a Gaussian kernel with σ and C tuned separately for each scheme; see Section 4 for details. We see that with the Gaussian kernel, overall performance is much stronger, and the differences between coding schemes disappear.

	OVA	AVA	COM	DEN	SPA
Allwein et al.	72.9	40.9	40.4	39.7	47.2
Rifkin & Klautau	40.3	41.0	40.3	40.1	38.6

Table 2: Multiclass classification error rates for the `yeast` data set. Allwein et al. used a polynomial kernel of degree four, an unknown value of C, and ten-fold cross-validation. Rifkin and Klautau used a Gaussian kernel with σ and C tuned separately for each scheme, and ten-fold cross-validation; see Section 4 for details. We see that with the Gaussian kernel, the performance of the one-vs-all scheme jumps substantially, and the differences between coding schemes disappear.

`satimage` data, noticeably better than *any* of the coding schemes when the underlying classifiers are weak.[11]

3.2.3 Crammer and Singer

Crammer and Singer develop a formalism for multiclass classification using *continuous* output coding (Crammer and Singer, 2000a,b, 2002). This formalism includes the single-machine approach discussed in Section 3.1.4 as a special case.

The Crammer and Singer framework begins by assuming that a collection of binary classifiers $f_1, \ldots, f_F$ is provided. The goal is then to *learn* the N-by-F error-correcting code matrix M. Crammer and Singer show (under some mild assumptions) that finding an optimal discrete code matrix is an NP-complete problem, so they relax the problem and allow the matrix M to contain real-valued entries. Borrowing ideas from regularization, they argue that we would like to find a matrix M that has good performance on the training set but also has a small norm. To simplify the presentation, we introduce the following notation. We let $\bar{f}(\mathbf{x})$ denote the vector $f_1(\mathbf{x}), \ldots, f_F(\mathbf{x})$, and we let M_i denote the ith row of the matrix M. Given a matrix M, we let $K(\bar{f}(\mathbf{x}), M_i)$ denote our confidence that point x is in class i; here K is an arbitrary positive definite kernel function satisfying Mercer's

11. In this context, weak is not used in a formal sense, but merely as a stand-in for poorly tuned.

Theorem. Then, the Crammer and Singer approach is:

$$\min_{M \in \mathbb{R}^{N \times F}} \quad \lambda ||M||_p + \sum_{i=1}^{\ell} \xi_i$$
$$K(\bar{f}(\mathbf{x_i}), M_{y_i}) \geq K(\bar{f}(\mathbf{x_i}), M_r) + 1 - \xi_i.$$

In the above formulation, the constraints range over all points $\mathbf{x_i}$, and all classes $r \neq y_i$. Simply put, we try to find a matrix with small norm with the property that the confidence for the correct class is greater by at least one than the confidence for any other class. Note that as in the approach discussed in Section 3.1.4, there is only a single slack variable ξ_i for each data point, rather than $N - 1$ slack variables per data point as in many of the other formulations we discuss.

In their general formulation, the norm in which the matrix is measured ($||M||_p$) is left unspecified. Crammer and Singer briefly show that if $p = 1$ or $p = \infty$ and $K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j}$, the resulting formulation is a linear program. They spend the majority of the paper considering $p = 2$ (technically, they penalize $||M||_2^2$, not $||M||_2$), and showing that this choice results in a quadratic programming problem. They take the dual (in order to introduce kernels; again, they start with the linear formulation in the primal), and indicate some algorithmic approaches to solving the dual problem.

In an interesting twist, Crammer and Singer also show that we can derive the one-machine multiclass SVM formulation used in a different paper of theirs (Crammer and Singer, 2001, discussed in Section 3.1.4) by taking $\bar{f}(\mathbf{x}) = \mathbf{x}$. In this case, the implicit assumption is that our "given" binary classifiers are d (the dimensionality of the input space) machines, where $f_i(\mathbf{x})$ is equal to the value of the ith dimension at point x. In the linear case ($K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j}$), the code matrix M becomes the N separating hyperplane functions $w_1, w_2, \ldots, w_N$. This formulation is discussed in greater detail in the previous section.

Experiments are performed on seven different data sets from the UCI repository, as well as a subset of the MNIST data set. The experiments compare the performance of the continuous output codes to discrete output codes, including the one-vs-all code, BCH codes, and random codes. Personal communication with Crammer indicates that the "base learners" for the continuous codes are linear SVMs. Seven different kernels are tested, although their identities are not disclosed (personal communication indicates that they were homogeneous and nonhomogeneous polynomials of degree one through three, and a Gaussian kernel with a γ that was not recorded). No performance results for individual experiments are given. Instead, for each data set, we find the improvement in performance for the "best kernel" (presumably the kernel with the largest difference in performance), and the average improvement in performance across the seven kernels. It is important to note that his comparison was not against an OVA system, but against using the error-correcting coding approach directly with *linear SVMs* as the underlying binary classifiers. Therefore, he was comparing the classification ability of nonlinear and linear systems on data sets for which it is well known that Gaussian classifiers perform very strongly. In this context, his results are unsurprising.

Crammer communicated to us personally the actual performance numbers, which allows us to compare their continuous codes to an OVA approach. For the `satimage` data, the *best* error rate they achieved (over all seven kernels and three coding schemes) was 9.8%,

compared to 8.2% for OVA in the current experiments (see Section 4). For the `shuttle` data, their best error rate was 0.5%; while we did not do experiments on this data set because of its unwieldy size, we note that Fürnrkanz (see below) achieved an error rate of 0.3% on this data set using an OVA system with Ripper as the underlying binary learner. We see that although the nonlinear system developed here greatly outperformed a linear system, it did not allow us to actually achieve better multiclass classification error rates than a simple well-tuned OVA system.

3.2.4 Fürnkranz

Relatively recently, Fürnkranz published a paper on round robin classification (Fürnkranz, 2002), which is another name for all-vs-all or all-pairs classification. He used Ripper (Cohen, 1995), a rule-based learner, as his underlying binary learner. He experimentally found that across a number of data sets from the UCI repository, an all-vs-all system had improved performance compared to a one-vs-all scheme. Fürnkranz used McNemar's test (McNemar, 1947) to decide when two classifiers were different. We hypothesize that Ripper is not as effective a binary learner as SVMs, and is therefore able to benefit from a scheme such as all-pairs; however, the scheme does not enable Fürnkranz to obtain better results than an OVA SVM scheme. In Section 4, we compare a variety of error-correcting schemes on the *specific data sets on which Fürnkranz found that AVA performed much better than OVA*; when the underlying classifiers are well-tuned SVMs, we find that the improved performance of AVA over OVA, observed by Fürnkranz, disappears.

3.2.5 Platt, Cristianini and Shawe-Taylor

The DAG (Directed Acyclic Graph) method of Platt, Cristianini, and Shawe-Taylor (2000) does not fit easily into the error-correcting code framework, but has much more in common with these methods than with the single-machine approaches, so is presented here. The DAG method is identical to AVA at training time—one SVM is trained for each pair of classes. At test time, an acyclic graph is used to determine which classifiers to test on a given point. First, classes i and j are compared, and whichever class achieves a lower score is removed from further consideration. By repeating this process $N-1$ times, $N-1$ classes are removed from consideration, and the final remaining class is predicted. Although the order in which classes are compared can affect the results, the authors observe that empirically, the ordering does not seem to affect the accuracy. The authors conduct well-controlled experiments on two data sets (`USPS` and `letter`), and observe that the OVA, AVA and DAG approaches have essentially identical accuracy, but that the DAG approach is substantially faster than OVA at testing time.

3.2.6 Hsu and Lin

Hsu and Lin (2002) present an empirical study comparing various methods of multiclass classification using SVMs as the binary learners. They conclude that "one-against-one and DAG methods are more suitable for practical use than the other methods." However, although they run a substantial number of experiments and their experimental protocol is sound, their data do not seem to support their conclusions. In particular, the table of

	Best Score	Worst Score	Difference	Size	Size * Diff
iris	97.333	96.667	.666	150	1.000
wine	99.438	98.876	.562	178	1.000
glass	73.832	71.028	2.804	214	6.000
vowel	99.053	98.485	.568	528	3.000
vehicle	87.470	86.052	1.418	746	10.58
segment	97.576	97.316	.260	2310	6.006
dna	95.869	95.447	.422	1186	5.005
satimage	92.35	91.3	1.050	2000	20.1
letter	97.98	97.68	.300	5000	15.0
shuttle	99.938	99.910	.028	14500	4.06

Table 3: A view of the multiclass results of Hsu and Lin (2002) for RBF kernels. The first three columns show the performance of the best and worst performing classifier for each data set, the third column shows the difference in performance between the best and worst, the fourth column the size of the data set, and the fifth column the difference expressed as a number of data points. Note that for the first six data sets, there is no training set and CV was used, so the fourth column reports the entire size of the data set. Columns 1-3 are percentages, columns four and five are numbers of points.

results for tuned RBF classifiers[12] shows That among the five methods they tried (OVA, AVA, DAG, the method of Crammer and Singer (2000b), and the method proposed by Vapnik (1998) and Weston and Watkins (1998)) are essentially identical. Table 3 presents one view of this data—we show, for each data set, the performance of the best and worst performing systems, and the difference between the best and worst both as a percentage and as a number of data points.

We see that the vast majority of these differences are quite small. Furthermore, visual inspection of Hsu and Lin's results show no clear pattern of which system is actually better across different data sets. Therefore, we must conclude that the Hsu and Lin results support the notion that at least as far as accuracy is concerned, when well-tuned binary classifiers (in this case SVMs with RBF kernels) are used as the underlying classifiers, a wide variety of multiclass classification schemes are essentially indistinguishable.

Hsu and Lin also examine the training and testing times of the various systems. Here again they find that the AVA and DAG systems have an advantage; although they are training $O(N^2)$ classifiers rather than $O(N)$ for an OVA system, the individual classifiers are much smaller, and given that the time required to train on ℓ points is generally superlinear in ℓ, we expect that AVA and DAG systems will train faster; this point is also explored in detail by Fürnkranz (2002). Their implementation is not heavily optimized, so it is difficult

12. Hsu and Lin also ran experiments where the underlying binary classifiers were linear, but this again corresponds to a situation where the underlying binary classifiers are poorly tuned, and the overall accuracy of all methods in this regime is lower than with RBF kernels on several data sets, and better on none.

to draw conclusions from this; in particular, their implementation does *not* share kernel products between different classifiers.[13] At testing time, the systems are relatively close together in speed (within a factor of 2), indicating that this argument is mostly about the time required for training. This is an interesting point, but it should be explored in a larger study involving a heavily optimized implementation on very large data sets. It is crucial, when comparing training times, to compare them on large data sets; for small data sets, all training times are short, so relative differences in training times are unimportant from a practical standpoint. Furthermore, differences on small training sets are not necessarily indicative of what will happen as larger problems are considered. In Hsu and Lin's study, the largest two problems are `letter`, with 15,000 training points in 26 dimensions, and `shuttle`, with 43,500 training points in 7 dimensions. For `letter`, the OVA approach trained 6 times slower than the AVA or DAG approaches, but for the much larger `shuttle`, the difference was only about 15%. Again, this indicates that a larger study with a more heavily optimized implementation would be necessary to untangle issues concerning the relative training times. However, it does seem likely that for large-scale problems, AVA will enjoy a speed advantage over OVA.

3.2.7 Summary

In the first paper exploring the use of error-correcting code approaches to multiclass classification, Dietterich and Bakiri were already fully aware of both the promise and the difficulty of this approach, which obviously relies heavily on the errors produced by different binary classifiers being (at least somewhat) decorrelated. In a companion paper, they address this issue for the specific case where the underlying binary classifiers are decision trees (Kong and Dietterich, 1995). We believe (and show experimentally in Section 4) that when the underlying classifiers are appropriately tuned regularization systems such as SVM or RLSC, that the errors produced by different binary classifiers will be extremely highly correlated, implying that a one-vs-all scheme will be as effective as an error-correcting code scheme. Furthermore, we will show that across several data sets, using SVM (or RLSC) in a one-vs-all framework yields results as good as error-correcting approaches.

4. Experimental Work

In previous work (Rifkin, 2002), we found that a simple OVA approach was extremely effective at multiclass classification. However, those experiments were on relatively few data sets, and furthermore, the kernel parameters were "illegally" tuned to the data sets. Therefore, we decided to perform a much larger set of more carefully controlled experiments, in order to better gauge the actual differences between multiclass data schemes. Fürnkranz (2002) reported that an AVA scheme substantially outperformed an OVA scheme over a wide variety of data sets; we decided to focus specifically on those data sets that Fürnkranz had found a large difference on. These data sets are all publicly available as part of the UCI Machine Learning Repository (Merz and Murphy, 1998). We tested the five error-correcting coding schemes suggested by Allwein et al. (2000): a one-vs-all scheme (OVA),

13. We note that the freely available SVM solver SvmFu (`http://fpn.mit.edu/SvmFu`) *does* share kernel products between SVMs while performing multiclass training (or testing).

an all-vs-all or all-pairs scheme (AVA), a complete code (COM), a dense code (DEN), and a sparse code (SPA). For further descriptions of these codes, see Section 3.2.2 or the paper by Allwein et al. (2000). In some cases, experiments could not be run because they were too computationally expensive.

4.1 Experimental Protocol

Of the data sets found by Fürnkrnaz to have a large performance difference between OVA and AVA schemes, we selected ten on which to perform experiments: `soybean-large`, `letter`, `satimage`, `abalone`, `optdigits`, `glass`, `car`, `spectrometer`, `yeast`, and `page-blocks`. For all data sets which have test sets, we use only the given train/test split (Fürnkranz tested three data sets in both the original train/test and cross-validation settings). We omitted the `covertype` data set because its very large test set made it unwieldy to work with, and we omitted the `vowel` data set because, although Fürnkranz found a significant performance difference under cross-validation, this difference disappeared under the original train/test split.[14]

By 10-fold cross-validation (CV), we refer to randomly breaking a data set into ten equal-sized (as closely as possible) subsets, respecting as closely as possible the class percentages in the original data, and then considering the ten train/test splits obtained by taking nine of the subsets as training data and the tenth as test data.

The only data set in our experiments which had any missing values was the `soybean-large` data set. For this data set, we first filled in missing elements in the data using the training set modes (all the attributes are nominal; we could use means if the attributes were numeric).

For data sets with a train/test split (`soybean-large`, `letter`, `satimage`, `abalone` and `optdigits`), each numeric attribute was normalized to have mean 0 and variance 1 on the training set (the same scaling was applied to the training and test sets, but the scaling was determined using only the training set). Finally, each nominal attribute taking on k different values was converted to k binary (0-1 valued) attributes, where the ith variable is set to 1 if and only if the nominal attribute takes on the ith possible value. The parameters γ and C (or, equivalently, λ for RLSC) were found by doing 10-fold cross-validation on the training set. We first set $C = 1$ and $\gamma = 1$ (a reasonable "rough guess" given the mean 0 variance 1 normalization), then increased or decreased γ by a factor of 2 until no improvements were seen for three consecutive attempts. Then γ was held fixed at the best value found and an identical optimization was performed over C. It would have been better to jointly optimize over C and γ, but this would have been computationally prohibitively expensive.

For data sets without a standard train/test split, we split the original data set using 10-fold CV, and then performed the procedure described in the previous paragraph. Note that because of this, different CV "splits" of a data set could end up with slightly different normalizations, C and γ values.

14. It is interesting to note that for the `vowel` data set, the standard train/test split is based on the speaker. Under cross-validation, instances of the same speaker are mixed into both the training and test sets, and we might expect performance in this scenario to be very different: indeed, in Fürnkranz's study, all methods perform much better under cross-validation than under the original train/test split.

Basic information about the data sets is summarized in Table 4. The baseline error for each data set is the error rate for a classification scheme which always chooses the class containing the largest number of examples. The number of classifiers required for each scheme is shown in Table 5. The large number of classifiers required for the COM code indicates that this approach is not feasible for data sets with many classes. In Section 5, we will show that when RLSC is used as the underlying binary classifier, the OVA and the COM approaches will produce *identical* classification decisions, indicating that the inability to directly implement this scheme is of little concern.

The simulations were implemented using a modified version of the WEKA package (Witten and Frank, 1999). The multiclass schemes were implemented as a new WEKA classifier and integrated into the original package. The underlying binary SVMs were trained using the SVMTorch package (Collobert and Bengio, 2001). The binary RLSCs were implemented directly in Java.

4.2 Results

Name	# train	# test	# classes	# attributes / # nominal attr.	average / min / max # examples per class	baseline error (%)
soybean-large	307	376	19	35 / 35	16.2 / 1 / 40	87.2
letter	16000	4000	26	16 / 0	615.4 / 576 / 648	96.4
satimage	4435	2000	6	36 / 0	739.2 / 409 / 1059	77.5
abalone	3133	1044	29	8 / 1	108.0 / 0 / 522	84.0
optdigits	3823	1797	10	64 / 0	382.3 / 376 / 389	89.9
glass	214	-	7	9 / 0	30.6 / 0 / 76	64.5
car	1728	-	4	6 / 6	432.0 / 65 / 1210	30.0
spectrometer	531	-	48	101 / 0	11.1 / 1 / 55	89.6
yeast	1484	-	10	8 / 0	148.4 / 5 / 463	68.8
page-blocks	5473	-	5	10 / 0	1094.6 / 28 / 4913	10.2

Table 4: Data sets used for the experiments.

Name	OVA	AVA	COM	DEN	SPA
soybean-large	19	171	262143	43	64
letter	26	325	33554431	48	71
satimage	6	15	31	26	39
abalone	29	406	268435455	49	73
optdigits	10	45	511	34	50
glass	6	15	31	26	39
car	4	6	7	20	30
spectrometer	48	1128	1.407e+014	56	84
yeast	10	45	511	34	50
page-blocks	5	10	15	24	35

Table 5: Number of possible binary classifiers for each code matrix.

Tables 6 through 9 show a comparison between the OVA method and each of the methods AVA, DEN, SPA, and COM. For each data set, we show the error rates of each system, the

Data Set	AVA	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	6.38	5.85	0.530	0.971	[-0.008, 0.019]
letter	3.85	2.75	1.09	0.978	[0.008, 0.015]
satimage	8.15	7.80	0.350	0.984	[-5E-4, 0.008]
abalone	72.32	79.69	-7.37	0.347	[-0.102, -0.047]
optdigits	3.78	2.73	1.05	0.982	[0.006, 0.016]
glass	30.37	30.84	-.470	0.818	[-0.047, 0.037]
car	0.41	1.50	-1.09	0.987	[-0.016, -0.006]
spectrometer	42.75	53.67	-10.920	0.635	[-0.143, -0.075]
yeast	41.04	40.30	0.740	0.855	[-0.006, 0.021]
page-blocks	3.38	3.40	-.020	0.991	[-0.002, 0.002]

Table 6: SVM test error rate (%), OVA vs. AVA.

Data Set	DEN	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	5.58	5.85	-0.270	0.963	[-0.019, 0.013]
letter	2.95	2.75	0.200	0.994	[5E-4, 0.004]
satimage	7.65	7.80	-0.150	0.985	[-0.006, 0.003]
abalone	73.18	79.69	-6.51	0.393	[-0.092, -0.039]
optdigits	2.61	2.73	-0.12	0.993	[-0.004, 0.002]
glass	29.44	30.84	-1.40	0.911	[-0.042, 0.014]
car	-	1.50	-	-	-
spectrometer	54.43	53.67	-0.760	0.866	[-0.011, 0.026]
yeast	40.30	40.30	0.00	0.900	[-0.011, 0.011]
page-blocks	-	3.40	-	-	-

Table 7: SVM test error rate (%), OVA vs. DENSE.

difference in error rates, the percentage of data points on which the two classifiers agree (predict the same output class), and finally a 90% bootstrap confidence interval for the difference in performance of the two methods (see Appendix A).[15]

Our primary observation for tables 6 through 9 is that in nearly all cases the results of the two methods compared are very close. In a majority of the comparisons, 0 is in the bootstrap interval, meaning we cannot conclude that the classifiers are statistically significantly different. For many of the remaining comparisons, the bootstrap interval includes numbers which are quite small. The main counterexample seems to be the `abalone` data set: on this data set we find that the OVA method performs noticeably worse than any of the other methods (except COM, which we could not run for computational reasons). It is perhaps worth noting that on this data set, the performance in general is quite poor—although there are 29 classes, a single class contains 16% of the data, yielding a baseline error rate of 84%,

15. As an additional check, we also ran McNemar's test for these experiments. With no exceptions, we found that McNemar's test reported a significant different (at the 5% level) between two classifiers if and only if the confidence interval for our bootstrap test did not include 0.

Data Set	SPA	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	6.12	5.85	0.270	0.968	[-0.011, 0.016]
letter	3.55	2.75	0.800	0.980	[0.005, 0.011]
satimage	8.85	7.80	1.05	0.958	[0.003, 0.018]
abalone	75.67	79.69	-4.02	0.352	[-0.067, -0.014]
optdigits	3.01	2.73	0.280	0.984	[-0.002, 0.008]
glass	28.97	30.84	-1.87	0.738	[-0.070, 0.033]
car	0.81	1.50	-0.69	0.988	[-0.011, -0.003]
spectrometer	52.73	53.67	-0.940	0.744	[-0.038, 0.019]
yeast	40.16	40.30	-0.140	0.855	[-0.015, 0.013]
page-blocks	3.84	3.40	0.440	0.979	[0.001, 0.007]

Table 8: SVM test error rate (%), OVA vs. SPARSE.

Data Set	COM	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	-	5.85	-	-	-
letter	-	2.75	-	-	-
satimage	7.80	7.80	0.00	0.999	[-1E-3, 1E-3]
abalone	-	79.69	-	-	-
optdigits	2.67	2.73	-0.060	0.996	[-0.003, 0.002]
glass	29.44	30.84	-1.340	0.911	[-0.042, 0.014]
car	1.68	1.50	-0.180	0.998	[5.79E-4, 0.003]
spectrometer	-	53.67	-	-	-
yeast	38.61	40.30	-1.690	0.906	[-0.028, -0.005]
page-blocks	3.49	3.40	-0.090	0.983	[-0.002, 0.004]

Table 9: SVM test error rate (%), OVA vs. COMPLETE.

and none of the systems achieve an error rate lower than 72%, indicating that this data set is very difficult to classify well. Additionally, on the `spectrometer` data set, the AVA method performs better than any of the remaining methods. On the remaining 8 of the 10 data sets, OVA performs essentially as well as the other methods.

Also interesting to note is the AGREE column of the classifiers. In many cases, the AGREE number is substantially higher than the error rates, indicating that not only do the classifiers make errors on the same point, *but that different systems often make identical "incorrect" predictions.* This is in keeping with the authors' intuition that when well-tuned SVMs are used as binary classifiers, points become errors not because of deficiencies in the method of combining binary classifiers, but simply because for all practical purposes, the points "look" more like a member of an incorrect class than their true class. These types of errors will in general be extremely difficult to correct.

It is critical to keep in mind that these results were obtained using only data sets on which Fürnkranz (2002) found a substantial difference in performance between OVA and AVA approaches. From this perspective, we feel that our results strongly support the

Data Set	FUR	OVA	DIFF	AGREE	BOOTSTRAP
soybean-large	13.3	5.85	7.45	0.891	[.056, .109]
letter	7.7	2.75	4.95	0.922	[.043, .057]
satimage	12.2	7.80	4.40	0.906	[.0345, .055]
abalone	74.1	79.69	-5.59	0.335	[-.083, -0.029]
optdigits	7.5	2.73	4.77	0.920	[0.035, 0.056]
glass	26.2	30.84	-4.64	0.734	[-0.098, 0.005]
car	2.8	1.50	1.3	0.969	[0.006, 0.020]
spectrometer	51.2	53.67	-2.47	0.488	[-0.060, 0.017]
yeast	41.6	40.3	1.29	0.765	[-0.005, -0.032]
page-blocks	2.6	3.40	-0.80	0.978	[-0.012, -0.005]

Table 10: SVM test error rate (%), OVA vs. Fürnkranz's R^3.

hypothesis that when the underlying binary classifiers are properly tuned SVMs, there is little difference between competing methods of multiclass classification. On the other hand, we fully expect that when weaker or poorly tuned binary classifiers are used, that combining the classifiers using an error-correcting approach will improve performance. The crucial question to the practitioner then becomes how to get the best performance. In Table 10 we compare our OVA SVM results to Fürnkranz's R^3 (essentially AVA) method. The code was obtained directly from Fürnkranz. Details of the comparison are given in Appendix B. We had to run our own experiments (rather than using Fürnkranz's numbers directly) because we needed the actual predictions at each data point in order to compute the AGREE and BOOTSTRAP columns of the table. Our results were similar to Fürnkranz's (error rates within 2%)[16] on 8 of the 10 data sets. We have no real explanation of the relatively large differences on `soybean` and `optdigits`.

Again, the primary impression from this table is that the methods are quite similar in performance; if anything, the SVMs seem to have a slight advantage. Certainly, these experiments do not support the idea that by intentionally using weaker classifiers and then combining them using an error-correcting approach, we can exceed the accuracy of an approach that begins with well-tuned binary classifiers and a simple OVA combination.

In Section 5, we present theoretical results on multiclass classification using RLSC as the underlying binary classifier. For this reason, it is important to gauge the empirical performance of RLSC on multiclass classification tasks. These results are presented in Tables 11 through 14. Note that because RLSC is substantially more computationally expensive than SVMs both to train and to test, we were only able to perform a subset of the SVM experiments; in particular, no experiments with the COM code were performed. In almost all cases, the differences between RLSC and SVM performance were very small. The vast majority of bootstrap intervals contained 0. Therefore, although the arguments we make in Section 5 apply directly to RLSC rather than SVMs, we can "infer experimentally" that the results are useful as well for describing SVM behavior.

16. It is interesting that the difference in the choice of random seed can lead to differences in accuracy of 1% or so. This is in keeping with our impressions that differences in performance of this magnitude can often be caused by "nuisance" issues such as random seeds, tolerance and accuracy parameters, etc.

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	6.12	5.85	0.270	0.984	[-0.008, 0.013]
letter	-	2.75	-	-	-
satimage	7.9	7.80	0.010	0.979	[-0.004, 0.006]
abalone	72.7	79.69	-7.000	0.284	[-0.099, -0.041]
optdigits	2.5	2.73	-0.230	0.980	[-0.007, 0.003]
glass	31.3	30.84	0.460	0.808	[-0.037, 0.047]
car	2.9	1.50	1.40	0.980	[0.009, 0.020]
spectrometer	52.3	53.67	-1.370	0.821	[-0.036, 0.009]
yeast	40.0	40.30	-0.300	0.872	[-0.016, 0.011]
page-blocks	3.25	3.40	-0.150	0.983	[-0.004, 0.001]

Table 11: OVA test error rate (%), RLSC vs. SVM.

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	8.2	6.38	1.820	0.941	[0.000, 0.037]
letter	-	3.85	-	-	-
satimage	7.4	8.15	-.750	0.974	[-0.013, -0.001]
abalone	73.66	72.32	1.340	0.560	[-0.009, 0.034]
optdigits	3.0	3.78	-.780	0.974	[-0.013, -0.002]
glass	29.4	30.37	-0.970	0.864	[-0.047, 0.028]
car	2.3	0.41	1.89	0.980	[0.013, 0.024]
spectrometer	49.1	42.75	6.350	0.738	[0.036, 0.092]
yeast	40.0	41.04	-1.040	0.838	[-0.025, 0.005]
page-blocks	3.4	3.38	0.020	0.981	[-0.003, 0.003]

Table 12: AVA test error rate (%), RLSC vs. SVM.

The parameter settings for SVM and RLSC, found via cross-validation over the training set, are given in Appendix C.

5. Multiclass Classification with RLSC: Theoretical Arguments

In this section, we make some simple yet powerful arguments concerning multiclass classification with RLSC as the underlying binary classifier. Recall that to solve an RLSC problem, we solve a linear system of the form

$$(K + \lambda \ell I)\mathbf{c} = \mathbf{y}.$$

Because this is a *linear* system, the vector $\mathbf{c}$ is a linear function of the right hand side $\mathbf{y}$. Define the vector $\mathbf{y^i}$ as

$$y^i_j = \begin{cases} 1 & \text{if } y_j = i, \\ 0 & \text{otherwise.} \end{cases}$$

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	8.0	5.58	2.41	0.971	[0.011, 0.040]
letter	8.0	7.65	0.350	0.976	[-0.002, 0.009]
abalone	72.8	73.18	-0.380	0.663	[-0.025, 0.017]
optdigits	2.5	2.61	-0.110	0.982	[-0.006, 0.003]
glass	29.9	29.44	-.460	0.864	[-0.037, 0.042]
car	-	-	-	-	-
spectrometer	52.9	54.43	-1.530	0.825	[-0.038, 0.008]
yeast	40.0	40.30	-.300	0.888	[-0.016, 0.009]
page-blocks	-	-	-	-	-

Table 13: DENSE test error rate (%), RLSC vs. SVM.

Data Set	RLSC	SVM	DIFF	AGREE	BOOTSTRAP
soybean-large	7.4	6.12	1.280	0.973	[0.000, 0.027]
letter	-	3.55	-	-	-
satimage	8.4	8.85	-0.450	0.958	[-0.011, 0.003]
abalone	73.3	75.67	-2.370	0.621	[-0.043, -0.005]
optdigits	3.6	3.01	0.590	0.977	[0.001, 0.011]
glass	29.4	28.97	-0.430	0.841	[-0.037, 0.047]
car	4.3	0.81	3.490	0.963	[0.028, 0.043]
spectrometer	52.5	52.73	-0.230	0.827	[-0.024, 0.021]
yeast	40.9	40.16	0.740	0.877	[-0.005, 0.020]
page-blocks	3.5	3.84	-0.340	0.980	[-0.006, 1.83E-4]

Table 14: SPARSE test error rate (%), RLSC vs. SVM.

Now, suppose that we solve N RLSC problems of the form

$$(K + \lambda \ell I)\mathbf{c^i} = \mathbf{y^i},$$

and denote the associated functions $f^{c_1}, \ldots, f^{c_N}$. Now, for *any* possible right hand side $\mathbf{y}^*$ for which the y_i and y_j are equal, whenever $\mathbf{x_i}$ and $\mathbf{x_j}$ are in the same class, we can calculate the associated $\mathbf{c}$ vector from the $\mathbf{c^i}$ *without solving a new RLSC system.* In particular, if we let m_i ($i \in \{1, \ldots, N\}$) be the y value for points in class i, then the associated solution vector $\mathbf{c}$ is given by

$$\mathbf{c} = \sum_{i=1}^{N} \mathbf{c^i} m_i.$$

For any code matrix M not containing any zeros, we can compute the output of the coding system using only the entries of the coding matrix and the outputs of the underlying one-vs-all classifiers. In particular, we do not need to actually train the classifiers associated with the coding matrix. We can simply use the appropriate linear combination of the "underlying" one-vs-all classifiers. For any $r \in \{1, \ldots, N\}$,

$$
\begin{aligned}
& \arg \min_{r\in\{1,\ldots,N\}} \sum_{i=1}^{F} L(M_{ri} f_i(\mathbf{x})) \\
= \; & \arg \min_{r\in\{1,\ldots,N\}} \sum_{i=1}^{F} L(M_{ri} \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x})) \\
= \; & \arg \min_{r\in\{1,\ldots,N\}} \sum_{i=1}^{F} (M_{ri} - \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x}))^2 \\
= \; & \arg \min_{r\in\{1,\ldots,N\}} \sum_{i=1}^{F} (1 - 2M_{ri} \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x}) + \sum_{j=1}^{N}\sum_{k=1}^{N} M_{ji} M_{ki} f^j(\mathbf{x}) f^k(\mathbf{x})) \\
= \; & \arg \min_{r\in\{1,\ldots,N\}} \sum_{i=1}^{F} (-M_{ri} \sum_{j=1}^{N} M_{ji} f^j(\mathbf{x})) \\
= \; & \arg \min_{r\in\{1,\ldots,N\}} \sum_{j=1}^{N} D_{rj} f^j(\mathbf{x}) \\
= \; & \arg \min_{r\in\{1,\ldots,N\}} -F f^r(\mathbf{x}) + \sum_{\substack{j=1 \\ j\neq r}} D_{rj} f^j(\mathbf{x}), \qquad (2)
\end{aligned}
$$

where we define define $D_{ri} \equiv -\sum_{j=1}^{F} M_{jr} M_{ji}$, and note that $D_{ii} = -F$ for all i.

We define a coding matrix to be *class-symmetric* if D_{ij} (and therefore C_{ij} as well) is independent of i and j (assuming $i \neq j$). Note that a sufficient (but not necessary) condition for a coding-matrix to be class-symmetric is if, whenever it contains a column containing k 1's and $n-k$ -1's, all $\frac{N!}{k!(N-k)!}$ such columns are included. The OVA and COMPLETE schemes are class-symmetric, while the DENSE scheme is in general not (the AVA and SPARSE schemes include zeros in the coding matrix, and are not addressed by this analysis). Other schemes generated by means of explicit error-correcting codes may be class-symmetric in this sense as well.

For class-symmetric schemes, D_{rj} is independent of the choice of r and j, and can be denoted simply as D^*. For these schemes, noting that $D^* > -F$ (assuming no identical rows in M),

$$
\begin{aligned}
f(\mathbf{x}) &= \arg \min_{r\in 1,\ldots,N} -F f^r(\mathbf{x}) + D^* \sum_{\substack{j=1 \\ j\neq r}}^{N} f^j(\mathbf{x}) \\
&= \arg \max_{r\in 1,\ldots,N} f^r(\mathbf{x}).
\end{aligned}
$$

We have shown that *when RLSC is used as the underlying binary learner for class-symmetric coding matrices containing no zeros, the predictions generated are identical to those of the one-vs-all scheme.*

For matrices which are not class-symmetric, we cannot use the above argument directly. However, we believe that Equation 2 is still highly indicative. If the D_{rj} are all close to

equal (which is generally the case for the matrices generated), we can see that if $f^i(\mathbf{x})$ is substantially larger than $f^j(\mathbf{x})$ for $j \neq i$, then i will be chosen by the multiclass RLSC system. Although this notion could potentially be quantified, it would probably not provide much additional insight.

It is important to note that the above arguments *only* apply to schemes in which all the data is used for every classifier, and there are no zeros in the coding matrix. This includes the OVA, DEN, and COM schemes of Allwein et al., but not their AVA or SPA schemes. Whether, in practice, the addition of zeros to the coding scheme can make a big difference is an interesting question. The experimental results of this paper seem to indicate that for most data sets, there is no difference, but on some small data sets with many classes and high error rates, there may be a modest improvement.

6. Discussion and Conclusions

It is hoped that this work will help practitioners. The viewpoint presented in this paper is that the most important step in good multiclass classification is to use the best binary classifier available. Once this is done, it seems to make little difference what multiclass scheme is applied, and therefore a simple scheme such as OVA (or AVA) is preferable to a more complex error-correcting coding scheme or single-machine scheme.

There seems to be some evidence that on small data sets with large numbers of classes (in our experiments, `abalone` and `spectrometer`) where the overall error rate is high, an AVA scheme can offer a moderate performance boost over an OVA scheme. However, this evidence is not strong—on the majority of data sets, we found the performance of all the schemes to be essentially identical. On the data sets where AVA did show an improvement over OVA, the error rates of all approaches were quite high, bringing into question the suitability of the task and representation to machine learning approaches.

We have also presented, using RLSC as the underlying binary classifiers, what we believe is the beginning of a theory demonstrating why simple approaches may perform just as well as error-correcting approaches. In Section 4 we showed empirically that across a number of data sets, RLSC and SVMs produce very similar results, thereby motivating the use of RLSC as a theoretical tool to study approaches to multiclass classification.[17] When Dietterich and Bakiri wrote their paper introducing error-correcting coding approaches (Dietterich and Bakiri, 1995), they were already aware that these methods would be useful only if the correlation between different binary classifiers was not too high. But the use of RLSC as the underlying classifier, with its simple linear structure, makes clear that the correlation is extremely high—once we've trained the OVA RLSC classifiers, we already know the RLSC outputs for any classifier that uses all the classes. It would be nice to extend this result to a situation where not every class is used by each binary classifier (for example, the AVA scheme), although how to do so is an open question.

This paper has focussed on multiclass classification accuracy, but a brief word on speed is in order. Comparing different machine learning algorithms for speed is notoriously difficult; we are no longer judging mathematical algorithms but are instead judging specific implementations. However, some possibly useful general observations can be made. Empir-

17. We are not suggesting the use of RLSC in practice, because on nonlinear problems, it is much slower than the SVM.

ically, SVM training time tends to be superlinear in the number of training points. Armed only with this assumption, it can be shown (see the analysis by Fürnkranz (2002) for an in-depth discussion) that an OVA scheme will train more slowly than an AVA scheme. This is certainly observed by Hsu and Lin (2002); however, on their largest data set (`shuttle`, 43,500 training points), they observe approximately a 15% difference in training times, whereas for their second largest data set (`letter`, 15,000 training points), OVA is six times slower, indicating that the size of the data set alone is not highly predictive of the size of the difference.

In many applications, the time required to test is of far greater importance than the time required to train. It is clear that, for any training scheme discussed here except the DAG scheme, the testing time is directly proportional to the number of unique support vectors. Hsu and Lin (2002) present the average number of unique SVs for their experiments, and the OVA scheme is often the worst performing from this perspective, but the differences are often not large. Nevertheless, it appears that from a speed perspective, an AVA or DAG scheme will likely have the advantage. Further work, especially on very large data sets using heavily-optimized implementations, could be useful here.

Additonally, we note that our belief that an OVA scheme is as accurate as any other scheme is predicated on the assumption that the classes are "independent"—that the classes do not belong to a natural hierarchy, and that we do not necessarily expect examples from class "A" to be closer to those in class "B" than those in class "C". In the latter situation, especially when few examples were available, we might suspect that an algorithm which exploited the relationships between classes could offer superior performance. This is an interesting open question.

Finally, we do not wish to overstate our claims. In particular, all of our experiments were done on data sets from the UCI repository, and some might well view these data sets as "toy data sets". In particular, it is unknown how OVA will compare to other schemes on very large, difficult problems that include label noise. Rather than view our experiments as proof that OVA is necessarily an ideal method in all possible situations, we instead view our experiments as demonstrating that there is no compelling evidence to date that either error-correcting coding or single-machine approaches outperform OVA when the underlying binary classifiers are properly tuned. This is interesting because it is in contrast to the work of Allwein et al. (2000) and Fürnkranz (2002), who concluded (using the *same* data sets that were used in this paper), respectively, that error-correcting approaches and AVA strongly outperformed OVA; in our own experiments, we show that when more care is taken to use properly tuned binary classifiers, the performance of all methods improves, and the difference in performance between methods largely disappears. While of course it remains open what will happen on more difficult tasks, we believe the burden should be on the developer of a new multiclass algorithm to show that it outperforms a naive approach such as OVA. To date, neither the single-machine approach nor the error-correcting code approach has resulted in an algorithm that demonstrably outperforms OVA in terms of accuracy. We welcome future work that offers practitioners a better algorithm than OVA—whether it is an entirely new algorithm, or a demonstration that any of the currently known algorithms outperforms OVA on more difficult data sets.

Appendix A. A Simple Bootstrap Approach to Comparing Classifiers

We briefly discuss McNemar's statistical test (McNemar, 1947, Everitt, 1977), and also present a simple bootstrap approach for comparing two different classifiers ("A" and "B") that were tested on the same test set. In both cases, by examining the paired outputs, we compute the empirical probabilities of the following four events over an individual data point x:

- both classifiers were correct on (CC),
- both classifiers were incorrect (II),
- A was correct, B incorrect (CI),
- B was correct, A incorrect (IC).

We let $n(CC)$ denote the number of observations falling into "class" CC, and similarly for the other "classes." McNemar's test rests on the assumption that under the null hypothesis (classifiers "A" and "B" have equivalent Bayes error rates), $n(CI)$ (as well as $n(IC)$) is binomially distributed with $p = 1/2$ and $N = n(IC) + n(CI)$. In practice, the test is carried out by means of a χ^2-approximation.

Instead of (or in addition to) using McNemar's test, we can use the following simple bootstrap procedure. We generate a large number (in our experiments, 10,000) of standard bootstrap data sets: samples of size ℓ where each data point independently belongs to the "class" CC, II, CI, or IC with probability equal to the empirical probability of the associated "class" on the original data set. For each bootstrap sample, we compute the difference in performance of the two classifiers. We generate a $k\%$ confidence interval (in our experiments, $k = 90\%$) by reporting the $\frac{1-k}{2}$'th and $\frac{1+k}{2}$'th percentiles of this distribution.

We feel that this technique represents an improvement over McNemar's test for two reasons. The first is that it takes into account the number of examples on which the two classifiers agree (and therefore the total size of the data set): with McNemar's test, only the number of points contributing to events CI and IC matter. Suppose we consider two scenarios. In the first scenario, there are 100 data points, and classifier "A" gets them all right and classifier "B" gets them all wrong. In the second scenario, there are 1,000,000 points, both classifiers get 999,900 points right, classifier "A" gets the remaining points right and classifier "B" gets them wrong. As far as McNemar's test is concerned, these two situations are *identical*.

The second, more important issue is that McNemar's test provides only an indication of whether the two classifiers are significantly different, but provides no indication of the *size* of the difference. This is obviously of crucial importance to practitioners—even if two methods are significantly different, if the actual difference in performance is quite small, a practitioner may well choose to implement the method which is less computationally intensive.

It is important to note that both this technique and McNemar's test require the paired predictions of the classifiers, and neither can be used if only the absolute error rates of the two systems are available.

Data Set	Current Experiments	Furnkranz's paper
soybean-large	13.3	6.30
letter	7.7	7.85
satimage	12.2	11.15
abalone	74.1	74.34
optdigits	7.5	3.74
glass	26.2	25.70
car	2.8	2.26
spectrometer	51.2	53.11
yeast	41.6	41.78
page-blocks	2.6	2.76

Table 15: Results for Furnkranz's R^3 ("double" all-pairs) using Ripper as the base learner.

Appendix B. Fürnkranz's Experiments Revisited

In this appendix, we describe attempts to "replicate" the results of Fürnkranz (2002). There are at least two good reasons for doing this. First, we wanted to see to what extents the results *were* replicable. Secondly, we wanted to obtain actual predictions of his algorithm on each data set, in order to calculate bootstrap confidence intervals against our other predictors (see Appendix A). Fürnkranz was kind enough to make his code available to us; we performed our own preprocessing as described in Section 4.1.

The results are shown in Table 15. The results are quite close on many of the data sets. The numbers are not identical because the Ripper (Cohen, 1995) implementation uses random numbers internally, and our random seed and Fürnkranz's differ—on our own machine, repeated runs give identical results. It is unclear what is causing the relatively large discrepancies on the `soybean` and `optdigits` data sets.

We are unable to compute bootstrap confidence intervals on the difference in performance between Fürnkranz's original results and our attempts to reproduce these results because we do not have his actual predictions available.

We were able to get broad agreement with Fürnkranz on the majority of the data sets. It is worth noting that in general, differing choices of the random seed seem to lead to differences in performance on the order of 1%; because many of the differences in performance between different classification systems on these data sets are of the same order, this is further evidence that such differences must be carefully tested for significance.

Appendix C. SVM and RLSC Parameter Settings

In this appendix we present the parameters selected for our experiments. All parameters were selected via 10-fold cross-validation. For each experiment, the Gaussian parameter γ was first selected, then the regularization parameter (C for SVMs, λ for RLSC) was selected. For those experiments with no test set, we performed 10-fold cross-validation on each of the 10 "train/test" splits; we did not constrain each split to have the same parameters. For

Data Set	OVA	AVA	DEN	SPA	COM
soybean-large	4	32	2	16	-
letter	4	32	4	16	-
satimage	4	4	2	16	4
abalone	2	4	4	4	-
optdigits	2	8	4	32	2
glass	8.4 (91.8)	26.0 (489.6)	16.4 (583.8)	11.2 (25.0)	18.0 (1.4e3)
car	3.2 (1.0)	6.8 (3.4)	-	10.0 (55.2)	3.6 (3.0)
spectrometer	3.0 (3.4)	46.4 (1.1E3)	5.2 (20.2)	5.8 (26.8)	-
yeast	1.4 (1.4)	5.8 (33.4)	3.0 (3.9)	5.0 (81.3)	2.0 (0.6)
page-blocks	36.0 (553.6)	82.4 (4.81E3)	-	50.4 (1.1E3)	14.6 (296.0)

Table 16: SVM configuration: the C parameter.

Data Set	OVA	AVA	DEN	SPA	COM
soybean-large	2^{-7}	2^{-7}	2^{-5}	2^{-6}	-
letter	2^{-1}	2^{-1}	2^{-1}	2^{-1}	-
satimage	2^{-2}	2^{-2}	2^{-2}	2^{-3}	2^{-2}
abalone	2^{4}	2^{-2}	2^{-1}	2^{-1}	-
optdigits	2^{-5}	2^{-6}	2^{-5}	2^{-6}	2^{-5}
glass	$2^{-1.8}$ (0.1)	$2^{-2.9}$ (1.4e-3)	$2^{-2.7}$ (4.1e-3)	$2^{-1.3}$ (1.5e-2)	$2^{-2.1}$ (3.4e-2)
car	2^{-3} (0.0)	2^{-4} (0.0)	-	2^{-4} (0.0)	2^{-3} (0.0)
spectrometer	$2^{-3.6}$ (8.2E-4)	2^{-6} (3.7E-5)	$2^{-3.9}$ (9.3E-4)	$2^{-3.9}$ (1.1E-3)	-
yeast	$2^{-1.2}$ (1.3E-2)	$2^{-3.5}$ (1.3E-3)	$2^{-1.5}$ (1.5E-2)	$2^{-3.1}$ (7.9E-4)	$2^{-1.8}$ (2.2E-2)
page-blocks	$2^{-2.4}$ (2.5E-2)	$2^{-2.9}$ (6.6E-3)	-	$2^{-1.2}$ (1.3E-2)	$2^{-0.8}$ (9.7E-2)

Table 17: SVM configuration: γ for the Gaussian kernel.

Data Set	OVA	AVA	DEN	SPA
soybean-large	0.5	2.0	2.0	2.0
letter	-	-	-	-
satimage	0.2	0.2	0.2	0.2
abalone	0.2	16.0	2.0	16.0
optdigits	0.1	1.56E-2	3.12E-2	0.1
glass	2.8 (20.0)	2.4 (20.8)	2.0 (5.2)	0.2 (3.3E-2)
car	0.2 (7.12E-3)	1.38E-2 (3.39E-4)	-	-
spectrometer	3.5 (18.8)	0.5 (1.4)	5.1 (84.6)	0.9 (0.2)
yeast	7.1 (88.4)	17.4 (559.2)	28.1 (1.68E3)	17.4 (559.2)
page-blocks	1.1 (0.6)	1.1 (9E-2)	-	1.0 (0.1)

Table 18: RLSC configuration: $\lambda * \ell$.

these data sets, we report the mean and variance of the parameters. The results are shown in Tables 16 to 18.

It is worth noting that many of the variances are quite large relative to the means, indicating that small changes in the data set (each pair of "training folds" shares 80% of the total data set) can lead to relatively large changes in the parameter settings. We tend to believe that changes of this magnitude do not induce a *large* difference in classifier accuracy (with SVMs or RLSC, parameter settings that are "close" will give classifier performances that are "close"), but this is a question for further study.

Data Set	OVA	AVA	DEN	SPA
soybean-large	2^{-7}	2^{-2}	2^{-2}	2^{-2}
letter	-	-	-	-
satimage	2^{-2}	2^{-1}	2^{-2}	2^{0}
abalone	2^{-4}	2^{-5}	2^{-3}	2^{-3}
optdigits	2^{-6}	2^{-4}	2^{-5}	2^{-4}
glass	$2^{2.5}$ (88.1)	$2^{-0.9}$ (0.2)	$2^{-1.5}$ (0.1)	$2^{-0.5}$ (6E-2)
car	$2^{-3.6}$ (8.2E-4)	$2^{-3.0}$ (0.0)	-	-
spectrometer	$2^{-3.5}$ (4.19E-3)	$2^{-3.9}$ (9.37E4)	$2^{-3.4}$ (3.91E-3)	$2^{-2.9}$ (1.78E-2)
yeast	$2^{-1.5}$ (6.7E-2)	$2^{-0.6}$ (0.4)	$2^{-1.3}$ (6.58E-2)	$2^{-0.6}$ (0.4)
page-blocks	$2^{1.0}$ (0.6)	$2^{-2.8}$ (3.16E-3)	-	$2^{-0.4}$ (8.06E-2)

Table 19: RLSC configuration: γ for the Gaussian kernel.

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

R. C. Bose and D. K. Ray-Chaudhuri. On a class of error-correcting binary group codes. *Information and Control*, 3:68–79, 1960.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, 1992.

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

E. Bredensteiner and K. P. Bennett. Multicategory classification by support vector machines. In *Computational Optimizations and Applications*, volume 12, pages 53–79, 1999.

W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

K. Crammer and Y. Singer. Improved output coding for classification using continuous relaxation. In *Proceedings of the Thirteenth Annual Conference on Neural Information Processing Systems*, 2000a.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learning Theory*, pages 35–46, 2000b.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-basd vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2):201–233, 2002.

T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

B. Everitt. *The analysis of contingency tables*. Chapman and Hall, 1977.

Theodoros Evgeniou, Massimiliano Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances In Computational Mathematics*, 13(1):1–50, 2000.

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. Fung and O. L. Mangasarian. Proximal support vector classifiers. In Provost and Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86. ACM, 2001a.

G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. Technical report, Data Mining Institue, 2001b.

J. Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

C. F. Gauss. Theoria combinationis obsevationum erroribus minimis obnoxiae. *Werke*, 1823.

F. Girosi and T. Poggio. Networks and the best approximation property. Technical Report A.I. Memo No. 1164, C.B.C.L Paper No. 45, Massachusetts Institute of Technology, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences, October 1989.

Y. Guermeur. Combining discriminant models with new multi-class svms. *Pattern Analysis and Applications*, 5(2):168–179, 2002.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

T. Joachims. Making large-scale SVM learning practical. Technical Report LS VIII-Report, Universität Dortmund, 1998.

E. B. Kong and T. G. Dietterich. Why error-correcting output coding works with decision trees. Technical report, Department of Computer Science, Oregon State University, Corvallis, OR, 1995.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines. Technical Report 1043, Department of Statistics, University of Wisconsin, 2001a.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines. In *Proceedings of the 33rd Symposium on the Interface*, 2001b.

Y. Lin. Support vector machines and the Bayes rule in classification. Technical Report Technical Report Numberr 1014, Department of Statistics, University of Wisconsin, 1999.

Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12:153–157, 1947.

C. J. Merz and P. M. Murphy. UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Hu, Larsen, Wilson, and Douglas, editors, *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IX*, pages 41–48, 1999.

E. Osuna. *Support Vector Machines: Training and Applications*. PhD thesis, Massachusetts Institute of Technology, 1998.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136, Los Alamitos, CA, USA, June 1997. IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence, IEEE Computer Society.

J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553. MIT Press, 2000.

T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, September 1990.

J. R. Quinlan. *C4.5: Programs for Empirical Learning*. Morgan Kaufmann, 1993.

R. M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches to Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.

C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5): 1651–1686, 1998.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

I. Schönberg. Spline functions and the problem of graduation. *Proceedings of the National Academy of Science*, pages 947–950, 1964.

T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Journal of Complex Systems*, 1(1):145–168, 1987.

J. A. K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle. Least squares support vector machine classifiers: a large scale algorithm. In *Proceedings of the European Conference on Circuit Theory and Design*, 1999.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999a.

J. A. K. Suykens and J. Vandewalle. Multiclass least squares support vector machines. In *Proceedings of the International Joint Conference on Neural Networks*, 1999b.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. W. H. Winston, 1977.

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

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1990.

G. Wahba and G. Kimeldorf. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis Applications*, 33(1):82–95, 1971.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science, 1998.

I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 1999.

Journal of Machine Learning Research 5 (2004) 143-151 Submitted 1/03; Revised 8/03; Published 2/04

Lossless Online Bayesian Bagging

Herbert K. H. Lee HERBIE@AMS.UCSC.EDU
Department of Applied Math and Statistics
School of Engineering
University of California, Santa Cruz
Santa Cruz, CA 95064, USA

Merlise A. Clyde CLYDE@STAT.DUKE.EDU
Institute of Statistics and Decision Sciences
Box 90251
Duke University
Durham, NC 27708, USA

Editor: Bin Yu

Abstract

Bagging frequently improves the predictive performance of a model. An online version has recently been introduced, which attempts to gain the benefits of an online algorithm while approximating regular bagging. However, regular online bagging is an approximation to its batch counterpart and so is not lossless with respect to the bagging operation. By operating under the Bayesian paradigm, we introduce an online Bayesian version of bagging which is exactly equivalent to the batch Bayesian version, and thus when combined with a lossless learning algorithm gives a completely lossless online bagging algorithm. We also note that the Bayesian formulation resolves a theoretical problem with bagging, produces less variability in its estimates, and can improve predictive performance for smaller data sets.

Keywords: Classification Tree, Bayesian Bootstrap, Dirichlet Distribution

1. Introduction

In a typical prediction problem, there is a trade-off between bias and variance, in that after a certain amount of fitting, any increase in the precision of the fit will cause an increase in the prediction variance on future observations. Similarly, any reduction in the prediction variance causes an increase in the expected bias for future predictions. Breiman (1996) introduced bagging as a method of reducing the prediction variance without affecting the prediction bias. As a result, predictive performance can be significantly improved.

Bagging, short for "Bootstrap AGGregatING", is an ensemble learning method. Instead of making predictions from a single model fit on the observed data, bootstrap samples are taken of the data, the model is fit on each sample, and the predictions are averaged over all of the fitted models to get the bagged prediction. Breiman (1996) explains that bagging works well for unstable modeling procedures, i.e. those for which the conclusions are sensitive to small changes in the data. He also gives a theoretical explanation of how bagging works, demonstrating the reduction in mean-squared prediction error for unstable

procedures. Breiman (1994) demonstrated that tree models, among others, are empirically unstable.

Online bagging (Oza and Russell, 2001) was developed to implement bagging sequentially, as the observations appear, rather than in batch once all of the observations have arrived. It uses an asymptotic approximation to mimic the results of regular batch bagging, and as such it is not a lossless algorithm. Online algorithms have many uses in modern computing. By updating sequentially, the update for a new observation is relatively quick compared to re-fitting the entire database, making real-time calculations more feasible. Such algorithms are also advantageous for extremely large data sets where reading through the data just once is time-consuming, so batch algorithms which would require multiple passes through the data would be infeasible.

In this paper, we consider a Bayesian version of bagging (Clyde and Lee, 2001) based on the Bayesian bootstrap (Rubin, 1981). This overcomes a technical difficulty with the usual bootstrap in bagging. It also leads to a theoretical reduction in variance over the bootstrap for certain classes of estimators, and a significant reduction in observed variance and error rates for smaller data sets. We present an online version which, when combined with a lossless online model-fitting algorithm, continues to be lossless with respect to the bagging operation, in contrast to ordinary online bagging. The Bayesian approach requires the learning base algorithm to accept weighted samples.

In the next section we review the basics of the bagging algorithm, of online bagging, and of Bayesian bagging. Next we introduce our online Bayesian bagging algorithm. We then demonstrate its efficacy with classification trees on a variety of examples.

2. Bagging

In ordinary (batch) bagging, bootstrap re-sampling is used to reduce the variability of an unstable estimator. A particular model or algorithm, such as a classification tree, is specified for learning from a set of data and producing predictions. For a particular data set $\mathbf{X_m}$, denote the vector of predictions (at the observed sites or at new locations) by $\boldsymbol{G(X_m)}$. Denote the observed data by $\boldsymbol{X = (x_1, \ldots, x_n)}$. A bootstrap sample of the data is a sample with replacement, so that $\boldsymbol{X_m = (x_{m_1}, \ldots, x_{m_n})}$, where $m_i \in \{1, \ldots, n\}$ with repetitions allowed. $\mathbf{X_m}$ can also be thought of as a re-weighted version of $\mathbf{X}$, where the weights, $\omega_i^{(m)}$ are drawn from the set $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$, i.e., $n\omega_i^{(m)}$ is the number of times that $\boldsymbol{x_i}$ appears in the mth bootstrap sample. We denote the weighted sample as $\boldsymbol{(X, \omega^{(m)})}$. For each bootstrap sample, the model produces predictions $\boldsymbol{G(X_m)} = G(\mathbf{X_m})_1, \ldots, G(\mathbf{X_m})_P$ where P is the number of prediction sites. M total bootstrap samples are used. The bagged predictor for the jth element is then

$$\frac{1}{M}\sum_{m=1}^{M} G(\mathbf{X_m})_j \;=\; \frac{1}{M}\sum_{m=1}^{M} G(\mathbf{X}, \boldsymbol{\omega^{(m)}})_j,$$

or in the case of classification, the jth element is predicted to belong to the most frequently predicted category by $G(\mathbf{X_1})_j, \ldots, G(\mathbf{X_M})_j$.

A version of pseudocode for implementing bagging is

1. For $m \in \{1, \ldots, M\}$,

(a) Draw a bootstrap sample, $\mathbf{X_m}$, from $\mathbf{X}$.

(b) Find predicted values $\boldsymbol{G(X_m)}$.

2. The bagging predictor is $\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{G(X_m)}$.

Equivalently, the bootstrap sample can be converted to a weighted sample $\boldsymbol{(X, \omega^{(m)})}$ where the weights $\omega_i^{(m)}$ are found by taking the number of times $\boldsymbol{x_i}$ appears in the bootstrap sample and dividing by n. Thus the weights will be drawn from $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$ and will sum to 1. The bagging predictor using the weighted formulation is $\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{G(X_m, \omega^{(m)})}$ for regression, or the plurality vote for classification.

2.1 Online Bagging

Online bagging (Oza and Russell, 2001) was recently introduced as a sequential approximation to batch bagging. In batch bagging, the entire data set is collected, and then bootstrap samples are taken from the whole database. An online algorithm must process observations as they arrive, and thus each observation must be resampled a random number of times when it arrives. The algorithm proposed by Oza and Russell resamples each observation according to a Poisson random variable with mean 1, i.e., $P(K_m = k) = \exp(-1)/k!$, where K_m is the number of resamples in "bootstrap sample" m, $K_m \in \{0, 1, \ldots\}$. Thus as each observation arrives, it is added K_m times to $\mathbf{X_m}$, and then $\boldsymbol{G(X_m)}$ is updated, and this is done for $m \in \{1, \ldots, M\}$.

Pseudocode for online bagging is

For $i \in \{1, \ldots, n\}$,

1. For $m \in \{1, \ldots, M\}$,
   (a) Draw a weight K_m from a *Poisson*(1) random variable and add K_m copies of x_i to $\mathbf{X_m}$.
   (b) Find predicted values $\boldsymbol{G(X_m)}$.
2. The current bagging predictor is $\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{G(X_m)}$.

Ideally, step 1(b) is accomplished with a lossless online update that incorporates the K_m new points without refitting the entire model. We note that n may not be known ahead of time, but the bagging predictor is a valid approximation at each step.

Online bagging is not guaranteed to produce the same results as batch bagging. In particular, it is easy to see that after n points have been observed, there is no guarantee that $\mathbf{X_m}$ will contain exactly n points, as the Poisson weights are not constrained to add up to n like a regular bootstrap sample. While it has been shown (Oza and Russell, 2001) that these samples converge asymptotically to the appropriate bootstrap samples, there may be some discrepancy in practice. Thus while it can be combined with a lossless online learning algorithm (such as for a classification tree), the bagging part of the online ensemble procedure is not lossless.

2.2 Bayesian Bagging

Ordinary bagging is based on the ordinary bootstrap, which can be thought of as replacing the original weights of $\frac{1}{n}$ on each point with weights from the set $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$, with the total of all weights summing to 1. A variation is to replace the ordinary bootstrap with the Bayesian bootstrap (Rubin, 1981). The Bayesian approach treats the vector of weights $\boldsymbol{\omega}$ as unknown parameters and derives a posterior distribution for $\boldsymbol{\omega}$, and hence $\boldsymbol{G(X, \omega)}$. The non-informative prior $\prod_{i=1}^{n} \omega_i^{-1}$, when combined with the multinomial likelihood, leads to a $Dirichlet_n(1, \ldots, 1)$ distribution for the posterior distribution of $\boldsymbol{\omega}$. The full posterior distribution of $\boldsymbol{G(X, \omega)}$ can be estimated by Monte Carlo methods: generate $\boldsymbol{\omega^{(m)}}$ from a $Dirichlet_n(1, \ldots, 1)$ distribution and then calculate $\boldsymbol{G(X, \omega^{(m)})}$ for each sample. The average of $\boldsymbol{G(X, \omega^{(m)})}$ over the M samples corresponds to the Monte Carlo estimate of the posterior mean of $\boldsymbol{G(X, \omega)}$ and can be viewed as a Bayesian analog of bagging (Clyde and Lee, 2001).

In practice, we may only be interested in a point estimate, rather than the full posterior distribution. In this case, the Bayesian bootstrap can be seen as a continuous version of the regular bootstrap. Thus Bayesian bagging can be achieved by generating M Bayesian bootstrap samples, and taking the average or majority vote of the $\boldsymbol{G(X, \omega^{(m)})}$. This is identical to regular bagging except that the weights are continuous-valued on $(0, 1)$, instead of being restricted to the discrete set $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$. In both cases, the weights must sum to 1. In both cases, the expected value of a particular weight is $\frac{1}{n}$ for all weights, and the expected correlation between weights is the same (Rubin, 1981). Thus Bayesian bagging will generally have the same expected point estimates as ordinary bagging. The variability of the estimate is slightly smaller under Bayesian bagging, as the variability of the weights is $\frac{n}{n+1}$ times that of ordinary bagging. As the sample size grows large, this factor becomes arbitrarily close to one, but we do note that it is strictly less than one, so the Bayesian approach does give a further reduction in variance compared to the standard approach. In practice, for smaller data sets, we often find a significant reduction in variance, possibly because the use of continuous-valued weights leads to fewer extreme cases than discrete-valued weights.

Pseudocode for Bayesian bagging is

1. For $m \in \{1, \ldots, M\}$,
   (a) Draw random weights $\boldsymbol{\omega^{(m)}}$ from a $Dirichlet_n(1, \ldots, 1)$ to produce the Bayesian bootstrap sample $\boldsymbol{(X, \omega^{(m)})}$.
   (b) Find predicted values $\boldsymbol{G(X, \omega^{(m)})}$.
2. The bagging predictor is $\frac{1}{M} \sum_{m=1}^{M} \boldsymbol{G(X, \omega^{(m)})}$.

Use of the Bayesian bootstrap does have a major theoretical advantage, in that for some problems, bagging with the regular bootstrap is actually estimating an undefined quantity. To take a simple example, suppose one is bagging the fitted predictions for a point y from a least-squares regression problem. Technically, the full bagging estimate is $\frac{1}{M_0} \sum_m \hat{y}_m$ where m ranges over all possible bootstrap samples, M_0 is the total number of possible bootstrap samples, and $\hat{y}_m$ is the predicted value from the model fit using the mth bootstrap sample. The issue is that one of the possible bootstrap samples contains the

first data point replicated n times, and no other data points. For this bootstrap sample, the regression model is undefined (since at least two different points are required), and so $\hat{y}$ and thus the bagging estimator are undefined. In practice, only a small sample of the possible bootstrap samples is used, so the probability of drawing a bootstrap sample with an undefined prediction is very small. Yet it is disturbing that in some problems, the bagging estimator is technically not well-defined. In contrast, the use of the Bayesian bootstrap completely avoids this problem. Since the weights are continuous-valued, the probability that any weight is exactly equal to zero is zero. Thus with probability one, all weights are strictly positive, and the Bayesian bagging estimator will be well-defined (assuming the ordinary estimator on the original data is well-defined).

We note that the Bayesian approach will only work with models that have learning algorithms that handle weighted samples. Most standard models either have readily available such algorithms, or their algorithms are easily modified to accept weights, so this restriction is not much of an issue in practice.

3. Online Bayesian Bagging

Regular online bagging cannot be exactly equivalent to the batch version because the Poisson counts cannot be guaranteed to sum to the number of actual observations. Gamma random variables can be thought of as continuous analogs of Poisson counts, which motivates our derivation of Bayesian online bagging. The key is to recall a fact from basic probability — a set of independent gamma random variables divided by its sum has a Dirichlet distribution, i.e.,

$$\text{If } w_i \sim \Gamma(\alpha_i, 1), \text{ then } \left(\frac{w_1}{\sum w_i}, \frac{w_2}{\sum w_i}, \ldots, \frac{w_k}{\sum w_i} \right) \sim \mathit{Dirichlet}_n(\alpha_1, \alpha_2, \ldots, \alpha_k) \; .$$

(See for example, Hogg and Craig, 1995, pp. 187–188.) This relationship is a common method for generating random draws from a Dirichlet distribution, and so is also used in the implementation of batch Bayesian bagging in practice.

Thus in the online version of Bayesian bagging, as each observation arrives, it has a realization of a $\mathit{Gamma}(1)$ random variable associated with it for each bootstrap sample, and the model is updated after each new weighted observation. If the implementation of the model requires weights that sum to one, then within each (Bayesian) bootstrap sample, all weights can be re-normalized with the new sum of gammas before the model is updated. At any point in time, the current predictions are those aggregated across all bootstrap samples, just as with batch bagging. If the model is fit with an ordinary lossless online algorithm, as exists for classification trees (Utgoff et al., 1997), then the entire online Bayesian bagging procedure is completely lossless relative to batch Bayesian bagging. Furthermore, since batch Bayesian bagging gives the same mean results as ordinary batch bagging, online Bayesian bagging also has the same expected results as ordinary batch bagging.

Pseudocode for online Bayesian bagging is

For $i \in \{1, \ldots, n\}$,

1. For $m \in \{1, \ldots, M\}$,

(a) Draw a weight $\omega_i^{(m)}$ from a $Gamma(1,1)$ random variable, associate weight with x_i, and add x_i to $\mathbf{X}$.
(b) Find predicted values $\boldsymbol{G(X, \omega^{(m)})}$ (renormalizing weights if necessary).

2. The current bagging predictor is $\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{G(X, \omega^{(m)})}$.

In step 1(b), the weights may need to be renormalized (by dividing by the sum of all current weights) if the implementation requires weights that sum to one. We note that for many models, such as classification trees, this renormalization is not a major issue; for a tree, each split only depends on the relative weights of the observations at that node, so nodes not involving the new observation will have the same ratio of weights before and after renormalization and the rest of the tree structure will be unaffected; in practice, in most implementations of trees (including that used in this paper), renormalization is not necessary. We discuss the possibility of renormalization in order to be consistent with the original presentation of the bootstrap and Bayesian bootstrap, and we note that ordinary online bagging implicitly deals with this issue equivalently.

The computational requirements of Bayesian versus ordinary online bagging are comparable. The procedures are quite similar, with the main difference being that the fitting algorithm must handle non-integer weights for the Bayesian version. For models such as trees, there is no significant additional computational burden for using non-integer weights.

4. Examples

We demonstrate the effectiveness of online Bayesian bagging using classification trees. Our implementation uses the lossless online tree learning algorithms (ITI) of Utgoff et al. (1997) (available at `http://www.cs.umass.edu/~lrn/iti/`). We compared Bayesian bagging to a single tree, ordinary batch bagging, and ordinary online bagging, all three of which were done using the minimum description length criterion (MDL), as implemented in the ITI code, to determine the optimal size for each tree. To implement Bayesian bagging, the code was modified to account for weighted observations.

We use a generalized MDL to determine the optimal tree size at each stage, replacing all counts of observations with the sum of the weights of the observations at that node or leaf with the same response category. Replacing the total count directly with the sum of the weights is justified by looking at the multinomial likelihood when written as an exponential family in canonical form; the weights enter through the dispersion parameter and it is easily seen that the unweighted counts are replaced by the sums of the weights of the observations that go into each count. To be more specific, a decision tree typically operates with a multinomial likelihood,

$$\prod_{\text{leaves } j} \prod_{\text{classes } k} p_{jk}^{n_{jk}},$$

where p_{jk} is the true probability that an observation in leaf j will be in class k, and n_{jk} is the count of data points in leaf j in class k. This is easily re-written as the product over all observations, $\prod_{i=1}^{n} p_i^*$ where if observation i is in leaf j and a member of class k then $p_i^* = p_{jk}$. For simplicity, we consider the case $k = 2$ as the generalization to larger k is straightforward. Now consider a single point, y, which takes values 0 or 1 depending on which class is it a member of. Transforming to the canonical parameterization, let $\theta = \frac{p}{1-p}$,

where p is the true probability that $y = 1$. Writing the likelihood in exponential family form gives $\exp\left\{\left(y\theta + \log\frac{1}{1+\exp\{\theta\}}\right)\Big/ a\right\}$ where a is the dispersion parameter, which would be equal to 1 for a standard data set, but would be the reciprocal of the weight for that observation in a weighted data set. Thus the likelihood for an observation y with weight w is $\exp\left\{\left(y\theta + \log\frac{1}{1+\exp\{\theta\}}\right)\Big/ (1/w)\right\} = p^{wy}(1-p)^{w(1-y)}$ and so returning to the full multinomial, the original counts are simply replaced by the weighted counts. As MDL is a penalized likelihood criterion, we thus use the weighted likelihood and replace each count with a sum of weights. We note that for ordinary online bagging, using a single Poisson weight K with our generalized MDL is exactly equivalent to including K copies of the data point in the data set and using regular MDL.

Table 1 shows the data sets we used for classification problems, the number of classes in each data set, and the sizes of their respective training and test partitions. Table 2 displays the results of our comparison study. All of the data sets, except the final one, are available online at `http://www.ics.uci.edu/~mlearn/MLRepository.html`, the UCI Machine Learning Repository. The last data set is described in Lee (2001). We compare the results of training a single classification tree, ordinary batch bagging, online bagging, and Bayesian online bagging (or equivalently Bayesian batch). For each of the bagging techniques, 100 bootstrap samples were used. For each data set, we repeated 1000 times the following procedure: randomly choose a training/test partition; fit a single tree, a batch bagged tree, an online bagged tree, and a Bayesian bagged tree; compute the misclassification error rate for each fit. Table 2 reports the average error rate for each method on each data set, as well as the estimated standard error of this error rate.

Data Set	Number of Classes	Size of Training Data Set	Size of Test Data Set
Breast cancer (WI)	2	299	400
Contraceptive	3	800	673
Credit (German)	2	200	800
Credit (Japanese)	2	290	400
Dermatology	6	166	200
Glass	7	164	50
House votes	2	185	250
Ionosphere	2	200	151
Iris	3	90	60
Liver	3	145	200
Pima diabetes	2	200	332
SPECT	2	80	187
Wine	3	78	100
Mushrooms	2	1000	7124
Spam	2	2000	2601
Credit (American)	2	4000	4508

Table 1: Sizes of the example data sets

Data Set	Single Tree	Batch Bagging	Online Bagging	Bayesian Online/Batch Bagging
Breast cancer (WI)	0.055 (.020)	0.045 (.010)	0.045 (.010)	0.041 (.009)
Contraceptive	0.522 (.019)	0.499 (.017)	0.497 (.017)	0.490 (.016)
Credit (German)	0.318 (.022)	0.295 (.017)	0.294 (.017)	0.285 (.015)
Credit (Japanese)	0.155 (.017)	0.148 (.014)	0.147 (.014)	0.145 (.014)
Dermatology	0.099 (.033)	0.049 (.017)	0.053 (.021)	0.047 (.019)
Glass	0.383 (.081)	0.357 (.072)	0.361 (.074)	0.373 (.075)
House votes	0.052 (.011)	0.049 (.011)	0.049 (.011)	0.046 (.010)
Ionosphere	0.119 (.026)	0.094 (.022)	0.099 (.022)	0.096 (.021)
Iris	0.062 (.029)	0.057 (.026)	0.060 (.025)	0.058 (.025)
Liver	0.366 (.036)	0.333 (.032)	0.336 (.034)	0.317 (.033)
Pima diabetes	0.265 (.027)	0.250 (.020)	0.247 (.021)	0.232 (.017)
SPECT	0.205 (.029)	0.200 (.030)	0.202 (.031)	0.190 (.027)
Wine	0.134 (.042)	0.094 (.037)	0.101 (.037)	0.085 (.034)
Mushrooms	0.004 (.003)	0.003 (.002)	0.003 (.002)	0.003 (.002)
Spam	0.099 (.008)	0.075 (.005)	0.077 (.005)	0.077 (.005)
Credit (American)	0.350 (.007)	0.306 (.005)	0.306 (.005)	0.305 (.006)

Table 2: Comparison of average classification error rates (with standard error)

We note that in all cases, both online bagging techniques produce results similar to ordinary batch bagging, and all bagging methods significantly improve upon the use of a single tree. However, for smaller data sets (all but the last three), online/batch Bayesian bagging typically both improves prediction performance and decreases prediction variability.

5. Discussion

Bagging is a useful ensemble learning tool, particularly when models sensitive to small changes in the data are used. It is sometimes desirable to be able to use the data in an online fashion. By operating in the Bayesian paradigm, we can introduce an online algorithm that will exactly match its batch Bayesian counterpart. Unlike previous versions of online bagging, the Bayesian approach produces a completely lossless bagging algorithm. It can also lead to increased accuracy and decreased prediction variance for smaller data sets.

Acknowledgments

This research was partially supported by NSF grants DMS 0233710, 9873275, and 9733013. The authors would like to thank two anonymous referees for their helpful suggestions.

References

L. Breiman. Heuristics of instability in model selection. Technical report, University of California at Berkeley, 1994.

L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

M. A. Clyde and H. K. H. Lee. Bagging and the Bayesian bootstrap. In T. Richardson and T. Jaakkola, editors, *Artificial Intelligence and Statistics 2001*, pages 169–174, 2001.

R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*. Prentice-Hall, Upper Saddle River, NJ, 5th edition, 1995.

H. K. H. Lee. Model selection for neural network classification. *Journal of Classification*, 18:227–243, 2001.

N. C. Oza and S. Russell. Online bagging and boosting. In T. Richardson and T. Jaakkola, editors, *Artificial Intelligence and Statistics 2001*, pages 105–112, 2001.

D. B. Rubin. The Bayesian bootstrap. *Annals of Statistics*, 9:130–134, 1981.

P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29:5–44, 1997.

Journal of Machine Learning Research 5 (2004) 153-188 Submitted 12/02; Published 2/04

Subgroup Discovery with CN2-SD

Nada Lavrač NADA.LAVRAC@IJS.SI
Jožef Stefan Institute
Jamova 39
1000 Ljubljana, Slovenia and
Nova Gorica Polytechnic
Vipavska 13
5100 Nova
Gorica, Slovenia

Branko Kavšek BRANKO.KAVSEK@IJS.SI
Jožef Stefan Institute
Jamova 39
1000 Ljubljana, Slovenia

Peter Flach PETER.FLACH@BRISTOL.AC.UK
University of Bristol
Woodland Road
Bristol BS8 1UB, United Kingdom

Ljupčo Todorovski LJUPCO.TODOROVSKI@IJS.SI
Jožef Stefan Institute
Jamova 39
1000 Ljubljana, Slovenia

Editor: Stefan Wrobel

Abstract

This paper investigates how to adapt standard classification rule learning approaches to subgroup discovery. The goal of subgroup discovery is to find rules describing subsets of the population that are sufficiently large and statistically unusual. The paper presents a subgroup discovery algorithm, *CN2-SD*, developed by modifying parts of the CN2 classification rule learner: its covering algorithm, search heuristic, probabilistic classification of instances, and evaluation measures. Experimental evaluation of *CN2-SD* on 23 UCI data sets shows substantial reduction of the number of induced rules, increased rule coverage and rule significance, as well as slight improvements in terms of the area under ROC curve, when compared with the CN2 algorithm. Application of *CN2-SD* to a large traffic accident data set confirms these findings.

Keywords: Rule Learning, Subgroup Discovery, UCI Data Sets, Traffic Accident Data Analysis

1. Introduction

Rule learning is most frequently used in the context of classification rule learning (Michalski et al., 1986, Clark and Niblett, 1989, Cohen, 1995) and association rule learning (Agrawal et al., 1996). While classification rule learning is an approach to *predictive induction* (or supervised learning), aimed at constructing a set of rules to be used for classification and/or prediction, association rule

learning is a form of *descriptive induction* (non-classificatory induction or unsupervised learning), aimed at the discovery of individual rules which define interesting patterns in data.

Descriptive induction has recently gained much attention of the rule learning research community. Besides mining of association rules (e.g., the APRIORI association rule learning algorithm (Agrawal et al., 1996)), other approaches have been developed, including clausal discovery as in the CLAUDIEN system (Raedt and Dehaspe, 1997, Raedt et al., 2001), and database dependency discovery (Flach and Savnik, 1999).

1.1 Subgroup Discovery: A Task at the Intersection of Predictive and Descriptive Induction

This paper shows how classification rule learning can be adapted to *subgroup discovery*, a task at the intersection of predictive and descriptive induction, that has first been formulated by Klösgen (1996) and Wrobel (1997, 2001), and addressed by rule learning algorithms EXPLORA (Klösgen, 1996) and MIDOS (Wrobel, 1997, 2001). In the work of Klösgen (1996) and Wrobel (1997, 2001), the problem of subgroup discovery has been defined as follows: Given a population of individuals and a property of those individuals we are interested in, find population subgroups that are statistically 'most interesting', e.g., are as large as possible and have the most unusual statistical (distributional) characteristics with respect to the property of interest.

In subgroup discovery, rules have the form $Class \leftarrow Cond$, where the property of interest for subgroup discovery is class value *Class* that appears in the rule consequent, and the rule antecedent *Cond* is a conjunction of features (attribute-value pairs) selected from the features describing the training instances. As rules are induced from labeled training instances (labeled positive if the property of interest holds, and negative otherwise), the process of subgroup discovery is targeted at uncovering properties of a selected *target* population of individuals with the given property of interest. In this sense, subgroup discovery is a form of supervised learning. However, in many respects subgroup discovery is a form of descriptive induction as the task is to uncover individual interesting *patterns* in data. The standard assumptions made by classification rule learning algorithms (especially the ones that take the covering approach), such as 'induced rules should be as accurate as possible' or 'induced rules should be as distinct as possible, covering different parts of the population', need to be relaxed. In our approach, the first assumption, implemented in classification rule learners by heuristic which aim at optimizing predictive accuracy, is relaxed by implementing new heuristics for subgroup discovery which aim at finding 'best' subgroups in terms of rule coverage and distributional unusualness. The relaxation of the second assumption enables the discovery of overlapping subgroups, describing some population segments in a multiplicity of ways. Induced subgroup descriptions may be redundant, if viewed from a classifier perspective, but very valuable in terms of their descriptive power, uncovering genuine properties of subpopulations from different viewpoints.

Let us emphasize the difference between subgroup discovery (as a task at the intersection of predictive and descriptive induction) and classification rule learning (as a form of predictive induction). The goal of standard rule learning is to generate models, one for each class, consisting of rule sets describing class characteristics in terms of properties occurring in the descriptions of training examples. In contrast, subgroup discovery aims at discovering individual rules or 'patterns' of interest, which must be represented in explicit symbolic form and which must be relatively simple in order to be recognized as actionable by potential users. Moreover, standard classification rule learning algorithms cannot appropriately address the task of subgroup discovery as they use the covering

algorithm for rule set construction which - as will be seen in this paper - hinders the applicability of classification rule induction approaches in subgroup discovery.

Subgroup discovery is usually seen as different from classification, as it addresses different goals (discovery of interesting population subgroups instead of maximizing classification accuracy of the induced rule set). This is manifested also by the fact that in subgroup discovery one can often tolerate many more false positives (negative examples incorrectly classified as positives) than in a classification task. However, both tasks, subgroup discovery and classification rule learning, can be unified under the umbrella of cost-sensitive classification. This is because when deciding which classifiers are optimal in a given context it does not matter whether we penalize false negatives as is the case in classification, or reward true positives as in subgroup discovery.

1.2 Overview of the *CN2-SD* Approach to Subgroup Discovery

This paper investigates how to adapt standard classification rule learning approaches to subgroup discovery. The proposed modifications of classification rule learners can, in principle, be used to modify any rule learner using the covering algorithm for rule set construction. In this paper, we illustrate the approach by modifying the well-known CN2 rule learning algorithm (Clark and Niblett, 1989, Clark and Boswell, 1991). Alternatively, we could have modified RL (Lee et al., 1998), RIPPER (Cohen, 1995), SLIPPER (Cohen and Singer, 1999) or other more sophisticated classification rule learners. The reason for modifying CN2 is that other more sophisticated learners include advanced techniques that make them more effective in classification tasks, improving their classification accuracy. Improved classification accuracy is, however, not of ultimate interest for subgroup discovery, whose main goal is to find interesting population subgroups.

We have implemented the new subgroup discovery algorithm *CN2-SD* by modifying CN2 (Clark and Niblett, 1989, Clark and Boswell, 1991). The proposed approach performs subgroup discovery through the following modifications of CN2: (a) replacing the accuracy-based search heuristic with a new weighted relative accuracy heuristic that trades off generality and accuracy of the rule, (b) incorporating example weights into the covering algorithm, (c) incorporating example weights into the weighted relative accuracy search heuristic, and (d) using probabilistic classification based on the class distribution of covered examples by individual rules, both in the case of unordered rule sets and ordered decision lists. In addition, we have extended the ROC analysis framework to subgroup discovery and propose a set of measures appropriate for evaluating the quality of induced subgroups.

This paper presents the *CN2-SD* subgroup discovery algorithm, together with its experimental evaluation on 23 data sets of the UCI Repository of Machine Learning Databases (Murphy and Aha, 1994), as well as its application to a real world problem of traffic accident analysis. The experimental comparison with CN2 demonstrates that the subgroup discovery algorithm *CN2-SD* produces substantially smaller rule sets, where individual rules have higher coverage and significance. These three factors are important for subgroup discovery: smaller size enables better understanding, higher coverage means larger support, and higher significance means that rules describe discovered subgroups that have significantly different distributional characteristics compared to the entire population. The appropriateness for subgroup discovery is confirmed also by slight improvements in terms of the area under ROC curve, without decreasing predictive accuracy.

The paper is organized as follows. Section 2 introduces the background of this work which includes the description of the CN2 rule learning algorithm, the weighted relative accuracy heuristic, and probabilistic classification of new examples. Section 3 presents the subgroup discovery algo-

rithm *CN2-SD* by describing the necessary modifications of CN2. In Section 4 we discuss subgroup discovery from the perspective of ROC analysis. Section 5 presents a range of metrics used in the experimental evaluation of *CN2-SD*. Section 6 presents the results of experiments on selected UCI data sets as well as an application of *CN2-SD* on a real-life traffic accident data set. Related work is discussed in Section 7. Section 8 concludes by summarizing the main contributions and proposing directions for further work.

2. Background

This section presents the background of our work: the classical CN2 rule induction algorithm, including the covering algorithm for rule set construction, the standard CN2 heuristic, weighted relative accuracy heuristic, and the probabilistic classification technique used in CN2.

2.1 The CN2 Rule Induction Algorithm

CN2 is an algorithm for inducing propositional classification rules (Clark and Niblett, 1989, Clark and Boswell, 1991). Induced rules have the form "`if` *Cond* `then` *Class*", where *Cond* is a conjunction of features (pairs of attributes and their values) and *Class* is the class value. In this paper we use the notation $Class \leftarrow Cond$.

CN2 consists of two main procedures: the bottom-level search procedure that performs beam search in order to find a single rule, and the top-level control procedure that repeatedly executes the bottom-level search to induce a rule set. The bottom-level performs beam search[1] using classification accuracy of the rule as a heuristic function. The accuracy of a propositional classification rule of the form $Class \leftarrow Cond$ is equal to the conditional probability of class *Class*, given that condition *Cond* is satisfied:

$$Acc(Class \leftarrow Cond) = p(Class|Cond) = \frac{p(Class.Cond)}{p(Cond)}.$$

Usually, this probability is estimated by relative frequency $\frac{n(Class.Cond)}{n(Cond)}$.[2] Different probability estimates, like the Laplace (Clark and Boswell, 1991) or the *m*-estimate (Cestnik, 1990, Džeroski et al., 1993), can be used in CN2 for estimating the above probability. The standard CN2 algorithm used in this work uses the Laplace estimate, which is computed as $\frac{n(Class.Cond)+1}{n(Cond)+k}$, where k is the number of classes (for a two-class problem, $k = 2$).

CN2 can also apply a significance test to an induced rule. A rule is considered to be significant, if it expresses a regularity unlikely to have occurred by chance. To test significance, CN2 uses the likelihood ratio statistic (Clark and Niblett, 1989) that measures the difference between the class probability distribution in the set of training examples covered by the rule and the class probability distribution in the set of all training examples (see Equation 2 in Section 5). The empirical evaluation in the work of Clark and Boswell (1991) shows that applying the significance test reduces the number of induced rules at a cost of slightly decreased predictive accuracy.

1. CN2 constructs rules in a general-to-specific fashion, specializing only the rules in the beam (the best rules) by iteratively adding features to condition *Cond*. This procedure stops when no specialized rule can be added to the beam, because none of the specializations is more accurate than the rules in the beam.
2. Here we use the following notation: $n(Cond)$ stands for the number of instances covered by rule $Class \leftarrow Cond$, $n(Class)$ stands for the number of examples of class *Class*, and $n(Class.Cond)$ stands for the number of correctly classified examples (true positives). We use $p(\ldots)$ for the corresponding probabilities.

Two different top-level control procedures can be used in CN2. The first induces an ordered list of rules and the second an unordered set of rules. Both procedures add a default rule (providing for majority class assignment) as the final rule in the induced rule set. When inducing an ordered list of rules, the search procedure looks for the most accurate rule in the current set of training examples. The rule predicts the most frequent class in the set of covered examples. In order to prevent CN2 finding the same rule again, all the covered examples are removed before a new iteration is started at the top-level. The control procedure invokes a new search, until all the examples are covered or no significant rule can be found. In the unordered case, the control procedure is iterated, inducing rules for each class in turn. For each induced rule, only covered examples belonging to that class are removed, instead of removing all covered examples, like in the ordered case. The negative training examples (i.e., examples that belong to other classes) remain.

2.2 The Weighted Relative Accuracy Heuristic

Weighted relative accuracy (Lavrač et al., 1999, Todorovski et al., 2000) is a variant of rule accuracy that can be applied both in the descriptive and predictive induction framework; in this paper this heuristic is applied for subgroup discovery. Weighted relative accuracy, a reformulation of one of the heuristics used in EXPLORA (Klösgen, 1996) and MIDOS (Wrobel, 1997), is defined as follows:

$$WRAcc(Class \leftarrow Cond) = p(Cond) \cdot (p(Class|Cond) - p(Class)). \quad (1)$$

Like most other heuristics used in subgroup discovery systems, weighted relative accuracy consists of two components, providing a tradeoff between rule *generality* (or the relative size of a subgroup $p(Cond)$) and distributional unusualness or *relative accuracy* (the difference between rule accuracy $p(Class|Cond)$ and default accuracy $p(Class)$). This difference can also be seen as the accuracy gain relative to the fixed rule $Class \leftarrow true$, which predicts that all instances belong to *Class*: a rule is interesting only if it improves upon this 'default' accuracy. Another aspect of relative accuracy is that it measures the difference between true positives and the expected true positives (expected under the assumption of independence of the left and right hand-side of a rule), i.e., the utility of connecting rule body *Cond* with a given rule head *Class*. However, it is easy to obtain high relative accuracy with highly specific rules, i.e., rules with low generality $p(Cond)$. To this end, generality is used as a 'weight', so that weighted relative accuracy trades off generality of the rule ($p(Cond)$, i.e., rule coverage) and relative accuracy ($p(Class|Cond) - p(Class)$).

In the work of Klösgen (1996), these quantities are referred to as g (generality), p (rule accuracy or precision) and p_0 (default rule accuracy) and different tradeoffs between rule generality and precision in the so-called *p-g* (precision-generality) space are proposed. In addition to function $g(p-p_0)$, which is equivalent to our weighted relative accuracy heuristic, other tradeoffs that reduce the influence of generality are proposed, e.g., $\sqrt{g}(p-p_0)$ or $\sqrt{g/(1-g)}(p-p_0)$. Here, we favor the weighted relative accuracy heuristic, because it has an intuitive interpretation in ROC space, discussed in Section 4.

2.3 Probabilistic Classification

The induced rules can be ordered or unordered. Ordered rules are interpreted as a decision list (Rivest, 1987) in a straightforward manner: when classifying a new example, the rules are sequentially tried and the first rule that covers the example is used for prediction.

if	legs = 2	&	feathers = yes	then	class = bird	[13,0]
if	beak = yes			then	class = bird	[20,0]
if	size = large	&	flies = no	then	class = elephant	[2,10]

Table 1: A rule set consisting of two rules for class 'bird' and one rule for class 'elephant'.

In the case of unordered rule sets, the distribution of covered training examples among classes is attached to each rule. Rules of the form:

if *Cond* then *Class* [*ClassDistribution*]

are induced, where numbers in the *ClassDistribution* list denote, for each individual class, how many training examples of this class are covered by the rule. When classifying a new example, all rules are tried and those covering the example are collected. If a clash occurs (several rules with different class predictions cover the example), a voting mechanism is used to obtain the final prediction: the class distributions attached to the rules are summed to determine the most probable class. If no rule fires, the default rule is invoked to predict the majority class of training instances not covered by the other rules in the list.

Probabilistic classification is illustrated on a sample classification task, taken from Clark and Boswell (1991). Suppose we need to classify an animal which is a two-legged, feathered, large, non-flying and has a beak,[3], and the classification is based on a rule set, listed in Table 1 formed of three probabilistic rules with the [bird, elephant] class distribution assigned to each rule (for simplicity, the rule set does not include the default rule). All rules fire for the animal to be classified, resulting in a [35,10] class distribution. As a result, the animal is classified as a bird.

3. The *CN2-SD* Subgroup Discovery Algorithm

The main modifications of the CN2 algorithm, making it appropriate for subgroup discovery, involve the implementation of the weighted covering algorithm, incorporation of example weights into the weighted relative accuracy heuristic, probabilistic classification also in the case of the 'ordered' induction algorithm, and the area under ROC curve rule set evaluation. This section describes the CN2 modifications, while ROC analysis and a novel interpretation of the weighted relative accuracy heuristic in ROC space are given in Section 4.

3.1 Weighted Covering Algorithm

If used for subgroup discovery, one of the problems of standard rule learners, such as CN2 and RIPPER, is the use of the covering algorithm for rule set construction. The main deficiency of the covering algorithm is that only the first few induced rules may be of interest as subgroup descriptions with sufficient coverage and significance. In the subsequent iterations of the covering algorithm, rules are induced from biased example subsets, i.e., subsets including only positive examples that are not covered by previously induced rules, which inappropriately biases the subgroup discovery process.

3. The animal being classified is a weka.

As a remedy to this problem we propose the use of a weighted covering algorithm (Gamberger and Lavrač, 2002), in which the subsequently induced rules (i.e., rules induced in the later stages) also represent interesting and sufficiently large subgroups of the population. The weighted covering algorithm modifies the classical covering algorithm in such a way that covered positive examples are not deleted from the current training set. Instead, in each run of the covering loop, the algorithm stores with each example a count indicating how often (with how many rules) the example has been covered so far. Weights derived from these example counts then appear in the computation of *WRAcc*. Initial weights of all positive examples e_j equal 1, $w(e_j,0)=1$. The initial example weight $w(e_j,0)=1$ means that the example has not been covered by any rule, meaning 'please cover this example, since it has not been covered before', while lower weights, $0<w<1$ mean 'do not try too hard on this example'. Consequently, the examples already covered by one or more constructed rules decrease their weights while the uncovered target class examples whose weights have not been decreased will have a greater chance to be covered in the following iterations of the algorithm.

For a weighted covering algorithm to be used, we have to specify the weighting scheme, i.e., how the weight of each example decreases with the increasing number of covering rules. We have implemented two weighting schemes described below.

3.1.1 MULTIPLICATIVE WEIGHTS

In the first scheme, weights decrease multiplicatively. For a given parameter $0<\gamma<1$, weights of covered positive examples decrease as follows: $w(e_j,i)=\gamma^i$, where $w(e_j,i)$ is the weight of example e_j being covered by i rules. Note that the weighted covering algorithm with $\gamma=1$ would result in finding the same rule over and over again, whereas with $\gamma=0$ the algorithm would perform the same as the standard CN2 covering algorithm.

3.1.2 ADDITIVE WEIGHTS

In the second scheme, weights of covered positive examples decrease according to the formula $w(e_j,i)=\frac{1}{i+1}$. In the first iteration all target class examples contribute the same weight $w(e_j,0)=1$, while in the following iterations the contributions of examples are inversely proportional to their coverage by previously induced rules.

3.2 Modified *WRAcc* Heuristic with Example Weights

The modification of CN2 reported in the work of Todorovski et al. (2000) affected only the heuristic function: weighted relative accuracy was used as a search heuristic, instead of the accuracy heuristic of the original CN2, while everything else remained the same. In this work, the heuristic function is further modified to handle example weights, which provide the means to consider different parts of the example space in each iteration of the weighted covering algorithm.

In the *WRAcc* computation (Equation 1) all probabilities are computed by relative frequencies. An example weight measures how important it is to cover this example in the next iteration. The modified *WRAcc* measure is then defined as follows:

$$WRAcc(Class \leftarrow Cond) = \frac{n'(Cond)}{N'} \cdot \left(\frac{n'(Class.Cond)}{n'(Cond)} - \frac{n'(Class)}{N'}\right).$$

if	legs = 2	&	feathers = yes	then	class = bird	[1, 0]
if	beak = yes			then	class = bird	[1, 0]
if	size = large	&	flies = no	then	class = elephant	[0.17,0.83]

Table 2: The rule set of Table 1 as treated by *CN2-SD*.

In this equation, N' is the sum of the weights of all examples, $n'(Cond)$ is the sum of the weights of all covered examples, and $n'(Class.Cond)$ is the sum of the weights of all correctly covered examples.

To add a rule to the generated rule set, the rule with the maximum *WRAcc* measure is chosen out of those rules in the search space, which are not yet present in the rule set produced so far (all rules in the final rule set are thus distinct, without duplicates).

3.3 Probabilistic Classification

Each CN2 rule returns a class distribution in terms of the number of examples covered, as distributed over classes. The CN2 algorithm uses class distribution in classifying unseen instances only in the case of unordered rule sets, where rules are induced separately for each class. In the case of ordered decision lists, the first rule that fires provides the classification. In our modified *CN2-SD* algorithm, also in the ordered case all applicable rules are taken into account, hence probabilistic classification is used in both classifiers. This means that the terminology 'ordered' and 'unordered', which in CN2 distinguished between decision list and rule set induction, has a different meaning in our setting: the 'unordered' algorithm refers to learning classes one by one, while the 'ordered' algorithm refers to finding best rule conditions and assigning the majority class in the rule head.

Note that *CN2-SD* does not use the same probabilistic classification scheme as CN2. Unlike CN2, where the rule class distribution is computed in terms of the numbers of examples covered, *CN2-SD* treats the class distribution in terms of probabilities (computed by the relative frequency estimate). Table 2 presents the three rules of Table 1 with the class distribution expressed with probabilities. A two-legged, feathered, large, non-flying animal with a beak is again classified as a bird but now the probabilities are averaged (instead of summing the numbers of examples), resulting in the final probability distribution [0.72,0.28]. By using this voting scheme the subgroups covering a small number of examples are not so heavily penalized (as is the case in CN2) when classifying a new example.

3.4 *CN2-SD* Implementation

Two variants of *CN2-SD* have been implemented. The *CN2-SD* subgroup discovery algorithm used in the experiments in this paper is implemented in C and runs on a number of UNIX platforms. Its predecessor, used in the experiments reported by Lavrač et al. (2002), is implemented in Java and incorporated in the WEKA data mining environment (Witten and Frank, 1999). The C implementation is more efficient and less restrictive than the Java implementation, which is limited to binary class problems and to discrete attributes.

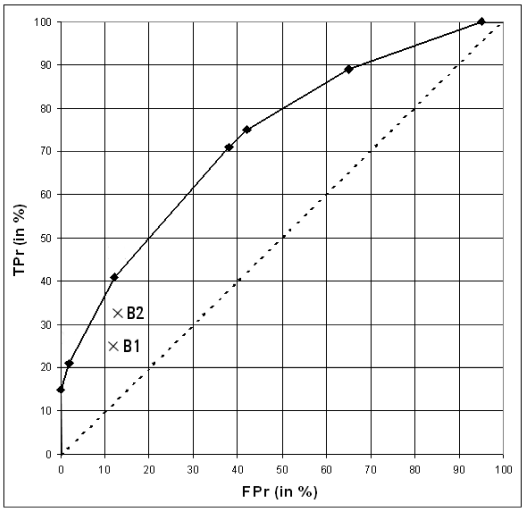


Figure 1: The ROC space with TPr on the X axis and FPr on the Y axis. The solid line connecting seven optimal subgroups (marked by ⧫) is the ROC convex hull. $B1$ and $B2$ denote suboptimal subgroups (marked by x). The dotted line – the diagonal connecting points (0,0) and (100,100) – indicates positions of insignificant rules with zero relative accuracy.

4. ROC Analysis for Subgroup Discovery

In this section we describe how ROC (Receiver Operating Characteristic) analysis (Provost and Fawcett, 2001) can be used to understand subgroup discovery and to visualize and evaluate discovered subgroups.

A point in *ROC space* shows classifier performance in terms of false alarm or *false positive rate* $FPr = \frac{FP}{TN+FP}$ (plotted on the X-axis), and sensitivity or *true positive rate* $TPr = \frac{TP}{TP+FN}$ (plotted on the Y-axis). In terms of the expressions introduced in Sections 2.1 and 2.2, TP (true positives), FP (false positives), TN (true negatives) and FN (false negatives) can be expressed as: $TP = n(Class.Cond)$, $FP = n(\overline{Class}.Cond)$, $TN = n(\overline{Class}.\overline{Cond})$ and $FN = n(Class.\overline{Cond})$, where $\overline{Class}$ and $\overline{Cond}$ stand for $\neg Class$ and $\neg Cond$, respectively.

The ROC space is appropriate for measuring the success of subgroup discovery, since rules/subgroups whose TPr/FPr tradeoff is close to the diagonal can be discarded as insignificant. Conversely, significant rules/subgroups are those sufficiently distant from the diagonal. Significant rules define the points in ROC space from which a convex hull can be constructed. The best rules define the ROC convex hull. Figure 1 shows seven rules on the convex hull (marked by ⧫), while two rules $B1$ and $B2$ below the convex hull (marked by x) are of lower quality.

4.1 The Interpretation of Weighted Relative Accuracy in ROC Space

Weighted relative accuracy is appropriate for measuring the quality of a single subgroup, because it is proportional to the distance from the diagonal in ROC space.[4] To see that this holds, note first that rule accuracy $p(Class|Cond)$ is proportional to the angle between the X-axis and the line connecting the origin with the point depicting the rule in terms of its TPr/FPr tradeoff in ROC space. So, for instance, the X-axis has always rule accuracy 0 (these are purely negative subgroups), the Y-axis has always rule accuracy 1 (purely positive subgroups), and the diagonal represents subgroups with rule accuracy $p(Class)$, the prior probability of the positive class. Consequently, all point on the diagonal represent insignificant subgroups.

Relative accuracy, $p(Class|Cond) - p(Class)$, re-normalizes this such that all points on the diagonal have relative accuracy 0, all points on the Y-axis have relative accuracy $1 - p(Class) = p(\overline{Class})$ (the prior probability of the negative class), and all points on the X-axis have relative accuracy $-p(Class)$. Notice that all points on the diagonal also have $WRAcc = 0$. In terms of subgroup discovery, the diagonal represents all subgroups with the same target distribution as the whole population; only the generality of these 'average' subgroups increases when moving from left to right along the diagonal. This interpretation is slightly different in classifier learning, where the diagonal represents random classifiers that can be constructed without any training.

More generally, *WRAcc* isometrics lie on straight lines parallel to the diagonal (Flach, 2003, Fürnkranz and Flach, 2003). Consequently, a point on the line $TPr = FPr + a$, where a is the vertical distance of the line to the diagonal, has $WRAcc = a.p(Class)p(\overline{Class})$. Thus, given a fixed class distribution, *WRAcc* is proportional to the vertical distance a to the diagonal. In fact, the quantity $TPr - FPr$ would be an alternative quality measure for subgroups, with the additional advantage that it allows for comparison of subgroups from populations with different class distributions.

4.2 Methods for Constructing ROC Curves and AUC Evaluation

Subgroups obtained by CN2-SD can be evaluated in ROC space in two different ways.

4.2.1 AUC-METHOD-1

The first method treats each rule as a separate subgroup which is plotted in ROC space in terms of its true and false positive rates (TPr and FPr). We then generate the convex hull of this set of points, selecting the subgroups which perform optimally under a particular range of operating characteristics. The area under this ROC convex hull (*AUC*) indicates the combined quality of the optimal subgroups, in the sense that it does evaluate whether a particular subgroup has anything to add in the context of all the other subgroups. However, this method does not take account of any overlap between subgroups, and subgroups not on the convex hull are simply ignored.

Figure 2 presents two ROC curves, showing the performance of CN2 and *CN2-SD* algorithms on the Australian UCI data set.

4.2.2 AUC-METHOD-2

The second method employs the combined probabilistic classifications of all subgroups, as indicated below. If we always choose the most likely predicted class, this corresponds to setting a fixed threshold 0.5 on the positive probability (the probability of the target class): if the positive probabil-

4. Some of the reasoning supporting this claim is further discussed in the last two paragraphs of Section 5.1.

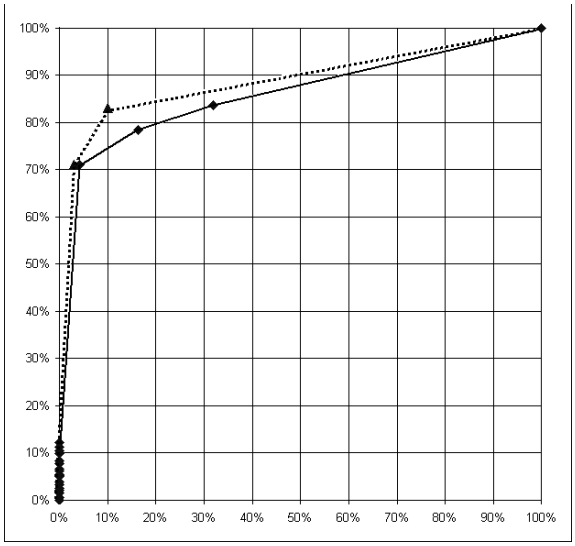


Figure 2: Example ROC curves (AUC-Method-1) on the Australian UCI data set: the solid curve for the standard CN2 classification rule learner, and the dotted curve for *CN2-SD*.

ity is larger than this threshold we predict positive, else negative. The ROC curve can be constructed by varying this threshold from 1 (all predictions negative, corresponding to (0,0) in ROC space) to 0 (all predictions positive, corresponding to (1,1) in ROC space). This results in $n+1$ points in ROC space, where n is the total number of classified examples (test instances). Equivalently, we can order all the classified examples by decreasing predicted probability of being positive, and tracing the ROC curve by starting in (0,0), stepping up when the example is actually positive and stepping to the right when it is negative, until we reach (1,1).[5] Each point on this curve corresponds to a classifier defined by a possible probability threshold, as opposed to AUC-Method-1, where a point on the ROC curve corresponds to one of the optimal subgroups. The ROC curve depicts a set of classifiers, whereas the area under this ROC curve indicates the combined quality of all subgroups (i.e., the quality of the entire rule set). This method can be used with a test set or in cross-validation, but the resulting curve is not necessarily convex.[6]

Figure 3 presents two ROC curves, showing the performance of the CN2 and *CN2-SD* algorithms on the Australian UCI data set. It is apparent from this figure that CN2 is badly overfitting on this data set, because almost all of its ROC curve is below the diagonal. This is because CN2 has learned many overly specific rules, which bias the predicted probabilities. These overly specific rules are visible in Figure 2 as points close to the origin.

5. In the case of ties, we make the appropriate number of steps up and to the right at once, drawing a diagonal line segment.
6. A description of this method applied to decision tree induction can be found in the paper by Ferri-Ramírez et al. (2002).

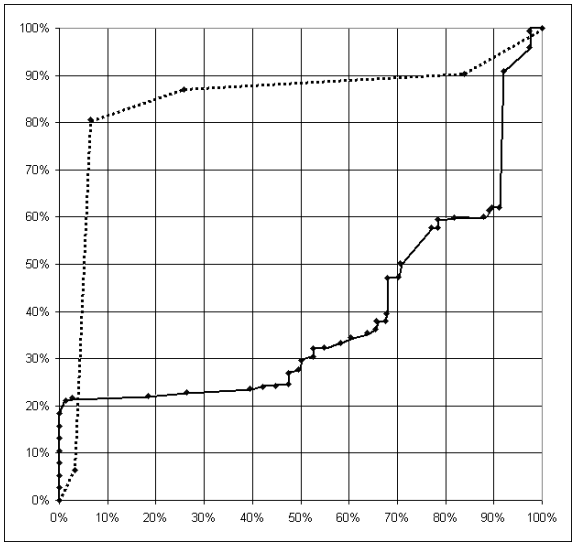


Figure 3: Example ROC curves (AUC-Method-2) on the Australian UCI data set: the solid curve for the standard CN2 classification rule learner, and the dotted curve for *CN2-SD*.

4.2.3 COMPARISON OF THE TWO AUC METHODS

Which of the two methods is more appropriate for subgroup discovery is open for debate. The second method seems more appropriate if the discovered subgroups are intended to be applied also in the predictive setting, as a rule set (a model) used for classification. Its advantage is also that it is easier to apply cross-validation. In the experimental evaluation in Section 6 we use AUC-Method-2 in the comparison of the predictive performance of rule learners.

An argument in favor of using AUC-Method-1 for subgroup evaluation is based on the observation that AUC-Method-1 suggests to eliminate, from the induced set of subgroup descriptions, those rules which are not on the ROC convex hull. This seems appropriate, as the 'best' subgroups according to the *WRAcc* evaluation measure, are subgroups most distant from the ROC diagonal. However, disjoint subgroups, either on or close to the convex hull, should not be eliminated, as (due to disjoint coverage and possibly different symbolic descriptions) they may represent interesting subgroups, regardless of the fact that there is another 'better' subgroup on the ROC convex hull, with a similar TPr/FPr tradeoff.

Notice that the area under ROC curve (AUC-Method-1) cannot be used as a predictive quality measure when comparing different subgroup miners, because it does not take into account the overlapping structure of subgroups. An argument against the use of this measure is here elaborated through a simple example.[7] Consider for instance two subgroup mining results, of say 3 subgroups in each resulting rule set. The first result set consists of three disjoint subgroups of equal size that together cover all the examples of the selected *Class* value and have a 100% accuracy. Thus these three subgroups are a perfect classifier for the *Class* value. In ROC space the three subgroups collapse at the point (0,1/3). The second result set consists of three equal subgroups (having a max-

7. We are grateful to the anonymous reviewer who provided this illustrative example.

imum overlap: with different descriptions, but equal extensions), also with a 100% accuracy and covering one third of the class examples. Clearly the first result is better, but the representation of the results in ROC space (and the area under ROC curve) is the same for both cases.

5. Subgroup Evaluation Measures

In this section we distinguish between *predictive* and *descriptive* evaluation measures, which is in-line with the distinction of predictive induction and descriptive induction made in Section 1. Descriptive measures are used to evaluate the quality of individual rules (individual patterns). These quality measures are the most appropriate for subgroup discovery, as the task of subgroup discovery is to induce individual patterns of interest. Predictive measures are used in addition to descriptive measures just to show that the *CN2-SD* subgroup discovery mechanisms perform well also in the predictive induction setting, where the goal is to induce a classifier.

5.1 Descriptive Measures of Rule Interestingness

Descriptive measures of rule interestingness evaluate each individual subgroup and are thus appropriate for evaluating the success of subgroup discovery. The proposed quality measures compute the average over the induced set of subgroup descriptions, which enables the comparison between different algorithms.

Coverage. The average coverage measures the percentage of examples covered on average by one rule of the induced rule set. Coverage of a single rule R_i is defined as

$$Cov(R_i) = Cov(Class \leftarrow Cond_i) = p(Cond_i) = \frac{n(Cond_i)}{N}.$$

The average coverage of a rule set is computed as

$$COV = \frac{1}{n_R} \sum_{i=1}^{n_R} Cov(R_i),$$

where n_R is the number of induced rules.

Support. For subgroup discovery it is interesting to compute the overall support (the target coverage) as the percentage of target examples (positives) covered by the rules, computed as the true positive rate for the union of subgroups. Support of a rule is defined as the frequency of correctly classified covered examples:

$$Sup(R_i) = Sup(Class \leftarrow Cond_i) = p(Class.Cond_i) = \frac{n(Class.Cond_i)}{N}.$$

The overall support of a rule set is computed as

$$SUP = \frac{1}{N} \sum_{Class_j} n(Class_j \cdot \bigvee_{Class_j \leftarrow Cond_i} Cond_i),$$

where the examples covered by several rules are counted only once (hence the disjunction of rule conditions of rules with the same $Class_j$ value in the rule head).

Size. Size is a measure of complexity (the syntactical complexity of induced rules). The rule set size is computed as the number of rules in the induced rule set (including the default rule):

$$SIZE = n_R.$$

In addition to rule set size used in this paper, complexity could be measured also by the average number of rules/subgroups per class, and the average number of features per rule.

Significance. Average rule significance is computed in terms of the likelihood ratio of a rule, normalized with the likelihood ratio of the significance threshold (99%); the average is computed over all the rules. Significance (or *evidence*, in the terminology of Klösgen, 1996) indicates how significant is a finding, if measured by this statistical criterion. In the CN2 algorithm (Clark and Niblett, 1989), significance is measured in terms of the likelihood ratio statistic of a rule as follows:

$$Sig(R_i) = Sig(Class \leftarrow Cond_i) = 2 \cdot \sum_j n(Class_j.Cond_i) \cdot \log \frac{n(Class_j.Cond_i)}{n(Class_j) \cdot p(Cond_i)} \qquad (2)$$

where for each class $Class_j$, $n(Class_j.Cond_i)$ denotes the number of instances of $Class_j$ in the set where the rule body holds true, $n(Class_j)$ is the number of $Class_j$ instances, and $p(Cond_i)$ (i.e., rule coverage computed as $\frac{n(Cond_i)}{N}$) plays the role of a normalizing factor. Note that although for each generated subgroup description one class is selected as the target class, the significance criterion measures the distributional unusualness unbiased to any particular class – as such, it measures the significance of rule condition only.

The average significance of a rule set is computed as:

$$SIG = \frac{1}{n_R} \sum_{i=1}^{n_R} Sig(R_i).$$

Unusualness. Average rule unusualness is computed as the average *WRAcc* computed over all the rules:

$$WRACC = \frac{1}{n_R} \sum_{i=1}^{n_R} WRAcc(R_i).$$

As discussed in Section 4.1, *WRAcc* is appropriate for measuring the unusualness of separate subgroups, because it is proportional to the vertical distance from the diagonal in the ROC space (see the underlying reasoning in Section 4.1).

As *WRAcc* is proportional to the distance to the diagonal in ROC space, *WRAcc* also reflects rule significance – the larger *WRAcc*, the more significant the rule, and vice versa. As both *WRAcc* and rule significance measure the distributional unusualness of a subgroup, they are the most important quality measures for subgroup discovery. However, while significance only measures distributional unusualness, *WRAcc* takes also rule coverage into account, therefore we consider *unusualness* – computed by the average *WRAcc* – to be the most appropriate measure for subgroup quality evaluation.

As pointed out in Section 4.1, the quantity $TPr - FPr$ could be an alternative quality measure for subgroups, with the additional advantage that we can use it to compare subgroups from populations with different class distributions. However, in this paper we are only concerned with comparing subgroups from the same population, and we prefer *WRAcc* because of its '*p-g*' (precision-generality) interpretation, which is particularly suitable for subgroup discovery.

5.2 Predictive Measures of Rule Set Classification Performance

Predictive measures evaluate a rule set, interpreting a set of subgroup descriptions as a predictive model. Despite the fact that optimizing accuracy is not the intended goal of subgroup discovery algorithms, these measures can be used in order to provide a comparison of *CN2-SD* with standard classification rule learners.

Predictive accuracy. The percentage of correctly predicted instances. For a binary classification problem, rule set accuracy is computed as follows:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN}.$$

Note that ACC measures the accuracy of the whole rule set on both positive and negative examples, while rule accuracy (defined as $Acc(Class \leftarrow Cond) = p(Class|Cond)$) measures the accuracy of a single rule on positives only.

Area under ROC curve. The AUC-Method-2, described in Section 4.2, applicable to rule sets is selected as the evaluation measure. It interprets a rule set as a probabilistic model, given all the different probability thresholds as defined through the probabilistic classification of test instances.

6. Experimental Evaluation

For subgroup discovery, expert evaluation of results is of ultimate interest. Nevertheless, before applying the proposed approach to a particular problem of interest, we wanted to verify our claims that the mechanisms implemented in the *CN2-SD* algorithm are indeed appropriate for subgroup discovery. For this purpose we tested it on selected UCI data sets. In this paper we use the same data sets as in the work of Todorovski et al. (2000). We have applied *CN2-SD* also to a real life problem of traffic accident analysis; these results were evaluated also by the expert.

6.1 The Experimental Setting

To test the applicability of *CN2-SD* to the subgroup discovery task, we compare its performance with the performance of the standard CN2 classification rule learning algorithm (referred to as *CN2-standard*, and described in the work of Clark and Boswell, 1991) as well as with the CN2 algorithm using *WRAcc* (*CN2-WRAcc*, described by Todorovski et al., 2000).

In this comparative study all the parameters of the CN2 algorithm are set to their default values (beam-size = 5, significance-threshold = 99%). The results of the *CN2-SD* algorithm are computed using both multiplicative weights (with γ = 0.5, 0.7, 0.9)[8] and additive weights.

We estimate the performance of the algorithms using stratified 10-fold cross-validation. The obtained estimates are presented in terms of their average values and standard deviations.

Statistical significance of the difference in performance compared to *CN2-standard* is tested using the paired t-test (exactly the same folds are used in all comparisons) with significance level of 95%: bold font and $\uparrow$ to the right of a result in all the tables means that the algorithm is significantly better than *CN2-standard* while $\downarrow$ means it is significantly worse. The same paired t-test is used to compare the different versions of our algorithm with *CN2-standard* over all the data sets.

8. Results obtained with γ = 0.7 are presented in the tables of Appendix A but not in the main part of the paper.

	Data set	#Att.	#D.att.	#C.att.	#Class	#Ex.	Maj. Class (%)
1	australian	14	8	6	2	690	56
2	breast-w	9	9	0	2	699	66
3	bridges-td	7	4	3	2	102	85
4	chess	36	36	0	2	3196	52
5	diabetes	8	0	8	2	768	65
6	echo	6	1	5	2	131	67
7	german	20	13	7	2	1000	70
8	heart	13	6	7	2	270	56
9	hepatitis	19	13	6	2	155	79
10	hypothyroid	25	18	7	2	3163	95
11	ionosphere	34	0	34	2	351	64
12	iris	4	0	4	3	150	33
13	mutagen	59	57	2	2	188	66
14	mutagen-f	57	57	0	2	188	66
15	tic-tac-toe	9	9	0	2	958	65
16	vote	16	16	0	2	435	61
17	balance	4	0	4	3	625	46
18	car	6	6	0	4	1728	70
19	glass	9	0	9	6	214	36
20	image	19	0	19	7	2310	14
21	soya	35	35	0	19	683	13
22	waveform	21	0	21	3	5000	34
23	wine	13	0	13	3	178	40

Table 3: Properties of the UCI data sets.

6.2 Experiments on UCI Data Sets

We experimentally evaluate our approach on 23 data sets from the UCI Repository of Machine Learning Databases (Murphy and Aha, 1994). Table 3 gives an overview of the selected data sets in terms of the number of attributes (total, discrete, continuous), the number of classes, the number of examples, and the percentage of examples of the majority class. These data sets have been widely used in other comparative studies (Todorovski et al., 2000). We have divided the data sets in two groups (Table 3), those with two classes (binary data sets 1–16) and those with more then two classes (multi-class data sets 17–23). This distinction is made as ROC analysis is applied only on binary data sets.[9]

6.2.1 RESULTS OF THE UNORDERED *CN2-SD*

Tables 4 and 5 present summary results of the UCI experiments, while details can be found in Tables 14–20 in Appendix A. For each performance measure, the summary table shows the average value over all the data sets, the significance of the results compared to *CN2-standard* (*p*-value), win/loss/draw in terms of the number of data sets in which the results are better/worse/equal compared with *CN2-standard*, as well as the number of significant wins and losses.

9. This is a simplification (as multi-class AUC could also be computed as the average of AUCs computed by comparing all pairs of classes (Hand and Till, 2001)) that still provides sufficient evidence to support the claims of this paper.

Performance Measure	Data Sets	**CN2 standard**	**CN2 WRAcc**	**CN2-SD ($\gamma = 0.5$)**	**CN2-SD ($\gamma = 0.9$)**	**CN2-SD (add.)**	**Detailed Results**
Coverage (COV)	23	0.131 ± 0.14	0.311 ± 0.17	0.403 ± 0.23	0.450 ± 0.26	**0.486** ± 0.30	Table 14
• significance – p value			0.000	0.000	0.000	**0.000**	
• win/loss/draw			22/1/0	22/1/0	23/0/0	**22/1/0**	
• sig.win/sig.loss			21/1	22/0	22/0	**21/1**	
Support (SUP)	23	0.84 ± 0.03	0.85 ± 0.03	0.90 ± 0.06	**0.92** ± 0.06	0.91 ± 0.06	Table 15
• significance – p value			0.637	0.000	**0.000**	0.001	
• win/loss/draw			13/10/0	18/5/0	**20/3/0**	16/7/0	
• sig.win/sig.loss			5/4	13/1	**18/0**	13/1	
Size (SIZE)	23	18.18 ± 21.77	**6.15** ± 4.49	6.25 ± 4.42	6.49 ± 4.57	6.35 ± 4.58	Table 16
• significance – p value			**0.006**	0.007	0.007	0.007	
• win/loss/draw			**22/1/0**	22/1/0	20/3/0	23/0/0	
• sig.win/sig.loss			**22/0**	21/0	19/2	18/0	
Significance (SIG)	23	2.11 ± 0.46	8.97 ± 4.66	15.57 ± 6.05	16.92 ± 8.90	**18.47** ± 9.00	Table 17
• significance – p value			0.000	0.000	0.000	**0.000**	
• win/loss/draw			22/1/0	23/0/0	22/1/0	**23/0/0**	
• sig.win/sig.loss			21/0	23/0	21/0	**23/0**	
Unusualness (WRACC)	23	0.017 ± 0.02	0.056 ± 0.05	0.079 ± 0.06	0.088 ± 0.06	**0.092** ± 0.07	Table 18
• significance – p value			0.001	0.000	0.000	**0.000**	
• win/loss/draw			20/1/2	22/1/0	22/1/0	**22/1/0**	
• sig.win/sig.loss			19/1	21/1	21/1	**21/1**	

Table 4: Summary of the experimental results on the UCI data sets (descriptive evaluation measures) for different variants of the unordered algorithm using 10-fold stratified cross-validation. The best results are shown in boldface.

The analysis shows that if multiplicative weights are used, most results improve with the increased value of the γ parameter. As in most cases the best *CN2-SD* variants are *CN2-SD* with $\gamma = 0.9$ and with additive weights, and as using additive weighs is the simpler method, the additive weights setting is recommended as default for experimental use.

The summary of results in terms of descriptive measures of interestingness is as follows.

- In terms of the average coverage per rule *CN2-SD* produces rules with significantly higher coverage (the higher the coverage the better the rule) than both *CN2-WRAcc* and *CN2-standard*. The coverage is increased by increasing the γ parameter and the best results are achieved by $\gamma = 0.9$ and by additive weights.

- *CN2-SD* induces rule sets with significantly larger overall support than *CN2-standard* meaning that it covers a higher percentage of target examples (positives) thus leaving a smaller number of examples unclassified.[10]

- *CN2-WRAcc* and *CN2-SD* induce rule sets that are significantly smaller than *CN2-standard* (smaller rule sets are better), while rule sets of *CN2-WRAcc* and *CN2-SD* are comparable, despite the fact that *CN2-SD* uses weights to 'recycle' examples and thus produces overlapping rules.

10. CN2 handles the unclassified examples by classifying them using the default rule – the rule predicting the majority class.

Performance Measure	Data Sets	**CN2 standard**	**CN2 WRAcc**	**CN2-SD ($\gamma = 0.5$)**	**CN2-SD ($\gamma = 0.9$)**	**CN2-SD (add.)**	**Detailed Results**
Accuracy (ACC)	23	**81.61** $\pm$ 11.66	78.12 $\pm$ 16.28	80.92 $\pm$ 16.04	81.07 $\pm$ 15.78	79.36 $\pm$ 16.24	Table 19
• significance – *p* value			0.150	0.771	0.818	0.344	
• win/loss/draw			10/12/1	17/6/0	19/4/0	15/8/0	
• sig.win/sig.loss			3/5	9/4	10/4	7/4	
AUC-Method-2 (AUC)	16	82.16 $\pm$ 16.81	84.37 $\pm$ 9.87	**86.75** $\pm$ 8.95	86.39 $\pm$ 10.32	86.33 $\pm$ 8.60	Table 20
• significance – *p* value			0.563	**0.175**	0.236	0.236	
• win/loss/draw			6/9/1	**10/6/0**	9/7/0	10/6/0	
• sig.win/sig.loss			5/5	**6/2**	7/4	6/3	

Table 5: Summary of the experimental results on the UCI data sets (predictive evaluation measures) for different variants of the unordered algorithm using 10-fold stratified cross-validation. The best results are shown in boldface.

- *CN2-SD* induces significantly better rules in terms of rule significance (rules with higher significance are better) computed by the average likelihood ratio: while the ratios achieved by *CN2-standard* are already significant at the 99% level, this is further pushed up by *CN2-SD* with maximum values achieved by additive weights. An interesting question, to be verified in further experiments, is whether the weighted versions of the CN2 algorithm improve the significance of the induced subgroups also in the case when CN2 rules are induced without applying the significance test.

- In terms of rule unusualness which is of ultimate interest to the subgroup discovery task, *CN2-SD* produces rules with significantly higher average weighted relative accuracy than *CN2-standard*. Like in the case of average coverage per rule the unusualness is increased by increasing the γ parameter and the best results are achieved by $\gamma = 0.9$ and by additive weights. Note that the unusualness of a rule, computed by its *WRAcc*, is a combination of rule accuracy, coverage and prior probability of the target class.

In terms of predictive measures of classification performance results can be summarized as follows.

- *CN2-SD* improves the accuracy in comparison with *CN2-WRAcc* and performs comparable to *CN2-standard* (the difference is insignificant). Notice however that while optimizing predictive accuracy is the ultimate goal of CN2, for *CN2-SD* the goal is to optimize the coverage/relative-accuracy tradeoff.

- In the computation of area under ROC curve (AUC-Method-2) due to the restriction of this method to binary class data sets, only 16 binary data sets are used in the comparisons. Notice that *CN2-SD* improves the area under ROC curve compared to *CN2-WRAcc* and compared to *CN2-standard*, but the differences are not significant. The area under ROC curve however seems not to be affected by the parameter γ or by the weighting approach of *CN2-SD*.

AUC performance is also illustrated by means of the results on the Australian UCI data set in Figures 2 and 3 of Section 4.2. The solid lines in these graphs indicate ROC curves obtained by *CN2-standard* while the dotted lines represent ROC curves for *CN2-SD* with additive weights.

6.2.2 RESULTS OF THE ORDERED *CN2-SD*

For completeness, the results for different versions of the ordered algorithm are summarized in Tables 6 and 7, without giving the results for individual data sets in Appendix A. In our view, the unordered *CN2-SD* algorithm is more appropriate for subgroup discovery than the ordered variant, as it induces a set of rules for each target class in turn.

Performance Measure	Data Sets	**CN2 standard**	**CN2 WRAcc**	**CN2-SD ($\gamma = 0.5$)**	**CN2-SD ($\gamma = 0.9$)**	**CN2-SD (add.)**
Coverage (COV)	23	0.174 ± 0.18	0.351 ± 0.18	0.439 ± 0.25	0.420 ± 0.23	**0.527** ± 0.32
• significance – p value			0.000	0.000	0.000	**0.000**
• win/loss/draw			21/2/0	23/0/0	23/0/0	**22/1/0**
• sig.win/sig.loss			20/1	22/0	22/0	**22/1**
Support (SUP)	23	0.85 ± 0.03	0.85 ± 0.03	0.87 ± 0.05	**0.91** ± 0.05	0.90 ± 0.06
• significance – p value			0.694	0.026	**0.000**	0.000
• win/loss/draw			12/11/0	14/9/0	**18/5/0**	19/4/0
• sig.win/sig.loss			4/4	11/2	**16/1**	14/0
Size (SIZE)	23	17.87 ± 28.10	**4.13** ± 2.73	4.30 ± 2.58	4.61 ± 2.64	4.27 ± 2.79
• significance – p value			**0.025**	0.026	0.030	0.025
• win/loss/draw			**21/1/1**	21/2/0	20/3/0	21/2/0
• sig.win/sig.loss			**21/0**	20/1	19/1	20/0
Significance (SIG)	23	1.87 ± 0.47	8.86 ± 4.81	12.70 ± 7.11	14.80 ± 8.31	**18.11** ± 9.84
• significance – p value			0.000	0.000	0.000	**0.000**
• win/loss/draw			22/1/0	22/1/0	23/0/0	**22/1/0**
• sig.win/sig.loss			22/0	22/0	22/0	**21/0**
Unusualness (WRACC)	23	0.024 ± 0.02	0.060 ± 0.05	0.080 ± 0.06	0.082 ± 0.06	**0.100** ± 0.07
• significance – p value			0.001	0.000	0.000	**0.000**
• win/loss/draw			18/5/0	21/2/0	21/2/0	**22/1/0**
• sig.win/sig.loss			17/2	20/1	20/2	**21/1**

Table 6: Summary of the experimental results on the UCI data sets (descriptive evaluation measures) for different variants of the ordered algorithm using 10-fold stratified cross-validation. The best results are shown in boldface.

6.3 Experiments in Traffic Accident Data Analysis

We have evaluated the *CN2-SD* algorithm also on a traffic accident data set. This is a large real-world database (1.5 GB) containing 21 years of police traffic accident reports (1979–1999). The analysis of this database is not straightforward because of the volume of the data, the amounts of noise and missing data, and the fact that there is no clearly defined data mining target. As described below, some preprocessing was needed before running the subgroup discovery experiments. Results of experiments were shown to the domain expert whose comments are included.

6.3.1 THE TRAFFIC ACCIDENT DATA SET

The traffic accident database contains data about traffic accidents and the vehicles and casualties involved. The data is organized in three linked tables: the ACCIDENT table, the VEHICLE table and the CASUALTY table. The ACCIDENT table consists of the records of all accidents that

Performance Measure	Data Sets	**CN2 standard**	**CN2 WRAcc**	**CN2-SD ($\gamma = 0.5$)**	**CN2-SD ($\gamma = 0.9$)**	**CN2-SD (add.)**
Accuracy (ACC)	23	**83.00** $\pm$ 10.30	78.34 $\pm$ 16.52	79.50 $\pm$ 16.68	81.10 $\pm$ 16.53	80.79 $\pm$ 16.61
• significance – p value			0.155	0.286	0.556	0.494
• win/loss/draw			8/15/0	14/9/0	15/8/0	15/8/0
• sig.win/sig.loss			3/6	15/4	8/4	7/3
AUC-Method-2 (AUC)	16	81.89 $\pm$ 10.07	82.28 $\pm$ 10.11	84.37 $\pm$ 9.19	**84.70** $\pm$ 8.53	83.79 $\pm$ 9.64
• significance – p value			0.721	0.026	**0.005**	0.049
• win/loss/draw			9/6/1	10/6/0	**12/4/0**	10/6/0
• sig.win/sig.loss			6/5	6/3	**8/4**	6/4

Table 7: Summary of the experimental results on the UCI data sets (predictive evaluation measures) for different variants of the ordered algorithm using 10-fold stratified cross-validation. The best results are shown in boldface.

happened over the given period of time (1979–1999), the VEHICLE table contains data about the vehicles involved in those accidents, and the CASUALTY table contains data about the casualties involved in the accidents. Consider the following example: "Two vehicles crashed in a traffic accident and three people were seriously injured in the crash". In terms of the traffic data set this is recorded as one record in the ACCIDENT table, two records in the VEHICLE table and three records in the CASUALTY table. The three tables are described in more detail below.

- The ACCIDENT table contains one record for each accident. The 30 attributes describing an accident can be divided in three groups: date and time of the accident, description of the road where the accident has occurred, and conditions under which the accident has occurred (such as weather conditions, light and junction details). In the ACCIDENT table there are more than 5 million records.

- The VEHICLE table contains one record for each vehicle involved in an accident from the ACCIDENT table. There can be one or many vehicles involved in a single accident. The VEHICLE table attributes describe the type of the vehicle, maneuver and direction of the vehicle (from and to), vehicle location on the road, junction location at impact, sex and age of the driver, alcohol test results, damage resulting from the accident, and the object that vehicle hit on and off carriageway. There are 24 attributes in the VEHICLE table which contains almost 9 million records.

- The CASUALTY table contains records about casualties for each of the vehicles in the VEHICLE table. There can be one or more casualties per vehicle. The CASUALTY table contains 16 attributes describing sex and age of casualty, type of casualty (e.g., pedestrian, cyclist, car occupant etc.), severity of casualty, if casualty type is pedestrian, what were his/her characteristics (location, movement, direction). This table contains almost 7 million records.

6.3.2 DATA PREPROCESSING

The large volume of data in the traffic data set makes it practically impossible to run any data mining algorithm on the whole set. Therefore we have taken samples of the data set and performed

PFC	Number of Examples	Percentage of Sampled Accidents	Class distribution (%) fatal / serious / slight
1	2555	0.3	1.76 / 24.85 / 73.39
2	2523	1.9	2.53 / 30.87 / 66.60
3	2501	4.8	0.56 / 12.35 / 87.09
4	2499	1.9	2.16 / 27.21 / 70.63
5	2522	9.2	1.90 / 23.39 / 74.71
6	2548	2.0	1.41 / 13.69 / 84.90
7	2788	1.4	0.97 / 16.25 / 82.78

Table 8: Properties of the traffic data set.

the experiments on these samples. We focused on the ACCIDENT table and examined only the accidents that happened in 7 districts (called Police Force Codes, or PFCs) across the UK.[11] The 7 PFCs were chosen by the domain expert and represent typical PFCs from clusters of PFCs with the same accident dynamics, analyzed by Ljubič et al. (2002). In this way we obtained 7 data sets (one for each PFC) with some hundred thousands of examples each. We further sampled this data to obtain approximately 2500 examples per data set. The sample percentages are listed in Table 8 together with the other characteristics of these 7 sampled data sets.

Among the 26 attributes describing each of the 7 data sets we chose the attribute 'accident severity' to be the class attribute. The task that we have addressed was therefore to find subgroups of accidents of a certain severity ('slight', 'serious' or 'fatal') and characterize them in terms of attributes describing the accident, such as: 'road class', 'speed limit', 'light condition', etc.

6.3.3 RESULTS OF EXPERIMENTS

We want to investigate if by running *CN2-SD* on the data sets, described in Table 8, we are able to get some rules that are typical and different for distinct PFCs.

We used the same methodology to perform the experiments as in the case of the UCI data sets of Section 6.2. The only difference is that here we do not perform the area under ROC curve analysis, because the data sets are not two-class. The results presented in Tables 9–13 show the same advantages of *CN2-SD* over *CN2-WRAcc* and *CN2-standard* as shown by the results of experiments on the UCI data sets.[12] In particular, *CN2-SD* produces substantially smaller rule sets, where individual rules have higher coverage and significance.

It should be noticed that these data sets have a very unbalanced class distribution (most accidents are 'slight' and only few are 'fatal', see Table 8). In terms of rule set accuracy, all algorithms achieved roughly default performance which is obtained by always predicting the majority class. Since classification was not the main interest of this experiment, we omit the results.

11. For the sake of anonymity, the code numbers 1 through 7 do not correspond to the PFCs 1 through 7 used for Police Force Codes in the actual traffic accident database.
12. Like in the UCI case, only the results of the unordered versions of the algorithm are presented here, although the experiments were done with both unordered and ordered variants of the algorithms.

#	**CN2 standard** *COV* ± *sd*	**CN2 WRAcc** *COV* ± *sd*	**CN2-SD (γ = 0.5)** *COV* ± *sd*	**CN2-SD (γ = 0.7)** *COV* ± *sd*	**CN2-SD (γ = 0.9)** *COV* ± *sd*	**CN2-SD (add.)** *COV* ± *sd*
1	0.056 ± 0.01	**0.108**↑ ± 0.00	**0.111**↑ ± 0.03	**0.111**↑ ± 0.03	**0.123**↑ ± 0.03	**0.110**↑ ± 0.03
2	0.050 ± 0.10	**0.113**↑ ± 0.04	**0.127**↑ ± 0.05	**0.127**↑ ± 0.04	**0.129**↑ ± 0.05	**0.151**↑ ± 0.04
3	0.140 ± 0.03	0.118 ± 0.03	0.126 ± 0.02	0.119 ± 0.02	0.118 ± 0.01	0.154 ± 0.02
4	0.052 ± 0.01	**0.105**↑ ± 0.03	**0.105**↑ ± 0.04	**0.120**↑ ± 0.04	**0.122**↑ ± 0.04	**0.116**↑ ± 0.04
5	0.075 ± 0.08	**0.108**↑ ± 0.04	**0.115**↑ ± 0.06	**0.121**↑ ± 0.05	**0.110**↑ ± 0.05	**0.127**↑ ± 0.04
6	0.078 ± 0.06	**0.118**↑ ± 0.03	**0.134**↑ ± 0.05	**0.122**↑ ± 0.06	**0.124**↑ ± 0.06	**0.120**↑ ± 0.05
7	0.116 ± 0.08	0.110 ± 0.11	0.118 ± 0.14	0.124 ± 0.13	0.122 ± 0.13	**0.143**↑ ± 0.12
Average	0.081 ± 0.03	0.111 ± 0.01	0.120 ± 0.01	0.121 ± 0.00	0.121 ± 0.01	**0.132** ± 0.02
• significance – *p* value		0.047	0.021	0.023	0.029	**0.003**
• win/loss/draw		5/2/0	6/1/0	6/1/0	6/1/0	**7/0/0**
• sig.win/sig.loss		5/0	5/0	5/0	5/0	**6/0**

Table 9: Experimental results on the traffic accident data sets. Average coverage per rule with standard deviation (*COV* ± *sd*) for different variants of the unordered algorithm.

#	**CN2 standard** *SUP* ± *sd*	**CN2 WRAcc** *SUP* ± *sd*	**CN2-SD (γ = 0.5)** *SUP* ± *sd*	**CN2-SD (γ = 0.7)** *SUP* ± *sd*	**CN2-SD (γ = 0.9)** *SUP* ± *sd*	**CN2-SD (add.)** *SUP* ± *sd*
1	0.86 ± 0.03	0.89 ± 0.02	0.83 ± 0.06	**0.93**↑ ± 0.04	**0.96**↑ ± 0.02	**0.95**↑ ± 0.03
2	0.84 ± 0.02	0.85 ± 0.09	0.85 ± 0.02	**0.92**↑ ± 0.04	**0.93**↑ ± 0.00	0.84 ± 0.08
3	0.81 ± 0.06	0.82 ± 0.04	**0.93**↑ ± 0.02	**0.90**↑ ± 0.05	**0.97**↑ ± 0.01	0.85 ± 0.06
4	0.80 ± 0.04	**0.87**↑ ± 0.05	0.82 ± 0.05	0.83 ± 0.00	**0.91**↑ ± 0.03	0.81 ± 0.10
5	0.87 ± 0.08	0.85 ± 0.03	0.80↓ ± 0.03	0.83 ± 0.06	**0.94**↑ ± 0.02	0.83 ± 0.08
6	0.84 ± 0.06	**0.88**↑ ± 0.07	0.81 ± 0.09	**0.91**↑ ± 0.06	**0.88**↑ ± 0.07	**0.98**↑ ± 0.01
7	0.81 ± 0.08	0.83 ± 0.05	**0.90**↑ ± 0.01	0.81 ± 0.01	**0.95**↑ ± 0.02	**0.99**↑ ± 0.00
Average	0.83 ± 0.03	0.85 ± 0.02	0.85 ± 0.05	0.88 ± 0.05	**0.93** ± 0.03	0.89 ± 0.08
• significance – *p* value		0.056	0.548	0.053	**0.001**	0.092
• win/loss/draw		6/1/0	4/3/0	6/1/0	**7/0/0**	6/1/0
• sig.win/sig.loss		2/0	2/1	4/0	**7/0**	3/0

Table 10: Experimental results on the traffic accident data sets. Overall support of rule sets with standard deviation (*SUP* ± *sd*) for different variants of the unordered algorithm.

6.3.4 EVALUATION BY THE DOMAIN EXPERT

We have further examined the rules induced by the *CN2-SD* algorithm (using additive weights). We focused on rules with high coverage and rules that cover a high percentage of the predicted class as these are the rules that are likely to reflect some regularity in the data.

One of the most interesting results concerned the following. One might expect that the number of people injured would increase with the severity of the accident (up to the total number of occupants in the vehicles). Furthermore, common sense would dictate that the number of vehicles

#	**CN2 standard** *SIZE* ± *sd*	**CN2 WRAcc** *SIZE* ± *sd*	**CN2-SD (γ = 0.5)** *SIZE* ± *sd*	**CN2-SD (γ = 0.7)** *SIZE* ± *sd*	**CN2-SD (γ = 0.9)** *SIZE* ± *sd*	**CN2-SD (add.)** *SIZE* ± *sd*
1	16.7 ± 0.60	**9.3**↑ ± 0.99	**10.0**↑ ± 0.51	**10.6**↑ ± 0.46	**10.6**↑ ± 0.73	9.5 ↑ ± 0.25
2	18.7 ± 1.28	**9.2**↑ ± 0.33	**10.0**↑ ± 0.20	**10.3**↑ ± 0.21	**10.3**↑ ± 0.56	**11.1**↑ ± 0.23
3	7.0 ± 0.30	8.6 ± 0.95	9.2 ± 0.19	10.2↓ ± 0.14	9.5 ± 0.35	9.8 ↓ ± 0.19
4	18.0 ± 1.39	**9.9**↑ ± 0.59	**10.4**↑ ± 0.31	**11.2**↑ ± 0.64	**11.2**↑ ± 0.24	**10.3**↑ ± 0.56
5	12.8 ± 1.44	**9.6**↑ ± 0.19	**10.1**↑ ± 0.51	11.2 ± 0.84	11.6 ± 0.96	9.7 ↑ ± 0.21
6	12.5 ± 0.31	**8.5**↑ ± 0.35	9.3 ↑ ± 0.51	8.7 ↑ ± 0.91	9.4 ↑ ± 0.60	8.5 ↑ ± 0.39
7	8.6 ± 1.41	9.3 ± 0.41	9.9 ± 0.90	10.8↓ ± 0.73	11.1↓ ± 0.13	10.4 ± 0.59
Average	13.47 ± 4.57	**9.20** ± 0.50	9.84 ± 0.44	10.42 ± 0.86	10.53 ± 0.84	9.90 ± 0.80
• significance – *p* value		**0.040**	0.066	0.123	0.127	0.075
• win/loss/draw		**5/2/0**	5/2/0	5/2/0	5/2/0	5/2/0
• sig.win/sig.loss		**5/0**	5/0	4/2	4/1	5/1

Table 11: Experimental results on the traffic accident data sets. Sizes of rule sets with standard deviation (*SIZE* ± *sd*) for different variants of the unordered algorithm.

#	**CN2 standard** *SIG* ± *sd*	**CN2 WRAcc** *SIG* ± *sd*	**CN2-SD (γ = 0.5)** *SIG* ± *sd*	**CN2-SD (γ = 0.7)** *SIG* ± *sd*	**CN2-SD (γ = 0.9)** *SIG* ± *sd*	**CN2-SD (add.)** *SIG* ± *sd*
1	1.9 ± 0.82	**7.0**↑ ± 0.31	**8.7**↑ ± 0.41	**9.7**↑ ± 0.59	**9.4**↑ ± 0.30	**9.6**↑ ± 0.45
2	1.9 ± 0.34	**6.2**↑ ± 0.25	**9.9**↑ ± 0.26	**9.8**↑ ± 0.20	**9.5**↑ ± 0.81	**9.8**↑ ± 0.36
3	1.3 ± 0.27	**6.6**↑ ± 0.61	**8.4**↑ ± 0.52	**9.2**↑ ± 0.54	**11.5**↑ ± 0.75	**9.3**↑ ± 0.17
4	1.6 ± 0.10	**7.6**↑ ± 0.14	**8.5**↑ ± 0.79	**11.0**↑ ± 0.84	**9.4**↑ ± 0.80	**11.1**↑ ± 0.24
5	1.6 ± 0.75	**6.0**↑ ± 0.23	**10.6**↑ ± 0.70	**9.6**↑ ± 0.76	**12.5**↑ ± 0.43	**9.1**↑ ± 0.74
6	1.5 ± 0.87	**8.5**↑ ± 0.41	**8.3**↑ ± 0.54	**9.8**↑ ± 0.24	**9.9**↑ ± 0.51	**12.5**↑ ± 0.35
7	1.7 ± 0.49	**6.8**↑ ± 0.75	**8.7**↑ ± 0.20	**9.9**↑ ± 0.63	**9.2**↑ ± 0.73	**9.7**↑ ± 0.40
Average	1.64 ± 0.20	6.95 ± 0.86	9.01 ± 0.89	9.85 ± 0.56	**10.20** ± 1.28	10.16 ± 1.21
• significance – *p* value		0.000	0.000	0.000	**0.000**	0.000
• win/loss/draw		7/0/0	7/0/0	7/0/0	**7/0/0**	7/0/0
• sig.win/sig.loss		7/0	7/0	7/0	**7/0**	7/0

Table 12: Experimental results on the traffic accident data sets. Average significance per rule with standard deviation (*SIG* ± *sd*) for different variants of the unordered algorithm.

involved would also increase with accident severity. Contrary to these expectations we found rules of the following two kinds:

- Rules that cover more than the average proportion of 'fatal' or 'serious' accidents when just one vehicle is involved in the accident. Examples of such rules are:
  `IF nv < 1.500 THEN sev = "1" [15 280 1024]`[13]
  `IF nv < 1.500 THEN sev = "2" [22 252 890]`

13. The rules in the example are given in the *CN2-SD* output format where `nv` stands for 'number of vehicles', `nc` is the 'number of casualties' and `"1"`, `"2"`, and `"3"` denote the class values 'fatal', 'serious' and 'slight' respectively.

#	**CN2 standard** *WRACC* ± *sd*	**CN2 WRAcc** *WRACC* ± *sd*	**CN2-SD (γ = 0.5)** *WRACC* ± *sd*	**CN2-SD (γ = 0.7)** *WRACC* ± *sd*	**CN2-SD (γ = 0.9)** *WRACC* ± *sd*	**CN2-SD (add.)** *WRACC* ± *sd*
1	0.013 ± 0.02	**0.025**↑ ± 0.05	**0.025**↑ ± 0.10	**0.026**↑ ± 0.02	**0.028**↑ ± 0.03	**0.025**↑ ± 0.09
2	0.009 ± 0.07	**0.018**↑ ± 0.05	**0.021**↑ ± 0.00	**0.021**↑ ± 0.04	**0.021**↑ ± 0.02	**0.025**↑ ± 0.04
3	0.052 ± 0.01	0.043 ± 0.00	0.046 ± 0.07	0.043 ± 0.03	0.043 ± 0.05	0.056 ± 0.02
4	0.010 ± 0.09	**0.021**↑ ± 0.06	**0.021**↑ ± 0.05	**0.024**↑ ± 0.09	**0.024**↑ ± 0.00	**0.023**↑ ± 0.07
5	0.019 ± 0.04	**0.026**↑ ± 0.06	**0.027**↑ ± 0.07	**0.029**↑ ± 0.08	**0.027**↑ ± 0.01	**0.030**↑ ± 0.07
6	0.027 ± 0.03	**0.041**↑ ± 0.06	**0.047**↑ ± 0.05	**0.042**↑ ± 0.05	**0.043**↑ ± 0.07	**0.042**↑ ± 0.07
7	0.038 ± 0.03	0.035 ± 0.01	0.038 ± 0.04	0.040 ± 0.00	0.039 ± 0.08	0.046 ± 0.04
Average	0.024 ± 0.02	0.030 ± 0.01	0.032 ± 0.01	0.032 ± 0.01	0.032 ± 0.01	**0.035** ± 0.01
• significance – *p* value		0.096	0.042	0.041	0.048	**0.000**
• win/loss/draw		5/2/0	5/2/0	6/1/0	6/1/0	**7/0/0**
• sig.win/sig.loss		5/0	5/0	5/0	5/0	**5/0**

Table 13: Experimental results on the traffic accident data sets. Unusualness of rule sets with standard deviation (*WRACC* ± *sd*) for different variants of the unordered algorithm.

- Rules that cover more than the average proportion of 'slight' accidents when two or more vehicles are involved and there are few casualties. An example of such a rule is:
  `IF nv > 1.500 AND nc < 2.500 THEN sev = "3" [8 140 1190]`

Having shown the induced results to the domain expert, he pointed out the following aspects of data collection for the data in the ACCIDENT table.[14]

- The severity code in the ACCIDENT table relates to the most severe injury among those reported for that accident. Therefore a multiple vehicle accident with 1 fatal and 20 slight injuries would be classified as fatal as one fatality occurred, while each individual casualty injury severity is coded in the CASUALTY table.
- Some (slight) injuries may be unreported at the accident scene: if the policeman compiled/revised the report after the event, new casualty/injury details can be reported (injuries that came to light after the event or reported for reasons relating to injury/insurance claims). However, these changes are not reflected in the ACCIDENT table.

The findings revealed by the rules were surprising to the domain expert and need further investigation. The analysis shows that examining the ACCIDENT table is not sufficient and that further examination of the VEHICLE and CASUALTY tables is needed in further work.

7. Related Work

Other systems have addressed the task of subgroup discovery, the best known being EXPLORA (Klösgen, 1996) and MIDOS (Wrobel, 1997, 2001). EXPLORA treats the learning task as a single relation problem, i.e., all the data are assumed to be available in one table (relation), whereas

14. We have also shown the *CN2-standard* and *CN2-WRAcc* results to the expert but he did not consider any of the rules to be interesting.

MIDOS extends this task to multi-relational databases. Other approaches deal with multi-relational databases using propositionalisation and aggregate functions can be found in the work of Knobbe et al. (2001, 2002).

Another approach to finding symbolic descriptions of groups of instances is symbolic clustering, which has been popular for many years (Michalski, 1980, Gowda and Diday, 1992). Moreover, learning of concept hierarchies also aims at discovering groups of instances, which can be induced in a supervised or unsupervised manner: decision tree induction algorithms perform supervised symbolic learning of concept hierarchies (Langley, 1996, Raedt and Blockeel, 1997), whereas hierarchical clustering algorithms (Sokal and Sneath, 1963, Gordon, 1982) are unsupervised and do not result in symbolic descriptions. Note that in decision tree learning, the rules which can be formed from paths leading from the root node to class labels in the leaves represent *discriminant descriptions*, formed from properties that best discriminate between the classes. As rules formed from decision tree paths form discriminant descriptions, they are inappropriate for solving subgroup discovery tasks which aim at describing subgroups by their characteristic properties.

Instance weights play an important role in boosting (Freund and Shapire, 1996) and alternating decision trees (Schapire and Singer, 1998). Instance weights have been used also in variants of the covering algorithm implemented in rule learning approaches such as SLIPPER (Cohen and Singer, 1999), RL (Lee et al., 1998) and DAIRY (Hsu et al., 1998). A variant of the weighted covering algorithm has been used in the subgroup discovery algorithm SD for rule subset selection (Gamberger and Lavrač, 2002).

A variety of rule evaluation measures and heuristics have been studied for subgroup discovery (Klösgen, 1996, Wrobel, 1997, 2001), aimed at balancing the size of a group (referred to as factor g) with its distributional unusualness (referred to as factor p). The properties of functions that combine these two factors have been extensively studied (the so-called 'p-g-space' Klösgen, 1996). An alternative measure $q = \frac{TP}{FP+par}$ was proposed in the SD algorithm for expert-guided subgroup discovery (Gamberger and Lavrač, 2002), aimed at minimizing the number of false positives FP, and maximizing true positives TP, balanced by generalization parameter par. Besides such 'objective' measures of interestingness, some 'subjective' measure of interestingness of a discovered pattern can be taken into account, such as actionability ('a pattern is interesting if the user can do something with it to his or her advantage') and unexpectedness ('a pattern is interesting to the user if it is surprising to the user') (Silberschatz and Tuzhilin, 1995).

Note that some approaches to association rule induction can also be used for subgroup discovery. For instance, the APRIORI-C algorithm (Jovanoski and Lavrač, 2001), which applies association rule induction to classification rule induction, outputs classification rules with guaranteed support and confidence with respect to a target class. If a rule satisfies also a user-defined significance threshold, an induced APRIORI-C rule can be viewed as an independent 'chunk' of knowledge about the target class (selected property of interest for subgroup discovery), which can be viewed as a subgroup description with guaranteed significance, support and confidence. This observation led to the development of a novel subgroup discovery algorithm APRIORI-SD (Kavšek et al., 2003).

It should be noticed that in the terminology 'patient vs. greedy' of Friedman and Fisher (1999), *WRAcc* is a 'patient' rule quality measure, favoring more general subgroups than those found by using 'greedy' quality measures. As shown by our experiments in Todorovski et al. (2000), *WRAcc* heuristic improves rule coverage compared to the standard CN2 heuristic. This observation is confirmed also in the experimental evaluation in Section 6 of this paper. Further evidence confirming this claim is provided by Kavšek et al. (2003), providing experimental comparison of results of *CN2-*

SD and our novel subgroup discovery algorithm APRIORI-SD with rule learners CN2, RIPPER and APRIORI-C.

8. Conclusions and Further Work

We have presented a novel approach to adapting standard classification rule learning to subgroup discovery. To this end we have appropriately adapted the covering algorithm, the search heuristic, the probabilistic classification and the area under the ROC curve (AUC) performance measure. We have also proposed a set of metrics appropriate for evaluating the quality of induced subgroup descriptions.

The experimental results on 23 UCI data sets demonstrate that *CN2-SD* produces substantially smaller rule sets, where individual rules have higher coverage and significance. These three factors are important for subgroup discovery: smaller size enables better understanding, higher coverage means larger support, and higher significance means that rules describe discovered subgroups that are significantly different from the entire population. We have evaluated the results of *CN2-SD* also in terms of AUC and shown a small (insignificant) increase in terms of the area under ROC curve.

We have applied *CN2-SD* also to a real-life problem of traffic accident analysis. The experimental results confirm the findings in the UCI data sets. The most interesting findings are due to interpretation by the domain expert. What was confirmed in this case study was that the result of a data mining process depends not only on the appropriateness of the selected method and the data that is at hand but also on how the data has been collected. In the traffic accident experiments examining the ACCIDENT table was not sufficient, and further examination of the VEHICLE and CASUALTY tables is needed. This will be performed using the RSD relational subgroup discovery algorithm (Lavrač et al., 2003), a recent upgrade of the *CN2-SD* algorithm which enables relational subgroup discovery.

In further work we plan to compare the results with the MIDOS subgroup discovery algorithm. We plan to investigate the behavior of *CN2-SD* in terms of AUC in multi-class problems (Hand and Till, 2001). An interesting question, to be verified in further experiments, is whether the weighted versions of the CN2 algorithm improve the significance of the induced subgroups also in the case when CN2 rules are induced without applying the significance test.

An important aspect of subgroup discovery performance, which is neglected in our study, is the degree of overlap of the induced subgroups. The challenge of our further research is to propose extensions of the weighted relative accuracy heuristic and ROC space evaluation metrics that will take into account the overlap of subgroups.

We are now moving the focus of our research in subgroup discovery from heuristic search toward exhaustive search of the space of patterns. An attempt of this kind is described by Kavšek et al. (2003) where the well known APRIORI association rule learner was adapted to the task of subgroup discovery.

Acknowledgments

Thanks to Dragan Gamberger for joint work on the weighted covering algorithm, and José Hernández-Orallo and Cesar Ferri-Ramírez for joint work on AUC. Thanks to Peter Ljubič and Damjan Demšar for the help in upgrading the C code of the original CN2 algorithm. We are grateful to

John Bullas for the evaluation of the results of traffic accident data analysis. Thanks are also due to the anonymous reviewers for their insightful comments. The work reported in this paper was supported by the Slovenian Ministry of Education, Science and Sport, the IST-1999-11495 project Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise, and the British Council project Partnership in Science PSP-18.

Appendix A. Tables with Detailed Results for Different Variants of the Unordered Algorithm in UCI Data Sets

The tables in this appendix show detailed results of the performance of different variants of the unordered algorithm. The comparisons are made on 23 UCI data sets listed in Table 3. The results shown in Tables 14–18 of Appendix A are summarized in the paper in Table 4, and the results of Tables 19–20 in Table 5.

#	**CN2 standard** *COV* ± *sd*	**CN2 WRAcc** *COV* ± *sd*	**CN2-SD ($\gamma = 0.5$)** *COV* ± *sd*	**CN2-SD ($\gamma = 0.7$)** *COV* ± *sd*	**CN2-SD ($\gamma = 0.9$)** *COV* ± *sd*	**CN2-SD (add.)** *COV* ± *sd*
1	0.071 ± 0.01	**0.416**↑ ± 0.00	**0.473**↑ ± 0.03	**0.492**↑ ± 0.03	**0.480**↑ ± 0.03	**0.424**↑ ± 0.03
2	0.079 ± 0.10	**0.150**↑ ± 0.04	**0.208**↑ ± 0.05	**0.174**↑ ± 0.04	**0.218**↑ ± 0.05	**0.260**↑ ± 0.04
3	0.625 ± 0.03	0.322↓ ± 0.03	0.612 ± 0.02	0.617 ± 0.02	0.721 ± 0.01	0.330↓ ± 0.02
4	0.048 ± 0.01	**0.496**↑ ± 0.03	**0.504**↑ ± 0.04	**0.513**↑ ± 0.04	**0.504**↑ ± 0.04	**0.507**↑ ± 0.04
5	0.057 ± 0.08	**0.275**↑ ± 0.04	**0.296**↑ ± 0.06	**0.344**↑ ± 0.05	**0.299**↑ ± 0.05	**0.381**↑ ± 0.04
6	0.312 ± 0.06	**0.576**↑ ± 0.03	**0.936**↑ ± 0.05	**1.039**↑ ± 0.06	**1.006**↑ ± 0.06	**1.295**↑ ± 0.05
7	0.053 ± 0.08	**0.092**↑ ± 0.11	**0.141**↑ ± 0.14	**0.153**↑ ± 0.13	**0.138**↑ ± 0.13	**0.151**↑ ± 0.12
8	0.107 ± 0.09	**0.240**↑ ± 0.07	**0.419**↑ ± 0.09	**0.376**↑ ± 0.12	**0.366**↑ ± 0.11	**0.435**↑ ± 0.09
9	0.207 ± 0.04	**0.430**↑ ± 0.06	**0.637**↑ ± 0.04	**0.829**↑ ± 0.04	**0.826**↑ ± 0.04	**0.686**↑ ± 0.03
10	0.093 ± 0.00	**0.495**↑ ± 0.00	**0.509**↑ ± 0.00	**0.509**↑ ± 0.00	**0.516**↑ ± 0.00	**0.513**↑ ± 0.00
11	0.099 ± 0.05	**0.168**↑ ± 0.08	**0.229**↑ ± 0.05	**0.234**↑ ± 0.04	**0.246**↑ ± 0.04	**0.354**↑ ± 0.06
12	0.378 ± 0.01	0.386 ± 0.01	**0.619**↑ ± 0.00	0.444 ± 0.00	**0.768**↑ ± 0.00	**0.668**↑ ± 0.01
13	0.160 ± 0.11	**0.408**↑ ± 0.09	**0.639**↑ ± 0.15	**0.467**↑ ± 0.16	**0.424**↑ ± 0.18	**0.621**↑ ± 0.17
14	0.142 ± 0.01	**0.356**↑ ± 0.07	**0.461**↑ ± 0.02	**0.668**↑ ± 0.03	**0.569**↑ ± 0.03	**0.720**↑ ± 0.03
15	0.030 ± 0.01	**0.113**↑ ± 0.07	**0.129**↑ ± 0.02	**0.146**↑ ± 0.03	**0.182**↑ ± 0.03	**0.117**↑ ± 0.03
16	0.129 ± 0.01	**0.650**↑ ± 0.07	**0.703**↑ ± 0.02	**0.711**↑ ± 0.03	**0.674**↑ ± 0.03	**0.831**↑ ± 0.03
17	0.021 ± 0.00	**0.216**↑ ± 0.00	**0.225**↑ ± 0.00	**0.270**↑ ± 0.00	**0.307**↑ ± 0.00	**0.324**↑ ± 0.00
18	0.022 ± 0.05	**0.146**↑ ± 0.08	**0.155**↑ ± 0.05	**0.157**↑ ± 0.04	**0.166**↑ ± 0.04	**0.200**↑ ± 0.06
19	0.066 ± 0.01	**0.331**↑ ± 0.01	**0.357**↑ ± 0.00	**0.628**↑ ± 0.00	**0.616**↑ ± 0.00	**0.759**↑ ± 0.01
20	0.039 ± 0.11	**0.139**↑ ± 0.09	**0.151**↑ ± 0.15	**0.159**↑ ± 0.16	**0.149**↑ ± 0.18	**0.169**↑ ± 0.17
21	0.040 ± 0.01	**0.076**↑ ± 0.07	**0.115**↑ ± 0.02	**0.177**↑ ± 0.03	**0.172**↑ ± 0.03	**0.216**↑ ± 0.03
22	0.004 ± 0.01	**0.185**↑ ± 0.07	**0.194**↑ ± 0.02	**0.185**↑ ± 0.03	**0.188**↑ ± 0.03	**0.191**↑ ± 0.03
23	0.231 ± 0.01	**0.477**↑ ± 0.07	**0.552**↑ ± 0.02	**0.715**↑ ± 0.03	**0.818**↑ ± 0.03	**1.022**↑ ± 0.03
Average	0.131 ± 0.14	0.311 ± 0.17	0.403 ± 0.23	0.435 ± 0.25	0.450 ± 0.26	**0.486** ± 0.30
• significance – p value		0.000	0.000	0.000	0.000	**0.000**
• win/loss/draw		22/1/0	22/1/0	22/1/0	23/0/0	**22/1/0**
• sig.win/sig.loss		21/1	22/0	21/0	22/0	**22/1**

Table 14: Relative average coverage per rule with standard deviation (*COV* ± *sd*) for different variants of the unordered algorithm using 10-fold stratified cross-validation.

#	**CN2 standard** *SUP* ± *sd*	**CN2 WRAcc** *SUP* ± *sd*	**CN2-SD (γ = 0.5)** *SUP* ± *sd*	**CN2-SD (γ = 0.7)** *SUP* ± *sd*	**CN2-SD (γ = 0.9)** *SUP* ± *sd*	**CN2-SD (add.)** *SUP* ± *sd*
1	0.81 ± 0.09	**0.89**↑ ± 0.02	**0.87**↑ ± 0.00	**0.97**↑ ± 0.01	**0.84**↑ ± 0.00	**0.89**↑ ± 0.04
2	0.88 ± 0.01	0.90 ± 0.02	0.89 ± 0.09	0.84 ± 0.04	**0.93**↑ ± 0.02	0.86 ± 0.05
3	0.87 ± 0.05	0.87 ± 0.09	0.84 ± 0.05	**0.93**↑ ± 0.02	0.84 ± 0.07	**0.95**↑ ± 0.01
4	0.87 ± 0.06	0.81↓ ± 0.09	0.90 ± 0.02	0.81↓ ± 0.04	**0.97**↑ ± 0.00	**0.93**↑ ± 0.02
5	0.80 ± 0.01	0.82 ± 0.03	**0.92**↑ ± 0.06	0.85 ± 0.01	**0.95**↑ ± 0.01	**0.87**↑ ± 0.05
6	0.90 ± 0.03	0.81↓ ± 0.01	**0.95**↑ ± 0.01	0.85 ± 0.03	**0.98**↑ ± 0.00	0.82↓ ± 0.02
7	0.89 ± 0.03	0.88 ± 0.03	0.90 ± 0.02	0.81↓ ± 0.07	**0.97**↑ ± 0.01	**0.96**↑ ± 0.01
8	0.84 ± 0.03	0.87 ± 0.04	**0.94**↑ ± 0.01	0.83 ± 0.03	0.89 ± 0.09	**0.98**↑ ± 0.00
9	0.87 ± 0.10	0.81↓ ± 0.02	0.85 ± 0.10	**0.94**↑ ± 0.00	0.90 ± 0.02	**0.99**↑ ± 0.00
10	0.84 ± 0.01	0.83 ± 0.08	0.82 ± 0.07	**1.00**↑ ± 0.00	**0.90**↑ ± 0.02	**0.95**↑ ± 0.02
11	0.83 ± 0.03	0.85 ± 0.07	**0.96**↑ ± 0.01	**0.95**↑ ± 0.01	**0.89**↑ ± 0.09	**0.98**↑ ± 0.01
12	0.82 ± 0.04	**0.89**↑ ± 0.00	0.83 ± 0.10	**0.91**↑ ± 0.01	**0.88**↑ ± 0.03	**0.95**↑ ± 0.01
13	0.87 ± 0.10	0.90 ± 0.06	0.81↓ ± 0.02	0.80↓ ± 0.09	0.85 ± 0.04	0.85 ± 0.03
14	0.84 ± 0.05	0.85 ± 0.07	0.83 ± 0.06	**0.89**↑ ± 0.06	**0.93**↑ ± 0.02	0.86 ± 0.05
15	0.83 ± 0.04	0.80 ± 0.07	**0.96**↑ ± 0.01	0.86 ± 0.09	0.80 ± 0.08	0.81 ± 0.00
16	0.85 ± 0.07	0.82 ± 0.02	**1.00**↑ ± 0.00	0.84 ± 0.06	**0.96**↑ ± 0.01	0.85 ± 0.10
17	0.86 ± 0.08	**0.90**↑ ± 0.03	0.86 ± 0.07	0.82 ± 0.06	**1.00**↑ ± 0.00	0.85 ± 0.06
18	0.81 ± 0.06	**0.85**↑ ± 0.07	**0.96**↑ ± 0.01	**0.89**↑ ± 0.05	**0.95**↑ ± 0.01	**0.97**↑ ± 0.00
19	0.83 ± 0.01	0.85 ± 0.05	**0.92**↑ ± 0.04	**0.95**↑ ± 0.01	**0.90**↑ ± 0.02	0.84 ± 0.05
20	0.90 ± 0.06	0.82↓ ± 0.07	**0.99**↑ ± 0.00	0.90 ± 0.03	**0.99**↑ ± 0.00	0.90 ± 0.04
21	0.81 ± 0.05	0.80 ± 0.04	**0.87**↑ ± 0.08	**0.90**↑ ± 0.04	**0.93**↑ ± 0.02	0.82 ± 0.06
22	0.81 ± 0.02	**0.89**↑ ± 0.06	**0.94**↑ ± 0.02	**0.96**↑ ± 0.01	**1.00**↑ ± 0.00	**0.96**↑ ± 0.01
23	0.82 ± 0.05	0.82 ± 0.04	**0.94**↑ ± 0.03	**0.87**↑ ± 0.07	**0.99**↑ ± 0.00	**0.99**↑ ± 0.00
Average	0.84 ± 0.03	0.85 ± 0.03	0.90 ± 0.06	0.89 ± 0.06	**0.92** ± 0.06	0.91 ± 0.06
• significance – *p* value		0.637	0.000	0.017	**0.000**	0.001
• win/loss/draw		13/10/0	18/5/0	14/9/0	**20/3/0**	16/7/0
• sig.win/sig.loss		5/4	13/1	11/3	**18/0**	13/1

Table 15: Overall rule set support with standard deviation (*SUP* ± *sd*) for different variants of the unordered algorithm using 10-fold stratified cross-validation.

#	**CN2 standard** *SIZE* ± *sd*	**CN2 WRAcc** *SIZE* ± *sd*	**CN2-SD ($\gamma = 0.5$)** *SIZE* ± *sd*	**CN2-SD ($\gamma = 0.7$)** *SIZE* ± *sd*	**CN2-SD ($\gamma = 0.9$)** *SIZE* ± *sd*	**CN2-SD (add.)** *SIZE* ± *sd*
1	12.4 ± 1.95	**2.0**↑ ± 0.75	**2.7**↑ ± 0.02	**2.6**↑ ± 0.87	**2.2**↑ ± 0.85	**3.5**↑ ± 0.79
2	12.6 ± 1.04	**8.8**↑ ± 0.95	**7.9**↑ ± 0.50	**8.5**↑ ± 1.75	**9.0**↑ ± 0.24	**9.2**↑ ± 1.24
3	1.8 ± 0.10	2.0 ± 0.41	2.0 ± 0.70	2.7 ↓ ± 0.44	1.9 ± 0.27	1.8 ± 0.29
4	14.6 ± 1.81	**7.9**↑ ± 1.78	**8.1**↑ ± 1.02	**7.9**↑ ± 0.97	**8.5**↑ ± 0.47	**8.5**↑ ± 0.41
5	12.8 ± 1.56	**5.2**↑ ± 0.79	**6.0**↑ ± 0.68	**5.6**↑ ± 1.35	**5.4**↑ ± 0.30	**4.6**↑ ± 0.86
6	3.7 ± 1.37	**2.5**↑ ± 0.79	3.1 ± 0.72	3.8 ± 1.61	4.7 ↓ ± 1.22	3.4 ± 0.02
7	15.1 ± 1.89	**7.8**↑ ± 1.49	**8.4**↑ ± 1.32	**8.7**↑ ± 0.46	**9.1**↑ ± 1.26	**8.8**↑ ± 1.13
8	6.4 ± 1.53	**3.0**↑ ± 1.20	**2.9**↑ ± 0.98	**2.7**↑ ± 0.67	**2.7**↑ ± 0.90	**1.8**↑ ± 0.38
9	3.0 ± 0.29	**2.1**↑ ± 0.50	**1.7**↑ ± 0.93	2.7 ± 0.53	3.6 ↓ ± 1.83	2.7 ± 0.00
10	10.1 ± 1.02	**3.9**↑ ± 0.31	**3.9**↑ ± 0.85	**3.4**↑ ± 1.10	**3.3**↑ ± 1.90	**2.5**↑ ± 0.54
11	7.6 ± 1.01	**3.0**↑ ± 1.78	**3.9**↑ ± 1.84	**4.0**↑ ± 0.18	**3.6**↑ ± 0.87	**4.2**↑ ± 0.41
12	3.8 ± 1.24	**3.0**↑ ± 1.24	**3.2**↑ ± 0.42	**3.4**↑ ± 0.39	**2.9**↑ ± 0.05	3.6 ± 0.69
13	4.7 ± 1.30	**3.1**↑ ± 1.15	**3.4**↑ ± 0.54	**3.9**↑ ± 0.98	4.6 ± 1.19	4.5 ± 0.71
14	5.2 ± 0.90	**2.7**↑ ± 0.91	**2.1**↑ ± 0.95	**1.9**↑ ± 0.10	**1.7**↑ ± 1.73	**2.1**↑ ± 0.78
15	21.2 ± 3.48	**10.5**↑ ± 1.85	**11.2**↑ ± 1.12	**10.3**↑ ± 1.99	**9.6**↑ ± 1.32	**10.2**↑ ± 1.30
16	7.1 ± 1.59	**2.0**↑ ± 0.81	**2.4**↑ ± 0.56	**2.4**↑ ± 0.75	**2.9**↑ ± 0.56	**1.8**↑ ± 0.45
17	28.7 ± 3.89	**9.9**↑ ± 1.22	**9.4**↑ ± 1.61	**8.9**↑ ± 1.80	**9.5**↑ ± 1.03	**8.3**↑ ± 1.17
18	83.8 ± 5.37	**10.9**↑ ± 2.37	**11.3**↑ ± 2.78	**11.8**↑ ± 1.45	**11.7**↑ ± 1.67	**12.8**↑ ± 1.74
19	12.9 ± 1.68	**7.7**↑ ± 1.00	**8.6**↑ ± 1.21	**9.1**↑ ± 1.85	**8.4**↑ ± 1.09	**10.1**↑ ± 1.83
20	32.8 ± 2.64	**8.7**↑ ± 1.82	**8.9**↑ ± 1.48	**9.8**↑ ± 1.01	**10.5**↑ ± 1.37	**9.2**↑ ± 1.49
21	35.1 ± 3.54	**19.6**↑ ± 1.80	**19.3**↑ ± 2.91	**19.7**↑ ± 2.99	**19.8**↑ ± 2.58	**19.2**↑ ± 2.90
22	77.3 ± 4.07	**12.2**↑ ± 1.79	**11.4**↑ ± 2.87	**12.4**↑ ± 2.29	**12.4**↑ ± 2.09	**11.7**↑ ± 2.81
23	5.5 ± 1.26	**3.0**↑ ± 0.36	**2.1**↑ ± 0.70	**2.1**↑ ± 0.57	**1.2**↑ ± 0.73	**1.4**↑ ± 0.90
Average	18.18 ± 21.77	**6.15** ± 4.49	6.25 ± 4.42	6.45 ± 4.48	6.49 ± 4.57	6.35 ± 4.58
• significance – *p* value		**0.006**	0.007	0.007	0.007	0.007
• win/loss/draw		**22/1/0**	22/1/0	21/2/0	20/3/0	23/0/0
• sig.win/sig.loss		**22/0**	21/0	20/1	19/2	18/0

Table 16: Average rule set sizes with standard deviation (*SIZE* ± *sd*) for different variants of the unordered algorithm using 10-fold stratified cross-validation.

#	**CN2 standard** *SIG* ± *sd*	**CN2 WRAcc** *SIG* ± *sd*	**CN2-SD (γ = 0.5)** *SIG* ± *sd*	**CN2-SD (γ = 0.7)** *SIG* ± *sd*	**CN2-SD (γ = 0.9)** *SIG* ± *sd*	**CN2-SD (add.)** *SIG* ± *sd*
1	2.0 ± 0.05	**7.8**↑ ± 1.49	**14.6**↑ ± 1.05	**24.0**↑ ± 1.01	**15.6**↑ ± 1.54	**4.6**↑ ± 0.52
2	2.7 ± 0.10	**13.3**↑ ± 1.69	**27.1**↑ ± 3.37	2.1 ± 0.02	**20.5**↑ ± 2.45	**26.6**↑ ± 3.43
3	2.1 ± 0.01	**7.8**↑ ± 0.64	**13.3**↑ ± 1.39	2.5 ± 0.01	**21.2**↑ ± 2.55	**22.9**↑ ± 2.43
4	2.4 ± 0.06	**9.1**↑ ± 0.58	**14.1**↑ ± 1.72	**16.9**↑ ± 1.28	**22.5**↑ ± 2.49	**30.2**↑ ± 3.98
5	2.0 ± 0.01	**15.8**↑ ± 1.07	**14.9**↑ ± 1.95	**11.0**↑ ± 1.43	**15.2**↑ ± 1.85	2.1 ± 0.01
6	1.9 ± 0.03	**10.0**↑ ± 1.63	**11.0**↑ ± 1.12	**30.5**↑ ± 2.12	**30.1**↑ ± 2.27	**23.1**↑ ± 2.97
7	2.0 ± 0.02	2.7 ± 0.83	**19.8**↑ ± 1.21	**17.7**↑ ± 1.63	**11.1**↑ ± 1.03	**16.3**↑ ± 1.49
8	1.9 ± 0.09	**4.6**↑ ± 0.59	**23.2**↑ ± 1.82	**5.3**↑ ± 0.36	**4.0**↑ ± 0.03	**30.6**↑ ± 2.96
9	2.7 ± 0.03	**9.7**↑ ± 0.86	**12.3**↑ ± 1.00	**9.3**↑ ± 0.65	**8.5**↑ ± 0.89	**25.0**↑ ± 2.60
10	1.4 ± 0.04	**3.6**↑ ± 0.74	**5.8**↑ ± 0.48	**28.3**↑ ± 2.27	**24.9**↑ ± 2.27	**13.5**↑ ± 1.84
11	2.0 ± 0.04	1.8 ± 0.07	**16.7**↑ ± 1.42	**23.9**↑ ± 2.41	**30.9**↑ ± 2.18	**14.9**↑ ± 1.52
12	1.9 ± 0.03	**7.1**↑ ± 0.07	**17.0**↑ ± 1.61	1.3 ± 0.09	**17.6**↑ ± 1.45	**4.0**↑ ± 0.00
13	2.1 ± 0.00	**15.1**↑ ± 1.80	**19.4**↑ ± 1.77	**21.9**↑ ± 2.38	**21.4**↑ ± 2.39	**9.7**↑ ± 0.61
14	2.5 ± 0.08	**14.9**↑ ± 1.93	**18.0**↑ ± 1.57	**13.9**↑ ± 1.28	3.0 ± 0.09	**18.1**↑ ± 1.73
15	2.5 ± 0.05	**4.2**↑ ± 0.42	**17.5**↑ ± 1.79	**5.7**↑ ± 0.46	**21.9**↑ ± 2.83	**26.5**↑ ± 2.22
16	2.6 ± 0.04	**11.7**↑ ± 1.90	**9.6**↑ ± 0.56	**22.7**↑ ± 2.59	2.3 ± 0.08	**6.0**↑ ± 0.00
17	2.7 ± 0.03	**4.8**↑ ± 0.53	**11.7**↑ ± 1.67	**21.8**↑ ± 2.55	**15.0**↑ ± 1.82	**24.3**↑ ± 2.26
18	1.5 ± 0.00	**14.1**↑ ± 1.11	**6.0**↑ ± 0.93	**26.8**↑ ± 2.53	**12.6**↑ ± 1.35	**19.3**↑ ± 1.09
19	1.0 ± 0.07	**2.4**↑ ± 0.01	**22.0**↑ ± 1.20	**17.0**↑ ± 1.78	**16.4**↑ ± 1.74	**9.1**↑ ± 0.02
20	1.5 ± 0.00	**16.0**↑ ± 2.52	**24.3**↑ ± 1.52	**11.4**↑ ± 1.25	**29.9**↑ ± 3.25	**21.7**↑ ± 2.88
21	2.4 ± 0.02	**6.8**↑ ± 0.88	**15.6**↑ ± 1.98	**12.9**↑ ± 1.47	**8.2**↑ ± 0.06	**30.6**↑ ± 2.39
22	2.6 ± 0.04	**9.7**↑ ± 1.56	**3.4**↑ ± 0.09	**14.2**↑ ± 1.20	**7.1**↑ ± 0.47	**20.2**↑ ± 2.71
23	2.0 ± 0.07	**13.5**↑ ± 1.57	**20.7**↑ ± 1.93	**2.7**↑ ± 0.02	**29.4**↑ ± 3.51	**25.7**↑ ± 2.48
Average	2.11 ± 0.46	8.97 ± 4.66	15.57 ± 6.05	14.95 ± 9.02	16.92 ± 8.90	**18.47** ± 9.00
• significance – *p* value		0.000	0.000	0.000	0.000	**0.000**
• win/loss/draw		22/1/0	23/0/0	21/2/0	22/1/0	**23/0/0**
• sig.win/sig.loss		21/0	23/0	20/0	21/0	**22/0**

Table 17: Average rule significance with standard deviation (*SIG* ± *sd*) for different variants of the unordered algorithm using 10-fold stratified cross-validation.

#	**CN2 standard** *WRACC* ± *sd*	**CN2 WRAcc** *WRACC* ± *sd*	**CN2-SD (γ = 0.5)** *WRACC* ± *sd*	**CN2-SD (γ = 0.7)** *WRACC* ± *sd*	**CN2-SD (γ = 0.9)** *WRACC* ± *sd*	**CN2-SD (add.)** *WRACC* ± *sd*
1	0.022 ± 0.09	**0.148**↑ ± 0.03	**0.186**↑ ± 0.09	**0.185**↑ ± 0.04	**0.181**↑ ± 0.07	**0.162**↑ ± 0.01
2	0.034 ± 0.04	**0.063**↑ ± 0.04	**0.095**↑ ± 0.02	**0.079**↑ ± 0.01	**0.093**↑ ± 0.07	**0.111**↑ ± 0.04
3	-0.016 ± 0.08	-0.012 ± 0.01	-0.005 ± 0.03	-0.006 ± 0.09	-0.001 ± 0.02	-0.012 ± 0.01
4	0.020 ± 0.04	**0.210**↑ ± 0.02	**0.228**↑ ± 0.02	**0.233**↑ ± 0.04	**0.224**↑ ± 0.03	**0.224**↑ ± 0.10
5	0.013 ± 0.06	**0.065**↑ ± 0.06	**0.085**↑ ± 0.07	**0.099**↑ ± 0.04	**0.086**↑ ± 0.07	**0.092**↑ ± 0.03
6	0.058 ± 0.07	**0.099**↑ ± 0.10	**0.174**↑ ± 0.05	**0.208**↑ ± 0.00	**0.213**↑ ± 0.01	**0.243**↑ ± 0.10
7	0.012 ± 0.02	**0.020**↑ ± 0.01	**0.034**↑ ± 0.00	**0.040**↑ ± 0.05	**0.034**↑ ± 0.08	**0.034**↑ ± 0.08
8	0.026 ± 0.04	**0.065**↑ ± 0.04	**0.124**↑ ± 0.02	**0.104**↑ ± 0.06	**0.104**↑ ± 0.09	**0.122**↑ ± 0.03
9	0.004 ± 0.07	**0.018**↑ ± 0.04	**0.057**↑ ± 0.10	**0.073**↑ ± 0.09	**0.066**↑ ± 0.04	**0.049**↑ ± 0.02
10	0.013 ± 0.04	**0.067**↑ ± 0.02	**0.076**↑ ± 0.01	**0.073**↑ ± 0.09	**0.076**↑ ± 0.04	**0.072**↑ ± 0.07
11	0.041 ± 0.02	**0.065**↑ ± 0.03	**0.099**↑ ± 0.04	**0.095**↑ ± 0.05	**0.104**↑ ± 0.10	**0.145**↑ ± 0.00
12	0.024 ± 0.04	0.024 ± 0.05	**0.062**↑ ± 0.02	**0.042**↑ ± 0.02	**0.052**↑ ± 0.03	**0.045**↑ ± 0.06
13	0.024 ± 0.03	**0.056**↑ ± 0.03	**0.114**↑ ± 0.10	**0.085**↑ ± 0.04	**0.065**↑ ± 0.07	**0.092**↑ ± 0.03
14	0.009 ± 0.10	**0.038**↑ ± 0.10	**0.053**↑ ± 0.03	**0.082**↑ ± 0.10	**0.082**↑ ± 0.02	**0.085**↑ ± 0.08
15	0.015 ± 0.07	**0.030**↑ ± 0.07	**0.036**↑ ± 0.09	**0.041**↑ ± 0.03	**0.055**↑ ± 0.08	**0.032**↑ ± 0.06
16	0.017 ± 0.00	**0.095**↑ ± 0.10	**0.117**↑ ± 0.04	**0.129**↑ ± 0.04	**0.127**↑ ± 0.06	**0.138**↑ ± 0.02
17	0.005 ± 0.03	**0.048**↑ ± 0.07	**0.051**↑ ± 0.02	**0.073**↑ ± 0.08	**0.083**↑ ± 0.02	**0.073**↑ ± 0.09
18	0.009 ± 0.06	**0.030**↑ ± 0.00	**0.037**↑ ± 0.01	**0.032**↑ ± 0.00	**0.034**↑ ± 0.07	**0.045**↑ ± 0.03
19	0.007 ± 0.07	**0.060**↑ ± 0.00	**0.081**↑ ± 0.08	**0.133**↑ ± 0.05	**0.132**↑ ± 0.03	**0.147**↑ ± 0.04
20	0.004 ± 0.01	-0.045↓ ± 0.10	-0.042↓ ± 0.04	-0.048↓ ± 0.02	-0.042↓ ± 0.03	-0.051↓ ± 0.06
21	0.015 ± 0.08	0.015 ± 0.03	**0.024**↑ ± 0.04	**0.039**↑ ± 0.08	**0.042**↑ ± 0.06	**0.045**↑ ± 0.05
22	0.001 ± 0.03	**0.045**↑ ± 0.06	**0.054**↑ ± 0.05	**0.054**↑ ± 0.09	**0.054**↑ ± 0.05	**0.049**↑ ± 0.05
23	0.033 ± 0.01	**0.076**↑ ± 0.05	**0.089**↑ ± 0.03	**0.144**↑ ± 0.05	**0.149**↑ ± 0.06	**0.167**↑ ± 0.01
Average	0.017 ± 0.02	0.056 ± 0.05	0.079 ± 0.06	0.086 ± 0.07	0.088 ± 0.06	**0.092** ± 0.07
• significance – *p* value		0.001	0.000	0.000	0.000	**0.000**
• win/loss/draw		20/1/2	22/1/0	22/1/0	22/1/0	**22/1/0**
• sig.win/sig.loss		19/1	21/1	21/1	21/1	**21/1**

Table 18: Average rule unusualness with standard deviation (*WRACC* ± *sd*) for different variants of the unordered algorithm using 10-fold stratified cross-validation.

#	**CN2 standard** *ACC* ± *sd*	**CN2 WRAcc** *ACC* ± *sd*	**CN2-SD (γ = 0.5)** *ACC* ± *sd*	**CN2-SD (γ = 0.7)** *ACC* ± *sd*	**CN2-SD (γ = 0.9)** *ACC* ± *sd*	**CN2-SD (add.)** *ACC* ± *sd*
1	81.62 ± 3.55	**85.53**↑ ± 0.14	**89.27**↑ ± 8.04	**87.61**↑ ± 8.71	**87.81**↑ ± 6.54	**88.35**↑ ± 8.60
2	92.28 ± 1.07	92.13 ± 5.95	95.80 ± 1.21	95.56 ± 3.15	92.53 ± 1.52	92.60 ± 2.28
3	82.45 ± 3.89	81.36 ± 1.30	84.13 ± 8.88	84.07 ± 6.11	84.81 ± 1.06	81.46 ± 2.24
4	94.18 ± 3.71	94.34 ± 2.25	97.19 ± 0.35	97.37 ± 0.42	96.54 ± 1.77	96.08 ± 1.22
5	72.77 ± 9.33	73.81 ± 0.91	**78.66**↑ ± 8.65	**78.80**↑ ± 0.04	**78.81**↑ ± 5.55	74.12 ± 9.97
6	68.71 ± 1.79	67.12 ± 6.55	68.62 ± 0.96	70.08 ± 6.28	**71.20**↑ ± 9.94	68.75 ± 5.32
7	72.40 ± 7.60	71.40 ± 7.57	73.73 ± 0.72	**75.82**↑ ± 8.07	74.67 ± 5.85	72.40 ± 7.36
8	74.10 ± 4.15	77.06 ± 7.06	**79.64**↑ ± 6.98	77.53 ± 4.77	**78.48**↑ ± 3.16	**78.03**↑ ± 2.70
9	80.74 ± 7.59	83.26 ± 0.83	**87.87**↑ ± 2.29	**87.75**↑ ± 0.36	**86.97**↑ ± 6.88	**86.14**↑ ± 1.99
10	98.58 ± 0.60	98.54 ± 0.11	99.86 ± 0.03	99.37 ± 0.06	99.77 ± 0.02	99.10 ± 0.40
11	91.44 ± 6.62	88.87↓ ± 7.26	93.25 ± 2.89	90.53 ± 1.44	92.41 ± 4.96	91.10 ± 3.76
12	91.33 ± 2.04	91.33 ± 7.02	**95.08**↑ ± 2.08	94.40 ± 0.94	91.77 ± 6.33	91.75 ± 2.28
13	80.87 ± 1.32	79.74 ± 1.74	83.81 ± 6.59	**84.23**↑ ± 7.59	81.41 ± 0.76	80.86 ± 7.26
14	72.28 ± 2.81	76.60 ± 3.10	**77.59**↑ ± 2.84	**78.35**↑ ± 5.11	**80.40**↑ ± 3.31	**77.74**↑ ± 1.69
15	98.01 ± 0.60	76.40↓ ± 3.75	77.59↓ ± 1.81	77.94↓ ± 0.63	80.26↓ ± 7.99	77.38↓ ± 4.97
16	94.24 ± 0.39	95.63 ± 1.83	**97.67**↑ ± 1.62	**99.09**↑ ± 0.14	**99.85**↑ ± 0.04	**97.62**↑ ± 1.05
17	74.71 ± 8.62	72.49 ± 0.48	72.55 ± 9.85	**77.08**↑ ± 8.89	76.90 ± 0.86	72.51 ± 5.30
18	89.82 ± 5.33	70.33↓ ± 7.94	74.21↓ ± 5.66	70.37↓ ± 7.81	70.56↓ ± 7.49	72.48↓ ± 1.62
19	60.60 ± 1.83	**68.13**↑ ± 3.76	**72.70**↑ ± 8.05	**71.12**↑ ± 5.40	**71.46**↑ ± 7.62	**69.32**↑ ± 0.08
20	58.88 ± 5.70	17.84↓ ± 2.33	22.47↓ ± 1.06	19.84↓ ± 1.48	21.98↓ ± 1.86	19.49↓ ± 1.18
21	88.73 ± 3.01	69.68↓ ± 4.14	70.71↓ ± 9.94	72.29↓ ± 8.70	74.23↓ ± 1.22	71.04↓ ± 7.45
22	69.18 ± 8.92	**74.26**↑ ± 1.32	**77.71**↑ ± 9.31	**79.11**↑ ± 1.26	**78.56**↑ ± 9.60	**75.70**↑ ± 7.67
23	89.16 ± 1.33	90.90 ± 1.18	91.08 ± 5.23	**95.12**↑ ± 1.01	**93.26**↑ ± 0.67	91.32 ± 2.97
Average	**81.61** ± 11.66	78.12 ± 16.28	80.92 ± 16.04	81.02 ± 16.44	81.07 ± 15.78	79.36 ± 16.24
• significance – *p* value		0.150	0.771	0.812	0.818	0.344
• win/loss/draw		10/12/1	17/6/0	18/5/0	19/4/0	15/8/0
• sig.win/sig.loss		3/5	9/4	11/4	10/4	7/4

Table 19: Average rule set accuracy with standard deviation (*ACC* ± *sd*) for different variants of the unordered algorithm using 10-fold stratified cross-validation.

#	**CN2 standard** *AUC* ± *sd*	**CN2 WRAcc** *AUC* ± *sd*	**CN2-SD ($\gamma = 0.5$)** *AUC* ± *sd*	**CN2-SD ($\gamma = 0.7$)** *AUC* ± *sd*	**CN2-SD ($\gamma = 0.9$)** *AUC* ± *sd*	**CN2-SD (add.)** *AUC* ± *sd*
1	33.39 ± 5.61	**86.12**↑ ± 0.05	**83.31**↑ ± 2.01	**84.27**↑ ± 9.44	**84.47**↑ ± 6.05	**85.12**↑ ± 5.16
2	90.74 ± 3.57	89.52 ± 7.26	**94.37**↑ ± 2.29	**96.28**↑ ± 1.47	**97.33**↑ ± 0.98	**94.52**↑ ± 1.67
3	84.51 ± 0.15	80.11↓ ± 9.84	82.58 ± 5.60	80.98↓ ± 8.12	78.38↓ ± 7.44	83.03 ± 1.88
4	96.22 ± 2.55	93.59 ± 2.26	97.19 ± 0.76	92.37↓ ± 2.33	96.54 ± 1.90	92.87↓ ± 2.66
5	71.33 ± 7.86	**80.75**↑ ± 0.51	**80.52**↑ ± 1.82	**80.56**↑ ± 8.17	**80.76**↑ ± 5.02	**80.06**↑ ± 3.49
6	70.53 ± 5.99	64.42↓ ± 3.29	68.09 ± 7.34	68.63 ± 2.44	64.02↓ ± 8.71	70.61 ± 2.46
7	71.99 ± 5.76	74.00 ± 7.19	73.99 ± 7.63	73.92 ± 6.01	**75.29**↑ ± 7.70	72.73 ± 3.84
8	74.17 ± 5.35	73.98 ± 0.90	**83.82**↑ ± 9.76	**84.69**↑ ± 0.63	**87.02**↑ ± 9.80	**85.62**↑ ± 1.84
9	78.81 ± 4.64	**85.65**↑ ± 0.33	**84.82**↑ ± 2.78	**82.80**↑ ± 5.19	78.66 ± 6.12	**81.29**↑ ± 0.23
10	96.22 ± 2.31	**98.59**↑ ± 0.10	97.13 ± 0.78	96.54 ± 0.13	**99.65**↑ ± 0.04	97.42 ± 0.24
11	94.46 ± 1.52	90.86↓ ± 0.32	93.17 ± 2.68	93.99 ± 2.83	94.30 ± 2.10	93.87 ± 1.07
12	99.17 ± 0.23	99.17 ± 0.16	99.96 ± 0.01	99.38 ± 0.15	99.92 ± 0.03	99.46 ± 0.06
13	83.20 ± 8.68	78.38↓ ± 2.33	82.11 ± 1.04	84.74 ± 4.51	80.12↓ ± 4.12	83.06 ± 6.97
14	75.06 ± 6.13	**79.41**↑ ± 5.12	**81.62**↑ ± 7.61	**79.97**↑ ± 1.29	**80.12**↑ ± 5.34	**78.51**↑ ± 1.15
15	97.90 ± 0.36	78.90↓ ± 6.95	91.88↓ ± 2.73	91.28↓ ± 2.63	90.87↓ ± 2.01	89.15↓ ± 4.32
16	96.88 ± 1.67	96.41 ± 1.63	93.44↓ ± 2.97	95.35 ± 0.18	94.82 ± 1.06	93.95↓ ± 2.06
Average	82.16 ± 16.81	84.37 ± 9.87	**86.75** ± 8.95	86.61 ± 8.81	86.39 ± 10.32	86.33 ± 8.60
• significance – *p* value		0.563	**0.175**	0.198	0.236	0.236
• win/loss/draw		6/9/1	**10/6/0**	10/6/0	9/7/0	10/6/0
• sig.win/sig.loss		5/5	**6/2**	6/3	7/4	6/3

Table 20: Area under the ROC curve (AUC-Method-2) with standard deviation (*AUC* ± *sd*) for different variants of the unordered algorithm using 10-fold stratified cross-validation.

References

Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*, AAAI Press:307–328, 1996.

Bojan Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–149, Pitman, 1990.

Peter Clark and Robin Boswell. Rule induction with cn2: Some recent improvements. In *Proceedings of the Fifth European Working Session on Learning*, pages 151–163, Springer, 1991.

Peter Clark and Tim Niblett. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

William W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, Morgan Kaufmann, 1995.

William W. Cohen and Yoram Singer. A simple, fast, and effective rule learner. In *Proceedings of AAAI/IAAI*, pages 335–342, AAAI Press, 1999.

Sašo Džeroski, Bojan Cestnik, and Igor Petrovski. Using the m-estimate in rule induction. *Journal of Computing and Information Technology*, 1(1):37–46, 1993.

Cesar Ferri-Ramírez, Peter A. Flach, and Jose Hernandez-Orallo. Learning decision trees using the area under the roc curve. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 139–146, Morgan Kaufmann, 2002.

Peter A. Flach. The geometry of roc space: Understanding machine learning metrics through roc isometrics. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 194–201, AAAI Press, 2003.

Peter A. Flach and Iztok Savnik. Database dependency discovery: A machine learning approach. *AI Communications*, 12(3):139–160, 1999.

Yoav Freund and Robert E. Shapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, Morgan Kaufmann, 1996.

Jerome H. Friedman and Nicholas I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9:123–143, 1999.

Johannes Fürnkranz and Peter A. Flach. An analysis of rule evaluation metrics. In *Proceedings of the Twetieth International Conference on Machine Learning*, pages 202–209, AAAI Press, 2003.

Dragan Gamberger and Nada Lavrač. Expert guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.

A.D. Gordon. *Classification*. Chapman and Hall, London, 1982.

K. Chidananda Gowda and Edwin Diday. Symbolic clustering using a new dissimilarity measure. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):567–578, 1992.

David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.

David Hsu, Oren Etzioni, and Stephen Soderland. A redundant covering algorithm applied to text classification. In *Proceedings of the AAAI Workshop on Learning from Text Categorization*, AAAI Press, 1998.

Viktor Jovanoski and Nada Lavrač. Classification rule learning with apriori-c. In *Progress in Artificial Intelligence: Proceedings of the Tenth Portuguese Conference on Artificial Intelligence*, pages 44–51, Springer, 2001.

Branko Kavšek, Nada Lavrač, and Viktor Jovanoski. Apriori-sd: Adapting association rule learning to subgroup discovery. In *Proceedings of the Fifth International Symposium on Intelligent Data Analysis*, pages 230–241, Springer, 2003.

Willi Klösgen. Explora: A multipattern and multistrategy discovery assistant. *Advances in Knowledge Discovery and Data Mining*, MIT Press:249–271, 1996.

Arno J. Knobbe, Marc de Haas, and Arno Siebes. Propositionalisation and aggregates. In *Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 277–288, Springer, 2001.

Arno J. Knobbe, Arno Siebes, and Bart Marseille. Involving aggregate functions in multi-relational search. In *Proceedings of the Sixth European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 287–298, Springer, 2002.

Pat Langley. *Elements of Machine Learning*. Morgan Kaufmann, 1996.

Nada Lavrač, Peter A. Flach, Branko Kavšek, and Ljupčo Todorovski. Adapting classification rule induction to subgroup discovery. In *Proceedings of the Second IEEE International Conference on Data Mining*, pages 266–273, IEEE Computer Society, 2002.

Nada Lavrač, Peter A. Flach, and Blaž Zupan. Rule evaluation measures: A unifying view. In *Proceedings of the Nineth International Workshop on Inductive Logic Programming*, pages 74–185, Springer, 1999.

Nada Lavrač, Filip Železný, and Peter A. Flach. Rsd: Relational subgroup discovery through first-order feature construction. In *Proceedings of the Twelfth International Conference on Inductive Logic Programming*, pages 149–165, Springer, 2003.

Yongwon Lee, Bruce G. Buchanan, and John M. Aronis. Knowledge-based learning in exploratory science: Learning rules to predict rodent carcinogenicity. *Machine Learning*, 30:217–240, 1998.

Peter Ljubič, Ljupčo Todorovski, Nada Lavrač, and John C. Bullas. Time-series analysis of uk traffic accident data. In *Proceedings of the Fifth International Multi-conference Information Society*, pages 131–134, 2002.

Ryszard S. Michalski. Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):349–361, 1980.

Ryszard S. Michalski, Igor Mozetič, Jiarong Hong, and N. Lavrač. The multi-purpose incremental learning system aq15 and its testing application on three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1041–1045, Morgan Kaufmann, 1986.

Patrick M. Murphy and David W. Aha. Uci repository of machine learning databases. Available electronically at http://www.ics.uci.edu/ mlearn/MLRepository.html, 1994.

Foster J. Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

Luc De Raedt and Hendrik Blockeel. Using logical decision trees for clustering. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 133–140, Springer, 1997.

Luc De Raedt, Hendrik Blockeel, Luc Dehaspe, and Wim Van Laer. Three companions for data mining in first order logic. *Relational Data Mining*, Springer:106–139, 2001.

Luc De Raedt and Luc Dehaspe. Clausal discovery. *Machine Learning*, 26:99–146, 1997.

Ronald L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Conference on Computational Learning Theory*, pages 80–91, ACM Press, 1998.

Avi Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proceedigns of the First International Conference on Knowledge Discovery and Data Mining*, pages 275–281, 1995.

R.R. Sokal and Peter H.A. Sneath. *Principles of Numerical Taxonomy*. Freeman, San Francisco, 1963.

Ljupčo Todorovski, Peter A. Flach, and Nada Lavrač. Predictive performance of weighted relative accuracy. In *Proceedings of the Fourth European Conference on Principles of Data Mining and Knowledge Discovery*, pages 255–264, Springer, 2000.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proceedings of the First European Conference on Principles of Data Mining and Knowledge Discovery*, pages 78–87, Springer, 1997.

Stefan Wrobel. Inductive logic programming for knowledge discovery in databases. *Relational Data Mining*, Springer:74–101, 2001.

Journal of Machine Learning Research 5 (2004) 189–217 Submitted 8/03; Published 2/04

Generalization Error Bounds for Threshold Decision Lists

Martin Anthony M.ANTHONY@LSE.AC.UK
Department of Mathematics
London School of Economics
London WC2A 2AE
United Kingdom

Editor: Yoram Singer

Abstract

In this paper we consider the generalization accuracy of classification methods based on the iterative use of linear classifiers. The resulting classifiers, which we call *threshold decision lists* act as follows. Some points of the data set to be classified are given a particular classification according to a linear threshold function (or hyperplane). These are then removed from consideration, and the procedure is iterated until all points are classified. Geometrically, we can imagine that at each stage, points of the same classification are successively chopped off from the data set by a hyperplane. We analyse theoretically the generalization properties of data classification techniques that are based on the use of threshold decision lists and on the special subclass of *multilevel threshold functions*. We present bounds on the generalization error in a standard probabilistic learning framework. The primary focus in this paper is on obtaining generalization error bounds that depend on the levels of separation—or *margins*—achieved by the successive linear classifiers. We also improve and extend previously published theoretical bounds on the generalization ability of perceptron decision trees.

Keywords: Threshold decision lists, generalization error, large margin bounds, growth function, covering numbers, perceptron decision trees

1. Introduction

This paper concerns the use of *threshold decision lists* for classifying data into two classes. The use of such methods has a natural geometrical interpretation and can be appropriate for an iterative or sequential approach to data classification, in which some points of the data set are given a particular classification, according to a linear threshold function (or hyperplane), are then removed from consideration, and the procedure iterated until all points are classified. We analyse theoretically the generalization properties of data classification techniques that are based on the use of threshold decision lists and the subclass of *multilevel threshold functions*. This analysis is carried out within the framework of the probabilistic PAC model of learning and its variants (see Valiant, 1984; Vapnik, 1998; Anthony and Biggs, 1992; Anthony and Bartlett, 1999; Blumer et al., 1989).

1.1 Outline of the Paper

Probabilistic approaches to the theory of machine learning can provide bounds on the 'generalization error' of classifiers. Such results give probabilistic guarantees on the future performance of a

classifier trained on a large random training set. This paper takes three main approaches to bounding the generalization error of threshold decision lists.

First, in the 'classical' approach to the PAC model, we present results on generalization that are obtained through bounding the growth function of these classes.

Secondly, we obtain bounds on the generalization error of threshold decision lists that depend on the levels of separation—or *margins*—achieved by the successive linear classifiers. We use techniques inspired by Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000), and also give generalization bounds for *perceptron decision trees*, improving upon and extending previous such results from Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000).

Thirdly, we focus specifically on the special subclass comprising the multilevel threshold functions. Here, a different and more specialized analysis results in generalization error bounds that are better than those that follow from the general results on threshold decision lists.

The rest of this section introduces the classes of threshold decision lists and multilevel threshold functions, and discusses related work. Section 2 discusses the definitions of generalization error. In Section 3, we derive bounds on the growth function and use these to bound the generalization error. Section 4 discusses the important idea of large-margin classification as it applies to threshold decision lists. Here, we give generalization error bounds that depend on the sizes of the margins and we also indicate some improved bounds for perceptron decision trees. Section 5 discusses the more specific margin-based analysis for multilevel threshold functions. Section 6 concludes the paper and suggests some possible directions for future work.

1.2 Threshold Decision Lists

Suppose that F is any set of functions from $\mathbb{R}^n$ to $\{0,1\}$, for some fixed $n \in \mathbb{N}$. A function $f : \mathbb{R}^n \to \{0,1\}$ is a *decision list* based on F if it can be evaluated as follows, for some $k \in \mathbb{N}$, some functions $f_1, f_2, \ldots, f_k \in F$, some $c_1, c_2, \ldots, c_k \in \{0,1\}$, and all $y \in \mathbb{R}^n$: if $f_1(y) = 1$, then $f(y) = c_1$; if not, we evaluate $f_2(y)$, and if $f_2(y) = 1$, then $f(y) = c_2$; otherwise we evaluate $f_3(y)$, and so on. If y fails to satisfy any f_i then $f(y)$ is given the default value 0. We can regard a decision list based on F as a finite sequence

$$f = (f_1, c_1), (f_2, c_2), \ldots, (f_r, c_r),$$

such that $f_i \in F$ and $c_i \in \{0,1\}$ for $1 \leq i \leq r$. The values of f are defined by $f(y) = c_j$ where $j = \min\{i : f_i(y) = 1\}$, or 0 if there are no j such that $f_j(y) = 1$. We call each f_i a *test*, and the pair (f_i, c_i) a *term* of the decision list. Decision lists were introduced by Rivest (1987), in the context of learning Boolean functions (and where the tests were conjunctions of literals).

A function $t : \mathbb{R}^n \to \{0,1\}$ is a *threshold function* if there are $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ such that

$$t(x) = \begin{cases} 1 & \text{if } \langle w, x \rangle \geq \theta \\ 0 & \text{if } \langle w, x \rangle < \theta, \end{cases}$$

where $\langle w, x \rangle$ is the standard inner product of w and x. Thus, $t(x) = \mathrm{sgn}(\langle w, x \rangle - \theta)$, where $\mathrm{sgn}(z) = 1$ if $z \geq 0$ and $\mathrm{sgn}(z) = 0$ if $z < 0$. Given such w and θ, we say that t is represented by $[w, \theta]$ and we write $t \leftarrow [w, \theta]$. The vector w is known as the *weight vector*, and θ is known as the *threshold*. Geometrically, a threshold function is defined by a hyperplane: all points lying to one side of the plane and on the plane are given the value 1, and all points on the other side are given the value 0.

Threshold decision lists are decision lists in which the tests are threshold functions. These have also been called *neural* decision lists by Marchand and Golea (1993) and *linear* decision lists by

Turan and Vatan (1997). Formally, a threshold decision list

$$f = (t_1, c_1), (t_2, c_2), \ldots, (t_r, c_r)$$

has each $t_i : \mathbb{R}^n \to \{0,1\}$ of the form $t_i(x) = \mathrm{sgn}(\langle w_i, x\rangle - \theta_i)$ for some $w_i \in \mathbb{R}^n$ and $\theta_i \in \mathbb{R}$. The value of f on $y \in \mathbb{R}^n$ is $f(y) = c_j$ if $j = \min\{i : t_i(y) = 1\}$ exists, or 0 otherwise (that is, if there are no j such that $t_j(y) = 1$).

There is a natural geometrical interpretation of the use of threshold decision lists. Suppose we are given some data points in $\mathbb{R}^n$, each one of which is labeled 0 or 1. It is unlikely that the positive and negative points can be separated by a hyperplane. However, we could use a hyperplane to separate off a set of points all of the same classification (either all are positive points or all are negative points). These points can then be removed from consideration and the procedure iterated until no points remain. This procedure is similar in nature to one of Jeroslow (1975), but at each stage in his procedure, only positive examples may be 'chopped off' (not positive *or* negative).

If we consider threshold decision lists in which the hyperplanes are parallel, we obtain a special subclass, known as the *multilevel threshold functions*. A *k-level threshold function* f is one that is representable by a threshold decision list of length k in which the test hyperplanes are parallel to each other. Any such function is defined by k parallel hyperplanes, which divide $\mathbb{R}^n$ into $k+1$ regions. The function assigns points in the same region the same value, either 0 or 1. Without any loss, we may suppose that the classifications assigned to points in neighboring regions are different (for, otherwise, at least one of the planes is redundant); thus, the classifications alternate as we traverse the regions in the direction of the normal vector common to the hyperplanes.

1.3 Related Work

The chopping procedure described above suggests that the use of threshold decision lists is fairly natural, if an iterative approach is to be taken to pattern classification. Other iterative approaches—which proceed by classifying some points, removing these from consideration, and proceeding recursively—have been taken, using different types of base classifier. For example, Magasarian's multisurface method (Mangasarian, 1968) finds, at each stage, two parallel hyperplanes (as close together as possible) such that the points not enclosed between the two planes all have the same classification. It then removes these points and repeats. This method may be regarded as constructing a decision list in which the set of base functions F are the indicator functions of the complements of regions enclosed between two parallel hyperplanes.

The focus of this paper is generalization error rather than learning algorithms. The 'chopping procedure' as we have described it is a useful device to help us see that threshold decision lists have a fairly natural geometric interpretation. However, the algorithmic practicalities of implementing such a procedure have been investigated by Marchand and Golea (1993). They propose a method that relies on an incremental approximation algorithm for the NP-hard problem of finding at each stage a hyperplane that chops off as many remaining points as possible (the 'densest hyperplane problem'). Reports on the experimental performance of their method can be found in Marchand and Golea (1993).

Threshold decision lists are special types of *perceptron decision trees*, decision trees in which the decision nodes compute threshold functions. Such trees have been studied by Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000), where the importance of large margins was emphasised. The techniques used to derive the results of Section 4 extend those used to derive generalization error

bounds in those papers. Bennett et al. (2000) consider learning algorithms for perceptron decision trees: specifically, they propose and test three variants of the OC1 perceptron decision tree learning algorithm (Murthy et al., 1994) that aim for a large margin of separation at each decision node. The theoretical generalization error bounds they derive apply to the case in which a perceptron decision tree is produced that is consistent with the training sample. The bounds for perceptron decision trees presented in this paper improve upon the bounds presented there and also apply, more generally, to the case in which some empirical error (measured with respect to the margins) is permitted.

The representational properties of threshold decision lists and multilevel threshold functions have been studied by a number of researchers, particularly in the context of Boolean functions. We mentioned above the paper of Jeroslow (1975). There, it is shown, essentially, that any Boolean function can be realized as a disjunction of threshold functions (and hence as a special type of threshold decision list). The general problem of decomposing a Boolean function into a disjunction of threshold functions has been considered independently of any machine learning considerations. Hammer et al. (1981) defined the *threshold number* of a Boolean function to be the minimum s such that f is a disjunction of s threshold functions; they and Zuev and Lipkin (1988) obtained results on the threshold numbers of increasing Boolean functions. Although any Boolean function can be expressed as a disjunction of threshold functions, threshold decision lists provide a more flexible representation. For instance, the parity function on n variables (in which the output is 1 precisely when the input to the function contains an odd number of entries equal to 1) can be represented by a threshold decision list with n terms; whereas, as observed by Jeroslow (1975), the shortest decomposition of parity into a disjunction of threshold functions involves 2^{n-1} threshold functions. Turan and Vatan (1997), by contrast, gave a specific example of a function with a necessarily long threshold decision list representation.

Decision lists in which the tests are defined with respect to points in the training sample have recently been investigated by Sokolova et al. (2003). They considered the case where the base class of tests consists of data-dependent balls (that is, the characteristic functions of balls centered on data points, and their complements). Additionally, Marchand et al. (2003) considered the use of disjunctions and conjunctions of functions constructed as threshold functions, possibly in some 'feature space'. Here, examples $x \in X$ are transformed by a fixed function ϕ into points of the feature space $\phi(X)$, and the classifiers used are disjunctions or conjunctions of functions that, acting in feature space and on transformed examples $\phi(x)$, are threshold functions with weight vectors defined by three of the transformed examples. The problems studied in this paper are rather different: here, we consider general threshold decision lists (rather than just conjunctions or disjunctions), and the individual tests need not be data-dependent (or, at least, not in the explicit way that they are in Marchand et al., 2003).

Multilevel threshold functions have been studied in a number of papers (Bohossian and Bruck, 1998; Olafsson and Abu-Mostafa, 1988; Takiyama, 1985, for instance). They originally were of interest as the sets of functions computed by devices knows as *multilevel threshold elements* (Takiyama, 1985), generalizations of the linear threshold elements. The 'capacity' (in our terminology, the growth function) has been of particular interest. Olafsson and Abu-Mostafa (1988) gave an upper bound on the capacity, correcting a claimed upper bound of Takiyama (1985). Subsequently, Ngom et al. (2003) claimed to have improved this bound, but were mistaken (Anthony, 2002). A bound improving upon that of Olafsson and Abu-Mostafa (1988), and which is used in this paper, was given in Anthony (2002). Just as threshold decision lists (and disjunctions of threshold functions) are 'universal' for Boolean functions, so too are the multilevel threshold functions. Bohossian and

Bruck (1998) observed that any Boolean function can be realized as a multilevel threshold function. (Specifically, they showed that every Boolean function is a 2^n-level threshold function, an appropriate weight-vector being $w = (2^{n-1}, 2^{n-2}, \ldots, 2, 1)$. For that reason, they paid particular attention to the question of whether a function can be computed by a multilevel threshold function where the number of levels is polynomial.) Functions similar to multilevel threshold functions have also been of interest in multiple-valued logic (Obradović and Parberry, 1994; Ngom et al., 2003) where, instead of classification labels alternating between 0 and 1, a partition by k parallel planes defines a $(k+1)$-valued function.

2. Generalization Error

Following a form of the PAC model of computational learning theory (see Anthony and Biggs, 1992; Vapnik, 1998; Blumer et al., 1989), we assume that labeled data points (x, b) (where $x \in \mathbb{R}^n$ and $b \in \{0,1\}$) have been generated randomly (perhaps from some larger corpus of data) according to a fixed probability distribution P on $Z = \mathbb{R}^n \times \{0,1\}$. (Note that this includes as a special case the situation in which x is drawn according to a fixed distribution μ on $\mathbb{R}^n$ and the label b is then given by $b = t(x)$ where t is some fixed function.) Thus, if there are m data points, we may regard the data set as a *sample* $s = ((x_1, b_1), \ldots, (x_m, b_m)) \in Z^m$, drawn randomly according to the product probability distribution P^m. Suppose that H is a set of functions from X to $\{0,1\}$. Given any function $f \in H$, we can measure how well f matches the sample s through its *sample error*,

$$\mathrm{er}_s(f) = \frac{1}{m} |\{i : f(x_i) \neq b_i\}|,$$

(the proportion of points in the sample incorrectly classified by f). An appropriate measure of how well f would perform on further examples is its *error*,

$$\mathrm{er}_P(f) = P\left(\{(x,b) \in Z : f(x) \neq b\}\right),$$

the probability that a further randomly drawn labeled data point would be incorrectly classified by f.

Much effort has gone into obtaining high-probability bounds on $\mathrm{er}_P(f)$ in terms of the sample error. A typical result would state that, for all $\delta \in (0,1)$, with probability at least $1-\delta$, for all $h \in H$, $\mathrm{er}_P(h) < \mathrm{er}_s(h) + \varepsilon(m,\delta)$, where $\varepsilon(m,\delta)$ (known as a *generalization error bound*) is decreasing in m and δ. Such results can be derived using uniform convergence theorems from probability theory (Vapnik and Chervonenkis, 1971; Pollard, 1984; Dudley, 1999), in which case $\varepsilon(m,\delta)$ would typically involve the growth function (see Vapnik and Chervonenkis, 1971; Blumer et al., 1989; Vapnik, 1998; Anthony and Bartlett, 1999). We indicate in the next section how this may be done for threshold decision lists.

Recently, emphasis has been placed in practical machine learning techniques such as Support Vector Machines (see Cristianini and Shawe-Taylor, 2000, for instance) on 'learning with a large margin'. (See Bartlett et al., 2000; Anthony and Bartlett, 1999, 2000; Shawe-Taylor et al., 1996, for example). Broadly speaking, the rationale behind margin-based generalization error bounds is that if a classifier has managed to achieve a 'wide' separation between (most of) the points of different classification, then this indicates that it is a good classifier, and it is possible that a better (that is, smaller) generalization error bound can be obtained. The classical example of this is linear separation, where the classifier is a linear threshold function. If we have found a linear threshold function

that classifies the points of a sample correctly *and*, moreover, the points of opposite classifications are separated by a wide margin (so that the hyperplane achieves not just a correct, but a 'definitely' correct classification), then this function might be a better classifier of future, unseen, points than one which 'merely' separates the points correctly, but with a small margin. In Section 4, we apply such ideas to threshold decision lists.

3. Generalization Bounds Based on the Growth Function

In this section, we use some by-now classical techniques of computational or statistical learning theory to bound the generalization error.

3.1 Bounding the Error

The *growth function* of a set of functions H mapping from $X = \mathbb{R}^n$ to $\{0,1\}$ is defined as follow (Blumer et al., 1989; Vapnik and Chervonenkis, 1971). Let $\Pi_H : \mathbb{N} \to \mathbb{N}$ be given by

$$\Pi_H(m) = \max\{|H|_S| : S \subseteq X, |S| = m\},$$

where $H|_S$ denotes H restricted to domain S. Note that $\Pi_H(m) \leq 2^m$ for all m. The key probability results we employ are the following bounds, due respectively to Vapnik and Chervonenkis (1971) and Blumer et al. (1989) (see also Anthony and Bartlett, 1999): for any $\varepsilon \in (0,1)$,

$$P^m(\{s \in Z^m : \text{there exists } f \in H,\ \mathrm{er}_P(f) \geq \mathrm{er}_s(f) + \varepsilon\}) < 4\Pi_H(2m)e^{-m\varepsilon^2/8},$$

and, for $m \geq 8/\varepsilon$,

$$P^m(\{s \in Z^m : \text{there exists } f \in H,\ \mathrm{er}_s(f) = 0 \text{ and } \mathrm{er}_P(f) \geq \varepsilon\}) < 2\Pi_H(2m)2^{-\varepsilon m/2}.$$

Thus, we can obtain (probabilistic) bounds on the error $\mathrm{er}_P(f)$ of a function from a class H when we know something about the growth function of H.

3.2 Growth Function Bounds

We first consider the set of threshold decision lists on $\mathbb{R}^n$ with some number k of terms. (So, the length of the list is no more than k.)

Theorem 1 *Let H be the set of threshold decision lists on $\mathbb{R}^n$ with k terms, where $n, k \in \mathbb{N}$. Then*

$$\Pi_H(m) < 4^k \left(\sum_{i=0}^{n} \binom{m-1}{i} \right)^k.$$

Proof: Let S be any set of m points in $\mathbb{R}^n$. Suppose we have two decision lists

$$f = (f_1, c_1), \ldots, (f_k, c_k),\ \ g = (g_1, d_1), \ldots, (g_k, d_k)$$

in H, where the f_i and g_j are threshold functions on $\mathbb{R}^n$. Clearly, f and g will agree on all points of S if (i) $c_i = d_i$ for each i and (ii) $f_i(x) = g_i(x)$ for all $x \in S$. For fixed i, condition (ii) is an equivalence relation among threshold functions. The number of equivalence classes is $|K|_S|$ where K is the set

of threshold functions. This is bounded by $\Pi_K(m)$, which, it is known (Cover, 1965; Blumer et al., 1989; Anthony and Bartlett, 1999), is bounded above as follows:

$$\Pi_K(m) \leq 2\sum_{i=0}^{n}\binom{m-1}{i}.$$

We can therefore upper bound $|H|_S|$ as follows:

$$|H|_S| \leq 2^k\left(2\sum_{i=0}^{n}\binom{m-1}{i}\right)^k.$$

Here, the first 2^k factor corresponds to the number of possible sequences of c_i and the remaining factor bounds the number of ways of choosing an equivalence class (with respect to S) of threshold functions, for each i from 1 to k. ■

There is a useful connection between certain types of decision list and threshold functions. We say that a decision list defined on $\{0,1\}^n$ is a 1-*decision list* if the Boolean function in each test is given by a formula that is a single literal. (So, for each i, there is some l_i such that *either* $f_i(y) = 1$ if and only if $y_{l_i} = 1$, *or* $f_i(y) = 1$ if and only if $y_{l_i} = 0$.) Then, it is known (Ehrenfeucht et al., 1989) (see also Anthony et al., 1995; Anthony, 2001) that any 1-decision list is a threshold function. In an easy analogue of this, any threshold decision list is a threshold function of threshold functions (Anthony, 2001). But a threshold function of threshold functions is nothing more than a two-layer threshold network, one of the simplest types of artificial neural network. (A similar observation was made by Marchand and Golea (1993) and Marchand et al. (1990), who construct a 'cascade' network from a threshold decision list.) So another way of bounding the growth function of threshold decision lists is to use this fact in combination with some known bounds (Baum and Haussler, 1989; Anthony and Bartlett, 1999) for the growth functions of linear threshold networks. This gives a similar, though slightly looser, upper bound.

To bound the growth function of the subclass consisting of k-level threshold functions, we use a result from (Anthony, 2002), which shows that the number of ways in which a set S of m points can be partitioned by k parallel hyperplanes is at most $\sum_{i=0}^{n+k-1}\binom{km}{i}$. (For fixed n and k, this bound is tight to within a constant, as a function of m.) Noting that we may assume adjacent regions to have different labels, there corresponds to each such partition at most two k-level threshold functions (defined on the domain restricted to S) and we therefore have the following bound.

Theorem 2 *Let H be the set of k-level threshold functions on $\mathbb{R}^n$. Then*

$$\Pi_H(m) \leq 2\sum_{i=0}^{n+k-1}\binom{km}{i}.$$

3.3 Generalization Error Bounds

Combining the results of the previous two subsections, we can obtain the following generalization error bounds.

Theorem 3 *Suppose that n and k are fixed positive integers and that s is a sample of m labeled points (x,b) of $Z = \mathbb{R}^n \times \{0,1\}$, each generated at random according to a fixed probability distribution P on Z. Let δ be any positive number less than one. Then the following hold with probability at least $1-\delta$:*

1. *If f is a threshold decision list with k terms, then the error $\mathrm{er}_P(f)$ of f and its sample error on s, $\mathrm{er}_s(f)$ are such that*

$$\mathrm{er}_P(f) < \mathrm{er}_s(f) + \sqrt{\frac{8}{m}\left(2k\ln 2 + nk\ln\left(\frac{e(2m-1)}{n}\right) + \ln\left(\frac{4}{\delta}\right)\right)}.$$

2. *If f is a k-level threshold function, then*

$$\mathrm{er}_P(f) < \mathrm{er}_s(f) + \sqrt{\frac{8}{m}\left((n+k-1)\ln\left(\frac{2emk}{n+k-1}\right) + \ln\left(\frac{4}{\delta}\right)\right)}.$$

Proof: We approximate the growth function of the class of k-term threshold decision lists by

$$\Pi_H(m) \le 4^k \left(\sum_{i=0}^{n} \binom{m-1}{i}\right)^k < 4^k \left(\frac{e(m-1)}{n}\right)^{nk},$$

for $m > n$. Similarly, when H is the class of k-level threshold functions,

$$\Pi_H(m) \le 2\sum_{i=0}^{n+k-1} \binom{km}{i} < 2\left(\frac{emk}{n+k-1}\right)^{n+k-1},$$

for $m \ge n+k$. The first part of the theorem is trivially true if $m \le n$ (since then the stated upper bound on the error is at least 1). If $m > n$, then

$$\varepsilon_0 = \sqrt{\frac{8}{m}\left(2k\ln 2 + nk\ln\left(\frac{e(2m-1)}{n}\right) + \ln\left(\frac{4}{\delta}\right)\right)} \ge \sqrt{\frac{8}{m}\left(\ln\left(\frac{4\Pi_H(2m)}{\delta}\right)\right)},$$

and so

$$P^m\left(\{s \in Z^m : \text{there exists } f \in H,\ \mathrm{er}_P(f) \ge \mathrm{er}_s(f) + \varepsilon_0\}\right) < 4\Pi_H(2m)e^{-m\varepsilon_0^2/8} \le \delta.$$

Thus, with probability at least $1-\delta$, for all $f \in H$, $\mathrm{er}_P(f) < \mathrm{er}_s(f) + \varepsilon_0$. The second part follows similarly. It is trivial for $m < n+k$ and it follows for $m \ge n+k$ on observing that, for H the class of k-level threshold functions,

$$\varepsilon_0' = \sqrt{\frac{8}{m}\left((n+k-1)\ln\left(\frac{2emk}{n+k-1}\right) + \ln\left(\frac{4}{\delta}\right)\right)} \ge \sqrt{\frac{8}{m}\left(\ln\left(\frac{4\Pi_H(2m)}{\delta}\right)\right)}.$$

■

For threshold decision lists that are consistent with a training sample, the following tighter bounds can be used.

Theorem 4 *Suppose that k and n are fixed positive integers and that s is a sample of m labeled points (x,b) of $Z = \mathbb{R}^n \times \{0,1\}$, each generated at random according to a fixed probability distribution P on Z. Let δ be any positive number less than one. Then the following hold with probability at least $1-\delta$:*

1. *If f is a threshold decision list with k terms and f is consistent with s (so that $\mathrm{er}_s(f) = 0$), then*
$$\mathrm{er}_P(f) < \frac{2}{m}\left(2k + nk\log_2\left(\frac{e(2m-1)}{n}\right) + \log_2\left(\frac{2}{\delta}\right)\right).$$
2. *If f is a k-level threshold function and f is consistent with s, then*
$$\mathrm{er}_P(f) < \frac{2}{m}\left((n+k-1)\log_2\left(\frac{2emk}{n+k-1}\right) + \log_2\left(\frac{2}{\delta}\right)\right),$$
for $n+k \geq 3$.

Proof: We use the growth function approximations of the proof of Theorem 3. For the class of threshold decision lists with k terms, and for $m > n$,
$$\varepsilon_0 = \frac{2}{m}\left(2k + nk\log_2\left(\frac{e(2m-1)}{n}\right) + \log_2\left(\frac{2}{\delta}\right)\right) \geq \frac{2}{m}\log_2\left(\frac{2\Pi_H(2m)}{\delta}\right),$$
and so
$$P^m(\{s \in Z^m : \text{there exists } f \in H,\ \mathrm{er}_s(f) = 0 \text{ and } \mathrm{er}_P(f) \geq \varepsilon_0\}) < 2\Pi_H(2m)\,2^{-\varepsilon_0 m/2} = \delta.$$
(Also, for $m \leq n$, the bound trivially holds.) The second part follows similarly on noting that, for the class of k-level threshold functions, if $m \geq n+k$, then
$$\varepsilon_0' = \frac{2}{m}\left((n+k-1)\log_2\left(\frac{2emk}{n+k-1}\right) + \log_2\left(\frac{2}{\delta}\right)\right) \geq \frac{2}{m}\log_2\left(\frac{2\Pi_H(2m)}{\delta}\right).$$
The bound is trivially true for $m < n+k$; and, for $m \geq n+k$, the condition $n+k \geq 3$ ensures that $m \geq 8/\varepsilon_0$, so that the bound of Blumer et al. (1989) applies. ■

The following variations of these results, in which k is not prescribed in advance, are perhaps more useful, since one does not necessarily know *a priori* how many terms a suitable threshold decision list will have.

Theorem 5 *With the notations as above, and for $n \geq 3$, the following holds with probability at least $1-\delta$:*

1. *If f is a threshold decision list, then*
$$\mathrm{er}_P(f) < \mathrm{er}_s(f) + \sqrt{\frac{8}{m}\left(2k\ln 2 + nk\ln\left(\frac{e(2m-1)}{n}\right) + \ln\left(\frac{14k^2}{\delta}\right)\right)},$$
where k is the number of terms of f.

2. *If f is a multilevel threshold function, then*

$$\mathrm{er}_P(f) < \mathrm{er}_s(f) + \sqrt{\frac{8}{m}\left((n+k-1)\ln\left(\frac{2emk}{n+k-1}\right)+\ln\left(\frac{14k^2}{\delta}\right)\right)},$$

where k is the number of levels (terms) of f.

3. *If f is a threshold decision list and $\mathrm{er}_s(f)=0$, then*

$$\mathrm{er}_P(f) < \frac{2}{m}\left(2k + nk\log_2\left(\frac{e(2m-1)}{n}\right)+\log_2\left(\frac{4k^2}{\delta}\right)\right)$$

where k is the number of terms of f;

4. *If f is a multilevel threshold function and $\mathrm{er}_s(f)=0$, then*

$$\mathrm{er}_P(f) < \frac{2}{m}\left((n+k-1)\log_2\left(\frac{2emk}{n+k-1}\right)+\ln\left(\frac{4k^2}{\delta}\right)\right),$$

where k is the number of terms of f.

Proof: We prove the last part, the other three being very similar. We use a well-known technique often found in discussions of 'structural risk minimisation' and model selection (see Vapnik, 1982; Shawe-Taylor et al., 1996; Anthony and Bartlett, 1999, for instance). From Theorem 4, for any $\delta \in (0,1)$ and any $k \in \mathbb{N}$, if $n+k \geq 3$, then the probability p_k that there is a k-level threshold function f such that $\mathrm{er}_s(f)=0$ and

$$\mathrm{er}_P(f) < \varepsilon'_k = \frac{2}{m}\left((n+k-1)\log_2\left(\frac{2emk}{n+k-1}\right)+\ln\left(\frac{2\pi^2k^2}{6\delta}\right)\right)$$

is less than $(\delta/k^2)(6/\pi^2)$. The fact that $n \geq 3$ ensures that $n+k \geq 3$. Hence the probability that, for some $k \in \mathbb{N}$, there is $f \in H$ with $\mathrm{er}_P(f) \geq \mathrm{er}_s(f)+\varepsilon_k$ is less than $\sum_{k=1}^{\infty} p_k < \delta(6/\pi^2)\sum_{k=1}^{\infty}(1/k^2) = \delta$. The result follows on noting that $2\pi^2/6 < 4$. ■

4. Margin-Based Error Bounds for Threshold Decision Lists

We now derive generalization error bounds dependent on the size of margins. The key qualitative difference between these bounds and those of Section 3 is that the margin-based bounds are dimension-independent, in that they do not depend on n.

4.1 Definition of Margin Error

Suppose that h is a threshold decision list, with k terms, and suppose that the tests in h are the threshold functions $t_1, t_2, \ldots, t_k$, and that t_i is represented by weight vector w_i and threshold θ_i. Assume also, without any loss of generality, that $\|w_i\| = 1$ for each i. We say that h classifies the labeled example (x,b) (correctly, and) with margin $\gamma > 0$ if $h(x) = b$ and, for all $1 \leq i \leq k$, $|\langle w_i, x\rangle - \theta_i| \geq \gamma$. In other words, h classifies x with margin γ if, overall, the classification of x given

by the threshold decision list h is correct and, additionally, x is distance at least γ from *all* of the k hyperplanes defining h.[1] Note that we do not simply stipulate that x is distance at least γ from the single hyperplane involved in the first test that x passes: rather, we require x to be distance at least γ from all of the hyperplanes. (In this sense, the classification given to x by h is not just correct, but 'definitely' correct.) Given a labeled sample $s = ((x_1,b_1),\ldots,(x_m,b_m))$, the error of h on s at margin γ, denoted $\mathrm{er}_s^\gamma(h)$, is the proportion of labeled examples in s that are *not* classified by h with margin γ. Thus, $\mathrm{er}_s^\gamma(h)$ is the fraction of the sample points that are either misclassified by h, or are classified correctly but are distance less than γ from one of the planes.

Following the analysis of perceptron decision trees in Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000), we may want to consider separate margin parameters $\gamma_1,\gamma_2,\ldots,\gamma_k$ for each of the k terms of the decision list. We have the following definition.

Definition 6 *Suppose $h = (t_1,c_1),\ldots,(t_k,c_k)$ is a threshold decision list, where t_i is represented by weight vector w_i and threshold θ_i, where $\|w_i\| = 1$. Given $\Gamma = (\gamma_1,\gamma_2,\ldots,\gamma_k)$, we say that h classifies the labeled example (x,b) (correctly and) with margin Γ if $h(x) = b$ and, for all $1 \le i \le k$, $|\langle w_i,x\rangle - \theta_i| \ge \gamma_i$. We define $\mathrm{er}_s^\Gamma(h)$ to be the proportion of labeled examples in the sample s that are not classified with margin Γ.*

4.2 Covering Numbers

A useful tool in the derivation of margin-based generalization error bounds is the *covering number* of a class of real functions. Suppose that $F : X \to \mathbb{R}$ is a set of real-valued functions with domain X, and that $x = (x_1,x_2,\ldots,x_m)$ is an unlabeled sample of m points of X. Then, for $\varepsilon > 0$, $C \subseteq F$ is an ε-*cover of F with respect to the d_∞^x-metric* if for all $f \in F$ there is $\hat{f} \in C$ such that $d_\infty^x(f,\hat{f}) < \varepsilon$, where, for $f,g \in F$,

$$d_\infty^x(f,g) = \max_{1\le i\le m} |f(x_i) - g(x_i)|.$$

(Coverings with respect to other metrics derived from x can also be defined, but this paper needs only the present definition.) The class F is said to be totally bounded if it has a finite ε-cover with respect to the d_∞^x metric, for all $\varepsilon > 0$ and all $x \in X^m$ (for all m). In this case, given $x \in X^m$, we define the d_∞^x-*covering number* $\mathcal{N}_\infty(F,\varepsilon,x)$ to be the minimum cardinality of an ε-cover of F with respect to the d_∞^x-metric. We then define the (uniform) d_∞-*covering numbers* $\mathcal{N}_\infty(F,\varepsilon,m)$ by

$$\mathcal{N}_\infty(F,\varepsilon,m) = \sup\{\mathcal{N}_\infty(F,\varepsilon,x) : x \in X^m\}.$$

Many bounds on covering numbers for specific classes have been obtained (see Anthony and Bartlett, 1999, for an overview), and general bounds on covering numbers in terms of a generalization of the VC-dimension, known as the *fat-shattering dimension*, have been given (Alon et al., 1997).

In this paper, we use a recent bound of Zhang (2002) for the d_∞-covering numbers of bounded linear mappings. For $R > 0$, let $B_R = \{x \in \mathbb{R}^n : \|x\| \le R\}$ be the closed ball in $\mathbb{R}^n$ of radius R, centred on the origin. For $w \in \mathbb{R}^n$, let $f_w : B_R \to \mathbb{R}$ be given by $f_w(x) = \langle w,x\rangle$, and let

$$L_R = \{f_w : w \in \mathbb{R}^n, \|w\| = 1\}.$$

1. The assumption that $\|w_i\| = 1$ ensures that the interpretation in terms of distance is valid; for, in this case the 'functional' and 'geometric' margins coincide. See the paper by Cristianini and Shawe-Taylor (2000).

Zhang (2002) has shown that

$$\log_2 \mathcal{N}_\infty(L_R, \varepsilon, m) \leq 36 \frac{R^2}{\varepsilon^2} \log_2 \left(2 \lceil 4R/\varepsilon + 2 \rceil m + 1\right). \tag{1}$$

One thing of note is that this bound is dimension-independent: it does not depend on n. This bound differs from previous bounds (Bartlett, 1998; Anthony and Bartlett, 1999; Shawe-Taylor et al., 1996) for the logarithm of the d_∞-covering numbers in that it involves a factor of order $\ln m$ rather than $(\ln m)^2$.[2]

4.3 Margin-based Bounds

Following a method used by Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000), together with the covering number bound of Zhang (2002), we can obtain the following two results. (In these results, it simplifies matters to assume that $R \geq 1$ and $\gamma_i \leq 1$, but it will be clear how to modify them otherwise.)

Theorem 7 *Suppose $R \geq 1$ and $Z = B_R \times \{0,1\}$, where $B_R = \{x \in \mathbb{R}^n : \|x\| \leq R\}$. Fix $k \in \mathbb{N}$ and let H be the set of all threshold decision lists with k terms, defined on domain B_R. Let $\gamma_1, \gamma_2, \ldots, \gamma_k \in (0,1]$ be given. Then, with probability at least $1-\delta$, the following holds for $s \in Z^m$: if $h \in H$ and $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)$, then*

$$\mathrm{er}_P(h) < \mathrm{er}_s^\Gamma(h) + \sqrt{\frac{8}{m}\left(576 R^2 D(\Gamma) \ln(8m) + \ln\left(\frac{2}{\delta}\right) + k\right)},$$

where $D(\Gamma) = \sum_{i=1}^k (1/\gamma_i^2)$ and where the margin error $\mathrm{er}_s^\Gamma(h)$ is as in Definition 6.

Proof: The proof extends a technique of Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000) (where the case of zero margin error was the focus), and is motivated by proofs of Anthony and Bartlett (1999, 2000), Bartlett (1998), Shawe-Taylor et al. (1996), which in turn are based on the original work of Vapnik and Chervonenkis (1971).

Given $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_n)$, it can fairly easily be shown that if

$$Q = \{s \in Z^m : \exists h \in H \text{ with } \mathrm{er}_P(h) \geq \mathrm{er}_s^\Gamma(h) + \varepsilon\}$$

and

$$T = \{(s, s') \in Z^m \times Z^m : \exists h \in H \text{ with } \mathrm{er}_{s'}(h) \geq \mathrm{er}_s^\Gamma(h) + \varepsilon/2\},$$

then for $m \geq 2/\varepsilon^2$, $P^m(Q) \leq 2P^{2m}(T)$. For, we have

$$\begin{aligned} P^{2m}(T) &\geq P^{2m}\left(\exists h \in H : \mathrm{er}_P(h) \geq \mathrm{er}_s^\Gamma(h) + \varepsilon \text{ and } \mathrm{er}_{s'}(h) \geq \mathrm{er}_P(h) - \varepsilon/2\right) \\ &= \int_Q P^m\left(\left\{s' : \exists h \in H, \mathrm{er}_P(h) \geq \mathrm{er}_s^\Gamma(h) + \varepsilon \text{ and } \mathrm{er}_{s'}(h) \geq \mathrm{er}_P(h) - \varepsilon/2\right\}\right) dP^m(s) \\ &\geq \frac{1}{2} P^m(Q), \end{aligned}$$

2. Previous approaches to bounding the d_∞-covering numbers first bounded the *fat-shattering dimension* and then used a result of Alon et al. (1997) that relates the covering numbers to the fat-shattering dimension. An additional $\ln m$ factor appears when this route is taken.

for $m \geq 2/\varepsilon^2$, where the final inequality follows from $P^m(\mathrm{er}_{s'}(h) \geq \mathrm{er}_P(h) - \varepsilon/2) \geq 1/2$, for any $h \in H$, true by Chebyshev's inequality.

Let G be the permutation group (the 'swapping group') on the set $\{1,2,\ldots,2m\}$ generated by the transpositions $(i, m+i)$ for $i = 1,2,\ldots,m$. Then G acts on Z^{2m} by permuting the coordinates: for $\sigma \in G$, $\sigma(z_1,z_2,\ldots,z_{2m}) = (z_{\sigma(1)},\ldots,z_{\sigma(m)})$. Now, by invariance of P^{2m} under the action of G, $P^{2m}(T) \leq \max\{\Pr(\sigma z \in T) : z \in Z^{2m}\}$, where Pr denotes the probability over uniform choice of σ from G. (See Vapnik and Chervonenkis, 1971, and Anthony and Bartlett, 1999, for instance.)

Given a threshold decision list on $B_R \subseteq \mathbb{R}^n$, each test is of the form $f_i \leftarrow [w_i, \theta_i]$; that is, the test is passed if and only if $\langle w_i, x\rangle \geq \theta_i$. An equivalent functionality is obtained by using inputs in B_R augmented by -1, and using *homogeneous* threshold functions of $n+1$ variables; that is, ones with zero threshold. So any threshold decision list of length k on B_R can be realized as one on $\mathbb{R}^{n+1}$, defined on the subset $B_R \times \{-1\}$, and with homogeneous threshold functions as its tests. Fix $z \in Z^{2m}$ and let $x = (x_1,x_2,\ldots,x_{2m}) \in X^{2m}$ be the corresponding vector of x_i, where $z_i = (x_i, b_i)$. For i between 1 and k, let C_i be a minimum-sized $\gamma_i/2$-cover of L with respect to the d_∞^x metric, where L is the set of linear functions $x \mapsto \langle w, x\rangle$ for $\|w\| = 1$, defined on the domain $D = \{(x,-1) : x \in \mathbb{R}^n, \|x\| \leq R\}$. Note that if $x \in \mathbb{R}^n$ satisfies $\|x\| \leq R$, then the corresponding $(x,-1)$ has length at most $\sqrt{R^2+1}$. So, by the covering number bound (1),

$$\log_2 |C_i| \leq \frac{144(R^2+1)}{\gamma_i^2} \log_2 \left(\left(\frac{32\sqrt{R^2+1}}{\gamma_i} + 14 \right) m \right) \leq \frac{288R^2}{\gamma_i^2} \log_2 \left(\frac{60Rm}{\gamma_i} \right). \tag{2}$$

Suppose that $h = (f_1,c_1),\ldots,(f_k,c_k)$ is a threshold decision list with k homogeneous threshold tests, defined on D. Denote the tests of the list by $f_1, f_2, \ldots, f_k$, where f_i corresponds to weight vector $w_i \in \mathbb{R}^{n+1}$. For each i, let $\hat{f}_i \in C_i$ satisfy $d_\infty^x(f_i, \hat{f}_i) < \gamma_i/2$, let $\hat{w}_i$ be the corresponding weight vector, and let $\hat{h}$ be the threshold decision list obtained from h by replacing each f_i by $\hat{f}_i$, leaving the c_i unchanged. The set $\hat{H}$ of all possible such $\hat{h}$ is of cardinality at most $2^k \prod_{i=1}^k |C_i|$ (where the 2^k factor corresponds to the choices of the values c_i). Suppose that $\sigma z = (s,s') \in T$ and that $\mathrm{er}_{s'}(h) \geq \mathrm{er}_s^\Gamma(h) + \varepsilon/2$. Let $\Gamma/2 = (\gamma_1/2,\ldots,\gamma_k/2)$. Then, because for all $1 \leq j \leq 2m$ and all $1 \leq i \leq k$, $|\langle w_i, x_j\rangle - \langle \hat{w}_i, x_j\rangle| < \gamma_i/2$, it can be seen that $\mathrm{er}_{s'}^{\Gamma/2}(\hat{h}) \geq \mathrm{er}_{s'}(h)$ and $\mathrm{er}_s^\Gamma(h) \geq \mathrm{er}_s^{\Gamma/2}(\hat{h})$. Explicitly (denoting any given x_i by x), $\mathrm{er}_{s'}^{\Gamma/2}(\hat{h}) \geq \mathrm{er}_{s'}(h)$ follows from the observation that if $\langle w_i, x\rangle < 0$, then $\langle \hat{w}_i, x\rangle < \gamma_i/2$ and if $\langle w_i, x\rangle > 0$, then $\langle \hat{w}_i, x\rangle > -\gamma_i/2$; and $\mathrm{er}_s^\Gamma(h) \geq \mathrm{er}_s^{\Gamma/2}(\hat{h})$ follows from the facts that if $\langle \hat{w}_i, x\rangle < \gamma_i/2$ then $\langle w_i, x\rangle < \gamma_i$, and if $\langle \hat{w}_i, x\rangle > -\gamma_i/2$ then $\langle w_i, x\rangle > -\gamma_i$. So, $\mathrm{er}_s^{\Gamma/2}(\hat{h}) \geq \mathrm{er}_s^{\Gamma/2}(\hat{h}) + \varepsilon/2$, and therefore, for any $z \in Z^{2m}$,

$$\Pr(\sigma z \in T) \leq \Pr\left(\sigma z \in \bigcup_{\hat{h} \in \hat{H}} S(\hat{h}) \right),$$

where $S(\hat{h}) = \{(s,s') \in Z^{2m} : \mathrm{er}_{s'}^{\Gamma/2}(\hat{h}) \geq \mathrm{er}_s^{\Gamma/2}(\hat{h}) + \varepsilon/2\}$. Fix $\hat{h} \in \hat{H}$ and let $v_i = 0$ if $\hat{h}$ classifies z_i with margin at least $\Gamma/2$, and 1 otherwise. Then

$$\Pr\left(\sigma z \in S(\hat{h})\right) = \Pr\left(\frac{1}{m} \sum_{i=1}^m (v_{m+i} - v_i) \geq \varepsilon/2 \right) = \Pr\left(\frac{1}{m} \sum_{i=1}^m \varepsilon_i |v_i - v_{m+i}| \geq \varepsilon/2 \right),$$

where the ε_i are independent (Rademacher) $\{-1,1\}$ random variables, each taking value 1 with probability $1/2$, and where the last probability is over the joint distribution of the ε_i. Hoeffding's

inequality bounds this probability by $\exp(-\varepsilon^2 m/8)$. (See Anthony and Bartlett, 1999, for instance, for details.)

We therefore have

$$\Pr\left(\sigma z \in \bigcup_{\hat{h}\in\hat{H}} T(\hat{h})\right) \leq \sum_{\hat{h}\in\hat{H}} \Pr\left(\sigma z \in T(\hat{h})\right) \leq |\hat{H}| \exp(-\varepsilon^2 m/8),$$

which gives

$$P^m(Q) \leq 2P^{2m}(T) \leq 2\,2^k \prod_{i=1}^{k} |C_i| \exp(-\varepsilon^2 m/8).$$

Using the bound (2), we see that, provided

$$\varepsilon \geq \varepsilon_0 = \sqrt{\frac{8}{m}\left(\sum_{i=1}^{k} \frac{288R^2}{\gamma_i^2} \ln\left(\frac{60Rm}{\gamma_i}\right) + \ln\left(\frac{2}{\delta}\right) + k\right)},$$

(in which case we certainly also have $m \geq 2/\varepsilon^2$) then the probability of Q is at most δ. So, with probability at most $1-\delta$, $\mathrm{er}_P(h) < \mathrm{er}_s^\Gamma(h) + \varepsilon_0$ for all $h \in H$. If, for each i, $m \geq R^2/\gamma_i^2$, then $\ln(60Rm/\gamma_i) \leq 2\ln(8m)$ and so, with probability at least $1-\delta$, for all $h \in H$,

$$\mathrm{er}_P(h) < \mathrm{er}_s^\Gamma(h) + \sqrt{\frac{8}{m}\left(\sum_{i=1}^{k} \frac{576R^2}{\gamma_i^2} \ln(8m) + \ln\left(\frac{2}{\delta}\right) + k\right)}. \tag{3}$$

If, however, for some i, $m < R^2/\gamma_i^2$, then the bound (3) is trivially true (since the term under the square root is greater than 1). The result follows. ■

A tighter bound can be given when the margin error is zero, as follows. (The bound involves $1/m$ rather than $1/\sqrt{m}$.)

Theorem 8 *Suppose $R \geq 1$ and $Z = B_R \times \{0,1\}$, where $B_R = \{x \in \mathbb{R}^n : \|x\| \leq R\}$. Fix $k \in \mathbb{N}$ and let H be the set of all threshold decision lists with k terms, defined on domain B_R. Let $\gamma_1, \gamma_2, \ldots, \gamma_k \in (0,1]$ be given. Then, with probability at least $1-\delta$, the following holds for $s \in Z^m$: if h is any threshold decision list with k terms, and h classifies s with margin $\Gamma = (\gamma_1, \ldots, \gamma_k)$, then*

$$\mathrm{er}_P(h) < \frac{2}{m}\left(576R^2 D(\Gamma)\log_2(8m) + \log_2\left(\frac{2}{\delta}\right) + k\right)$$

where $D(\Gamma) = \sum_{i=1}^{k}(1/\gamma_i^2)$.

Proof: This proof is similar to that of Theorem 7. It uses, first, the fact[3] that if

$$Q = \{s \in Z^m : \exists h \in H \text{ with } \mathrm{er}_s^\Gamma(h) = 0, \mathrm{er}_P(h) \geq \varepsilon\}$$

and

$$T = \{(s,s') \in Z^m \times Z^m : \exists h \in H \text{ with } \mathrm{er}_s^\Gamma(h) = 0, \mathrm{er}_{s'}(h) \geq \varepsilon/2\},$$

3. For similar results, see Vapnik and Chervonenkis (1971); Blumer et al. (1989); and Anthony and Bartlett (1999).

then, for $m \geq 8/\varepsilon$, $P^m(Q) \leq 2P^{2m}(T)$. This is so, because

$$\begin{aligned} P^{2m}(T) &\geq P^{2m}\left(\exists h \in H : \mathrm{er}_s^{\Gamma}(h) = 0,\ \mathrm{er}_P(h) \geq \varepsilon \text{ and } \mathrm{er}_{s'}(h) \geq \varepsilon/2\right) \\ &= \int_Q P^m\left(\left\{s' : \exists h \in H,\ \mathrm{er}_s^{\Gamma}(h) = 0,\ \mathrm{er}_P(h) \geq \varepsilon \text{ and } \mathrm{er}_{s'}(h) \geq \varepsilon/2\right\}\right) dP^m(s) \\ &\geq \frac{1}{2}P^m(Q), \end{aligned}$$

for $m \geq 8/\varepsilon$. The final inequality follows from the fact that if $\mathrm{er}_P(h) = 0$, then for $m \geq 8/\varepsilon$, $P^m(\mathrm{er}_{s'}(h) \geq \varepsilon/2) \geq 1/2$, for any $h \in H$, something that follows for $m \geq 8/\varepsilon$ by Chebyshev's inequality or a Chernoff bound (Anthony and Biggs, 1992, for instance). As before, $P^{2m}(T) \leq \max_{z \in Z^{2m}} \Pr(\sigma z \in T)$, where Pr denotes the probability over uniform choice of σ from the 'swapping group' G. A very similar argument to that given in the proof of Theorem 3 establishes that for any $z \in Z^{2m}$,

$$\Pr(\sigma z \in T) \leq \Pr\left(\sigma z \in \bigcup_{\hat{h} \in \hat{H}} S(\hat{h})\right),$$

where $S(\hat{h}) = \{(s,s') \in Z^{2m} : \mathrm{er}_s^{\Gamma/2}(h) = 0, \mathrm{er}_{s'}^{\Gamma/2}(\hat{h}) \geq \varepsilon/2\}$. Now, suppose $S(\hat{h}) \neq \emptyset$, so that for some $\tau \in G$, $\tau z = (s,s') \in S(\hat{h})$, meaning that $\mathrm{er}_s^{\Gamma/2}(\hat{h}) = 0$ and $\mathrm{er}_{s'}^{\Gamma/2}(\hat{h}) \geq \varepsilon/2$. Then, by symmetry, $\Pr\left(\sigma z \in S(\hat{h})\right) = \Pr\left(\sigma(\tau z) \in S(\hat{h})\right)$. Suppose that $\mathrm{er}_{s'}^{\Gamma/2}(\hat{h}) = r/m$, where $r \geq \varepsilon m/2$ is the number of x_i in s' not classified with margin $\Gamma/2$ by $\hat{h}$. Then those permutations σ such that $\sigma(\tau z) \in S(\hat{h})$ are precisely those that 'swap' elements other than these r, and there are $2^{m-r} \leq 2^{m-\varepsilon m/2}$ such σ. It follows that, for each fixed $\hat{h} \in \hat{H}$,

$$\Pr\left(\sigma z \in S(\hat{h})\right) \leq \frac{2^{m(1-\varepsilon/2)}}{|G|} = 2^{-\varepsilon m/2}.$$

The proof then proceeds as does the proof of Theorem 7, using the bound (2). ■

4.4 Uniform Margin-based Bounds

One difficulty with Theorems 7 and 8 is that the number, k, of terms, and the margins γ_i are specified *a priori*. A more useful generalization error bound would enable us to choose, tune, or observe these parameters after learning. We now derive such a result. The approach we take to obtaining a 'uniform' result of this type differs from that taken by Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000), and gives a slightly better bound.

We first need a generalization of a result from (Bartlett, 1998), where the following is shown. Suppose $\mathbb{P}$ is any probability measure and that $\{E(\alpha_1,\alpha_2,\delta) : 0 < \alpha_1,\alpha_2,\delta \leq 1\}$ is a set of events such that:

- for all α, $\mathbb{P}(E(\alpha,\alpha,\delta)) \leq \delta$,
- if $0 < \alpha_1 \leq \alpha \leq \alpha_2 < 1$ and $0 < \delta_1 \leq \delta \leq 1$, then $E(\alpha_1,\alpha_2,\delta_1) \subseteq E(\alpha,\alpha,\delta)$.

Then $\mathbb{P}\left(\bigcup_{\alpha \in (0,1]} E(\alpha/2,\alpha,\delta\alpha/2)\right) \leq \delta$ for $0 < \delta < 1$.

We modify and extend this result as follows.

Theorem 9 *Suppose $\mathbb{P}$ is any probability measure, $k \in \mathbb{N}$, and that*

$$\{E(\Gamma_1, \Gamma_2, \delta) : \Gamma_1, \Gamma_2 \in (0,1]^k, \delta \leq 1\}$$

is a set of events such that:

(a) for all $\Gamma \in (0,1]^k$, $\mathbb{P}(E(\Gamma, \Gamma, \delta)) \leq \delta$,

(b) $\Gamma_1 \leq \Gamma \leq \Gamma_2$ (component-wise) and $0 < \delta_1 \leq \delta \leq 1$ imply $E(\Gamma_1, \Gamma_2, \delta_1) \subseteq E(\Gamma, \Gamma, \delta)$.

Then

$$\mathbb{P}\left(\bigcup_{\Gamma \in (0,1]^k} E\left((1/2)\Gamma, \Gamma, \delta c(\Gamma)\right) \right) \leq \delta$$

for $0 < \delta < 1$, where

$$c(\Gamma) = \left\{ \prod_{i=1}^{k} \log_2 \left(\frac{4}{\gamma_i} \right) \right\}^{-2}.$$

Proof: Denoting by $\mathbf{u} = (1, 1, \ldots, 1)$ the all-1 vector of length k, we have

$$\begin{aligned}
& \mathbb{P}\left(\bigcup_{\Gamma \in (0,1]^k} E\left((1/2)\Gamma, \Gamma, \delta c(\Gamma)\right) \right) \\
\leq\ & \mathbb{P}\left(\bigcup_{i_1, \ldots, i_k = 0}^{\infty} \left\{ E\left((1/2)\Gamma, \Gamma, \delta c(\Gamma)\right) : \text{for } j = 1, \ldots, k, \gamma_j \in \left(\left(\frac{1}{2}\right)^{i_j+1}, \left(\frac{1}{2}\right)^{i_j} \right] \right\} \right) \\
\leq\ & \mathbb{P}\left(\bigcup_{i_1, \ldots, i_k = 0}^{\infty} E\left(\left(\frac{1}{2}\right)^{i_1+1} \mathbf{u}, \left(\frac{1}{2}\right)^{i_1+1} \mathbf{u}, \delta \prod_{j=1}^{k} \frac{1}{i_j+1} \frac{1}{i_j+2} \right) \right).
\end{aligned}$$

Here, we have used property (b) of the events $E(\Gamma_1, \Gamma_2, \delta)$, together with the following two observations: if $\gamma_j \in ((1/2)^{i_j+1}, (1/2)^{i_j}]$, then $(1/2)\Gamma \leq (1/2)^{i_j+1}\mathbf{u}$ and $\Gamma \geq (1/2)^{i_j+1}\mathbf{u}$; and $\gamma_i \in ((1/2)^{i_j+1}, (1/2)^{i_j}]$ implies

$$\left(\log_2 \left(\frac{4}{\gamma_j} \right) \right)^2 \geq (i_j+2)^2 \geq (i_j+1)(i_j+2),$$

so that

$$c(\Gamma) \leq \prod_{j=1}^{k} \frac{1}{(i_j+1)(i_j+2)}.$$

Now, by property (a),

$$\begin{aligned}
& \mathbb{P}\left(\bigcup_{i_1,\ldots,i_k=0}^{\infty} E\left(\left(\frac{1}{2}\right)^{i_1+1}\mathbf{u},\left(\frac{1}{2}\right)^{i_1+1}\mathbf{u},\delta\prod_{j=1}^{k}\frac{1}{i_j+1}\frac{1}{i_j+2}\right)\right)\\
\leq & \sum_{i_1,i_2,\ldots,i_k=0}^{\infty}\mathbb{P}\left(E\left(\left(\frac{1}{2}\right)^{i_1+1}\mathbf{u},\left(\frac{1}{2}\right)^{i_1+1}\mathbf{u},\delta\prod_{j=1}^{k}\frac{1}{i_j+1}\frac{1}{i_j+2}\right)\right)\\
\leq & \sum_{i_1,i_2,\ldots,i_k=0}^{\infty}\delta\prod_{j=1}^{k}\left(\frac{1}{(i_j+1)(i_j+2)}\right)\\
= & \delta\prod_{j=1}^{k}\sum_{i_j=0}^{\infty}\left(\frac{1}{(i_j+1)(i_j+2)}\right)\\
= & \delta\prod_{j=1}^{k}\sum_{i_j=0}^{\infty}\left(\frac{1}{i_j+1}-\frac{1}{i_j+2}\right)\\
= & \delta\prod_{j=1}^{k}1=\delta.
\end{aligned}$$

■

We can now obtain the following 'uniform' result.

Theorem 10 *Suppose $R\geq 1$ and $Z=B_R\times\{0,1\}$, where $B_R=\{x\in\mathbb{R}^n:\|x\|\leq R\}$. Let H be the set of all threshold decision lists (with any number of terms) defined on domain B_R. With probability at least $1-\delta$, the following statements hold for $s\in Z^m$:*

1. for all $k\in\mathbb{N}$ *and* for all $\gamma_1,\gamma_2,\ldots,\gamma_k\in(0,1]$, *if $h\in H$ has k terms, and* $\Gamma=(\gamma_1,\gamma_2,\ldots,\gamma_k)$, *then*

$$\mathrm{er}_P(h)<\mathrm{er}_s^{\Gamma}(h)+\sqrt{\frac{8}{m}\left(2304R^2D(\Gamma)\ln(8m)+\ln\left(\frac{2}{\delta}\right)+2k+2\sum_{i=1}^{k}\ln\left(\log_2\left(\frac{4}{\gamma_i}\right)\right)\right)},$$

   where $D(\Gamma)=\sum_{i=1}^{k}(1/\gamma_i^2)$.

2. for all $k\in\mathbb{N}$, *and* for all $\gamma_1,\gamma_2,\ldots,\gamma_k\in(0,1]$, *if $h\in H$ has k terms, and h classifies s with margin $\Gamma=(\gamma_1,\gamma_2,\ldots,\gamma_k)$, then*

$$\mathrm{er}_P(h)<\frac{2}{m}\left(2304R^2D(\Gamma)\log_2(8m)+\log_2\left(\frac{2}{\delta}\right)+2k+2\sum_{i=1}^{k}\ln\left(\log_2\left(\frac{4}{\gamma_i}\right)\right)\right),$$

   where $D(\Gamma)=\sum_{i=1}^{k}(1/\gamma_i^2)$.

Proof: Fix $k\in\mathbb{N}$. If $\Gamma_1=(\gamma_1^{(1)},\ldots,\gamma_k^{(1)})$ and $\Gamma_2=(\gamma_1^{(2)},\ldots,\gamma_k^{(2)})$, let $E(\Gamma_1,\Gamma_2,\delta)$ be the event that there exists a threshold decision list h with k terms such that

$$\mathrm{er}_P(h)\geq\mathrm{er}_s^{\Gamma_2}(h)+\sqrt{\frac{8}{m}\left(576R^2D(\Gamma_1)\ln(8m)+\ln\left(\frac{2}{\delta}\right)+k\right)},$$

where $D(\Gamma_1) = \sum_{i=1}^{k} (1/\gamma_i^{(1)})^2$. Then, by Theorem 7, $P^m(E(\Gamma,\Gamma,\delta)) \leq \delta$, and it is easily seen that $\Gamma_1 \leq \Gamma \leq \Gamma_2$ and $0 < \delta_1 \leq \delta \leq 1$ imply $E(\Gamma_1,\Gamma_2,\delta_1) \subseteq E(\Gamma,\Gamma,\delta)$. It follows that

$$P^m\left(\bigcup_{\Gamma\in(0,1]^k} E\left((1/2)\Gamma,\Gamma,\delta c(\Gamma)\right)\right) \leq \delta,$$

where

$$c(\Gamma) = \left\{\prod_{i=1}^{k} \log_2\left(\frac{4}{\gamma_i}\right)\right\}^{-2}.$$

So, with probability at least $1-\delta$, *for all* $\gamma_1,\gamma_2,\ldots,\gamma_k \in (0,1]$, if h is any threshold decision list with k terms, and $\Gamma = (\gamma_1,\gamma_2,\ldots,\gamma_k)$, then

$$\mathrm{er}_P(h) < \mathrm{er}_s^{\Gamma}(h) + \sqrt{\frac{8}{m}\left(2304R^2D(\Gamma)\ln(8m) + \ln\left(\frac{2}{\delta}\right) + k + \ln\left(\frac{1}{c(\Gamma)}\right)\right)},$$

where $D(\Gamma) = \sum_{i=1}^{k}(1/\gamma_i^2)$. This holds for any *fixed* k. Replacing δ by $\delta/2^k$, we see that, with probability at least $1-\delta/2^k$, for any h with k terms and any Γ,

$$\mathrm{er}_P(h) < \mathrm{er}_s^{\Gamma}(h) + \sqrt{\frac{8}{m}\left(2304R^2D(\Gamma)\ln(8m) + \ln\left(\frac{2\,2^k}{\delta}\right) + k + \ln\left(\frac{1}{c(\Gamma)}\right)\right)},$$

and so, with probability at least $1-\sum_{k=1}^{\infty}(\delta/2^k) = 1-\delta$, for all k, for all h of length k, and for all Γ,

$$\mathrm{er}_P(h) < \mathrm{er}_s^{\Gamma}(h) + \sqrt{\frac{8}{m}\left(2304R^2D(\Gamma)\ln(8m) + \ln\left(\frac{2}{\delta}\right) + 2k + 2\sum_{i=1}^{k}\ln\left(\log_2\left(\frac{4}{\gamma_i}\right)\right)\right)}.$$

(Note that we could have replaced δ by $\delta\alpha_k$ where (α_k) is any sequence such that $\sum_{i=1}^{\infty}\alpha_k = 1$.) The second part of the theorem is proved similarly, using Theorem 8. ■

4.5 Comparison with Related Results

Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000) proved a margin-based generalization result for the more general class of perceptron decision trees, in the case where there is zero Γ-margin error on the sample. The special case of their result that applies to threshold decision lists gives a bound (with probability at least $1-\delta$) of the form

$$\mathrm{er}_P(h) < O\left(\frac{1}{m}\left(D(\Gamma)(\ln m)^2 + k\ln m + \ln\left(\frac{1}{\delta}\right)\right)\right). \tag{4}$$

(The O-notation indicates that constants have been suppressed.)

By comparison, the bound given in Theorem 10 is of order

$$\mathrm{er}_P(h) < O\left(\frac{1}{m}\left(D(\Gamma)\ln m + k + \sum_{i=1}^{k}\ln\ln\left(\frac{1}{\gamma_i}\right) + \ln\left(\frac{1}{\delta}\right)\right)\right). \tag{5}$$

The first term of bound (5) is a $\ln m$ factor better than the corresponding term of (4). That this is so is because we have used Zhang's covering number bound, (1), rather than bounding the covering number by using results on fat-shattering dimension, coupled with the bound ofAlon et al. (1997). Additionally, since all these probability bounds are trivial (greater than 1) unless $m > (R/\gamma_i)^2$ for all i, the remaining terms of the bound (5) are of order no more than $O(k + \ln\ln m)$ rather than the $O(k \ln m)$ of (4), and they are potentially much smaller. This improvement results from the use of Theorem 9. Theorem 10 is therefore an improvement over the results implied by Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000).

4.6 Bounds for Perceptron Decision Trees

Although the focus of this paper is threshold decision lists, we now show how the analysis here can be used to improve and extend results on perceptron decision trees given by Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000). Recall that these are decision trees in which the decision nodes compute threshold functions. The definition of margin error $\mathrm{er}^{\Gamma}(h)$ for a perceptron decision tree classifier h is defined in a straightforward way by extending Definition 6. Suppose the threshold functions computed at the decision nodes are $t_1, \ldots, t_k$, where k is the number of decision nodes, and suppose that t_i is represented by weight vector w_i and threshold θ_i, where $\|w_i\| = 1$. Given $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)$, we say that the tree h classifies the labeled example (x, b) with margin Γ if $h(x) = b$ and, for all $1 \leq i \leq k$, $|\langle w_i, x\rangle - \theta_i| \geq \gamma_i$. Then, for a labeled sample s, $\mathrm{er}_s^{\Gamma}(h)$ is the proportion of labeled examples in s that are not classified with margin Γ.

Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000) obtain a generalization error bound for the special case in which the margin error is zero. The following theorem improves that bound and provides a bound applicable in the case of non-zero margin error. (We have stated only a 'uniform' result; that is, one in which the margin parameters and tree size are not fixed *a priori*. However, embedded in the proof are the corresponding non-uniform results.) The proof is a modification of the proofs of the theorems for threshold decision lists, in which we make use of the fact that the number of binary trees with k vertices—and hence the number of decision tree skeletons with k decision nodes (as noted in Quinlan and Rivest, 1989; Bennett et al., 2000)—is given by the Catalan number $N_k = \frac{1}{k+1}\binom{2k}{k}$.

Theorem 11 *For $k \in \mathbb{N}$, let $N_k = \dfrac{1}{k+1}\binom{2k}{k}$. Suppose $R \geq 1$ and $Z = B_R \times \{0,1\}$, where $B_R = \{x \in \mathbb{R}^n : \|x\| \leq R\}$. Let H be the set of all perceptron decision trees (of any size and structure) defined on domain B_R. With probability at least $1-\delta$, the following statements hold for $s \in Z^m$:*

1. for all $k \in \mathbb{N}$ *and* for all $\gamma_1, \gamma_2, \ldots, \gamma_k \in (0,1]$, *if $h \in H$ has k decision nodes, and $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)$, then*

$$\mathrm{er}_P(h) < \mathrm{er}_s^{\Gamma}(h) + \sqrt{\frac{8}{m}\left(2304R^2D(\Gamma)\ln(8m) + \ln\left(\frac{2^{k+2}N_k}{\delta}\right) + k + 2\sum_{i=1}^{k}\ln\left(\log_2\left(\frac{4}{\gamma_i}\right)\right)\right)},$$

where $D(\Gamma) = \sum_{i=1}^{k}(1/\gamma_i^2)$.

2. for all $k \in \mathbb{N}$, *and* for all $\gamma_1, \gamma_2, \ldots, \gamma_k \in (0,1]$, *if* $h \in H$ *has* k *decision nodes, and* h *classifies* s *with margin* $\Gamma = (\gamma_1, \gamma_2, \ldots, \gamma_k)$, *then*

$$\mathrm{er}_P(h) < \frac{2}{m}\left(2304R^2D(\Gamma)\log_2(8m) + \log_2\left(\frac{2^{k+2}N_k}{\delta}\right) + k + 2\sum_{i=1}^{k}\ln\left(\log_2\left(\frac{4}{\gamma_i}\right)\right)\right),$$

where $D(\Gamma) = \sum_{i=1}^{k}(1/\gamma_i^2)$.

Proof: The proof is similar to that of Theorems 7, 8 and 10, so we will omit some of the detail. As in the proof of Theorem 7, for any $k \in \mathbb{N}$, for H the class of perceptron decision trees (or, rather, the functions represented by such trees) with k decision nodes, for any Γ, if

$$Q = \{s \in Z^m : \exists h \in H \text{ with } \mathrm{er}_P(h) \geq \mathrm{er}_s^{\Gamma}(h) + \varepsilon\}$$

and

$$T = \{(s,s') \in Z^m \times Z^m : \exists h \in H \text{ with } \mathrm{er}_{s'}(h) \geq \mathrm{er}_s^{\Gamma}(h) + \varepsilon/2\},$$

then for $m \geq 2/\varepsilon^2$, $P^m(Q) \leq 2P^{2m}(T)$. With G the swapping permutation group, we have, as before, $P^{2m}(T) \leq \max\{\Pr(\sigma z \in T) : z \in Z^{2m}\}$, where Pr denotes the probability over uniform choice of σ from G. Given a perceptron decision tree on B_R, we may (as discussed in the proof of Theorem 7) realize the tree as one defined on $D = \{(x,-1) : x \in \mathbb{R}^n, \|x\| \leq R\}$, in which the decision nodes compute homogeneous threshold functions. Fixing $z \in Z^{2m}$, and arguing as in the proof of Theorem 7, for i between 1 and k, let C_i be a minimal-cardinality $\gamma_i/2$-cover of L with respect to the d_∞^x metric, where L is the set of linear functions $x \mapsto \langle w, x\rangle$ for $\|w\| = 1$, defined on D. Then $|C_i|$ is bounded as in (2). Suppose that, in a given perceptron decision tree h, and at a given decision node, the test is given by the threshold function f_i, represented by weight vector w_i and let $\hat{w}_i$ be an element of the cover C_i which is distance less than $\gamma_i/2$ from w_i. Then a very similar analysis to that in Theorem 7 establishes that if $\hat{h}$ is the tree obtained by replacing each f_i by $\hat{f}_i$, we have

$$\Pr(\sigma z \in T) \leq \Pr\left(\sigma z \in \bigcup_{\hat{h} \in \hat{H}} S(\hat{h})\right),$$

where $S(\hat{h}) = \{(s,s') \in Z^{2m} : \mathrm{er}_{s'}^{\Gamma/2}(\hat{h}) \geq \mathrm{er}_s^{\Gamma/2}(\hat{h}) + \varepsilon/2\}$. Now, the set $\hat{H}$ of all such $\hat{h}$ will have cardinality bounded as follows:

$$|\hat{H}| \leq 2^{k+1}N_k\prod_{i=1}^{k}|C_i| \leq 2^{k+1}N_k\prod_{i=1}^{k}2^{(288R^2/\gamma_i^2)\log_2(60Rm/\gamma_i)},$$

where the factor of 2^{k+1} accounts for the possible binary values at the $k+1$ leaves of the tree, and N_k accounts for the number of skeletons of trees with k decision nodes. By arguing precisely as in Theorem 7, we can then establish that for $\delta \in (0,1)$, for fixed k and fixed Γ, with probability at least $1-\delta$, for all perceptron decision trees with k decision nodes, $\mathrm{er}_P(h) < \mathrm{er}_s^{\Gamma}(h) + \varepsilon(\Gamma,\delta,k,m)$, where

$$\varepsilon(\Gamma,\delta,k,m) = \sqrt{\frac{8}{m}\left(576R^2D(\Gamma)\ln(8m) + \ln\left(\frac{2^{k+2}N_k}{\delta}\right)\right)},$$

Next, we apply Theorem 9. Fixing k and taking $E(\Gamma_1,\Gamma_2,\delta)$ to be the event that there exists a perceptron decision tree h with k decision nodes such that $\mathrm{er}_P(h) \geq \mathrm{er}_s^{\Gamma_2}(h) + \varepsilon(\Gamma_1,\delta,k,m)$, we establish that with probability at least $1-\delta$, for *all* Γ,

$$\mathrm{er}_P(h) < \mathrm{er}_s^{\Gamma}(h) + \sqrt{\frac{8}{m}\left(2304R^2D(\Gamma)\ln(8m) + \ln\left(\frac{2^{k+2}N_k}{\delta}\right) + 2\sum_{i=1}^{k}\ln\left(\log_2\left(\frac{4}{\gamma_i}\right)\right)\right)}.$$

Finally, replacing δ by $\delta/2^k$ and proceeding as in the final part of the proof of Theorem 9, we obtain the desired result. The proof of the second part of the theorem is similar. ■

The result given in Shawe-Taylor and Cristianini (1998) and Bennett et al. (2000) corresponds to the second case given in Theorem 11 and takes the form: with probability at least $1-\delta$, for all Γ and for any perceptron decision tree such that $\mathrm{er}^{\Gamma}(h) = 0$,

$$\mathrm{er}_P(h) < O\left(\frac{1}{m}\left(D(\Gamma)(\ln m)^2 + k\ln m + \ln N_k + \ln\left(\frac{1}{\delta}\right)\right)\right),$$

where we have suppressed the constants. Theorem 11 improves upon this, as can be seen by similar considerations to those made in comparing bounds (4) and (5) above. In particular, an expression of order $D(\Gamma)(\ln m)^2 + k\ln m$ is replaced by one of order $D(\Gamma)\ln m + k + \ln\ln m$.

5. Margin-Based Error Bounds for Multilevel Threshold Functions

Suppose that h is a k-level threshold function, represented by weight vector w with $\|w\| = 1$ and threshold vector $\theta = (\theta_1,\theta_2,\ldots,\theta_k)$ (where $\theta_1 \leq \theta_2 \cdots \leq \theta_k$). Regarded as a threshold decision list, the tests are the threshold functions t_i, where $t_i(y) = \mathrm{sgn}(\langle w,x\rangle - \theta_i)$. Recall that we say h classifies the labeled example (x,b) with margin $\gamma > 0$ if $h(x) = b$ and, for all $1 \leq i \leq k$, $|\langle w,x\rangle - \theta_i| \geq \gamma$. (In other words, h classifies x correctly, and x is distance at least γ from any of the hyperplanes defining the multilevel threshold function h.) As above, for a labeled sample s, $\mathrm{er}_s^{\gamma}(h)$, the sample error at margin γ, is the proportion of labeled examples in s that are *not* correctly classified with margin γ.

To bound generalization error in this special case, we take a slightly different approach to the one used above for general threshold decision lists. Rather than take a cover for each term of the decision list, a more 'global' approach can be taken, exploiting the fact that the planes are parallel. In taking this approach, however, the analysis considers only one margin parameter, γ, rather than k possibly different margin parameters, one for each plane. (As before, for the sake of simplicity, we assume that $R \geq 1$ and $\gamma \leq 1$.)

5.1 Generalization Error Bounds for k-level Threshold Functions

We have the following result.

Theorem 12 *Suppose $R \geq 1$ and $Z = B_R \times \{0,1\}$, where $B_R = \{x \in \mathbb{R}^n : \|x\| \leq R\}$. Fix $k \in \mathbb{N}$ and let H be the set of all k-level threshold functions defined on domain B_R. Let P be any probability distribution on Z, and suppose $\gamma \in (0,1]$ and $\delta \in (0,1)$. Then, with P^m-probability at least $1-\delta$, a*

sample s is such that if $h \in H$, then

$$\mathrm{er}_P(h) < \mathrm{er}_s^\gamma(h) + \sqrt{\frac{8}{m}\left(\frac{1152R^2}{\gamma^2}\ln(9m) + k\ln\left(\frac{10R}{\gamma}\right) + \ln\left(\frac{4}{\delta}\right)\right)}.$$

Proof: Fix $\gamma \in (0,1]$. As earlier, with H the set of k-level threshold functions on B_R, if

$$Q = \{s \in Z^m : \exists h \in H \text{ with } \mathrm{er}_P(h) \geq \mathrm{er}_s^\gamma(h) + \varepsilon\}$$

and

$$T = \{(s,s') \in Z^m \times Z^m : \exists h \in H \text{ with } \mathrm{er}_{s'}(h) \geq \mathrm{er}_s^\gamma(h) + \varepsilon/2\},$$

then $P^m(Q) \leq 2P^{2m}(T)$. Also as before, $P^{2m}(R) \leq \max\{\Pr(\sigma z \in R) : z \in Z^{2m}\}$, where Pr denotes the probability over uniform choice of σ from the 'swapping group' G. Let L_R be the set of all functions of the form $x \mapsto \langle w,x\rangle$, where $w \in \mathbb{R}^n$ satisfies $\|w\| = 1$, and where the domains of the functions are B_R. Now fix $z \in Z^{2m}$, let $x \in X^{2m}$ be the corresponding x_i-vector, and let C be a $\gamma/4$-cover of minimum size of L with respect to the d_∞^x metric. By (1),

$$\begin{aligned}
\log_2|C| &\leq \log_2 \mathcal{N}_\infty(L_R, \gamma/4, 2m) \\
&\leq \frac{576R^2}{\gamma^2}\log_2\left(2\lceil 16R/\gamma + 2\rceil 2m + 1\right) \\
&\leq \frac{576R^2}{\gamma^2}\log_2\left(\frac{80Rm}{\gamma}\right).
\end{aligned}$$

Each function in C is represented by a weight vector, and we shall denote the set of these weight vectors by $\hat{W}$. For each $w \in \mathbb{R}^n$, denote by $\hat{w}$ a member of $\hat{W}$ such that for $i = 1,2,\ldots,2m$, $|\langle w,x_i\rangle - \langle \hat{w},x_i\rangle| < \gamma/4$. Let

$$D = \{\theta \in \mathbb{R} : \exists n \in \mathbb{Z} \cap [-(4R/\gamma) - 1, (4R/\gamma) + 1] \text{ such that } \theta = n(\gamma/4)\},$$

and let $\hat{\Theta} = D^k$. Then

$$|\hat{\Theta}| \leq \left(\frac{8R}{\gamma} + 2\right)^k \leq \left(\frac{10R}{\gamma}\right)^k.$$

Now, suppose h is a k-level threshold function defined on B_R. Then, of course, h is represented by a weight vector $w \in \mathbb{R}^n$ with $\|w\| = 1$ and a threshold vector $\theta \in \mathbb{R}^k$. Since, for all $x \in B_R$, $|\langle w,x\rangle| \leq \|w\|\|x\| = \|x\| \leq R$, we can assume that each θ_i satisfies $|\theta_i| \leq R$. Then, denote by $\hat{\theta}$ a member of $\hat{\Theta}$ such that for $i = 1,2,\ldots,k$, $|\theta_i - \hat{\theta}_i| \leq \gamma/4$. (Such a $\hat{\theta}$ exists by the way in which $\hat{\Theta}$ is defined.) Let $\hat{H}$ be the set of all k-level threshold functions representable by weight vectors $\hat{w} \in \hat{W}$ and threshold vectors $\hat{\theta} = (\theta_1,\ldots,\theta_k) \in \hat{\Theta}$. Then

$$|\hat{H}| \leq 2\,2^{(576R^2/\gamma^2)\log_2(80Rm/\gamma)}\left(\frac{10R}{\gamma}\right)^k.$$

(Here, the first factor of 2 accounts for the two different ways in which the classifications can alternate as we traverse the planes is a normal direction.) For each $h \in H$, let $\hat{h}$ be the k-level threshold vector with weight vector $\hat{w} \in \hat{W}$ and threshold vector $\hat{\theta} \in \theta$, where $\hat{w}$ and $\hat{\theta}$ satisfy the properties indicated above. For each $i = 1,2,\ldots,2m$, for each $j = 1,2,\ldots,k$,

$$|(\langle w,x_i\rangle - \theta_j) - (\langle \hat{w},x_i\rangle - \hat{\theta}_j)| \leq |\langle w,x_i\rangle - \langle \hat{w},x_i\rangle| + |\theta_i - \hat{\theta}_i| \leq \gamma/4 + \gamma/4 = \gamma/2.$$

This means that, when x is any one of the x_i, and $1 \leq j \leq k$,

$$\begin{aligned} \langle w,x\rangle < \theta_j &\implies \langle \hat{w},x\rangle < \hat{\theta}_j + \gamma/2, \\ \langle w,x\rangle > \theta_j &\implies \langle \hat{w},x\rangle > \hat{\theta}_j - \gamma/2, \\ \langle \hat{w},x\rangle \leq \hat{\theta}_j + \gamma/2 &\implies \langle w,x\rangle < \theta_j + \gamma, \\ \langle \hat{w},x\rangle \geq \hat{\theta}_j - \gamma/2 &\implies \langle w,x\rangle > \theta_j - \gamma. \end{aligned}$$

It follows that $\mathrm{er}_{s'}^{\gamma/2}(\hat{h}) \geq \mathrm{er}_{s'}(h)$ and $\mathrm{er}_s^{\gamma}(h) \geq \mathrm{er}_s^{\gamma/2}(\hat{h})$. So, if we have $\sigma z = (s,s') \in T$ and $\mathrm{er}_{s'}(h) \geq \mathrm{er}_s^{\gamma}(h) + \varepsilon/2$, then

$$\mathrm{er}_{s'}^{\gamma/2}(\hat{h}) \geq \mathrm{er}_{s'}(h) \geq \mathrm{er}_s^{\gamma}(h) + \varepsilon/2 \geq \mathrm{er}_s^{\gamma/2}(\hat{h}) + \varepsilon/2.$$

The proof now proceeds as the proof of Theorem 7. For any $z \in Z^{2m}$,

$$\Pr(\sigma z \in T) \leq \Pr\left(\sigma z \in \bigcup_{\hat{h}\in\hat{H}} S(\hat{h})\right),$$

where

$$S(\hat{h}) = \{(s,s') \in Z^{2m} : \mathrm{er}_{s'}^{\gamma/2}(\hat{h}) \geq \mathrm{er}_s^{\gamma/2}(\hat{h}) + \varepsilon/2\}.$$

Fixing $\hat{h} \in \hat{H}$, we find that, by Hoeffding's inequality,

$$\Pr\left(\sigma z \in S(\hat{h})\right) \leq \exp(-\varepsilon^2 m/8).$$

Therefore,

$$P^m(Q) < 2|\hat{H}|\exp(-\varepsilon^2 m/8) \leq 4\,2^{576R^2/\gamma^2 \log_2(80Rm/\gamma)} \left(\frac{10R}{\gamma}\right)^k \exp(-\varepsilon^2 m/8).$$

So, with probability at least $1-\delta$, for all $h \in H$,

$$\mathrm{er}_P(h) < \mathrm{er}_s(h) + \sqrt{\frac{8}{m}\left(\left(\frac{576R^2}{\gamma^2}\right)\ln\left(\frac{80Rm}{\gamma}\right) + k\ln\left(\frac{10R}{\gamma}\right) + \ln\left(\frac{4}{\delta}\right)\right)}.$$

The result follows on noting that the bound stated in the theorem is trivially true if $m < R^2/\gamma^2$, and is implied by the bound just derived if $m \geq R^2/\gamma^2$. ■

For the case in which the margin error is zero, a better bound can be derived.

Theorem 13 *Suppose $R > 0$ and $Z = B_R \times \{0,1\}$, where $B_R = \{x \in \mathbb{R}^n : \|x\| \leq R\}$. Fix $k \in \mathbb{N}$ and let H be the set of all k-level threshold functions defined on domain B_R. Let P be any probability distribution on Z, and suppose $\gamma \in (0,1]$ and $\delta \in (0,1)$. Then, with P^m-probability at least $1-\delta$, a sample s is such that if $h \in H$ and $\mathrm{er}_s^{\gamma}(h) = 0$, then*

$$\mathrm{er}_P(h) < \frac{2}{m}\left(\frac{1152R^2}{\gamma^2}\log_2(9m) + k\log_2\left(\frac{10R}{\gamma}\right) + \log_2\left(\frac{2}{\delta}\right)\right).$$

Proof: This result is obtained by modifying the proof of Theorem 12, just in the same way as Theorem 8 is obtained by modifying the proof of Theorem 7. First, one uses the fact that if

$$Q = \{s \in Z^m : \exists h \in H \text{ with } \mathrm{er}_s^\gamma(h) = 0, \mathrm{er}_P(h) \geq \varepsilon\}$$

and

$$T = \{(s,s') \in Z^m \times Z^m : \exists h \in H \text{ with } \mathrm{er}_s^\gamma(h) = 0, \mathrm{er}_{s'}(h) \geq \varepsilon/2\},$$

then, for $m \geq 8/\varepsilon$, $P^m(Q) \leq 2P^{2m}(T)$. As before, $P^{2m}(T) \leq \max_{z \in Z^{2m}} \Pr(\sigma z \in T)$, where Pr denotes the probability over uniform choice of σ from the 'swapping group' G. Then, it can be seen that for any $z \in Z^{2m}$,

$$\Pr(\sigma z \in T) \leq \Pr\left(\sigma z \in \bigcup_{\hat{h} \in \hat{H}} S(\hat{h})\right),$$

where $S(\hat{h}) = \{(s,s') \in Z^{2m} : \mathrm{er}_s^{\gamma/2}(h) = 0, \mathrm{er}_{s'}^{\gamma/2}(\hat{h}) \geq \varepsilon/2\}$ and where $\hat{H}$ is as in the proof of Theorem 12. Arguing as in the proof of Theorem 8, if $S(\hat{h}) \neq \emptyset$, so that for some $\tau \in G$, $\tau z = (s,s') \in S(\hat{h})$, then $\Pr\left(\sigma z \in S(\hat{h})\right) = \Pr\left(\sigma(\tau z) \in S(\hat{h})\right)$. Supposing that $\mathrm{er}_{s'}^{\gamma/2}(\hat{h}) = r/m$, where $r \geq \varepsilon m/2$ is the number of x_i in s' not classified with margin $\gamma/2$ by $\hat{h}$, we see that there are at most $2^{m-r} \leq 2^{m-\varepsilon m/2}$ σ such that $\sigma(\tau z) \in S(\hat{h})$. Hence, for each $\hat{h} \in \hat{H}$,

$$\Pr\left(\sigma z \in S(\hat{h})\right) \leq \frac{2^{m(1-\varepsilon/2)}}{|G|} = 2^{-\varepsilon m/2}.$$

The proof then proceeds as does the proof of Theorem 12. ■

5.2 Uniform Margin-based Bounds for Multilevel Threshold Functions

It is straightforward to remove the *a priori* specification of γ and k, using Theorem 9. The following bounds are obtained.

Theorem 14 *Suppose $R > 0$ and $Z = B_R \times \{0,1\}$, where $B_R = \{x \in \mathbb{R}^n : \|x\| \leq R\}$. Let H be the set of all multilevel threshold functions defined on domain B_R. Let P be any probability distribution on Z. Then, with P^m-probability at least $1-\delta$, the following hold:*

1. for all $k \in \mathbb{N}$ *and* for all $\gamma \in (0,1]$, *if $h \in H$ is a k-level threshold function, then*

$$\mathrm{er}_P(h) < \mathrm{er}_s^\gamma(h) + \varepsilon(\gamma, \delta, k, m)$$

where $\varepsilon = \varepsilon(\gamma, \delta, k, m)$ is given by

$$\varepsilon = \sqrt{\frac{8}{m}\left(\frac{4608R^2}{\gamma^2}\ln(9m) + k + k\ln\left(\frac{20R}{\gamma}\right) + \ln\left(\frac{4}{\delta}\right) + 2\ln\left(\log_2\left(\frac{4}{\gamma}\right)\right)\right)}.$$

2. for all $k \in \mathbb{N}$, *and* for all $\gamma \in (0,1]$, *if $h \in H$ is a k-level threshold function and h classifies s with margin γ, then*

$$\mathrm{er}_P(h) < \frac{2}{m}\left(\frac{4608R^2}{\gamma^2}\log_2(9m) + k + k\log_2\left(\frac{20R}{\gamma}\right) + \log_2\left(\frac{4}{\delta}\right) + 2\ln\left(\log_2\left(\frac{4}{\gamma}\right)\right)\right).$$

Proof: Let $E(\gamma_1,\gamma_2,\delta)\subseteq Z^m$ be the event that there exists $h\in H$ with k terms such that

$$\mathrm{er}_P(h)\geq \mathrm{er}_s^{\gamma_2}(h)+\varepsilon'(\gamma_1,\delta,k,m),$$

where

$$\varepsilon'(\gamma,\delta,k,m)=\sqrt{\frac{8}{m}\left(\frac{1152R^2}{\gamma^2}\ln(9m)+k\ln\left(\frac{10R}{\gamma}\right)+\ln\left(\frac{4}{\delta}\right)\right)}.$$

Then, by Theorem 12, $P^{2m}(E(\gamma,\gamma,\delta))\leq\delta$. It is also clear that $0<\gamma_1\leq\gamma\leq\gamma_2<1$ and $0<\delta_1\leq\delta\leq 1$ imply $E(\gamma_1,\gamma_2,\delta_1)\subseteq E(\gamma,\gamma,\delta)$. By Theorem 9, with $\delta/2^k$ in place of δ, we therefore have that, for any fixed $k\in\mathbb{N}$, with probability at least $1-\delta/2^k$, for *all* $\gamma\in(0,1]$, every k-level threshold function h satisfies

$$\mathrm{er}_P(h)<\mathrm{er}_s^{\gamma}(h)+\varepsilon'(\gamma/2,\delta c(\gamma)/2^k,k,m),$$

where $c(\gamma)=1/\left(\log_2(4/\gamma)\right)^2$. Thus, with probability at least $1-\delta$, *for all* $\gamma\in(0,1]$ and *all* $k\in\mathbb{N}$, every k-level threshold function has

$$\mathrm{er}_P(h)<\mathrm{er}_s^{\gamma}(h)+\varepsilon'(\gamma/2,\delta c(\gamma)/2^k,k,m)\leq\mathrm{er}_s^{\gamma}(h)+\varepsilon(\gamma,\delta,k,m).$$

The first part of the result now follows, and the second is proved similarly, using Theorem 13. ■

5.3 Comparison with the Bounds for General Threshold Decision Lists

The generalization error bound implied by Theorem 7 in the case in which $\gamma_i=\gamma$ for all i is, suppressing constants,

$$\mathrm{er}_P(h)<\mathrm{er}_s^{\gamma}(h)+O\left(\sqrt{\frac{1}{m}\left(\frac{R^2k}{\gamma^2}\ln m+\ln\left(\frac{1}{\delta}\right)\right)}\right)$$

(with probability at least $1-\delta$), whereas that of Theorem 12 is

$$\mathrm{er}_P(h)<\mathrm{er}_s^{\gamma}(h)+O\left(\sqrt{\frac{1}{m}\left(\frac{R^2}{\gamma^2}\ln m+k\ln\left(\frac{R}{\gamma}\right)+\ln\left(\frac{1}{\delta}\right)\right)}\right),$$

so there is some advantage in the more particular analysis that has been carried out for multi-level threshold functions. Similar comments apply to the respective 'uniform' bounds of Theorem 10 and Theorem 14.

6. Conclusions and Further Work

This paper has derived different types of theoretical bounds on the generalization error of threshold decision lists. Applying the standard PAC model, by bounding the growth functions, we have given bounds for threshold decision lists and multilevel threshold functions. We then derived generalization error bounds that involve the margins by which successive planes in the threshold decision list 'clear' the training examples. These bounds improve upon those that follow (for the special case in which the margin error is zero) from earlier results of Bennett et al. (2000) and Shawe-Taylor

and Cristianini (1998). Although threshold decision lists have been the focus of this paper, we have also presented generalization error bounds for perceptron decision trees that improve and extend (to the case in which margin error need not be zero) previous such bounds from Bennett et al. (2000) and Shawe-Taylor and Cristianini (1998). For the subclass of multilevel threshold functions (those threshold decision lists in which the defining hyperplanes may be taken to be parallel), a different approach to constructing empirical covers has been shown to lead to better margin-based bounds than those that would follow from the general bounds obtained for threshold decision lists.

There are several possible directions for further investigation.

We used upper bounds on the growth functions of threshold decision lists and multilevel threshold functions to upper bound generalization error. An interesting combinatorial question concerns the VC-dimension of these classes. Lower bounds on the VC-dimension would provide worst-case lower bounds on generalization error (see Ehrenfeucht et al., 1989; Anthony and Biggs, 1992; Anthony and Bartlett, 1999; Blumer et al., 1989). Certainly, upper bounds on the VC-dimensions follow from the bounds we obtained on the growth functions, but these are quite likely to be loose and a more direct attempt might be productive in obtaining not only better upper bounds, but also lower bounds, on the VC-dimension.

There are other approaches to deriving generalization error bounds. Of particular importance recently have been methods using Rademacher complexity and local Rademacher complexity, together with concentration-of-measure results (Bartlett and Mendelson, 2001; Mendelson, 2003; Bartlett et al., 2002; Bousquet et al., 2002; Bousquet, 2003). It would be interesting to investigate such approaches for threshold decision lists.

The margin-based results obtained here for multilevel threshold functions only involve a single margin parameter rather than separate ones for each plane, and it is possible that a different approach might permit such added flexibility.

We have not considered in this paper the algorithmics of learning threshold decision lists. As mentioned, heuristics for learning threshold decision lists were studied by Marchand and Golea (1993), and although no theoretical generalization error bounds were derived there, the techniques appeared to perform well in experiments. Furthermore, the perceptron decision tree algorithms FAT, MOC1, and MOC2 due to Bennett et al. (2000) are variants of the OC1 algorithm (Murthy et al., 1994) that are explicitly driven by the aim of maximising the margins at the decision nodes. It would be interesting to modify the techniques of Marchand and Golea (1993) with a view to obtaining large margins, and to modify the algorithms of Bennett et al. (2000) so as to learn a threshold decision list rather than a perceptron decision tree.

Acknowledgements

I am grateful to the referees for a number of helpful suggestions and, in particular, for comments leading to an improvement of Theorem 9.

References

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler (1997). Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM* 44(4): 615–631.

M. Anthony (2001). *Discrete Mathematics of Neural Networks: Selected Topics*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathemat-

ics, Philadelphia, PA.

M. Anthony (2002). Partitioning points by parallel planes. RUTCOR Research Report RRR-39-2002, Rutgers Center for Operations Research. (Also, CDAM research report LSE-CDAM-2002-10, Centre for Discrete and Applicable Mathematics, London School of Economics.) To appear, *Discrete Mathematics*.

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

M. Anthony and P. L. Bartlett (2000). Function learning from interpolation. *Combinatorics, Probability and Computing*, 9: 213–225.

M. Anthony and N. L. Biggs (1992). *Computational Learning Theory: An Introduction*. Cambridge Tracts in Theoretical Computer Science, 30. Cambridge University Press, Cambridge, UK.

M. Anthony, G. Brightwell and J. Shawe-Taylor (1995). On specifying Boolean functions by labelled examples. *Discrete Applied Mathematics*, 61: 1–25.

P. L. Bartlett (1998). 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.

P. L. Bartlett, O. Bousquet and S. Mendelson (2002), Localized Rademacher complexities. *Proceedings of the* 15*th Annual Conference on Computational Learning Theory*, ed. J. Kivinen and R. H. Sloan. Springer Lecture Notes in Artificial Intelligence 2375.

P. Bartlett and S. Mendelson (2001), Rademacher and Guassian complexities: risk bounds and structural results. In *Proceedings of the* 14*th Annual Conference on Computational Learning Theory*, Lecture Notes in Artificial Intelligence, Springer, 224-240.

E. Baum and D. Haussler (1989). What size net gives valid generalization? *Neural Computation*, 1(1): 151–160.

K. Bennett, N. Cristianini, J. Shawe-Taylor and D. Wu (2000). Enlarging the Margins in Perceptron Decision Trees. *Machine Learning*, 41: 295–313.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4): 929–965.

V. Bohossian and J. Bruck (1998). Multiple threshold neural logic. In *Advances in Neural Information Processing, Volume 10: NIPS'1997*, Michael Jordan, Michael Kearns, Sara Solla (eds), MIT Press.

O. Bousquet (2003). New Approaches to Statistical Learning Theory. Annals of the Institute of Statistical Mathematics 55 (2): 371-389.

S. Boucheron, G. Lugosi and P. Massart (2000). A sharp concentration inequality with applications. *Random Structures and Algorithms*, 16: 277–292.

O. Bousquet, V. Koltchinskii and D. Panchenko (2002). Some local measures of complexity on convex hulls and generalization bounds. *Proceedings of the* 15*th Annual Conference on Computational Learning Theory*, ed. J. Kivinen and R. H. Sloan. Springer Lecture Notes in Artificial Intelligence 2375.

T. M. Cover (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Trans. Electronic Computers* 14: 326–334.

N. Cristianini and J. Shawe-Taylor (2000). *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK.

R. M. Dudley (1999). *Uniform Central Limit Theorems*, Cambridge Studies in Advanced Mathematics, 63, Cambridge University Press, Cambridge, UK.

A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant (1989). A general lower bound on the number of examples needed for learning. *Information and Computation*, 82: 247–261.

P. L Hammer, T. Ibaraki and U. N. Peled (1981). Threshold numbers and threshold completions. *Annals of Discrete Mathematics* 11: 125–145.

R. G. Jeroslow (1975). On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11: 119–124.

O. L. Mangasarian (1968). Multisurface method of pattern separation. *IEEE Transactions on Information Theory* IT-14 (6): 801–807.

M. Marchand and M. Golea (1993). On learning simple neural concepts: from halfspace intersections to neural decision lists. *Network: Computation in Neural Systems*, 4: 67–85.

M. Marchand, M. Golea and P. Ruján (1990). A convergence theorem for sequential learning in two-layer perceptrons. *Europhys. Lett.* 11: 487–492.

M. Marchand, M. Shah, J. Shawe-Taylor and M. Sokolova (2003). The Set Covering Machine with Data-Dependent Half-Spaces. Proceedings of the Twentieth International Conference on Machine Learning (ICML'2003), 520–527, Morgan Kaufmann, San Francisco CA.

S. Mendelson (2003). A few notes on Statistical Learning Theory. In *Advanced Lectures in Machine Learning*, (S. Mendelson, A. J. Smola Eds), LNCS 2600, 1-40, Springer.

S. K. Murthy, S.Kasif and S. Salzberg (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* 2: 1–32.

A. Ngom, I. Stojmenović and J. Žunić (2003). On the number of multilinear partitions and the computing capacity of multiple-valued multiple-threshold perceptrons, IEEE Transactions on Neural Networks 14(3): 469–477.

Z. Obradović and I. Parberry (1994). Learning with discrete multivalued neurons. *Journal of Computer and System Sciences* 49: 375–390.

S. Olafsson and Y. S. Abu-Mostafa (1988). The capacity of multilevel threshold functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10 (2): 277–281.

D. Pollard (1984). *Convergence of Stochastic Processes*. Springer-Verlag.

J. R. Quinlan and R. Rivest (1989). Inferring decision trees using the minimum description length principle. *Information and Computation* 80: 227–248.

R. L. Rivest (1987). Learning Decision Lists. *Machine Learning* 2 (3): 229–246.

J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson and M. Anthony (1996). Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5): 1926–1940.

J. Shawe-Taylor and N. Cristianini (1998). Data-Dependent Structural Risk Minimisation for Perceptron Decision Trees. Neurocolt Technical Report NC2-TR-1998-003.

A. J. Smola, P. L. Bartlett, B. Schölkopf and D. Schuurmans (editors) (2000). *Advances in Large-Margin Classifiers (Neural Information Processing)*, MIT Press.

M. Sokolova, M. Marchand, N. Japkowicz, and J. Shawe-Taylor (2003). The Decision List Machine. Advances in Neural Information Processing Systems 15, 921–928, MIT-Press, Cambridge, MA, USA.

R. Takiyama (1985). The separating capacity of a multi-threshold element. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7: 112–116.

G. Turán and F. Vatan (1997). Linear decision lists and partitioning algorithms for the construction of neural networks. Foundations of Computational Mathematics: selected papers of a conference held at Rio de Janeiro, Springer, 414-423.

L. G. Valiant (1984). A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142.

V. N. Vapnik (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York..

V. N. Vapnik (1998). *Statistical Learning Theory*, Wiley.

V. N. Vapnik and A. Y. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2): 264–280.

T. Zhang (2002). Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2: 527–550.

A. Zuev and L. I. Lipkin (1988). Estimating the efficiency of threshold representations of Boolean functions. *Cybernetics* 24: 713–723. (Translated from Kibernetika (Kiev), 6, 1988: 29–37.)

Journal of Machine Learning Research 5 (2004) 219–238 Submitted 12/02; Published 3/04

On the Importance of Small Coordinate Projections

Shahar Mendelson SHAHAR.MENDELSON@ANU.EDU.AU
Petra Philips PETRA.PHILIPS@ANU.EDU.AU
RSISE, The Australian National University
Canberra, ACT 0200, Australia

Editor: Peter Bartlett

Abstract

It has been recently shown that sharp generalization bounds can be obtained when the function class from which the algorithm chooses its hypotheses is "small" in the sense that the Rademacher averages of this function class are small. We show that a new more general principle guarantees good generalization bounds. The new principle requires that random coordinate projections of the function class evaluated on random samples are "small" with high probability and that the random class of functions allows symmetrization. As an example, we prove that this geometric property of the function class is exactly the reason why the two lately proposed frameworks, the *luckiness* (Shawe-Taylor et al., 1998) and the *algorithmic luckiness* (Herbrich and Williamson, 2002), can be used to establish generalization bounds.

Keywords: Statistical learning theory, generalization bounds, data-dependent complexities, coordinate projections

1. Introduction

Generalization bounds are used to show that, with high probability, functions produced by a learning algorithm have a small error, and as such, can be used to approximate the unknown target. For many years, such bounds were obtained by deviation estimates of empirical means from the actual mean, *uniformly* over the whole class of functions from which the algorithm produces its hypothesis. Thus, classes which satisfy the uniform law of large numbers (so-called uniform Glivenko-Cantelli or GC classes) have played a central role in Machine Learning literature. More recently, other methods of deriving generalization bounds were developed, in which the "size" of the function class from which the algorithm chooses a hypothesis is not specified *a priori*. Examples of such methods are the *luckiness* (Shawe-Taylor et al., 1998) and the *algorithmic luckiness* (Herbrich and Williamson, 2002) frameworks, but although in both cases one can obtain generalization bounds, they seem to be based on completely different arguments.

In this article, we show that the bounds obtained in all these frameworks follow from the same general principle. This principle requires that coordinate projections of a function subclass on random samples are "small" with high probability.

We consider the following setting for the learning problem: let Ω be a measurable input space, $t : \Omega \longrightarrow \mathbb{R}$ an unknown real-valued target function, $H = \{h | h : \Omega \longrightarrow \mathbb{R}\}$ a class of hypothesis functions, and μ an unknown probability distribution on Ω. Let $(X_1, ..., X_n) \in \Omega^n$ be a finite training sample, where each X_i is generated randomly, independently, according to μ. Based on the values of the target function on this sample, $(t(X_1), ..., t(X_n))$, the goal of a learning algorithm is to choose a function $h^* \in H$ which is a good estimator of the target function t. A quantitative measure of how

well a function $h \in H$ approximates t is given by a loss function $l : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$. Typical examples of loss functions are the 0-1 loss for classification defined by $l(r,s) = 0$ if $r = s$ and $l(r,s) = 1$ if $r \neq s$ or the square-loss for regression tasks $l(r,s) = (r-s)^2$. In what follows we will assume a bounded loss function and therefore, without loss of generality, $l : \mathbb{R} \times \mathbb{R} \longrightarrow [-1,1]$. For every $h \in H$ we define the associated loss function $l_h : \Omega \longrightarrow [-1,1]$, $l_h(x) := l(h(x),t(x))$ and denote by $F = \{l_h : \Omega \longrightarrow [-1,1] \,|\, h \in H\}$ the loss class associated with the learning problem. If h^* is the best estimate for t in H, we call $F' = \{l_h - l_{h^*} | \, h \in H\}$ the excess loss class.

The best estimate for t is defined to be the $h^* \in H$ for which the expected loss (also called risk) over all possible observations is as small as possible, that is, $\int l_{h*}(x)d\mu(x) = \mathbb{E}_\mu l_{h*} \approx \inf_{h \in H} \mathbb{E}_\mu l_h$. Empirical risk minimization algorithms are based on the philosophy that it is possible to approximate this expectation with the empirical mean, and choose instead a hypothesis $\hat{h} \in H$ for which $\frac{1}{n}\sum_{i=1}^n l_{\hat{h}}(x_i) \approx \inf_{h \in H} \frac{1}{n}\sum_{i=1}^n l_h(x_i)$. Therefore, the relationship between expected and empirical loss is of crucial importance for the performance of these learning algorithms.

In the classical approach to obtain generalization bounds, called GC-type bounds, one investigates the probability that, for *any* hypothesis in the class, the deviation of the empirical mean from the actual mean of its associated loss function is larger than a given threshold, that is,

$$Pr\Big\{\sup_{f \in F} \big|\mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^n f(X_i)\big| \geq t\Big\}, \tag{1.1}$$

where μ is a probability measure on Ω, $X_1, ..., X_n$ are independent random variables distributed according to μ, F is the loss class associated with the learning problem, and $\mathbb{E}_\mu f$ denotes $\mathbb{E} f(X_i)$.

Classes of functions F which, independently of the underlying measure μ, satisfy the law of large numbers, that is, the probability (1.1) tends to 0 as n goes to infinity uniformly in μ, are called *uniform Glivenko-Cantelli classes*. For these classes learning is possible. Historically, uniform Glivenko-Cantelli classes were characterized by a finite combinatorial dimension (e.g., a finite VC dimension in the 0,1-case) (see Vapnik and Chervonenkis, 1971; Alon et al., 1997). The ability to bound the tails of the random variable in (1.1) is therefore due to the fact that the class F is "small" in some sense.

In Mendelson (2002a,b) it has been shown that parameters which also characterize the uniform Glivenko-Cantelli property are the *uniform Rademacher averages* of the class F, defined as

$$R_n(F) := \sup_{\{x_1,...,x_n\} \subset \Omega^n} \mathbb{E}_\varepsilon \sup_{f \in F} \Big|\sum_{i=1}^n \varepsilon_i f(x_i)\Big|,$$

where $(\varepsilon_i)_{i=1}^n$ are independent, symmetric, $\{-1,1\}$-valued random variables, also known as Rademacher random variables. The necessary and sufficient condition for a class F to be a uniform Glivenko-Cantelli class is that $R_n(F) = o(n)$. In this case, tail estimates for (1.1) which are independent of the underlying measure μ can be as fast as the order of e^{-cnt^2}. Therefore, the Rademacher averages $R_n(F)$ seem to be a reasonable notion of "size" for a function class F in the context of learning via the uniform law of large numbers.

It was shown in Mendelson (2002b) that—if F satisfies mild structural assumptions—it is possible to derive sharper bounds on the learning problem by bounding the probability

$$Pr\Big\{\exists f \in F,\ \frac{1}{n}\sum_{i=1}^n f(X_i) \leq t,\ \mathbb{E}_\mu f \geq 2t\Big\} \tag{1.2}$$

instead of (1.1). The difference in the two cases is that in (1.1) one controls the deviation of the empirical means from the actual one for *all* the functions in the class, whereas in (1.2) the control is only for functions with "small" empirical mean, that is, potential minimizers for the actual mean. The tail estimates for (1.2) can be as good as e^{-cnt} instead of e^{-cnt^2} and are governed by the Rademacher averages $R_n(F^t)$ of the class $F^t := \{f \in F : \mathbb{E}_\mu f^2 \leq t\}$.

In other words, in both these cases measure independent estimates depend on the fact that for *all* coordinate projections, the projected sets $\big\{\big(f(x_1),...,f(x_n)\big) : \ f \in F\big\}$ and respectively $\big\{\big(f(x_1),...,f(x_n)\big) : \ f \in F^t\big\}$ are small in the sense that they have small Rademacher averages. Clearly, this is a property of the class F, and if one wishes to obtain useful bounds, one has to assume *a priori* that F is small.

In this article we show that the fact that classes have "small" coordinate projections with high probability (for a fixed probability measure) is the reason that the tails of (1.1) and (1.2) are well behaved. More surprising is the fact that this is also the reason why several other (seemingly very different) approaches yield generalization bounds.

The method of analysis we use is a combination of a symmetrization with respect to a random subclass and sharp concentration results. The need to investigate random subclasses is simple, as the starting point is that, ultimately, one wants to control the generalization ability *only* for the hypothesis functions which are reachable by the specific learning algorithm when presented with the actual training sample. Therefore, it suffices to obtain estimates on

$$Pr\Big\{\sup_{f\in F'}\big|\mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^{n} f(X_i)\big| \geq t\Big\}, \tag{1.3}$$

where $F' \subset F$ and $\hat{f} \in F'$, where $\hat{f}$ is the loss function associated with the hypothesis produced by the algorithm from the sample $(X_1,\ldots,X_n)$. Although one can hope that F' has a smaller "size" than F, it is not possible to use the classical uniform generalization bounds because F' depends on the random training sample and could change with the sample.

The outline of this paper is as follows: in Section 2 we will first present a symmetrization procedure which is performed with respect to a random subclass of functions. The proof is a modification of the original proof of the standard symmetrization argument. The probability in Equation (1.3) can be therefore related to the probability of having large Rademacher sums for the (random) coordinate projections of this random subclass. Sharp generalization bounds can be obtained when the "size" of the set of coordinate projections of the random subclass is "small" in the sense that with high probability the Rademacher averages associated with a random projection of a random subclass are small (Corollary 2.5). We conclude that the general principle which ensures learnability and fast error rates consists of the combination of three main ingredients: a symmetrization procedure, a sharp concentration inequality, and a small "size" for the set of random coordinate projections.

In Section 3 we show how apparently different approaches fall within this general framework. In Section 3.1 we present the GC case (1.1) as an easy corollary of the symmetrization procedure. It is considerably more difficult to obtain the sharp rates for (1.2), a fact which is demonstrated in Section 3.2. In both these examples, the class whose coordinate projections have to be controlled is not random, but a deterministic subset of F.

There are several examples in which one really requires random subclasses of F. The examples we shall present are of the luckiness and algorithmic luckiness frameworks. The reason we chose these examples and not others is because our methods give considerably shorter proofs that seem to clarify the reason why luckiness and algorithmic luckiness work to the extent that they do.

In the luckiness and algorithmic luckiness frameworks, it is possible to avoid the detour via the worst-case quantity in (1.1) and to derive bounds using additional prior knowledge on the learning algorithm or on the training sample. The bounds stated in these approaches are, as opposed to the existing classical ones, data- or algorithm-dependent.

In the luckiness framework (Shawe-Taylor et al., 1998), prior knowledge about the connection between the actual sample and the functions in F is quantified through a luckiness function. A "fortunate" property of this luckiness function (ω-smallness) ensures good tail estimates for (1.3). One example for a luckiness function is the size of the margin for linear classifiers.

The algorithmic luckiness framework (Herbrich and Williamson, 2002) generalizes the luckiness framework. Prior knowledge about the link between the functions learned by the algorithm and the actual sample are formulated through an algorithmic luckiness function whose property of "ω-smallness" enables one to bound the generalization error.

Although there are no examples in which these methods yield results clearly better than other approaches, one merit of these two frameworks is that they set up a possible theoretical basis that could enable one to directly estimate data-dependent and algorithm-dependent generalization bounds. Unfortunately, the "ω-smallness" property is somewhat technically complicated and seems unnatural. The notion of complexity for the function class employed in these frameworks is covering numbers, and it was unexplored how these frameworks link to approaches using Rademacher averages as a notion of size.

We show that the ω-smallness condition is just a way of ensuring that a random coordinate projection of the random set is "small" and this suffices to recover the original generalization bounds. Hence, both luckiness and algorithmic luckiness fall within the general framework.

Note also that other examples of generalization bounds for data-dependent hypothesis classes that could be obtained using our methods are presented in Gat (1999) and Cannon et al. (2002). In fact, the proofs in Gat (1999) and Cannon et al. (2002) are based on a similar symmetrization argument. The notion of "size" which is employed is simply that of cardinality, and thus these results easily fall within the framework presented here (for details see Mendelson and Philips, 2003).

We end this introduction with some notation which will be used throughout the article. In the following, F is a class of real-valued functions defined on a measurable space Ω which take values in $[-1,1]$ and μ is a probability measure on Ω. Ω^n denotes the product space $\Omega \times \cdots \times \Omega$. Let $X_1,...,X_n$ be independent random variables distributed according to μ and let $(Y_1,...,Y_n)$ be an independent copy of $(X_1,...,X_n)$. μ_n denotes the random empirical probability measure supported on $\{X_1,...,X_n\}$, that is, $\mu_n := n^{-1}\sum_{i=1}^n \delta_{X_i}$. Pr_μ and $\mathbb{E}_\mu$ are the probability and the expectation with respect to μ and Pr_X and $\mathbb{E}_X$ denote the probability and the expectation with respect to the random vector $X = (X_1,\ldots,X_n)$ (and therefore with respect to μ^n). $\mathbb{E}_\mu f$ is the expectation and $\mathrm{var}(f)$ is the variance of the random variable $f(X_i)$. In general, for any random variable Z, Pr_Z and $\mathbb{E}_Z$ are the probability and the expectation with respect to the distribution of Z.

Let F/X be the random set $\{(f(X_1),...,f(X_n)) : f \in F\}$, that is, the coordinate projection of the set F onto the random set of coordinates X. $VC(F)$ is the VC-dimension of F if F is a boolean class of functions.

Set ℓ_p^n to be $\mathbb{R}^n$ with the norm $\|x\|_p := \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$ and put B_p^n to be the unit ball of ℓ_p^n. ℓ_∞^n is $\mathbb{R}^n$ endowed with the norm $\|x\|_\infty := \sup_{1\le i\le n}|x_i|$, let $L_\infty(\Omega)$ be the set of bounded functions on Ω with respect to the norm $\|f\|_\infty := \sup_{\omega\in\Omega}|f(\omega)|$, and denote its unit ball by $B\big(L_\infty(\Omega)\big)$. For a probability measure μ on a measurable space Ω and $1 \le p < \infty$, let $L_p(\mu)$ be the space of measurable functions on Ω with a finite norm $\|f\|_{L_p(\mu)} := (\int |f|^p d\mu)^{1/p}$.

Let (Y,d) be a metric space. If $F \subset Y$ then for every $\varepsilon > 0$, $N(\varepsilon,F,d)$ is the minimal number of open balls (with respect to the metric d) needed to cover F. A corresponding set $\{y_1,...,y_m\} \subset Y$ of minimal cardinality such that for every $f \in F$ there is some y_i with $d(f,y_i) < \varepsilon$ is called an ε-cover of F. For $1 \leq p < \infty$, denote by $N\big(\varepsilon,F,L_p(\mu_n)\big)$ the covering number of F at scale ε with respect to the $L_p(\mu_n)$ norm. Similarly, one can define the packing number at scale ε, which is the maximal cardinality of a set $\{y_1,...,y_k\} \subset F$ such that for every $i \neq j$, $d(y_i,y_j) \geq \varepsilon$. Denote the ε-packing numbers by $M(\varepsilon,F,d)$ and note that for every $\varepsilon > 0$, $N(\varepsilon,F,d) \leq M(\varepsilon,F,d) \leq N(\varepsilon/2,F,d)$. If S is a set, we denote its complement by S^c.

Finally, throughout this article all absolute constants are denoted by c, C or K. Their values may change from line to line, or even within the same line.

2. Symmetrization

For every integer n, let F_n and F_n^{sym} be set-valued functions which assign to each $\sigma_n \in \Omega^n$ a subset of F. We assume that F_n^{sym} is invariant to permutations, that is, for every $\sigma_n \in \Omega^n$ and every permutation $\pi(\sigma_n)$ of σ_n, $F_n^{\mathrm{sym}}(\sigma_n) = F_n^{\mathrm{sym}}(\pi(\sigma_n))$, in which case we say that F_n^{sym} is symmetric.

The question we wish to address in this section is how to estimate the probability

$$Pr_X\Big\{ \sup_{f \in F_n(\sigma_n)} \big|\mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^n f(X_i)\big| \geq t \Big\}, \tag{2.1}$$

where $\sigma_n = (X_1,...,X_n)$ is a random sample. Note that the worst-case probability (1.1) is a special case of (2.1), where F_n is the constant function mapping every sample to the whole function class F, $F_n(\sigma_n) = F$. Another extreme case occurs when $F_n(\sigma_n) = \{\hat{f}\}$, where $\hat{f}$ is the loss function associated with the hypothesis produced by a learning algorithm from the sample σ_n.

We will show that by employing an additional assumption on the functions F_n and F_n^{sym} which relates the random subsets $F_n(\sigma_n)$ to *symmetric* random subsets dependent on a double-sample, it is possible to upper bound (2.1) in terms of Rademacher sums associated with sets of coordinate projections.

Assumption 2.1 *There exists a constant $\delta > 0$ such that for every $t > 0$,*

$$\begin{aligned} &Pr_{X\times Y}\Big\{ \exists f \in F_n(\sigma_n),\ \Big|\frac{1}{n}\sum_{i=1}^n \big(f(X_i) - f(Y_i)\big)\Big| \geq t \Big\} \leq \\ &Pr_{X\times Y}\Big\{ \exists f \in F_{2n}^{sym}(\sigma_n,\tau_n),\ \Big|\frac{1}{n}\sum_{i=1}^n \big(f(X_i) - f(Y_i)\big)\Big| \geq t \Big\} + \delta, \end{aligned} \tag{2.2}$$

where $\sigma_n = (X_1,...,X_n)$ and $\tau_n = (Y_1,...,Y_n)$.

This assumption quantifies that by replacing the original random subset of hypotheses with another symmetric random subset dependent on the double-sample—and which is therefore invariant under permutations of this double-sample—the probability of having large deviations of empirical means evaluated on the sample and ghost sample increases by at most δ and therefore not "too much".

Indeed, in all the applications we present δ can be made as small as we require. One extreme case occurs when for every double-sample (σ_n,τ_n),

$$F_n(\sigma_n) \subseteq F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n),$$

in which case Assumption 2.1 holds trivially with a constant $\delta = 0$. For example, if both set-valued maps are the constant function $F_n(\sigma_n) = F_n^{\rm sym}(\sigma_n) = F$, then $\delta = 0$.

Given F_n, one can always define a mapping $F_{2n}^{\rm sym}$ to satisfy Assumption 2.1 as the symmetric extension of F_n: for every double-sample (σ_n, τ_n), $F_{2n}^{\rm sym}(\sigma_n, \tau_n)$ is defined to be the union of all subsets corresponding to the first half of permutations of the double-sample (σ_n, τ_n),

$$F_{2n}^{\rm sym}(\sigma_n, \tau_n) := \bigcup_{\pi \in S_{2n}} F_n\big(\pi(\sigma_n, \tau_n)|_{i=1}^n\big), \tag{2.3}$$

where S_{2n} is the set of permutations on $(1, \ldots, 2n)$, and $\pi(\sigma_n, \tau_n)|_{i=1}^n$ is the first half of the permuted double-sample. However, Assumption 2.1 allows us to replace the original subset $F_n(\sigma_n)$ even with a potentially "smaller" symmetric subset $F_{2n}^{\rm sym}(\sigma_n, \tau_n)$ as long as the change in probabilities can be controlled.

The importance of Assumption 2.1 lies in the fact that it enables one to bound the probability (2.1) by proving a similar symmetrization argument to that employed in the original proof of the uniform Glivenko-Cantelli property.

The following symmetrization theorem is the main result of this section.

Theorem 2.1 *If Assumption 2.1 holds then for every $t > 0$,*

$$\Big(1 - \frac{4}{nt^2} \sup_{f \in F} {\rm var}(f)\Big) \cdot Pr_X\Big\{\exists f \in F_n(\sigma_n),\ \Big|\mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^n f(X_i)\Big| \geq t\Big\}$$
$$\leq 2Pr_{X\times Y} Pr_\varepsilon\Big\{\exists f \in F_{2n}^{sym}(\sigma_n, \tau_n),\ \Big|\sum_{i=1}^n \varepsilon_i f(X_i)\Big| \geq \frac{nt}{4}\Big\} + \delta, \tag{2.4}$$

where $\sigma_n = (X_1, ..., X_n)$ and $\tau_n = (Y_1, ..., Y_n)$.

As the examples we present in the next section show, most of the standard methods used in Learning Theory fall within the general framework of Theorem 2.1. The advantage of Theorem 2.1 is that it reduces the analysis to a geometric problem, namely, estimating the Rademacher sums associated with the coordinate projection onto $\sigma_n = (X_1, ..., X_n)$ of the random class $F_{2n}^{\rm sym}(\sigma_n, \tau_n)$,

$$F_{2n}^{\rm sym}(\sigma_n, \tau_n)/\sigma_n := \Big\{\big(f(X_1), ..., f(X_n)\big) :\ f \in F_{2n}^{\rm sym}(\sigma_n, \tau_n)\Big\}.$$

The proof is done in two steps which are the same as in the standard symmetrization procedure: a symmetrization by a ghost sample which relates the deviation of the mean from the empirical mean to the deviation of the empirical means evaluated on two different samples; and a symmetrization by signs which relates the latter deviation to the probability of having "large" Rademacher sums $\sup_{f \in F_{2n}^{\rm sym}(\sigma_n, \tau_n)} |\sum_{i=1}^n \varepsilon_i f(X_i)|$. We present the proof for the sake of completeness.

Lemma 2.2 (Symmetrization by a Ghost Sample) *For every $t > 0$,*

$$\Big(1 - \frac{4n}{t^2} \sup_{f \in F} {\rm var}(f)\Big) Pr_X\Big\{\exists f \in F_n(\sigma_n),\ \Big|\sum_{i=1}^n \big(f(X_i) - \mathbb{E}_\mu f\big)\Big| \geq t\Big\}$$
$$\leq Pr_{X\times Y}\Big\{\exists f \in F_n(\sigma_n),\ \Big|\sum_{i=1}^n \big(f(X_i) - f(Y_i)\big)\Big| \geq \frac{t}{2}\Big\}.$$

Proof. Define the random processes $Z_i(f) = f(X_i) - \mathbb{E}_\mu f$ and $W_i(f) = f(Y_i) - \mathbb{E}_\mu f$, and fix $t > 0$. Let $\sigma_n = (X_1, ..., X_n)$, put $\beta = \inf_{f \in F} Pr\{|\sum_{i=1}^n Z_i(f)| \le t/2\}$ and set $A = \{\sigma_n : \exists f \in F_n(\sigma_n),\ |\sum_{i=1}^n Z_i(f)| \ge t\}$. For every element in A there is some $f \in F_n(\sigma_n)$ and a realization of Z_i such that $|\sum_{i=1}^n Z_i(f)| \ge t$. Fix this realization and f and observe that by the triangle inequality, if $|\sum_{i=1}^n W_i(f)| \le t/2$ then $\left|\sum_{i=1}^n \big(Z_i(f) - W_i(f)\big)\right| \ge t/2$. Since $(W_i)_{i=1}^n$ is an independent copy of $(Z_i)_{i=1}^n$,

$$\begin{aligned}\beta &\le Pr_Y\Big\{\Big|\sum_{i=1}^n W_i(f)\Big| \le \frac{t}{2}\Big\} \le Pr_Y\Big\{\Big|\sum_{i=1}^n W_i(f) - \sum_{i=1}^n Z_i(f)\Big| \ge \frac{t}{2}\Big\} \\ &\le Pr_Y\Big\{\exists f \in F_n(\sigma_n),\ \Big|\sum_{i=1}^n W_i(f) - \sum_{i=1}^n Z_i(f)\Big| \ge \frac{t}{2}\Big\}.\end{aligned}$$

Since the two extreme sides of this inequality are independent of the specific selection of f, this inequality holds on the set A. Integrating with respect to X on A it follows that

$$\begin{aligned}&\beta Pr_X\Big\{\exists f \in F_n(\sigma_n),\ \Big|\sum_{i=1}^n Z_i(f)\Big| \ge t\Big\} \\ \le &Pr_X Pr_Y\Big\{\exists f \in F_n(\sigma_n),\ \Big|\sum_{i=1}^n \big(W_i(f) - Z_i(f)\big)\Big| \ge \frac{t}{2}\Big\}.\end{aligned}$$

Finally, to estimate β, note that by Chebyshev's inequality

$$Pr\Big\{\Big|\sum_{i=1}^n Z_i(f)\Big| \ge \frac{t}{2}\Big\} \le \frac{4n}{t^2}\mathrm{var}(f)$$

for every $f \in F$, and thus, $\beta \ge 1 - (4n/t^2)\sup_{f\in F}\mathrm{var}(f)$. ∎

Proposition 2.3 (Symmetrization by Random Signs) *Let F_{2n}^{sym} be a symmetric map. Then, for any probability measure μ and every $t > 0$,*

$$\begin{aligned}&Pr_{X\times Y}\Big\{\exists f \in F_{2n}^{sym}(\sigma_n, \tau_n),\ \Big|\frac{1}{n}\sum_{i=1}^n \big(f(X_i) - f(Y_i)\big)\Big| \ge t\Big\} \\ &\le 2Pr_{X\times Y}Pr_\varepsilon\Big\{\exists f \in F_{2n}^{sym}(\sigma_n, \tau_n),\ \Big|\sum_{i=1}^n \varepsilon_i f(X_i)\Big| \ge \frac{nt}{2}\Big\},\end{aligned}$$

where $\sigma_n = (X_1, ..., X_n)$, $\tau_n = (Y_1, ..., Y_n)$, and $(\varepsilon_i)_{i=1}^n$ are independent Rademacher variables.

Proof. By the symmetry of F_{2n}^{sym} it follows that for every $\{\varepsilon_1, ..., \varepsilon_n\} \in \{-1, 1\}^n$,

$$\begin{aligned}&Pr_{X\times Y}\Big\{\exists f \in F_{2n}^{\mathrm{sym}}(\sigma_n, \tau_n),\ \Big|\frac{1}{n}\sum_{i=1}^n \big(f(X_i) - f(Y_i)\big)\Big| \ge t\Big\} \\ &= Pr_{X\times Y}\Big\{\exists f \in F_{2n}^{\mathrm{sym}}(\sigma_n, \tau_n),\ \Big|\frac{1}{n}\sum_{i=1}^n \varepsilon_i\big(f(X_i) - f(Y_i)\big)\Big| \ge t\Big\}.\end{aligned}$$

Taking the expectation with respect to the random signs (that is, with respect to the Rademacher random variables), the proof follows from the triangle inequality and the fact that $(X_1, ..., X_n)$ has the same distribution as $(Y_1, ..., Y_n)$. ∎

Proof of Theorem 2.1. The claim follows immediately by combining Lemma 2.2, Assumption 2.1 and Proposition 2.3. ∎

We can now relate the Rademacher sums from Theorem 2.1 to the Rademacher averages of $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)/\sigma_n$ by employing concentration inequalities for the random variable

$$Z = \sup_{f\in F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)} \Big|\sum_{i=1}^n \varepsilon_i f(X_i)\Big| = \sup_{v\in F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)/\sigma_n} \Big|\sum_{i=1}^n \varepsilon_i v_i\Big|$$

around its conditional mean $\mathbb{E}_\varepsilon(Z|X_1,...,X_n,Y_1,...,Y_n)$.

For example, we state one particular concentration result which follows directly from martingale methods (see, e.g., McDiarmid, 1989) for functions with bounded differences:

Theorem 2.4 (Concentration) *For every set $V\subset B_\infty^n$ and every $t>0$,*

$$Pr_\varepsilon\Big\{\sup_{v\in V}\Big|\sum_{i=1}^n \varepsilon_i v_i\Big| - \mathbb{E}_\varepsilon\sup_{v\in V}\Big|\sum_{i=1}^n \varepsilon_i v_i\Big| > t\Big\} \le e^{-\frac{t^2}{2n}}. \tag{2.5}$$

Proof. Define $h(\varepsilon_1,\dots,\varepsilon_n) := \sup_{v\in V}\Big|\sum_{i=1}^n \varepsilon_i v_i\Big|$. By the triangle inequality, for every $1\le i\le n$,

$$\sup_{\{\varepsilon_1,\dots,\varepsilon_n,\tilde{\varepsilon}_i\}} |h(\varepsilon_1,\dots,\varepsilon_n) - h(\varepsilon_1,\dots,\varepsilon_{i-1},\tilde{\varepsilon}_i,\varepsilon_{i+1},\dots,\varepsilon_n)| \le 2,$$

and the claim follows directly from McDiarmid's inequality for h. ∎

In particular, setting V to be the (random) coordinate projection of $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$ onto σ_n,

$$V := F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)/\sigma_n = \Big\{\big(f(X_1),...,f(X_n)\big) :\ f\in F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)\Big\}$$

and A_t to be the set of double-samples (σ_n,τ_n) with small Rademacher averages for the projections onto σ_n,

$$A_t := \big\{(\sigma_n,\tau_n)\ :\ \mathbb{E}_\varepsilon\sup_{v\in V}\Big|\sum_{i=1}^n \varepsilon_i v_i\Big| \le nt/8\big\},$$

it follows by the union bound and Equation (2.5) that

$$\begin{aligned} Pr_{X\times Y}Pr_\varepsilon\Big\{\exists f\in F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n),\ \Big|\sum_{i=1}^n \varepsilon_i f(X_i)\Big| > \frac{nt}{4}\Big\} \\ \le Pr_{X\times Y}\{A_t^c\} + e^{-\frac{nt^2}{128}}. \end{aligned}$$

Corollary 2.5 *If Assumption 2.1 holds, then for every $t>0$*

$$\begin{aligned} \Big(1-\frac{4}{nt^2}\sup_{f\in F}\mathrm{var}(f)\Big)\cdot Pr_X\Big\{\exists f\in F_n(\sigma_n),\ \Big|\mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^n f(X_i)\Big| \ge t\Big\} \\ \le 2\big(Pr_{X\times Y}\{A_t^c\} + e^{-\frac{nt^2}{128}}\big) + \delta, \end{aligned}$$

where $V := F_{2n}^{sym}(\sigma_n,\tau_n)/\sigma_n$ and $A_t := \big\{(\sigma_n,\tau_n)\ :\ \mathbb{E}_\varepsilon\sup_{v\in V}\big|\sum_{i=1}^n \varepsilon_i v_i\big| \le nt/8\big\}$.

This corollary illustrates how Assumption 2.1 is sufficient to guarantee a generalization bound with tails of order e^{-cnt^2} for a learning algorithm drawing its hypotheses from the random set $F_n(\sigma_n)$, as soon as the Rademacher averages of the projection of $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$ onto σ_n are "small" with high probability.

3. Examples

In the previous section we have proved that, under the Assumption 2.1, a small "size" for the set of random coordinate projections together with a sharp concentration inequality allow us to derive tail estimates for random subsets F_n. In particular, in order to obtain tail estimates of the order of e^{-cnt^2}, by Corollary 2.5, it is sufficient to find symmetric random subsets $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$ which satisfy Assumption 2.1 and for which the probability

$$Pr_{X\times Y}\Big\{\mathbb{E}_\varepsilon\Big(\sup_{f\in F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)}|\sum_{i=1}^n\varepsilon_i f(X_i)|\Big|X_1,...,X_n,Y_1,...,Y_n\Big)>\frac{nt}{8}\Big\},$$

is small. Now we are ready to show how apparently different approaches fall within the framework of Theorem 2.1.

Indeed, we will show that the tail estimate on (1.1) and the generalization bounds in the luckiness and the algorithmic luckiness frameworks can be derived directly from Corollary 2.5. We will illustrate this by specifying the corresponding maps F_n and F_{2n}^{sym}, and showing that for every fixed double-sample (σ_n,τ_n),

$$\mathbb{E}_\varepsilon\sup_{f\in F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)}\big|\sum_{i=1}^n\varepsilon_i f(x_i)\big|,$$

are sufficiently small (of the order $o(n)$).

As we present below, to recover the better estimates for (1.2) as in Mendelson (2002b) is more delicate because it requires a sharper concentration result than (2.5). We will show that these estimates as well follow from Theorem 2.1, by proving a different concentration inequality which will enable us to obtain the desired rates of the order of e^{-cnt}.

3.1 Glivenko-Cantelli Classes

In this section we will demonstrate how one can recover the optimal deviation estimates for uniform Glivenko-Cantelli classes directly from Corollary 2.5.

F is called a *uniform Glivenko-Cantelli class (GC class)* if for every $t>0$

$$\lim_{n\to\infty}\sup_\mu Pr\Big\{\sup_{f\in F}\big|\mathbb{E}_\mu f-\frac{1}{n}\sum_{i=1}^n f(X_i)\big|\geq t\Big\}=0.$$

If F is a uniform GC class, by selecting the constant functions

$$F_n(\sigma_n)=F,\ \ F_n^{\mathrm{sym}}(\sigma_n)=F,$$

Assumption 2.1 is trivially satisfied with $\delta=0$ and

$$F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)/\sigma_n=\Big\{\big(f(X_1),...,f(X_n)\big):\ f\in F\Big\}$$

for every double-sample (σ_n,τ_n). The fact that these coordinate projections are "small" follows from the characterization of uniform GC classes, an observation we shall return to later.

Theorem 3.1 *(Mendelson, 2002a) A class of uniformly bounded functions is a uniform GC class if and only if*

$$\lim_{n\to\infty}\sup_{\{x_1,...,x_n\}\subset\Omega^n}\frac{1}{n}\mathbb{E}_\varepsilon\sup_{f\in F}\big|\sum_{i=1}^n\varepsilon_i f(x_i)\big|=0.$$

Recall that

$$R_n(F) = \sup_{\{x_1,\ldots,x_n\}\subset\Omega^n} \mathbb{E}_\varepsilon \sup_{f\in F} \Big| \sum_{i=1}^n \varepsilon_i f(x_i) \Big|$$

and note that Theorem 3.1 ensures that the bound obtained from Theorem 2.1 is nonempty.

In particular, for every $t > 0$ let n_0 be such that for every $n \geq n_0$, $R_n(F) \leq nt/8$. Since $F \subset B\big(L_\infty(\Omega)\big)$ then $\sup_{f\in F} \mathrm{var}(f) \leq 1$, and thus $1 - 4\sup_{f\in F} \mathrm{var}(f)/nt^2 \geq 1/2$, provided that $n \geq 8/t^2$. Thus, by Corollary 2.5 and selecting $N = \max\{8/t^2, n_0\}$ it follows that for every integer $n > N$ and for any probability measure μ,

$$Pr\Big\{ \sup_{f\in F} \Big| \mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^n f(X_i) \Big| \geq t \Big\} \leq 8e^{-\frac{nt^2}{128}} .$$

In cases where one has *a priori* estimates on the size of the class (e.g., the shattering dimension or the uniform entropy), one can recover the optimal GC deviation results. For example, if $VC(F) = d$, then $R_n(F) \leq C\sqrt{dn}$ where C is an absolute constant, implying that one can take $n_0 = Cd/t^2$, and thus, for every $n \geq Ct^{-2}\max\{d, \log(1/\delta)\}$,

$$Pr\Big\{ \sup_{f\in F} \Big| \mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^n f(X_i) \Big| \geq t \Big\} \leq \delta.$$

Similar estimates can be recovered for classes with a polynomial shattering dimension by applying the bounds on $R_n(F)$ from Mendelson (2002a).

3.2 Learning Sample Complexity and Error Bounds

The learning sample complexity is governed by the probability that the empirical risk minimization algorithm fails, that is, it is the probability that an empirical minimizer of the loss functional (or more generally, an "almost empirical minimizer") will have a relatively large expectation. Formally, our aim is to estimate

$$Pr\Big\{ \exists f \in F,\ \frac{1}{n}\sum_{i=1}^n f(X_i) \leq t,\ \mathbb{E}_\mu f \geq 2t \Big\}, \tag{3.1}$$

where F is the excess squared-loss class.

The required tail estimates follow from two principles: The first is a mild structural assumption on F, namely that F is star-shaped around 0 (i.e., for every $f \in F$ and $0 \leq t \leq 1$, $tf \in F$); the second is that there is some $B > 0$ such that for every $f \in F$, $\mathbb{E}_\mu f^2 \leq B\mathbb{E}_\mu f$. Note that there are many examples of loss classes for which this second assumption could be verified. For example, for nonnegative bounded loss functions, the associated loss function classes satisfy this property. For convex classes of functions bounded by 1, the associated excess squared-loss class satisfies this property as well, a result that was first shown in Lee, Bartlett, and Williamson (1998) and improved and extended in Bartlett, Jordan, and McAuliffe (2003) and Mendelson (2002b).

Under these two assumptions, one can show (Mendelson, 2002b) that for every $t > 0$,

$$\begin{aligned} &Pr\Big\{ \exists f \in F,\ \frac{1}{n}\sum_{i=1}^n f(X_i) \leq t,\ \mathbb{E}_\mu f \geq 2t \Big\} \\ &\leq 2Pr\Big\{ \sup_{f\in F,\ \mathbb{E}_\mu f^2 \leq Bt} \Big| \mathbb{E}_\mu f - \frac{1}{n}\sum_{i=1}^n f(X_i) \Big| \geq t \Big\}. \end{aligned} \tag{3.2}$$

For the sake of simplicity we present our results for $B = 1$, which is the case if F consists of nonnegative functions. The general case follows an identical path.

It is possible to obtain sharper deviation estimates for (3.2)—of the order of e^{-cnt} instead of e^{-cnt^2} like in the uniform GC case—as long as the largest variance of a class member is of the same order of magnitude as the required deviation. This follows directly from Talagrand's inequality (see Mendelson, 2002b; Bartlett, Bousquet, and Mendelson, 2004), and was the basis for the estimates on (3.1) in Mendelson (2002b). Unfortunately, the method of proof used in Mendelson (2002b) cannot be used directly in a way which fits our general principle. We thus present a different proof which uses the fact that "most" coordinate projections of $\{f \in F : \mathbb{E}_\mu f^2 \leq t\}$ are contained in a "small" Euclidean ball. Although the proof is slightly more complicated than the original one, the significance of having "small" coordinate projections is better exhibited.

Theorem 3.2 below is the main result of this section.

Theorem 3.2 *There are absolute constants K, c and c_1 for which the following holds. Let $F \subset B\big(L_\infty(\Omega)\big)$ be star-shaped around* 0 *such that for every $f \in F$, $\mathbb{E}_\mu f^2 \leq \mathbb{E}_\mu f$. If $t \geq c_1/n$ satisfies that*

$$\mathbb{E} \sup_{f \in F,\ \mathbb{E}_\mu f^2 \leq t} \left| \sum_{i=1}^n \varepsilon_i f(X_i) \right| \leq \frac{nt}{16}, \tag{3.3}$$

then

$$Pr\Big\{ \exists f \in F,\ \frac{1}{n}\sum_{i=1}^n f(X_i) \leq t,\ \mathbb{E}_\mu f \geq 2t \Big\} \leq Ke^{-cnt}.$$

For example, if t_0 is the minimal t which satisfies (3.3), F is a loss class in a proper learning scenario, and f^* denotes the loss function associated with the empirical minimizer, then with probability larger than $1 - Ke^{-cnt_0}$,

$$\mathbb{E}_\mu f^* \leq 2t_0.$$

Note that it is possible to estimate t_0, either via *a priori* assumptions on the function class, such as assumptions on the shattering dimension or the uniform entropy as in Mendelson (2002b), or from the sampled data as in Bartlett, Bousquet, and Mendelson (2004). For example, one can show (Mendelson, 2002b; Bartlett, Bousquet, and Mendelson, 2004) that if F is the star-shaped hull of a Boolean class G and 0, and if $VC(G) = d$, then $t_0 = O\big(\frac{d}{n}\log\big(\frac{en}{d}\big)\big)$. Hence, there are absolute constants c and C such that with probability larger that $1 - c(d/n)^d$,

$$\mathbb{E}_\mu f^* \leq C\frac{d}{n}\log\Big(\frac{en}{d}\Big).$$

The rest of this section will be devoted to the proof of Theorem 3.2.

First, let us denote the subset of functions in F with variance bounded by t by

$$F^t := \{f \in F,\ \mathbb{E}_\mu f^2 \leq t\}.$$

From (3.2) it follows that by setting $F_n = F_{2n}^{\mathrm{sym}} := F^t$, and applying Theorem 2.1 (Assumption 2.1 holds trivially with $\delta = 0$), the probability we want to estimate is bounded by the probability $Pr_{X\times Y} Pr_\varepsilon \Big\{ \exists f \in F^t,\ \Big|\sum_{i=1}^n \varepsilon_i f(X_i)\Big| \geq \frac{nt}{4} \Big\}$.

We will show that the condition (3.3) is just a way of ensuring that, with high probability, the coordinate projections of F^t onto a random sample are small. This, together with a sharp concentration result will yield the desired result.

Let

$$Z_t(X_1,...,X_n) := \mathbb{E}_\varepsilon \sup_{f\in F^t} \Big| \sum_{i=1}^n \varepsilon_i f(X_i)\Big|.$$

The first step in the proof is based on an inequality due to Boucheron, Lugosi, and Massart which will allow us to bound the probability that Z_t deviates from its expectation.

Theorem 3.3 *(Boucheron, Lugosi, and Massart, 2003) Let $V_1,...,V_n$ be independent, identically distributed random variables which take values in a Banach space B, and assume that $\|V_i\| \leq 1$ almost surely. Set*

$$Z := \mathbb{E}\big(\|\sum_{i=1}^n \varepsilon_i V_i\| \;\big|\; V_1,...,V_n\big).$$

Then, for any $t>0$,

$$Pr\big(Z \geq \mathbb{E}Z + t\big) \leq e^{-\frac{t^2}{2\mathbb{E}Z+2t/3}}.$$

To apply this theorem to $Z = Z_t$, let $B = \ell_\infty(F)$, which is the set of all bounded functions $z : F \to \mathbb{R}$ such that $\|z\|_{\ell_\infty(F)} = \sup_{f\in F}|z(f)|$. Let $V_i := X_i$ and define $X_i(f) := f(X_i)$. Hence, $\|X_i\|_B \leq 1$ and $\|\sum_{i=1}^n \varepsilon_i V_i\|_B = \sup_{f\in F}|\sum_{i=1}^n \varepsilon_i f(X_i)|$.

If t is such that $\mathbb{E}Z_t \leq nt/16$ then by Theorem 3.3,

$$Pr_X\Big\{\mathbb{E}_\varepsilon \sup_{f\in F^t}\Big|\sum_{i=1}^n \varepsilon_i f(X_i)\Big| > \frac{nt}{8}\Big\} \leq e^{-cnt}, \tag{3.4}$$

where c is an absolute constant. Hence, with $F_{2n}^{\rm sym}(\sigma_n,\tau_n) = F^t$ and

$$A_t = \{(\sigma_n,\tau_n) :\ \mathbb{E}_\varepsilon \sup_{f\in F^t} |\sum_{i=1}^n \varepsilon_i f(X_i)| \leq nt/8\},$$

where $\sigma_n = (X_1,...,X_n)$ and $\tau_n = (Y_1,...,Y_n)$ as before, it follows that

$$Pr_{X\times Y}\{A_t^c\} \leq e^{-cnt}.$$

For a fixed $(\sigma_n,\tau_n) \in A$, we require a sharp concentration result for

$$W_t(\varepsilon_1,...,\varepsilon_n) := \sup_{f\in F^t} |\sum_{i=1}^n \varepsilon_i f(X_i)|,$$

since (2.5) leads only to a tail estimate of e^{-cnt^2}. To that end, we use Talagrand's convex-distance inequality (Talagrand, 1995) (let us mention that our estimates also follow from an earlier result due to Johnson and Schechtman 1991). We will formulate the concentration result only in the context we require (see Ledoux, 2001, pg. 76).

Theorem 3.4 *Let $T \subset \ell_2^n$ and set $\sigma := \sup_{t\in T}\|t\|_{\ell_2^n}$. Define the random variable $G := \sup_{t\in T}|\sum_{i=1}^n \varepsilon_i t_i|$, and denote its median by M_G. Then for every $r>0$,*

$$Pr\big\{|G - M_G| > r\big\} \leq 4e^{-r^2/4\sigma^2},$$

and $|\mathbb{E}G - M_G| \leq 4\pi\sigma$.

In our case, T is the image of F under the coordinate projection onto the random sample $\sigma_n = (X_1, ..., X_n)$ where $(\sigma_n, \tau_n) \in A$. In order to bound σ we shall estimate the probability that a coordinate projection has a small diameter in ℓ_2^n.

Theorem 3.5 *Let F be a class of functions which map Ω into $[-1,1]$. For every $x > 0$ and r which satisfies that*

$$\mathbb{E} \sup_{f \in F,\ \mathbb{E}_\mu f^2 \le r} \Big| \sum_{i=1}^n \varepsilon_i f(X_i) \Big| \le \frac{nr}{20} - \frac{11x}{20},$$

then with probability at least $1 - 2e^{-x}$,

$$\Big\{ f \in F :\ \mathbb{E}_\mu f^2 \le r \Big\} \subset \Big\{ f \in F :\ \sum_{i=1}^n f^2(X_i) \le 2rn \Big\}.$$

Proof. The proof follows directly from the contraction inequality (see, e.g., Theorem 2.8 in Bartlett, Bousquet, and Mendelson, 2004) for $\phi(x) = x^2$ combined with Corollary 2.7 in Bartlett, Bousquet, and Mendelson (2004). ∎

By our selection of t, it is easy to see that there is an absolute constant c, such that if $x = cnt$, then with probability larger that $1 - 2e^{-cnt}$, the radius of the projected set $F^t/\sigma_n \subset \ell_2^n$ is smaller than $\sqrt{2nt}$. In particular, we have

Corollary 3.6 *There are absolute constants c and c_1 for which the following holds. For every $t \ge c_1/n$ such that $\mathbb{E}Z_t \le nt/16$, there is a set A_t' of samples (σ_n, τ_n) which has probability larger than $1 - 3e^{-cnt}$, on which the set $V = \big\{ \big(f(X_1), ..., f(X_n)\big) :\ f \in F^t \big\}$ is such that $\mathbb{E}_\varepsilon \sup_{v \in V} |\sum_{i=1}^n \varepsilon_i v_i| \le nt/8$ and $\sup_{v \in V} \|v\|_{\ell_2^n} \le \sqrt{2nt}$.*

Combining this corollary with Theorem 3.4, for every such set V,

$$Pr_\varepsilon \Big\{ \sup_{v \in V} \Big| \sum_{i=1}^n \varepsilon_i v_i \Big| \ge \frac{nt}{4} \Big\} \le Pr_\varepsilon \Big\{ \sup_{v \in V} \Big| \sum_{i=1}^n \varepsilon_i v_i \Big| \ge \mathbb{E}_\varepsilon \sup_{v \in V} \Big| \sum_{i=1}^n \varepsilon_i v_i \Big| + \frac{nt}{8} \Big\} \le 4e^{-cnt}$$

for an absolute constant c. Hence, there are absolute constants c and K such that

$$Pr_{X \times Y} Pr_\varepsilon \Big\{ \exists f \in F^t,\ \Big| \sum_{i=1}^n \varepsilon_i f(X_i) \Big| \ge \frac{nt}{4} \Big\} \le Ke^{-cnt}. \tag{3.5}$$

Proof of Theorem 3.2. For every $t > 0$ let $F_n = F_{2n}^{\mathrm{sym}} := F^t$, and thus Assumption 2.1 holds with $\delta = 0$. Since for every $f \in F^t$, $\mathbb{E}_\mu f^2 \le t$, then

$$\big(1 - \frac{4}{nt^2} \sup_{f \in F^t} \mathrm{var}(f)\big) \ge \frac{1}{2}$$

provided that $t \ge 8/n$. Now the assertion follows from (3.2), Theorem 2.1, and (3.5). ∎

3.3 Luckiness

In the luckiness framework introduced in Shawe-Taylor et al. (1998), bounds on the generalization error of functions are formulated *a posteriori*, after having seen a sample σ_n. The bounds are given in terms of an upper bound on some empirical, computable quantity dependent on the sample.

In the following, let n be a fixed sample size, d is a given fixed integer, and set $\delta \in (0,1]$. Three concepts are used in the luckiness framework. The first is the luckiness function $L : F \times \cup_k \Omega^k \longrightarrow \mathbb{R}$ which is invariant under permutations of the sample, that is, it depends only on the set $\{x_1,...,x_k\}$.

Using the luckiness function one can construct sample dependent subsets of F, called *lucky sets* in the following manner; for every sample ζ and $f \in F$, the lucky set consists of all the functions luckier on this sample than the given function, that is,

$$H(f,\zeta) := \big\{g \in F : L(g,\zeta) \geq L(f,\zeta)\big\}.$$

Observe that the luckiness function imposes a structure of increasing subsets of F, because $H(g,\zeta) \subseteq H(f,\zeta)$ if and only if $L(g,\zeta) \geq L(f,\zeta)$, a fact which will allow us to define $F_{2n}^{\rm sym}(\sigma_n,\tau_n)$.

Lemma 3.7 *For every integer d and sample ζ there is a unique set $H_d(\zeta)$ with the following properties:*

1. *$M\big(\frac{1}{n}, H_d(\zeta), L_1(\mu_n)\big) \leq 2^d$, where μ_n is the empirical measure supported on ζ.*
2. *If $f \in F$ satisfies that $M\big(\frac{1}{n}, H(f,\zeta), L_1(\mu_n)\big) \leq 2^d$ then $f \in H_d(\zeta)$.*

Proof. Let $A := \{f \in F : M\big(\frac{1}{n}, H(f,\zeta), L_1(\mu_n)\big) \leq 2^d\}$ and set $H_d(\zeta) := \bigcup_{f\in A} H(f,\zeta)$. To see that $H_d(\zeta)$ has the required properties, note that if $K \subset H_d(\zeta)$ is a finite $1/n$-separated set with respect to $L_1(\mu_n)$, then there is some $f \in A$ such that $K \subset H(f,\zeta)$, implying that $|K| \leq 2^d$. The second property and the uniqueness are easily verified. ∎

For every double-sample $\zeta = (\sigma_n,\tau_n)$ we set

$$F_{2n}^{\rm sym}(\sigma_n,\tau_n) := H_d(\sigma_n,\tau_n), \tag{3.6}$$

and observe that this random class is permutation invariant, implying that $F_{2n}^{\rm sym}$ is symmetric.

The second ingredient in the luckiness framework, the ω-function, $\omega : \mathbb{R} \times \mathbb{N} \times (0,1] \to \mathbb{N}$, is used to define $F_n(\sigma_n)$. Given a luckiness function L and an ω-function, then for a fixed integer d and $\delta \in (0,1]$, define

$$F_n(\sigma_n) := \big\{f \in F : \ \omega\big(L(f,\sigma_n), n, \delta\big) \leq 2^d\big\}. \tag{3.7}$$

The third ingredient is the ω-smallness condition, which is a joint property of the luckiness and ω functions. It states that for every $n \in \mathbb{N}$, every $\delta \in (0,1]$ and every probability measure μ

$$Pr_{X\times Y}\Big\{\exists f \in F : M\big(\tfrac{1}{n}, H(f,(\sigma_n,\tau_n)), L_1(\mu_{2n})\big) > \omega\big(L(f,\sigma_n), n, \delta\big)\Big\} < \delta. \tag{3.8}$$

Examples for luckiness functions are the empirical VC-dimension of a binary function class with respect to a sample—in this case all lucky sets are equal to the whole set F—and the margin of linear classifiers. Their corresponding ω functions can be found in Shawe-Taylor et al. (1998). Although the luckiness framework gives a unified proof for existing generalization bounds, finding a pair of

luckiness and ω-functions seems to be difficult, because of the quite technical and counterintuitive ω-smallness condition.

The following result shows that the ω-smallness of L ensures that Assumption 2.1 holds, and that, with high probability, $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)/\sigma_n$ is sufficiently small. Therefore, it is just a way of requiring that $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$ has, with high probability, small random coordinate projections.

Lemma 3.8 *For fixed integers n and d, and $\delta\in(0,1]$, let F_n and F_{2n}^{sym} be defined as in* (3.7) *and* (3.6)*. If a luckiness function L and an ω-function satisfy the ω-smallness condition* (3.8)*, then for every $t>0$,*

$$Pr_{X\times Y}\Big\{\exists f\in F_n(\sigma_n),\ \Big|\frac{1}{n}\sum_{i=1}^{n}\big(f(X_i)-f(Y_i)\big)\Big|\geq t\Big\}$$
$$\leq Pr_{X\times Y}\Big\{\exists f\in F_{2n}^{sym}(\sigma_n,\tau_n),\ \Big|\frac{1}{n}\sum_{i=1}^{n}\big(f(X_i)-f(Y_i)\big)\Big|\geq t\Big\}+\delta.$$

Proof. For a fixed double sample $\zeta=(\sigma_n,\tau_n)$ let μ_{2n} be the empirical measure supported on ζ. Put

$$A_\zeta:=\Big\{f\in F\ :M\big(\tfrac{1}{2n},H(f,(\sigma_n,\tau_n)),L_1(\mu_{2n})\big)\leq\omega\big(L(f,\sigma_n),n,\delta\big)\Big\}$$

and

$$B_\zeta:=\Big\{f\in F:M\big(\tfrac{1}{2n},H(f,(\sigma_n,\tau_n)),L_1(\mu_{2n})\big)\leq 2^d\Big\}.$$

Note that $F_n(\sigma_n)\cap A_\zeta\subseteq B_\zeta\subseteq F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$. By the ω-smallness condition,

$$Pr_{X\times Y}\{\exists f\in(A_\zeta)^c\}\leq\delta,$$

and by the union bound for disjoint sets,

$$Pr_{X\times Y}\Big\{\exists f\in F_n(\sigma_n),\ \Big|\frac{1}{n}\sum_{i=1}^{n}\big(f(X_i)-f(Y_i)\big)\Big|\geq t\Big\}$$
$$=Pr_{X\times Y}\Big\{\exists f\in F_n(\sigma_n)\cap A_\zeta,\ \Big|\frac{1}{n}\sum_{i=1}^{n}\big(f(X_i)-f(Y_i)\big)\Big|\geq t\Big\}$$
$$+Pr_{X\times Y}\Big\{\exists f\in F_n(\sigma_n)\cap(A_\zeta)^c,\ \Big|\frac{1}{n}\sum_{i=1}^{n}\big(f(X_i)-f(Y_i)\big)\Big|\geq t\Big\},\quad(3.9)$$

and our claim follows. ∎

Now, we are ready to formulate the generalization bound for the luckiness framework.

Theorem 3.9 *Let L and ω be functions satisfying the ω-smallness condition* (3.8)*. Then, for every probability measure μ, every $d\in\mathbb{N}$ and every $\delta\in(0,1]$, there is a set of probability larger than $1-12\delta$ such that if $\omega\big(L(f,\sigma_n),n,\delta\big)\leq 2^d$, then*

$$\Big|\mathbb{E}_\mu f-\frac{1}{n}\sum_{i=1}^{n}f(X_i)\Big|\leq C\sqrt{\frac{d}{n}\log\frac{1}{\delta}},$$

where C is an absolute constant.

Proof. Let F_n and $F_{2n}^{\rm sym}$ be defined as above, and observe that

$$M\big(\tfrac{1}{n}, F_{2n}^{\rm sym}(\sigma_n,\tau_n), L_1(\mu_{2n})\big) \le 2^d \tag{3.10}$$

for every (σ_n,τ_n). By Corollary 2.5 we have to estimate

$$Pr_{X\times Y}\Big\{\mathbb{E}_\varepsilon \sup_{f\in F_{2n}^{\rm sym}(\sigma_n,\tau_n)}\Big|\sum_{i=1}^n \varepsilon_i f(X_i)\Big| > \frac{nt}{8}\Big\},$$

where $\sigma_n=(X_1,...,X_n)$ and $\tau_n=(Y_1,...,Y_n)$. Let

$$V := \Big\{\big(f(X_1),...,f(X_n)\big) : f\in F_{2n}^{\rm sym}(\sigma_n,\tau_n)\Big\}\subset \ell_2^n,$$

put μ_{2n} to be the empirical measure supported on $\zeta=(\sigma_n,\tau_n)$ and set ν_n to be the empirical measure supported on σ_n. Note that for every f,g, $\mathbb{E}_{\mu_{2n}}|f-g|\ge \mathbb{E}_{\nu_n}|f-g|/2$. Thus, every $1/n$-cover of $F_{2n}^{\rm sym}(\sigma_n,\tau_n)$ in $L_1(\mu_{2n})$ is a $2/n$-cover of the same set in $L_1(\nu_n)$. In particular, if A is a maximal $1/n$-packing of $F_{2n}^{\rm sym}(\sigma_n,\tau_n)$ in $L_1(\mu_{2n})$, it is a $2/n$ cover of that set in $L_1(\nu_n)$. It is easy to verify that $B\big(L_1(\nu_n)\big)=nB_1^n$, and in particular, $V\subset A+\frac{2}{n}\cdot nB_1^n = A+2B_1^n$, where $A+B=\{a+b\,:a\in A,\,b\in B\}$. By the triangle inequality,

$$\begin{aligned}\mathbb{E}_\varepsilon \sup_{f\in F_{2n}^{\rm sym}(\sigma_n,\tau_n)}\Big|\sum_{i=1}^n \varepsilon_i f(x_i)\Big| &= \mathbb{E}_\varepsilon \sup_{v\in V}\Big|\sum_{i=1}^n \varepsilon_i v_i\Big| = \mathbb{E}_\varepsilon \sup_{a\in A,\,b\in B_1^n}\Big|\sum_{i=1}^n \varepsilon_i(a_i+2b_i)\Big| \\ &\le \mathbb{E}_\varepsilon \sup_{a\in A}\Big|\sum_{i=1}^n \varepsilon_i a_i\Big| + 2\mathbb{E}_\varepsilon \sup_{b\in B_1^n}\Big|\sum_{i=1}^n \varepsilon_i b_i\Big|.\end{aligned}$$

The first term can be bounded by a corollary of Slepian's inequality (Pisier, 1989), which states that there is an absolute constant C such that for every $A\subset\ell_2^n$,

$$\mathbb{E}_g \sup_{a\in A}\Big|\sum_{i=1}^n g_i a_i\Big| \le C\sqrt{\log|A|}\sup_{u,v\in A}\|u-v\|_2,$$

where $(g_i)_{i=1}^n$ are independent standard gaussian random variables.

Since our class consists of functions bounded by 1, then $V\subset B_\infty^n\subset\sqrt{n}B_2^n$ and since the Rademacher averages are upper bounded (up to an absolute constant) by the gaussian ones (Milman and Schechtman, 2001), then

$$\mathbb{E}_\varepsilon \sup_{a\in A}\Big|\sum_{i=1}^n \varepsilon_i a_i\Big| \le C\mathbb{E}_g \sup_{a\in A}\Big|\sum_{i=1}^n g_i a_i\Big| \le C\sqrt{\log|A|}\sqrt{n} \le C\sqrt{nd},$$

where the final inequality holds because $|A|\le 2^d$ by (3.10).

In order to estimate the second term, one can apply the triangle inequality to show that

$$\mathbb{E}_\varepsilon \sup_{b\in B_1^n}\Big|\sum_{i=1}^n \varepsilon_i b_i\Big| \le 1.$$

In conclusion

$$\mathbb{E}_\varepsilon \sup_{f\in F_{2n}^{\rm sym}(\sigma_n,\tau_n)}\Big|\sum_{i=1}^n \varepsilon_i f(x_i)\Big| \le C\sqrt{nd}.$$

To complete the proof, apply Corollary 2.5 for $t=C\sqrt{\frac{d}{n}\log(1/\delta)}$. ∎

3.4 Algorithmic Luckiness

In the algorithmic luckiness framework (Herbrich and Williamson, 2002), the generalization error bound is also formulated *a posteriori*, after having seen a sample. It differs from the luckiness framework because it gives bounds on the generalization error of the function learned by the learning algorithm from the sample at hand. Again, the bound is given in terms of a computable quantity dependent on the sample and on the algorithm.

In a similar fashion to the luckiness framework, an algorithmic luckiness function and an ω-function are introduced in order to define the functions F_n and F_{2n}^{sym}. The functions L and ω satisfy a joint smallness condition which ensures that Assumption 2.1 holds, and that the size of the projection $F_{2n}^{\text{sym}}(\sigma_n,\tau_n)/\sigma_n$ is sufficiently small.

As we did before, fix a sample size n, an integer d and some $\delta \in (0,1]$. Denote by $\mathcal{A}$ a fixed learning algorithm, by $\mathcal{A}(\zeta)$ the loss function associated with the hypothesis produced by the algorithm from the sample ζ, and set $\mathcal{A}(F) = \{f = \mathcal{A}(\zeta) : \zeta \in \Omega^n\}$.

The algorithmic luckiness function is a function $L : \mathcal{A}(F) \longrightarrow \mathbb{R}$. For a sample ζ of size $2n$, the *lucky set* $G(\zeta)$ is defined as the subset of losses of functions learned by the algorithm on the first half of the sample, when permuting the whole sample, as long as the function the algorithm produced on the first half of the permuted sample is "luckier" than on the original one. Formally, let S_{2n} be the set of permutations on $\{1,...,2n\}$, and for every $\zeta = (\zeta_1,...,\zeta_{2n})$, set $\zeta|_{i=1}^n = (\zeta_1,...,\zeta_n)$. Define the lucky set as

$$G(\zeta) := \Big\{ \mathcal{A}\big(\pi(\zeta)|_{i=1}^n\big) : L\big(\mathcal{A}(\pi(\zeta)|_{i=1}^n)\big) \geq L\big(\mathcal{A}(\zeta|_{i=1}^n)\big),\ \pi \in S_{2n} \Big\}.$$

If $G_{\mathcal{A}}(\zeta)$ is the subset of losses corresponding to functions learned by $\mathcal{A}$ on the first half of all the permutations of the double-sample ζ, then $G(\zeta) \subset G_{\mathcal{A}}(\zeta)$, and clearly, $|G_{\mathcal{A}}(\zeta)| \leq (2n)! < \infty$. Therefore, we can order the functions in decreasing order according to their luckiness. Define the ordered set

$$G_{\mathcal{A}}(\zeta) := \Big[\underbrace{f_1, f_2, f_3, \ldots, f_{k-1}, f_k}_{G(\zeta)}, f_{k+1}, \ldots, f_m \Big],$$

and for the sake of simplicity, assume that for every $i < j$, $L(f_i) > L(f_j)$. Only a small modification is required in the general case, where some functions might have the same luckiness.

Set $f_k = \mathcal{A}(\zeta|_{i=1}^n)$ and let $G_{\mathcal{A}}^{\ell}(\zeta)$ be the subset consisting of the first ℓ functions in $G_{\mathcal{A}}(\zeta)$, that is, $G_{\mathcal{A}}^{\ell}(\zeta) = \{f_1, f_2, f_3, \ldots, f_\ell\}$.

For the given integer d and the double-sample (σ_n,τ_n) put k^* to be the largest integer such that

$$M\big(\tfrac{1}{n}, G_{\mathcal{A}}^{k^*}((\sigma_n,\tau_n)), L_1(\mu_{2n})\big) \leq 2^d \text{ and } M\big(\tfrac{1}{n}, G_{\mathcal{A}}^{k^*+1}((\sigma_n,\tau_n)), L_1(\mu_{2n})\big) > 2^d.$$

Then, by setting

$$F_{2n}^{\text{sym}}(\sigma_n,\tau_n) := G_{\mathcal{A}}^{k^*}((\sigma_n,\tau_n)) \tag{3.11}$$

it follows that $F_{2n}^{\text{sym}}$is symmetric, since the learning algorithm is permutation invariant.

The ω-function, $\omega : \mathbb{R} \times \mathbb{N} \times (0,1] \longrightarrow \mathbb{N}$ is used to define $F_n(\sigma_n)$. Indeed, define

$$F_n(\sigma_n) := \begin{cases} \{\mathcal{A}(\sigma_n)\} & \text{if } \omega\big(L(\mathcal{A}(\sigma_n)), n, \delta\big) \leq 2^d \\ \emptyset & \text{otherwise,} \end{cases} \tag{3.12}$$

and note that $|F_n(\sigma_n)| \leq 1$.

Finally, the ω-smallness condition states that for every integer n, every $\delta \in (0,1]$, and every probability measure μ,

$$P_{X\times Y}\Big\{M\big(\tfrac{1}{n}, G((\sigma_n,\tau_n)), L_1(\mu_{2n})\big) \geq \omega\big(L(\mathcal{A}(\sigma_n)), n, \delta\big)\Big\} < \delta, \tag{3.13}$$

and as we show, it assures that Assumption 2.1 holds.

Lemma 3.10 *Let $\mathcal{A}$ be a learning algorithm, fix an integer d and some $\delta \in (0,1]$, and let F_n and F_{2n}^{sym} be as in* (3.12) *and* (3.11)*. If a luckiness function and ω-function satisfy the ω-smallness condition* (3.13)*, then for every $t > 0$*

$$Pr_{X\times Y}\Big\{\exists f \in F_n(\sigma_n) : \Big|\frac{1}{n}\sum_{i=1}^{n}(f(X_i) - f(Y_i))\Big| \geq t\Big\}$$
$$\leq Pr_{X\times Y}\Big\{\exists f \in F_{2n}^{sym}(\sigma_n,\tau_n) : \Big|\frac{1}{n}\sum_{i=1}^{n}(f(X_i) - f(Y_i))\Big| \geq t\Big\} + \delta.$$

Proof. For every double sample $\zeta = (\sigma_n, \tau_n)$, let μ_{2n} be the empirical measure supported on (σ_n,τ_n) and define two random sets in the following manner. Let $A_\zeta := \{\mathcal{A}(\sigma_n)\}$ if $M\big(\frac{1}{n}, G((\sigma_n,\tau_n)), L_1(\mu_{2n})\big) < \omega\big(L(\mathcal{A}(\sigma_n)), n, \delta\big)$ and the empty set otherwise, and put $B_\zeta := \{\mathcal{A}(\sigma_n)\}$ if $M\big(\frac{1}{n}, G((\sigma_n,\tau_n)), L_1(\mu_{2n})\big) \leq 2^d$ and the empty set otherwise. Note that for every ζ, $F_n(\sigma_n) \cap A_\zeta \subset B_\zeta \subset F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$. Moreover, if $F_n(\sigma_n) \cap (A_\zeta)^c \neq \emptyset$, then $F_n(\sigma_n) = \{\mathcal{A}(\sigma_n)\}$ and $A_\zeta = \emptyset$. Thus, by the ω-smallness condition,

$$Pr_{X\times Y}\Big\{F_n(\sigma_n) \cap (A_\zeta)^c \neq \emptyset\Big\} \leq Pr_{X\times Y}\Big\{A_\zeta = \emptyset\Big\} < \delta.$$

Finally, for every $t > 0$,

$$P_{X\times Y}\Big\{\exists f \in F_n(\sigma_n),\ \Big|\frac{1}{n}\sum_{i=1}^{n}(f(X_i) - f(Y_i))\Big| \geq t\Big\}$$
$$= Pr_{X\times Y}\Big\{\exists f \in F_n(\sigma_n)\cap A_\zeta,\ \Big|\frac{1}{n}\sum_{i=1}^{n}(f(X_i) - f(Y_i))\Big| \geq t\Big\}$$
$$+ Pr_{X\times Y}\Big\{\exists f \in F_n(\sigma_n)\cap (A_\zeta)^c,\ \Big|\frac{1}{n}\sum_{i=1}^{n}(f(X_i) - f(Y_i))\Big| \geq t\Big\}$$
$$\leq Pr_{X\times Y}\Big\{\exists f \in F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n),\ \Big|\frac{1}{n}\sum_{i=1}^{n}(f(X_i) - f(Y_i))\Big| \geq t\Big\} + \delta,$$

as claimed. ∎

The definition of $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$ assures that the covering numbers of $F_{2n}^{\mathrm{sym}}(\sigma_n,\tau_n)$ are small, and by Corollary 2.5 we obtain a result analogous to Theorem 3.9, which recovers the main result of Herbrich and Williamson (2002).

Theorem 3.11 *Let $\mathcal{A}$ be a learning algorithm which takes values in $B\big(L_\infty(\Omega)\big)$, and let L and ω be functions satisfying the ω-smallness condition* (3.13)*. Then, for every probability measure*

μ, every $d \in \mathbb{N}$ and every $\delta \in (0,1]$, there is a set of probability at least $1-12\delta$ such that if $\omega\big(L(\mathcal{A}(\sigma_n)), n, \delta\big) \leq 2^d$, then

$$\left|\mathbb{E}_\mu\big(\mathcal{A}(\sigma_n)\big) - \mathbb{E}_{\mu_n}\big(\mathcal{A}(\sigma_n)\big)\right| \leq C\sqrt{\frac{d}{n}\log\frac{1}{\delta}},$$

where C is an absolute constant.

Acknowledgements

We would like to thank Ran Bachrach, Olivier Bousquet, Gideon Schechtman, and Bob Williamson for their valuable suggestions and comments.

References

N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, vol. 44(4), 615-631, 1997.

P. L. Bartlett, O. Bousquet, and S. Mendelson. Local Rademacher complexities. *The Annals of Statistics*, 2004, to appear.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. Technical Report 638, Department of Statistics, UC Berkeley, 2003.

S. Boucheron, G. Lugosi, and P. Massart. Concentration inequalities using the entropy method. *The Annals of Probability*, vol 31(3), 1583–1614, 2003.

O. Bousquet. A Bennett concentration inequality and its application to suprema of empirical processes. *Comptes-Rendus de l'Académie Scientifique de Paris*, Ser. I, 334, 495–500, 2002.

A. H. Cannon, J. M. Ettinger, D. R. Hush, and J. C. Scovel. Machine learning with data dependent hypothesis classes. *Journal of Machine Learning Research*, 2, 335–358, 2002.

Y. Gat. A bound concerning the generalization ability of a certain class of learning algorithms. Technical Report 548, UC Berkeley, March 1999.

R. Herbrich and R. C. Williamson. Algorithmic luckiness. *Journal of Machine Learning Research*, 3, 175–212, 2002.

W. B. Johnson and G. Schechtman. A remark on Talagrand's deviation inequality for Rademacher functions. In *Functional Analysis (Austin, TX, 1987/1989)*, Lecture Notes in Mathematics 1470, 72–77, Springer, 1991.

M. Ledoux. *The concentration of measure phenomenon*. Mathematical Surveys and Monographs, Vol 89, AMS, 2001.

W. S. Lee, P. L. Bartlett, and R. C. Williamson. The importance of convexity in learning with squared loss. *IEEE Transactions on Information Theory*, 44(5), 1974–1980, 1998.

C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, London Mathematical Society Lecture Note Series 141, 148–188, Cambridge University Press, 1989.

S. Mendelson. Rademacher averages and phase transitions in Glivenko-Cantelli classes. *IEEE Transactions on Information Theory*, 48(1), 251–263, 2002a.

S. Mendelson. Improving the sample complexity using global data. *IEEE Transactions on Information Theory*, 48(7), 1977–1991, 2002b.

S. Mendelson. A few notes on Statistical Learning Theory. In *Proceedings of the Machine Learning Summer School, Canberra 2002, S. Mendelson and A. J. Smola (Eds.)*, Lecture Notes in Computer Sciences LNCS 2600, 1–40, Springer, 2003.

S. Mendelson and P. Philips. Random subclass bounds. In *Proceedings of the 16th Annual Conference on Computational Learning Theory*, 329–343, Springer, 2003.

S. Mendelson and R. Vershynin. Entropy and the combinatorial dimension. *Inventiones Mathematicae*, 152, 37–55, 2003.

V. D. Milman and G. Schechtman. *Asymptotic Theory of Finite Dimensional Normed Spaces*. Lecture Notes in Mathematics 1200, Springer, 2001.

G. Pisier. *The Volume of Convex Bodies and Banach Space Geometry*. Cambridge University Press, 1989.

J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5), 1926–1940, 1998.

M. Talagrand. Majorizing measures: The generic chaining. *The Annals of Probability*, 24, 1049–1103, 1996.

M. Talagrand. Concentration of measure and isoperimetric inequalities in product spaces. *Publications Mathématiques de l'I.H.E.S.* 81, 73–205, 1995.

V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2), 264–280, 1971.

Journal of Machine Learning Research 5 (2004) 239–253 Submitted 9/02; Revised 7/03; Published 3/04

Weather Data Mining Using Independent Component Analysis

Jayanta Basak BJAYANTA@IN.IBM.COM
IBM India Research Lab
Block I, Indian Institute of Technology
Hauz Khas, New Delhi - 110016, India

Anant Sudarshan, Deepak Trivedi
Department of Mechanical Engineering
Indian Institute of Technology
Hauz Khas, New Delhi - 110016, India

M. S. Santhanam SANTH@MPIPKS-DRESDEN.MPG.DE
Max Planck Institute for Physics of Complex Systems
Nothnitzer Strasse 38
D-01187 Dresden, Germany

Editors: Te-Won Lee and Erkki Oja

Abstract

In this article, we apply the independent component analysis technique for mining spatio-temporal data. The technique has been applied to mine for patterns in weather data using the North Atlantic Oscillation (NAO) as a specific example. We find that the strongest independent components match the observed synoptic weather patterns corresponding to the NAO. We also validate our results by matching the independent component activities with the NAO index.

Keywords: North Atlantic Oscillation, ICA, spatio-temporal pattern mining

1. Introduction

Classical laws of fluid motion govern the states of the atmosphere. Atmospheric states exhibit a great deal of correlations at various spatial and temporal scales. Numerical models for predicting weather attempt to capture the dynamics of various atmospheric variables (like temperature, pressure etc.) and how physical processes (like convection, radiation etc.) influence the future state of these variables. Thus, the weather system can be thought of as a complex system whose various components interact in various spatial and temporal scales. It is also known that the atmospheric system is chaotic and there are limits to the predictability of its future state (Lorenz, 1963, 1965). Nevertheless, even though daily weather may, under certain conditions, exhibit symptoms of chaos, long-term climatic trends are still meaningful and their study can provide significant information about climate changes.

Statistical approaches to weather and climate prediction have a long and distinguished history that predates modeling based on physics and dynamics (Wilks, 1995; Santhanam and Patra, 2001). This trend continues today with newer approaches based on machine learning algorithms (Hsieh and

Tang, 1998; Monahan, 2000). The central problem in weather and climate modeling is to predict the future states of the atmospheric system. Since the weather data are generally voluminous, they can be mined for occurrence of particular patterns that distinguish specific weather phenomena. For instance, the wind fields of tropical cyclones and certain low-pressure systems are characterized by the anti-clockwise circulation pattern in the northern hemisphere. The strength of these patterns provides information about the particular weather phenomenon.

It is therefore possible to view the weather variables as sources of spatio-temporal signals. The information from these spatio-temporal signals can be extracted using data mining techniques. The variation in the weather variables can be viewed as a mixture of several independently occurring spatio-temporal signals with different strengths. Independent component analysis (ICA) has been widely studied in the domain of signal and image processing where each signal is viewed as a mixture of several independently occurring source signals. Under the assumption of non-Gaussian mixtures, it is possible to extract the independently occurring signals from the mixtures under certain well known constraints. Therefore, if the assumption of independent stable activity in the weather variables holds true then it is also possible to extract them using the same technique of ICA.

One basic assumption of our approach is that we view the weather phenomenon as a mixture of a certain number of signals with independent stable activity. By 'stable activity', we mean spatio-temporal stability, i.e., the activities that do not change over time and are spatially independent. The observed weather phenomenon is only a mixture of these stable activities. The weather changes due to the changes in the mixing patterns of these stable activities over time. For linear mixtures, the change in the mixing coefficients gives rise to the changing nature of the global weather.

The purpose of the present article is to investigate if there exist any such set of spatio-temporal stable patterns such that the variation of the mixture gives rise to the observed weather or climate phenomena. Our conjecture is that there exist independent stable spatio-temporal activities, the mixture of which give rise to the weather variables; and these stable activities can be extracted by independent component analysis (ICA) of the data arising from the weather and climate patterns, viewing them as spatio-temporal signals (Stone, Porrill, Buchel, and Friston, 1999; Hyvarinen, 2001). If our conjecture about the existence of stable spatio-temporal activity in the weather is true, then the mixing coefficients will vary in accordance with the changes in the weather variables. For instance, in this work, we take as our canonical weather activity, the North Atlantic Oscillation (NAO) (Lamb and Peppler, 1987), characterised by a stable dipole pattern in the north Atlantic ocean as reflected in the sea level pressure data displayed in Figure 2. The NAO has been extensively studied and documented in the atmospheric sciences literature (Lamb and Peppler, 1987; Wallace and Gutzler, 1981; Hurrell, 1995; Bell and Visbeck). The strength of the NAO pattern is indicated by the measured (scaled) quantity called the NAO index. In this paper, we validate our conjecture about the existence of stable spatio-temporal patterns in the weather by comparing the varying mixing coefficients with the changes in the strength of NAO, i.e., the NAO index. Our results here show that the ICA techniques can play a vital role in mining spatio-temporal patterns. Here it may be mentioned that the independent component analysis has also been applied in analyzing the fMRI images where activations vary spatially as well as temporally (Stone, Porrill, Buchel, and Friston, 1999).

The rest of the paper is organized as follows. In Section 2, we describe the particular weather variables that are considered for analysis. In Section 3, we provide a brief description of ICA and then present the techniques for mining the weather patterns and validating the independent stable

components obtained. In Section 4, we summarize the results from numerical experiments on the weather data (NOAA-CIRES). Section 5 concludes the paper.

2. Weather Phenomena

In this section, we provide a brief description of the weather variables in the north Atlantic region of earth that we considered.

2.1 Atmospheric Correlation

Atmospheric correlations play a significant role in determining the climate trends. These correlations are crucial in understanding the short- and long-term trends in climate. Examples of such trends are the well known El Nino-Southern oscillation and its global implications, predictability of the Asian summer monsoon, etc. Most significant correlations that have a bearing on the climatic conditions are documented as 'teleconnection' patterns, *i.e.*, the simultaneous correlations in the fluctuations of the large scale atmospheric parameters at widely separated points on the earth (Wallace and Gutzler, 1981). They could be thought of as the dominant modes of atmospheric variability.

For instance, the North Atlantic Oscillation (NAO) (Lamb and Peppler, 1987) refers to the large-scale exchange of the atmospheric mass between the Greenland and Iceland regions and the regions of the North Atlantic ocean between 35^oN and 40^oN, and is characterized by a north-south dipole pattern as shown in Figure 2. The positive phase of the NAO pattern features anomalously high pressure over central Atlantic, eastern United States and western Europe and below-normal pressure over high latitude North Atlantic regions. It has been observed that the positive phases of NAO are linked with the above-average temperatures in the eastern United States and northern Europe. It is also linked with the anomalous rainfall patterns and shifts in storm tracks in almost the entire Western Europe including Scandinavia (Hurrell, 1995). Its negative phase has an opposite effect to that during a positive phase. The transition between these phases is not periodic and is still a matter of current research. Thus, it is important to understand these simultaneous correlations or *teleconnection patterns* since they lead to better seasonal forecasts and have considerable economic implications.

The strength of the NAO pattern is given by the measured quantity called the NAO index, which is the normalized difference in sea level pressures (SLP) between two fixed positions in the north Atlantic region. For instance, Hurrell's NAO index is the difference in SLP values between Ponta Delgada, Azores and Stykkisholmur/Reykjavik, Iceland (Hurrell, 1995). The NAO index (Figure 3 as available in Bell and Visbeck) provides a time-series of the strength of NAO over the years.

2.2 Spatio-Temporal Data

Here, we use the time-series of monthly mean sea level pressure (SLP) data obtained from the NCEP reanalysis archives (NOAA-CIRES). We use the data for the Atlantic domain $(0-90^o\text{N}, 120^o\text{W}-30^o\text{E})$ as shown in Figure 1, from 1948 to 1957. The SLP data is on a uniform spatial grid of 2.5^o along both the latitude and longitude and thus the spatial grid size is 61 by 29 grid points. Figure 2 illustrates one such data frame of average sea level pressure in the north Atlantic region for the month of January in 1948. The figure shows the contours of SLP, after subtracting the long-term average, plotted with the geographical map of the Atlantic region in the background. The contour

lines connect the points having the same SLP values. Note that NAO is characterized by the dipole pattern shown in Figure 2. Continuous contour lines represent isobars of above average pressure and dashed contour lines represent isobars of below average pressure values.

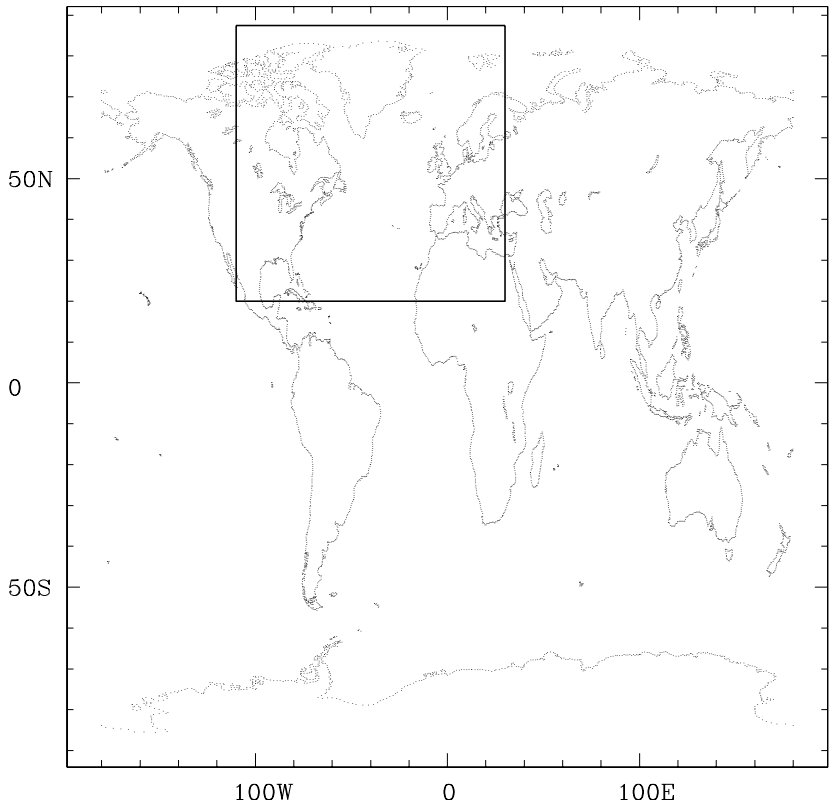


Figure 1: Map of the world with the region of interest marked within the box

3. Weather Data Mining

In this section we describe how independent component analysis has been used to mine for the spatio-temporal stable activities in the sea level pressure in the north Atlantic region.

3.1 Principal Component Analysis

Given a set of data vectors $[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(N)}]$, the principal component represents the direction along which the data vectors have the maximum variation (Dejviver and Kittler, 1982). Mathematically, it is the largest eigenvector of the data covariance matrix

$$C = \sum_i (\mathbf{x}^{(i)} - \mu)(\mathbf{x}^{(i)} - \mu)^T,$$

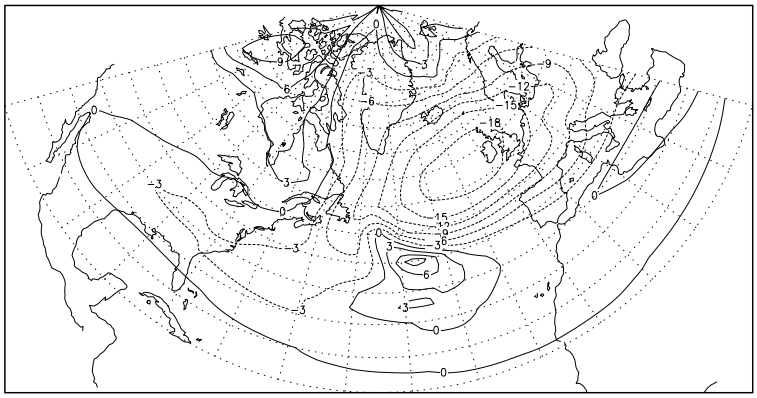


Figure 2: Data plot of the average sea level pressure for January, 1948

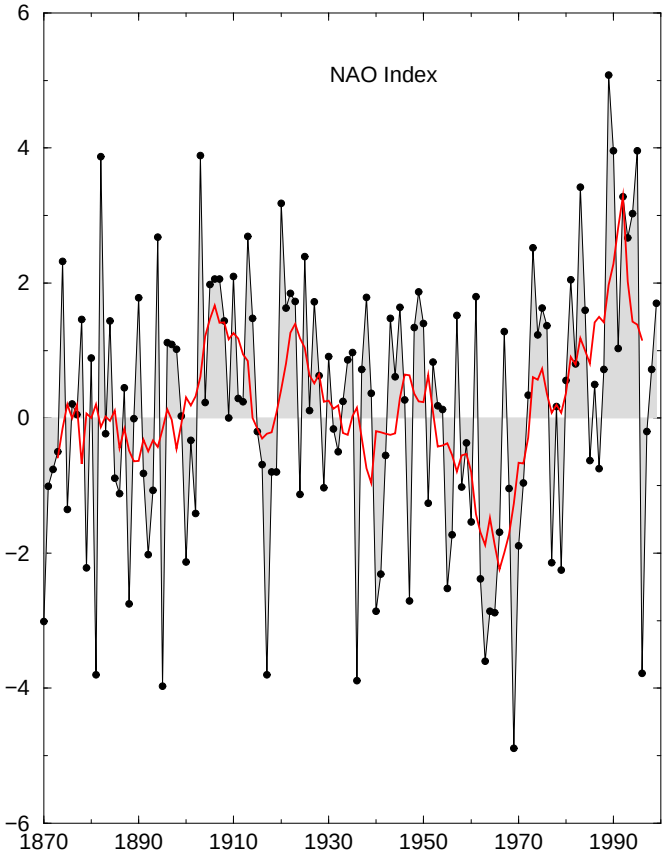


Figure 3: Time series of Hurrell's NAO index

where μ is the sample mean over the data vectors. Variants of principal component analysis such as on-line computation of the principal components (Oja, 1982; Oja, Karhunen, Wang, and Vigario, 1995), nonlinear principal component analysis (Oja, 1995), have also been proposed in the literature of neural networks.

Principal component analysis – also referred to as the Karhunen-Loeve transform (Dejviver and Kittler, 1982)– has been widely used in the literature of pattern recognition and feature extraction and dimensionality reduction. The principal component and other orthogonal major components (in the sense of having a large eigenvalue) are extracted and treated as the derived features. The principal components also reveal the major characteristics of the data set as in the case of human face recognition (Turk and Pentland, 1991). However, if the data comes from more than one class then the principal component analysis technique (being an unsupervised technique) does not preserve the class conditional information of the data set. Characteristrics of chaotic systems were also analyzed by nonlinear principal component analysis technique (Monahan, 2000).

3.2 Independent Component Analysis

Given a set of n-dimensional data vectors $[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(N)}]$, the independent components are the directions (vectors) along which the statistics of projections of the data vectors are independent of each other. Formally, if $\mathbf{A}$ is a transformation from the given reference frame to the independent component reference frame then

$$\mathbf{x} = \mathbf{A}\mathbf{s}$$

such that

$$p(\mathbf{s}) = \Pi p_a(s_i),$$

where $p_a(\cdot)$ is the marginal distribution and $p(\mathbf{s})$ is the joint distribution over the n-dimensional vector $\mathbf{s}$. Various algorithms (Jutten and Herault, 1991) are proposed for performing the independent component analysis including maximization of the conditional entropy in the output (Bell and Sejnowski, 1995, 1997) (i.e., the information content in the output that, in general, increases if the output components become independent), minimization of the divergence measure between the joint density and the product of marginal densities (Amari, Cichocki, and Yang, 1996; Amari, 1998; Yang and Amari, 1997; Basak and Amari, 1999) using natural gradient and relative gradient techniques (Cardoso and Laheld, 1996), using nonlinear principal component analysis (Karhunen and Joutsensalo, 1994; Hyvarinen and Oja, 1998) and many others.

Usually, the technique for performing independent component analysis (ICA) is expressed as the technique for deriving one particular $\mathbf{W}$,

$$\mathbf{y} = \mathbf{W}\mathbf{x},$$

such that each component of $\mathbf{y}$ (i.e., each y_i) becomes independent of each other. If the individual marginal distributions are non-Gaussian then the derived marginal densities become a scaled permutation of the original density functions if one such $\mathbf{W}$ can be obtained. One general learning technique (Amari, 1998; Yang and Amari, 1997) for finding one $\mathbf{W}$ (as derived from the natural gradient descent of Kullback-Leibler divergence between joint density and the product of marginal densities) is

$$\Delta\mathbf{W} = \eta(I - \phi(\mathbf{y})\mathbf{y}^T)\mathbf{W},$$

where $\phi(\mathbf{y})$ is a nonlinear function of the output vector $\mathbf{y}$ (such as a cubic polynomial or a polynomial of odd degree, or a sum of polynomials of odd degrees, or a sigmoidal function).

Analogous to principal component analysis (PCA), independent component analysis (ICA) can also be used for feature extraction (Amari, Cichocki, and Yang, 1996; Bell and Sejnowski, 1997), where each data vector is the result of a mixture of multiple independent sources. In the next section, we describe the process of extracting the viable independent components from the weather data.

3.3 Feature Extraction from Weather Data

The weather data are represented in terms of frames where each frame is composed of a grid structure over certain region on earth (for example, the particular region is divided into $M \times N$ grid points). The sea level pressure data averaged over months are used in our study. The data over certain number of years (Y) is thus represented by a certain number of frames, say K, where $K = Y/T$. T is the period over which the data is averaged. For example, if we use monthly averaged data then T is a period of one month, i.e., $1/12$ year. Each frame consists of $M \times N$ data points (the data can be normalized for the sake of uniformity in the representation).

We applied the fast independent component analysis technique (Hyvarinen and Oja, 1996) to extract the independent stable components from the data sets. Note that, we intend to extract spatio-temporal stable activities in the weather. The independent component analysis assumes that the activities are spatially independent. The temporal behavior is captured in the changes of the mixing coefficients of the spatial activities. In the usual algorithms for ICA, an inherent assumption is that the mixing matrix does not change with time and signals are changing. Therefore, we consider several frames of spatial data to extract the independent components with an assumption that the number of spatially independent activities is less than or equal to the number of frames being considered. Later in the experimental section, we demonstrate the effectiveness of the choice of number of frames in capturing all such spatially independent activities. The spatio-temporal data sets (a total of $M \times N \times K$ data points) can be represented as input in different ways to the ICA computing algorithm and thus various interpretations can be obtained from the output. Here we consider two different representations of the data set.

The first representation is a spatial representation. Here each individual location of the grid is considered as a separate mixture signal (i.e., x_i). Thus the output extracted represents an independent signal in each grid location. This kind of representation has certain shortcomings. First, in reality, each grid location is not independent of the other location (in the neighborhood). Second, sea level pressure (the variable considered here) is a slowly varying variable. Thus if the number of frames is not sufficient then it is difficult to capture the statistical nature of the variables.

The second one is a temporal representation. Here each time frame is considered as a signal and all K frames are considered as input to the ICA computing module. Thus this kind of representation will extract certain independent stable activity across the frames that are not changing in nature over a period of time. Since each frame is represented as a signal, the assumption about the spatial independence of the activity across the grid locations is relaxed (there can be correlation between the activities in the grid location). Thus although we are investigating the existence of stable independent spatial activities, we convert the weather data into spatial signals and each frame over time is considered as a separate mixture.

We use the second representation with each frame being converted into a signal by sampling all the $M \times N$ grid locations randomly. It is not necessary that each frame be sampled sequentially, and

in fact, a random sampling can exhibit a better result because it will enhance the statistical measures over a smaller number of samples during the online computation of the ICs. However, if the ICs are computed in a batch mode then each frame can be sampled sequentially. Once the independent output signals are computed they are restored into the output frames in the same order as that of the input signals. Thus if $u(\mathbf{z},t)$ is the activity in a particular frame where $\mathbf{z}$ is the two-dimensional coordinates and t is the time at which the data frame is being considered then the input to the ICA computing block is given as

$$x_t(i) = u(\mathbf{z}_k,t),$$

where $\mathbf{z}_k$ is the two-dimensional coordinates at point k in the t^{th} frame, $x_t(i)$ is the i^{th} instance of the signal x_t corresponding to t^{th} frame. The variable i is a certain permutation of k, i.e., $i = P(k)$ where P is a one-to-one permutation function. Thus each input signal is represented by $M \times N$ discrete points, i.e., k runs from $1,\cdots,M \times N$. There are K such mixed signals, i.e., t runs from $1,\cdots,K$. Thus the input to the ICA computing algorithm is a K dimensional signal vector, each component signal of the vector has $1,\cdots,M \times N$ instances. Once the output is computed by the ICA computing block, they are restored as

$$v(\mathbf{z}_l,\tau) = y_\tau(j),$$

where $l = P^{-1}(j)$ is the spatial coordinate corresponding to the j^{th} instance of the signal. Thus, after computation, we obtain K different independent signals which represent the spatially independent activities. The mixing matrix A is a $K \times K$ non-singular matrix.

3.4 Validation of the Existence of Independent Components

The extracted stable independent components $v(\mathbf{z},t)$ are validated against the observed phenomenon and index (NAO index as described in Section 2) obtained by the weather scientists. Each input data frame can be represented as

$$u(\mathbf{z},t) = \sum_{\tau=1}^{K} a_{t\tau} v(\mathbf{z},\tau),$$

where $A = [a_{t\tau}]$ is the inverse of W.

Let us now present the way we validate the existence of independent components. Correspondig to K frames, we obtain K spatially independent stable activities in the weather. Let us represent them as $v(\mathbf{z},\tau)$ where τ indexes the independent spatial activities such that $\tau \in [1,\cdots,K]$. Note that τ does not have any correspondence with the time of the weather phenomenon and it is just an index of the independent components. Thus the columns of the mixing matrix $\mathbf{A} = [a_{t\tau}]$ represent the varying nature of the mixing coefficients for the corresponding independent components over time. That is, for a given independent component τ_0, $a_{1\tau_0},\cdots,a_{K\tau_0}$ represent the variations of the contribution of the independent component τ_0 over K time frames.

The strongest independent components can be obtained by maximum nongaussianity measure or some other measure (Hyvarinen and Oja, 1996). However, it is difficult to obtain a quantified index to characterize the overall changing nature of the weather phenomenon. Since we derive the independent components from the overall weather data, the columns for the strongest independent components (derived from the nongaussianity measure) may not exactly match the north Atlantic oscillation index which is a partial view of the overall weather. In order to correspond to the NAO phenomena in weather, we find those independent components that contribute maximally

to the NAO. These independent components are obtained by having a linear fit of the mixing coefficients with the NAO index. After obtaining a linear fit, if we find that the independent components maximally contributing to NAO index correspond to the fixed points on earth where the sea level pressures are measured for obtaining the NAO index then we establish our proposition that the ICA can provide an insight into the weather about the fact that such spatio-temporally stable activities can possibly exist in the nature.

In order to do so, first we obtain a linear fit of the mixing coefficients (columns of **A**) with the NAO index. Then we obtain the top two strongest components that provide dominant contribution in the linear fit. Then we observe how these two spatially independent components match with the real-world dipoles where the sea level pressures are measured in order to obtain the NAO index. First we find the independent components that contribute maximum to the NAO. If $g(t)$ represents the NAO index value at time t, then $g(t)$ can be expressed as

$$g(t) = \sum_{\tau} c_{\tau} a_{t\tau},$$

where c_τ is invariant over time. The coefficient $|c_\tau|$ indicates the strength of the corresponding independent component $\hat{v}(\mathbf{z}, \tau)$ in contributing to the NAO signal $g(t)$, where $\hat{v}(\cdot)$ is the normalized independent component. By linear regression, we obtain the coefficients **c** such that

$$\mathbf{c} = < aa^{'} >^{-1} < ga >,$$

where $< \cdot >$ is the sample mean over all time instances t. We then computed the variance of the linear fit as

$$V = \frac{1}{K} \sum_{t} (g(t) - \mathbf{c}^T \mathbf{a}(t))^2. \quad (1)$$

The strongest active stable phenomenon in the weather can be found by considering the largest component of **c**. The contribution of the stable components to the weather phenomenon is characterized by the strength of the coefficients **c**.

4. Experimental Results

We used data frames over 2 years, 4 years, 6 years, 8 years, and 10 years. Since the NAO activity is generally strong throughout the year and more or less repeats itself every year, a few years' data are sufficient to extract the major NAO features. Monthly averaging of the SLP data ensures that the daily transients are smoothed out and only the significant monthly behavior stands out in the SLP data. Therefore, we considered averaged phenomena over one month time period (it could have been with a higher resolution also, but in that case the number of data frames will be large) and made the total duration up to 10 years (even a much larger duration can also be considered). The total number of frames is therefore 24, 48, 72, 96, and 120 respectively in the different sets of experiments.

We obtained the independent components in two ways. In one experiment, we projected the data onto 10-dimensional space (which are the top 10 eigenvectors after the Karhunen-Loeve transform), and then performed the ICA on the projected signals. It was done in order to reduce the computational time. In the second experiment, we preserved the original signals and performed the ICA on them. Thus for the first experiment we always obtain 10 independent components and in the second case we obtain 24, 48, 72, 96, and 120 independent components for 2 years, 4 years, 6 years,

8 years, and 10 years of data respectively. Subsequently, the top two independent components are found that contribute maximum to the NAO (as described in Section 3.4).

We then obtained the normalized variance (Equation 1) of the linear fit for the top two components with the NAO index for 2-10 years of data set. Figure 4 illustrates the variance of the fit (Equation 1) for data sets of different number of years. We also illustrate the two stable independent components (strongest and the second strongest ones) obtained for the four years data set (as an example) in Figure 5. The strongest stable independent component (as extracted by the proposed algorithm) as illustrated in Figure 5, perfectly match with the observed dipoles of the NAO (low-pressure and high-pressure regions). The other data sets were also analyzed in the same way and similar results were obtained. As a comparison, we also illustrate the obtained stable oscillation patterns in the first experiment (where we computed ICA for the spatio-temporal data set projected onto the first 10 principal components) in the Figure 6. Note that, in the second experiment, since the independent components were computed from the original data set, it extracted the two dipoles separately. On the other hand, in the first experiment, ICA was performed on the projected data set. The strongest ICA component exhibits one dipole properly, however, the second strongest component exhibits an average of both the dipoles.

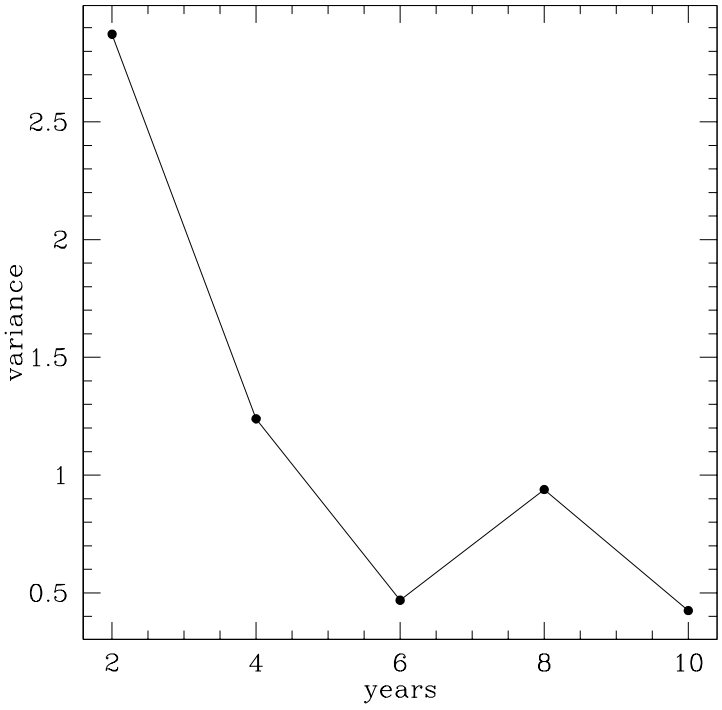
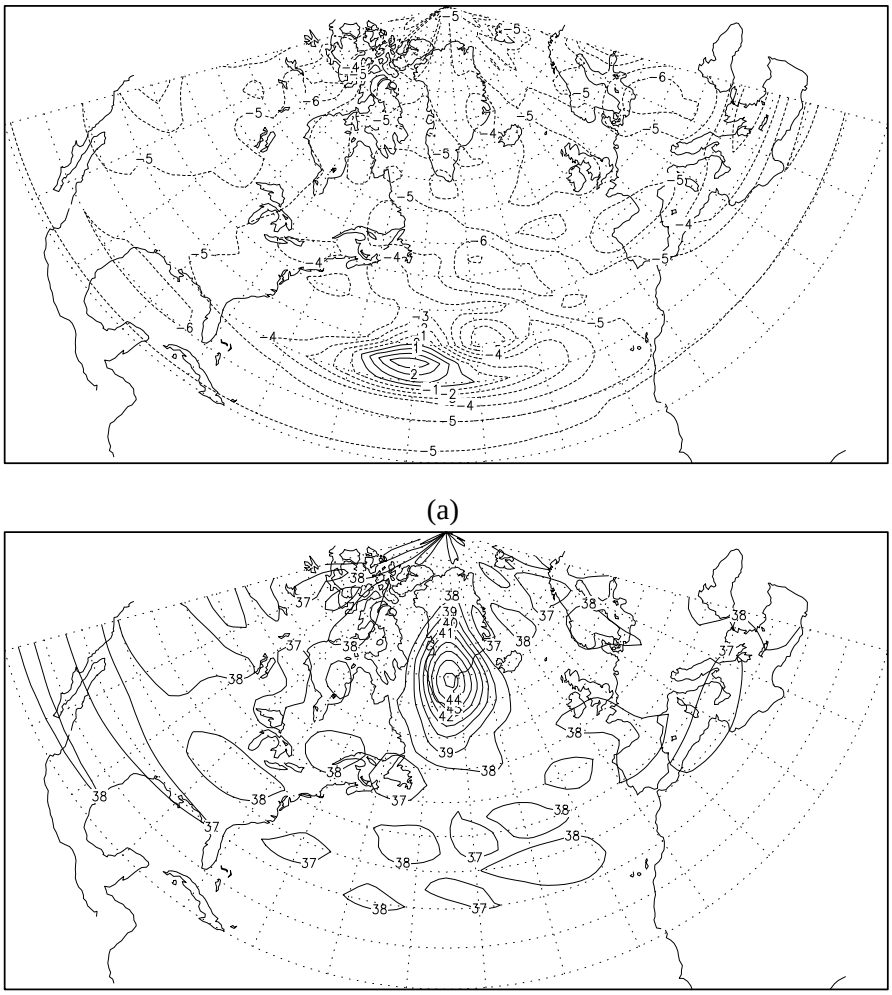


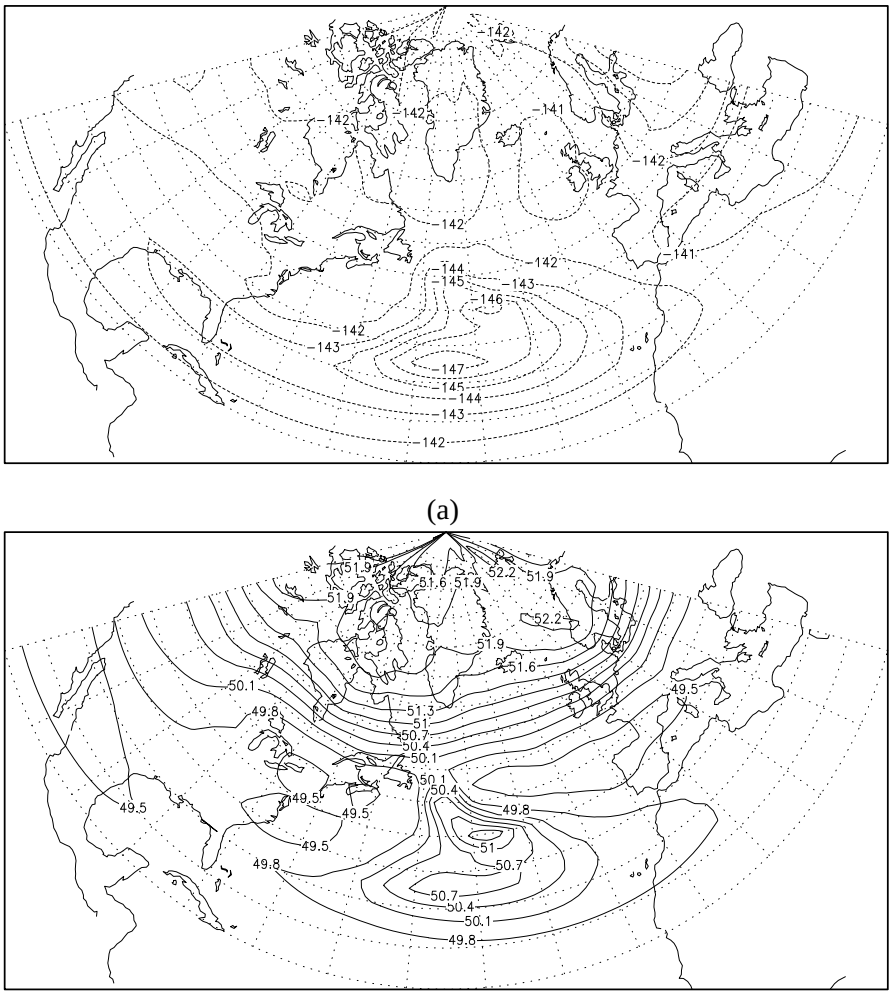
Figure 4: Variance of the linear fit of the top two strongest independent components with the NAO index for different number of years in the second experiment



(a)

(b)

Figure 5: (a) and (b) illustrate the stable oscillation patterns represented by the top two strongest independent components as obtained in the second experiment. Note that the oscillation patterns have strong resemblance to the dipoles observed in NAO.



(a)

(b)

Figure 6: (a) and (b) illustrate the stable oscillation patterns represented by the top two strongest independent components as obtained in the first experiment where independent components were computed from the data projected onto top 10 principal components.

5. Discussion and Conclusions

In this work, we have provided a new way of viewing the physical phenomena of changing weather and climate by mining spatio-temporal data of weather and climate variables. We consider the NAO as a typical example and mine the SLP data using independent component analysis. We provided techniques for determining the strongest independent components in the multidimensional data set, and observed that the strongest stable patterns as obtained by ICA matched with the physical patterns of oscillation in SLP. The results are also verified by finding a linear fit of the independent components with the standard NAO index as provided by the meteorological measurements.

The method of mining spatio-temporal data is generic in nature and is not subject only to the weather phenomenon. The same method can be applied to find certain stable characteristics in other spatio-temporal systems. Even when a spatio-temporal system is chaotic, the method may be appled to extract meaningful patterns if the system embeds some such stable patterns (possibly weather is a natural example of a physical chaotic system).

The method can be further investigated in the following manner. First, it extracts certain stable patterns whose temporal trend perfectly matches with the physical phenomenon. Therefore, the individual stable oscillations (obtained as independent components from the spatio-temporal data) can be analyzed further to predict the time-series behavior of the oscillation. Second, it is very difficult to analyze the NAO in order to find the physical correlations between various modes that interact to produce the NAO phenomenon. However, ICA gives a mixing matrix that provides an indication about how the various modes interact (in a linear manner). Third, we assumed a linear mixture of various independent components. In further investigation, this assumption can be relaxed and nonlinear independent component analysis can be performed on these kind of spatio-temporal data sets in order to find even more meaningful characteristics.

Acknowledgments

This work was done when the fourth author was affiliated with the IBM India Research Lab, Delhi. The authors acknowledge Dr. Ashwin Srinivasan for his kind effort in proof-reading this article.

References

S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10:251–276, 1998.

S.-I. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and E. Hasselmo, editors, *Neural Information Processing Systems : Natural and Synthetic, NIPS'96*, pages 757–763, MIT Press, 1996.

J. Basak and S.-I. Amari. Blind separation of a mixture of uniformly distributed signals. *Neural Computation*, 11:1011–1034, 1999.

A. J. Bell and T. J. Sejnowski. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

I. Bell and M. Visbeck. North Atlantic Oscillation. URL `http://www.ldeo.columbia.edu/NAO`.

J. F. Cardoso and B. Laheld. Equivariant adaptive source separation. *IEEE Transactions on Signal Processing*, 44:3017–3030, 1996.

P. A. Dejviver and J. Kittler. *Pattern Recognition : A Statistical Approach*. Prentice Hall International, 1982.

W. W. Hsieh and B. Tang. Applying neural network models to prediction and data analyis in meteorology and oceanography. *Bulletin of America Meteorological Society*, 79:1855–1870, 1998.

J. W. Hurrell. Decadal trends in the North Atlantic Oscillation region temperatures and precipitation. *Science*, 269:676–679, 1995.

A. Hyvarinen. Complexity pursuit: Separating interesting components from time-series. *Neural Computation*, 13:883–898, 2001.

A. Hyvarinen and E. Oja. A fast fixed point algorithm for ICA. Technical Report A-35, Faculty of Information Technology, Helsinki University of Technology, Finland, 1996.

A. Hyvarinen and E. Oja. Independent component analysis by general nonlinear Hebbian-like learning rules. *Signal Processing*, 64:301–313, 1998.

C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–20, 1991.

J. Karhunen and J. Joutsensalo. Representation and separation of signals using nonlinear pca type learning. *Neural Networks*, 7:113–127, 1994.

P. J. Lamb and R. A. Peppler. North Atlantic Oscillation - concept and an application. *Bulletin of American Meteorological Society*, 68:1218–1225, 1987.

E. N. Lorenz. Deterministic non-periodic flow. *Journal of Atmospheric Sciences*, 20:130–141, 1963.

E. N. Lorenz. A study of the predictability of a 28-variable atmospheric model. *Tellus*, 17:321–329, 1965.

A. H. Monahan. Nonlinear principal component analysis by neural networks: Theory and applications to the Lorentz system. *Journal of Climate*, 13:821–835, 2000.

Climate Diagnostics Center : NOAA-CIRES. URL `http://www.cdc.noaa.gov`.

E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

E. Oja. The nonlinear PCA learning rule and signal separation - mathematical analysis. Technical Report A26, Helsinki University of Technology, Lab. of Computer and Information Science, 1995.

E. Oja, J. Karhunen, L. Wang, and R. Vigario. Principal and independent components in neural networks - recent developments. In *Proc. Italian Workshop on Neural Networks, WIRN'95*, Vietri, Italy, 1995.

M. S. Santhanam and P. K. Patra. Statistics of atmospheric correlations. *Physical Review E*, 64: 016102–1–7, 2001.

J. V. Stone, J. Porrill, C. Buchel, and K. Friston. Spatial, temporal, and spatiotemporal independent component analysis of fMRI data. In *18th Leeds Statistical Research Workshop on Spatio-Temporal Modeling and its Applications, University of Leeds*, 1999.

M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

J. M. Wallace and D. S Gutzler. Teleconnections in the geopotential height field during the northern hemisphere winter. *Monthly Weather Review*, 109:784–812, 1981.

D. S. Wilks. *Statistical Methods in Atmospheric Sciences*. Academic Press, London, 1995.

H. H. Yang and S.-I. Amari. Adaptive on-line learning algorithms for blind separation - maximum entropy and minimum mutual information. *Neural Computation*, 9:1457–1482, 1997.

Journal of Machine Learning Research 5 (2004) 255–291 Submitted 10/03; Published 3/04

Online Choice of Active Learning Algorithms

Yoram Baram BARAM@CS.TECHNION.AC.IL
Ran El-Yaniv RANI@CS.TECHNION.AC.IL
Kobi Luz KOBIL@CS.TECHNION.AC.IL
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000
Israel

Editor: Manfred Warmuth

Abstract

This work is concerned with the question of how to combine online an ensemble of active learners so as to expedite the learning progress in pool-based active learning. We develop an active-learning master algorithm, based on a known competitive algorithm for the multi-armed bandit problem. A major challenge in successfully choosing top performing active learners online is to reliably estimate their progress during the learning session. To this end we propose a simple maximum entropy criterion that provides effective estimates in realistic settings. We study the performance of the proposed master algorithm using an ensemble containing two of the best known active-learning algorithms as well as a new algorithm. The resulting active-learning master algorithm is empirically shown to consistently perform almost as well as and sometimes outperform the best algorithm in the ensemble on a range of classification problems.

Keywords: Active learning, kernel machines, online learning, multi-armed bandit, entropy maximization

1. Introduction

The goal in *active learning* is to design and analyze learning algorithms that can effectively choose the samples for which they ask the teacher for a label. The incentive for using active learning is mainly to expedite the learning process and reduce the labeling efforts required by the teacher. While there is a lack of theoretical understanding of active learning (in particular, the generalization power of computationally practical active-learning algorithms is not well understood[1]), there is substantial empirical evidence that active learning can dramatically expedite the learning process.

We focus on *pool-based* active learning by classifiers, which can be viewed as the following game composed of *trials*. The learner is presented with a fixed pool of unlabeled instances. On each trial the learner chooses one instance from the pool to be labeled by the teacher. The teacher provides the learner with the true label of this instance and then the learner induces a (new) classifier based on all the labeled samples seen so far and possibly on unlabeled instances in the pool. Then a new trial begins, etc. Other variants of the active-learning problem have also been considered.

1. In noise-free settings, the Query-by-Committee (QBC) algorithm of Freund et al. (1997) can provably provide exponential speedups in the learning rate over a random selection. However, at this time a "practically efficient" implementation of this technique seems to be beyond reach (see, for example, Bachrach et al., 1999).

Two important variants are *stream-based* active learning (Freund et al., 1997) and learning with *membership queries* (Angluin, 1988); see Section 3 for more details. We do not consider these variants in this work.

Seeking top performing active-learning algorithms among the numerous algorithms proposed in the literature, we found two algorithms that appear to be among the best performers, based on empirical studies. The first algorithm relies on kernel machines and was independently proposed by three research groups (Tong and Koller, 2001, Schohn and Cohn, 2000, Campbell et al., 2000). The algorithm, called SIMPLE by Tong and Koller (2001), uses the current SVM classifier to query the instance closest to the decision hyperplane (in kernel space). A theoretical motivation for SIMPLE is developed by Tong and Koller (2001) in terms of version space bisection. The second algorithm we consider, proposed by Roy and McCallum (2001), is based on a different motivation: the algorithm chooses its next example to be labeled while attempting to reduce the generalization error probability. Since true future error rates are unknown, the learner attempts to estimate them using a "self-confidence" heuristic that utilizes its current classifier for probability measurements. Throughout this paper we, therefore, call this algorithm SELF-CONF. The original SELF-CONF proposed by Roy and McCallum (2001) bases its probability estimates (and classification) on Naive Bayes calculations and is shown to significantly outperform other known active-learning algorithms on text categorization tasks. In our studies we used an SVM-based variant of SELF-CONF, which appears to be stronger than the original.

Empirical studies we have conducted reveal that neither SIMPLE nor SELF-CONF is a consistent winner across problems. Moreover, both algorithms exhibit a severe pitfall that seems to appear in learning problems with a "XOR-like" structure (see Section 4.4). While perhaps no single active-learning algorithm should be expected to consistently perform better than others on all problems, some problems clearly favor particular algorithms. This situation motivates an online learning approach whereby one attempts to utilize an *ensemble* of algorithms online aiming to achieve a performance that is close to the best algorithm in hindsight. This scheme has been extensively studied in computational learning theory, mainly in the context of 'online prediction using expert advice' (see, for example, Ceza-Bianchi et al., 1997). Our main contribution is an algorithm that actively learns by combining active learners.

A reasonable approach to combining an ensemble of active-learning algorithms (or 'experts') might be to evaluate their individual performance and dynamically switch to the best performer so far. However, two obstacles stand in the way of successfully implementing this scheme. First, standard classifier evaluation techniques, such as cross-validation, leave-one-out, or bootstrap, tend to fail when used to estimate the performance of an active learner based on the labeled examples chosen by the learner. The reason is that the set of labeled instances selected by a good active learner tends to be acutely biased towards 'hard' instances that do not reflect the true underlying distribution. In Section 6 we show an example of this phenomenon. Second, even if we overcome the first problem, each time we choose to utilize a certain expert, we only get to see the label of the example chosen by this expert and cannot observe the consequence of choices corresponding to other experts without "wasting" more labeled examples.

We overcome these two obstacles by using the following two ideas. Instead of using standard statistical techniques, such as cross-validation, we use a novel maximum entropy semi-supervised criterion, which utilizes the pool of unlabeled samples and can faithfully evaluate the relative progress of the various experts; second, we cast our problem as an instance of the *multi-armed bandit* problem, where each expert corresponds to one slot machine and on each trial we are allowed to

play one machine (that is, choose one active-learning algorithm to generate the next query). We then utilize a known online multi-armed bandit algorithm proposed by Auer et al. (2002). This algorithm enjoys strong performance guarantees without any statistical assumptions.

We present an extensive empirical study of our new active-learning master algorithm and compare its performance to its ensemble members consisting of three algorithms: SIMPLE, SELF-CONF and a novel active-learning heuristic based on "furthest-first traversals" (Hochbaum and Shmoys, 1985). Our master algorithm is shown to consistently perform almost as well as the best algorithm in the ensemble, and on some problems it outperforms the best algorithm.

2. Active Learning

In this section we first define pool-based active learning and then discuss performance measures for active learners. We consider a binary classification problem and are given a pool $\mathcal{U} = \{x_1, \ldots, x_n\}$ of unlabeled instances where each x_i is a vector in some space $\mathcal{X}$. Instances are assumed to be i.i.d. distributed according to some unknown fixed distribution $P(x)$. Each instance x_i has a label $y_i \in \mathcal{Y}$ (where in our case $\mathcal{Y} = \{\pm 1\}$) distributed according to some unknown conditional distribution $P(y|x)$. At the start, none of the labels is known to the learner. At each stage of the active-learning game, let $\mathcal{L}$ be the set of labeled instances already known to the learner.[2] An *active learner* consists of a classifier learning algorithm $\mathcal{A}$, and a *querying function* Q, which is a mapping $Q : \mathcal{L} \times \mathcal{U} \to \mathcal{U}$. The querying function determines one unlabeled instance in $\mathcal{U}$ to be labeled by the teacher. On each *trial* $t = 1, 2, \ldots$, the active learner first applies Q to choose one unlabeled instance x from $\mathcal{U}$. The label y of x is then revealed and the pair (x, y) is added to $\mathcal{L}$ and x is removed from $\mathcal{U}$. Then the learner applies $\mathcal{A}$ to induce a new classifier C_t using $\mathcal{L}$ (and possibly $\mathcal{U}$) as a training set and a new trial begins, etc. Thus, the active learner generates a sequence of classifiers $C_1, C_2, \ldots$. Clearly, the maximal number of trials is the initial size of $\mathcal{U}$. In this paper we focus on active learners $(\mathcal{A}, Q)$ where $\mathcal{A}$ is always an SVM induction algorithm.

To the best of our knowledge, in the literature there is no consensus on appropriate performance measures for active learning. A number of performance measures for active-learning algorithms make sense. For example, some authors (for instance, Tong and Koller, 2001) test the accuracy achieved by the active learner after a predetermined number of learning trials. Other authors (for example, Schohn and Cohn, 2000, Campbell et al., 2000, Roy and McCallum, 2001) simply show active-learning curves to visually demonstrate advantages in learning speed. Here we propose the following natural performance measure, which aims to quantify the "deficiency" of the querying function with respect to random sampling from the pool (which corresponds to standard "passive learning"), while using a fixed inductive learning algorithm ALG. Fix a particular classification problem. Let $\mathcal{U}$ be a random pool of n instances. For each $1 \leq t \leq n$ let $\mathrm{Acc}_t(\mathrm{ALG})$ be the true average accuracy achievable by ALG using a training set of size t that is randomly and uniformly chosen from $\mathcal{U}$. A hypothetical $\mathrm{Acc}_t(\mathrm{ALG})$ is depicted by the lower learning curve in Figure 1. Let ACTIVE be an active-learning algorithm that uses ALG as its inductive learning component $\mathcal{A}$. Define $\mathrm{Acc}_t(\mathrm{ACTIVE})$ to be the average accuracy achieved by ACTIVE after t active-learning trials starting with the pool $\mathcal{U}$ (see Figure 1 for a hypothetical $\mathrm{Acc}_t(\mathrm{ACTIVE})$, which is depicted by the

2. We assume that initially $\mathcal{L}$ contains two examples, one from each class.

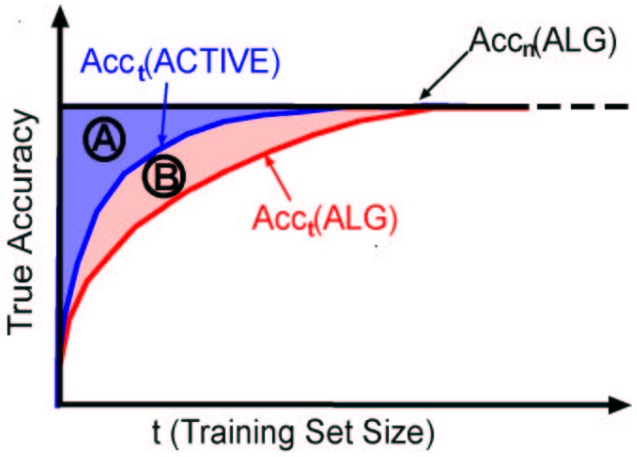


Figure 1: The definition of an active learner's *deficiency*. $\mathrm{Acc}_n(\mathrm{ALG})$ is the maximal achievable accuracy. $\mathrm{Acc}_t(\mathrm{ACTIVE})$ is the average accuracy achieved by hypothetical active learner ACTIVE. $\mathrm{Acc}_t(\mathrm{ALG})$ is the average accuracy achieved by a "passive" learner ALG (which queries randomly and uniformly from the pool). In this case (when the active-learning curve is above the passive learning curve) the *deficiency*, as defined in Equation (1), is the ratio of areas $\frac{A}{A+B}$.

higher learning curve). Then, the *deficiency* of ACTIVE is defined to be

$$\mathrm{Def}_n(\mathrm{ACTIVE}) = \frac{\sum_{t=1}^{n}(\mathrm{Acc}_n(\mathrm{ALG}) - \mathrm{Acc}_t(\mathrm{ACTIVE}))}{\sum_{t=1}^{n}(\mathrm{Acc}_n(\mathrm{ALG}) - \mathrm{Acc}_t(\mathrm{ALG}))}. \tag{1}$$

In words, the deficiency is simply the ratio of areas $\frac{A}{A+B}$, as depicted in Figure 1. This measure captures the "global" performance of an active learner throughout the learning session. Notice that the numerator is simply the area between the "maximal"[3] achievable accuracy $\mathrm{Acc}_n(\mathrm{ALG})$ using the entire pool, and the learning curve of the active-learning algorithm (area A in Figure 1). The denominator is the area between the same maximal accuracy and the learning curve of the "passive" algorithm (the sum of areas A and B in Figure 1). The purpose of the denominator is to normalize the measure so as to be "problem independent". Thus, this measure is always non-negative and ***smaller*** values in $[0,1)$ indicate more *efficient* active learning. $\mathrm{Def}_n(\mathrm{ACTIVE})$ has the desired property that if n is sufficiently large so that $\mathrm{Acc}_n(\mathrm{ALG})$ achieves the maximal accuracy (for this classifier), then for any $n' > n$, $\mathrm{Def}_{n'}(\mathrm{ACTIVE}) = \mathrm{Def}_n(\mathrm{ACTIVE})$.

3. Related Work

This paper is the first to consider a utilization of an ensemble of active learners. Here we discuss selected results related to the present work. Our main focus is on techniques for devising querying functions and methods that can be used to evaluate the progress of an active learner. Note that, in

3. In some cases (see for example, Schohn and Cohn, 2000), better accuracy can be achieved using "early stopping" (see Section 3).

general, a method estimating the "value" of an *unlabeled point* can be used to construct a querying function, and a method that evaluates the gain of a newly acquired *labeled point* can be used as an estimator for an active learner's progress.

We begin with a presentation of various techniques used to construct querying functions. Although our work focuses on pool-based active learning, a number of interesting and relevant ideas appear within other active-learning frameworks. In addition to the pool-based active-learning setting introduced in Section 2, we consider two other settings:

- *Stream-based* active learning (see, for example, Freund et al., 1997): the learner is provided with a stream of unlabeled points. On each trial, a new unlabeled point is drawn and introduced to the learner who must decide whether or not to request its label. Note that the stream-based model can be viewed as an online version of the pool-based model.

- Active learning with *membership queries* (Angluin, 1988), also called *selective sampling*: On each trial the learner constructs a point in input space and requests its label. This model can be viewed as a pool-based game where the pool consists of *all possible points* in the domain.

Within the *stream-based* setting, Seung et al. (1992) present the Query by Committee (QBC) algorithm. This algorithm is based on a seminal theoretical result stating that by halving the version space after each query, the generalization error decreases exponentially (relative to random active learning). To approximately bisect the version space, the proposed method randomly samples the version space and induces an even number of classifiers. The label of a (stream) point is requested whenever a voting between the classifiers on this point's label results in a tie. The main obstacle in implementing this strategy is the sampling of the version space. Currently, it is not known how to efficiently sample version spaces uniformly at random for many hypothesis classes of interest (see also discussions in Bachrach et al., 1999, Herbrich et al., 2001).

A variation of the QBC algorithm was proposed by McCallum and Nigam. Expectation Maximization (EM) is used to create a committee of classifiers to implement the querying function using the QBC voting idea: A diversified committee of classifiers is created by sampling a population of Naive Bayes distributions. Then an EM-like procedure is used to iteratively classify the unlabelled data and rebuild the classifiers until the process converges. Experimental results show that this algorithm has an advantage over random sampling and QBC.[4]

Interesting ideas were also considered within the *learning with membership queries* setting of Angluin (1988). Cohn et al. (1994) use the following version space reduction strategy. Two "distant" hypotheses are selected from the version space by finding a "most specific" concept (that classifies as 'negative' as much of the domain as possible) and a "most general" one (that classifies as 'positive' as many points as possible).[5] Selective sampling is done by randomly searching for a domain point on which the most specific and most general hypotheses disagree. This method ensures that the version space size is reduced after every query, though it might be reduced by only one hypothesis. The algorithm was implemented using a hypothesis class consisting of feed-forward neural networks trained with the back-propagation algorithm. According to the authors, the method works well only for "simple" concepts.

4. The QBC used for comparison utilizes the committee created by sampling of the Naive Bayes distributions without employing the EM-like procedure.
5. The most specific (resp. general) concept is found by classifying many (unlabeled) examples as 'negative' (resp. 'positive').

Again, within the selective sampling model, Lindenbaum et al. (2004) propose an active-learning algorithm for the nearest-neighbor classifier. The paper proposes using a random field model to estimate class probabilities. Using the class probabilities of the unlabeled examples a "utility function" of a training set is defined. The querying function of this algorithm is constructed using a game-tree that models a game between the learner and the teacher. For each unlabeled example, its expected utility is measured using the utility function on the training set and using expected probabilities for the possible classes of the unlabeled example. A set of domain points is constructed randomly and the example with highest expected utility is chosen. This algorithm suffers from extensive time complexity that depends both on the depth of the game-tree and on the number of near examples taken to construct the random field.

In the *pool-based* setting, three independent working groups (Schohn and Cohn, 2000, Campbell et al., 2000, Tong and Koller, 2001) propose the same querying function for active learners based on support vector machines (SVMs). This querying function chooses, for the next query, the unlabeled point closest to the decision hyperplane in kernel space; namely, the point with the smallest margin. Some theoretical motivation is given to this function in terms of version space reduction along the lines of QBC ideas (Tong and Koller, 2001); see Section 4.1 for further details. Experimental results presented in these papers show that the resulting active learner can provide significant sample complexity speedups compared to random sampling. Schohn and Cohn (2000) encountered a phenomenon where the true accuracy of the learner *decreases* (after reaching a certain peak) as the active session progresses. This phenomenon motivates the idea of "early stopping" and the authors suggest to stop querying when no example lies within the SVM margin. Campbell et al. (2000) offer to use this stopping criterion as a trigger for running a test for the current classifier. In this test they randomly and uniformly choose a small subsample of (unlabeled) pool points, request the labels of these points from the teacher and then test the current classifier on these points. The algorithm decides to stop if the error (as estimated using this test) is "satisfying" according to a user defined threshold.

Zhang and Oles (2000) analyze the value of unlabeled data in both active and transductive learning settings focusing on SVM learning. The value of unlabeled data is evaluated using Fisher information matrices. It is proved that selecting unlabeled data *with low confidence* (i.e. small margin) that is not subsumed by (i.e., not too close to) previous choices is likely to cause a large change in the model (once the true label is known). A few numerical examples presented in this paper show that an SVM-based active learner using this criterion is superior to random selection.

Roy and McCallum (2001) offer a pool-based active-learning algorithm that attempts to "directly" optimize the true generalization error rate (rather than reduce the version space). The algorithm chooses to query points that strengthen the belief of the current classifier about the pool classification. The learner estimates its error on the unlabeled pool, where the current classifier's posterior class probabilities are taken as proxies for the true ones. For each example in the pool and for each possible label, the expected loss of a classifier trained after adding this example to the training set is calculated. The example from the pool with the lowest total expected loss is chosen as the next query. The computational complexity required to implement this scheme is "hopelessly impractical," as noted by the authors. However, various heuristic approximations and optimizations are suggested. Some of these optimizations (for example, subsampling) are general and some are designed for the specific implementation provided by Roy and McCallum (2001), which is based on Naive Bayes. The authors report that the resulting active learner exhibits excellent performance over

text categorization problems.[6] In Section 4 we further elaborate on the two pool-based algorithms mentioned above.

Active learning has also been explored in the contexts of regression and probabilistic estimation. Saar-Tsechansky and Provost (2002) examine the use of active learning for *class probability estimation* (in contrast to exact, or 'hard' classification). The algorithm proposed is based on creating a number of subsamples from the training set and training one soft classifier (capable of giving class probability estimates) on each subsample. All these generated classifiers form an "ensemble". The querying function works by drawing a pool point according to a probability distribution that is proportional to the variance of the probability estimates computed by the various ensemble members. This variance is normalized by the average probability of the minority class. Empirical results indicate that this algorithm performs better than random sampling (both as a soft as well as a hard classifier).

In the context of regression, Cohn et al. (1996) assume that the learner is approximately unbiased and propose an active-learning algorithm that chooses to query the instance that minimizes the average expected variance of the learner (integrated over the input domain). This algorithm is outlined for general feed-forward neural networks, a mixture of Gaussians and locally weighted regression. For the latter two learning schemes, this method is efficient. However, this method ignores the learner's bias. The algorithm requires a closed form calculation of the learner's variance, which has been developed only for the three learning schemes considered.

In the context of ensemble methods for regression, Krogh and Vedelsby (1995) show a nice equality giving the (squared error loss) regression generalization error as a (weighted) sum of the generalization errors of the individual members minus a diversity measure called 'ambiguity', which is related to the anti-correlation between ensemble members. Increasing this ambiguity (without increasing the weighted sum of individual errors) will reduce the overall error. Increasing the ambiguity is done by dividing the training set into a number of disjoint subsets and training one ensemble member over a distinct training subset. One nice property of the ambiguity measure is that it can be computed without the labels. Krogh and Vedelsby (1995) apply this relation (which guarantees that the weighted ambiguity is a lower bound on the weighted sum of individual ensemble generalization errors) in the context of pool-based active learning as follows. Assuming that the most beneficial pool point to sample is the one that increases the weighted sum of individual ensemble errors, they search for the point whose contribution to the ensemble ambiguity is maximal.

We now turn to discuss some approaches to measuring the "value" of *labeled* points. Such methods are potentially useful for obtaining information on the progress of active learners, and may be useful for implementing our approach that combines an ensemble of active learners. Within a Bayesian setting, MacKay (1992) attempts to measure the information that can be gained about the unknown target hypothesis using a new *labeled* point. Two gain measures are considered, both based on Shannon's entropy. It is assumed that given a training set, a probability distribution over the possible hypotheses is defined. The first measure is how much the entropy of the hypothesis distribution has decreased using a new labeled point. A decrease in this entropy can indicate how much the support of this hypothesis distribution shrinks. The second measure is the cross-entropy between the hypothesis distributions before and after the new labeled point is added. An increase in this cross-entropy can indicate how much the support has changed due to the new information. It is proved that these two gain measures are equivalent in the sense that their expectations are equal.

6. Our implementation of this algorithm, as discussed in Section 4.2, uses SVMs rather than Naive Bayes and only utilizes the subsampling approximation idea suggested by Roy and McCallum (2001).

Guyon et al. (1996) propose methods to measure the information gain of labeled points in a data set. The information gain of a labeled point with respect to a trained probabilistic (soft) classifier is defined as the probability that the classification of this point is correct. This probability equals the Shannon "information gain" of the class probability of that point as measured by the probabilistic classifier. On a given labeled data set, the information gain of each point can be measured using the above definition with a leave-one-out estimator. The paper also addresses the information gain of labeled points in the context of minimax learning algorithms such as SVM. The SVM measures the information gain of a labeled point by its ("alpha") coefficient in the induced weights vector. The coefficient of a point that is not a support vector is zero, and so is its information gain. Support vectors with larger coefficients are more informative. These ideas are discussed in the context of "data cleaning" of large databases. We note that, in the context of SVM active learning it is not in general the case that support vectors identified during early trials will remain support vectors thereafter.

Finally, we note that active learning is only one way of exploiting the availability of unlabeled data. In recent years the broader topic of "semi-supervised" or "learning with labeled-unlabeled examples" has been gaining popularity. One can distinguish between inductive semi-supervised methods and transductive ones (Vapnik, 1998); see a recent survey by Seeger (2002) for a recent review.

4. On Three Active-Learning Algorithms

The main purpose of this section is to motivate the challenge considered in this work. In contrast to what may be implied in various recent papers in the field, when closely examining state-of-the-art active-learning algorithms on various learning problems, there is no consistent single winner. While not necessarily surprising, it remains unclear how to choose the best active-learning algorithm for the problem at hand.

To demonstrate this point we focus on two known active-learning algorithms: SIMPLE (Tong and Koller, 2001, Schohn and Cohn, 2000, Campbell et al., 2000) and the algorithm of Roy and McCallum (2001), which we call here SELF-CONF. We selected these algorithms because they are both reasonably well motivated, achieve high performance on real-world data (and surpass various other known algorithms) and appear to complement each other. In particular, as we show below, neither algorithm consistently tops the other.[7] Moreover, both algorithms fail in learning problems with "XOR-like" structures (in the sense that they perform considerably worse than random sampling).[8] We thus consider another novel algorithm, which excels on 'XOR' problems but exhibits quite poor performance on problems with simple structures. Altogether, these three algorithms form the ensemble on which our new "master" algorithm is later experimentally applied (Section 8). Throughout the rest of the paper we assume basic familiarity with SVMs.[9]

4.1 Algorithm SIMPLE

This algorithm uses an SVM as its induction component. The querying function of SIMPLE at trial t uses the already induced classifier C_{t-1} to choose an unlabeled instance, which is closest to the

7. To the best of our knowledge, no previous attempts to compare these two algorithms have been conducted.
8. This deficiency (of SIMPLE) was already observed by Tong and Koller (2001).
9. Good textbooks on the subject were written by Cristianini and Shawe-Taylor (2000) and Schölkopf and Smola (2002).

decision boundary of C_{t-1}. Thus, the querying function can be computed in time linear in $|\mathcal{U}|$ (assuming the number of support vectors is a constant).

An intuitive interpretation of SIMPLE is that it chooses to query the instance whose label is the "least certain" according to the current classifier. A more formal theoretical motivation for SIMPLE is given by Tong and Koller (2001) in terms of *version space* bisection. This argument utilizes a nice duality between instances and hypotheses. The *version space* (Mitchell, 1982) is defined as the set of all hypotheses consistent with the training set. Let $\mathcal{L} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be a training set, and let $\mathcal{H}$ be the set of all hypotheses (in our case $\mathcal{H}$ is the set of all hyperplanes in kernel space). The version space $\mathcal{V}$ is defined as

$$\mathcal{V} = \{h \in \mathcal{H} | \forall (x,y) \in \mathcal{L}, h(x) = y\}.$$

Let w^* be a hyperplane constructed in kernel space by training an SVM on the entire pool with the true labeling (clearly w^* cannot be computed by an active learner without querying the entire pool). A good strategy for an active learner would be to choose a query that bisects the version space as it brings us faster to w^*. The question is how to find the instance that approximately bisects the version space. The version space can be viewed as a subsurface of the unit hypersphere in parameter space (the entire surface contains all possible hypotheses). Every unlabeled instance x corresponds to a hyperplane $\Phi(x)$ in the parameter space. The hypotheses $\{w|(w \cdot x) > 0\}$ that classify x 'positively' lie on one side of this hyperplane and the hypotheses $\{w|(w \cdot x) < 0\}$ that classify x 'negatively' lie on the other side. Let w be a hyperplane constructed in kernel space by training an SVM on the current set of labeled instances. Tong and Koller (2001) show that w is the center of the largest hypersphere that can be placed in version space and whose surface does not intersect the hyperplanes corresponding to the labeled instances. Moreover, the radius of this hypersphere is the margin of w and the hyperplanes that "touch" this hypersphere are those corresponding to the support vectors. Therefore, in cases where this hypersphere sufficiently approximates the version space, w approximates the center of mass of the version space. The closer an unlabeled instance is to w, the closer the hyperplane $\Phi(x)$ is to the center of mass of the version space. This motivates the strategy of querying the closest instance to w, as it can bisect the version space and quickly advance the algorithm toward the (unknown) target hypothesis w^*.

Tong and Koller (2001) require that each of the classifiers (i.e., SVMs) computed during the operation of SIMPLE is consistent with its training set. This is essential to guarantee a non-vacuous version space. This requirement is easy to fulfil using the "kernel trick" discussed by Shaw-Taylor and Christianini (1999), which guarantees that the training set is linearly separable in kernel space. Further details are specified in Appendix A.

When considering simple binary classification problems, this strategy provides rapid refinement of the decision boundary and results in very effective active learning (Tong and Koller, 2001, Schohn and Cohn, 2000, Campbell et al., 2000). In our experience, algorithm SIMPLE is a high-performance active-learning algorithm, which is hard to beat on a wide range of real-world problems. In particular, this algorithm performs quite well on text categorization problems (Tong and Koller, 2001).

We note that two more querying functions (called "Maxmin" and "Maxratio") are proposed by Tong and Koller (2001). These are also based on the version space bisection principle and achieve more efficient version space reduction than SIMPLE. The main drawback of these functions is their computational intensity, which makes them impractical. Specifically, for each query, the

computation of each of these functions requires the induction of two SVMs for each unlabeled point in the pool.[10]

4.2 Algorithm SELF-CONF

The second algorithm we discuss, proposed by Roy and McCallum (2001), is based on a different motivation. The algorithm chooses its next example to be labeled while "directly" attempting to reduce future generalization error probability. Since true future error rates are unknown, the learner attempts to estimate them using a "self-confidence" heuristic that utilizes its current classifier for probability measurements. Throughout this paper we therefore call this algorithm SELF-CONF.[11]

At the start of each trial this algorithm already holds a trained probabilistic (soft) classifier denoted by $\hat{P}(y|x)$. For each $x \in \mathcal{U}$ and $y \in \mathcal{Y}$ the algorithm trains a new classifier $\hat{P}'$ over $\mathcal{L}'(x,y) = \mathcal{L} \cup \{(x,y)\}$ and estimates the resulting "self-estimated expected log-loss"[12] defined to be

$$E(\hat{P}'_{\mathcal{L}'(x,y)}) = -\frac{1}{|\mathcal{U}|} \sum_{y' \in \mathcal{Y}, x' \in \mathcal{U}} \hat{P}'(y'|x') \log \hat{P}'(y'|x'). \tag{2}$$

Then, for each $x \in \mathcal{U}$ it calculates the self-estimated average expected loss

$$\sum_{y \in \mathcal{Y}} \hat{P}(y|x) E(\hat{P}'_{\mathcal{L}'(x,y)}).$$

The x with the lowest self-estimated expected loss is then chosen to be queried.

The original SELF-CONF algorithm proposed by Roy and McCallum (2001) bases its probability estimates (and classification) on Naive Bayes calculations[13] and is shown to outperform other known active-learning algorithms on text categorization tasks. In our studies, we used an SVM-based variant of SELF-CONF, which appears to be stronger than the original Naive Bayes algorithm.[14] Thus, in our case, probabilistic estimates are obtained in a standard way, using logistic regression.[15] Also note that the algorithm, as presented, is extremely inefficient, since for each query, two SVMs are induced for each unlabeled point in the pool. Various optimizations and approximations are proposed by Roy and McCallum (2001), to make its running time practically feasible. From all these methods we only use random subsampling in our implementation of the algorithm: On each trial we estimate the expression (2) for only a random subset of $\mathcal{U}$. The subsample in the first active session trial contains 100 points; on each subsequent trial we decrement the subsample size by one point until we reach a minimum of 10 points, which we keep for the remaining trials.

Observe that the motivation for SELF-CONF and SIMPLE is in some sense complementary. While SIMPLE queries instances in $\mathcal{U}$ with the least confidence, as measured by its current model, SELF-CONF queries instances that provide the maximal "reassurance" of its current model.

10. Approximation heuristics such as subsampling of the pool as well as using incremental/decremental SVM algorithms (Cauwenberghs and Poggio, 2000) can be used to speed up these algorithms. We have not explored these possibilities in the present work.
11. This self-confidence approach is somewhat similar to the one proposed by Lindenbaum et al. (2004).
12. Roy and McCallum (2001) also considered using zero-one loss instead of log-loss.
13. The empirical results of Roy and McCallum (2001) considered multinomial models (in particular, text categorization), which enabled the required calculations.
14. However, we have not performed an extensive comparison between the Naive Bayes and SVM variants of this algorithm.
15. If $h(x)$ is the (confidence-rated) SVM value for a point x whose label is y, we take $\hat{P}(y|x) = \frac{1}{1+\exp\{-yh(x)\}}$.

4.3 Algorithm Kernel Farthest-First (KFF)

Here we propose a simple active-learning heuristic based on "farthest-first" traversal sequences in kernel space. *Farthest-first (FF)* sequences have been previously used for computing provably approximate optimal clustering for k-center problems (Hochbaum and Shmoys, 1985). The FF traversal of the points in a data set is defined as follows. Start with any point x and find the farthest point from x. Then find the farthest point from the first two (where the distance of a point from a set is defined to be the minimum distance to a point in the set), etc. In any metric space, FF traversals can be used for computing 2-approximation solutions to the *k-center clustering* problem in which one seeks an optimal k-clustering of the data and optimality is measured by the maximum diameter of the clusters. In particular, by taking the first k elements in the FF traversal as "centroids" and then assigning each other point to its closest "centroid", one obtains a k-clustering whose cost is within a factor 2 of the optimal (Hochbaum and Shmoys, 1985).

We use FF traversals for active learning in the following way. Given the current set $\mathcal{L}$ of labeled instances, we choose as our next query an instance $x \in \mathcal{U}$, which is farthest from $\mathcal{L}$. Using $\mathcal{L} \cup \{(x,y)\}$ as a training set, the active learner then induces the next classifier. This heuristic has a nice intuitive appeal in our context: the next instance to query is the farthest (and in some sense the most dissimilar) instance in the pool from those we have already observed. Unlike SIMPLE (and other algorithms) whose querying function is based on the classifier, the above FF querying function can be applied with any classifier learning algorithm. We apply it with an SVM. For compatibility with the SVM, we compute the FF traversals in kernel space as follows. Given any kernel K, if x and y are instances in input space, and $\Phi(x)$ and $\Phi(y)$ are their embedding in kernel space (so that $\langle \Phi(x), \Phi(y) \rangle = K(x,y)$), then we measure $d_K(x,y)$, the distance between x and y in kernel space, as $d^2(x,y) = \|\Phi(x) - \Phi(y)\|^2 = K(x,x) + K(y,y) - 2K(x,y)$. We call the resulting algorithm KFF.

4.4 Some Examples

Figure 2 shows the learning curves of SIMPLE, SELF-CONF, KFF and of a random sampling "active learner" (which we call RAND)[16] for the artificial 'XOR' problem of Figure 2 (top left) as well as two other UCI problems - 'Diabetis' and 'Twonorm'. These three learning problems demonstrate that none of the three active-learning algorithms discussed here is consistently better than the rest. In the two UCI problems SIMPLE performs better than SELF-CONF on the 'Twonorm' data set, while SELF-CONF performs better than SIMPLE on the 'Diabetis' data set. KFF is inferior to the other two in both cases. In the 'XOR' problem SIMPLE's and SELF-CONF's performances are significantly worse than that of random sampling. On the other hand, KFF clearly shows that active learning can expedite learning also in this problem. This weakness of both SIMPLE and SELF-CONF typically occurs when confronting problems with "XOR-like" structure. The inferiority of KFF relative to SIMPLE and SELF-CONF (and even RAND) typically occurs in learning problems with a "simple" structure. The main advantage in considering KFF is its use within an ensemble of active-learning algorithms. The "master" active algorithm we present later benefits from KFF in "XOR-like" problems without significant compromises in problems where KFF is weaker.

16. The querying function of RAND chooses the next example to be labeled uniformly at random from $\mathcal{U}$.

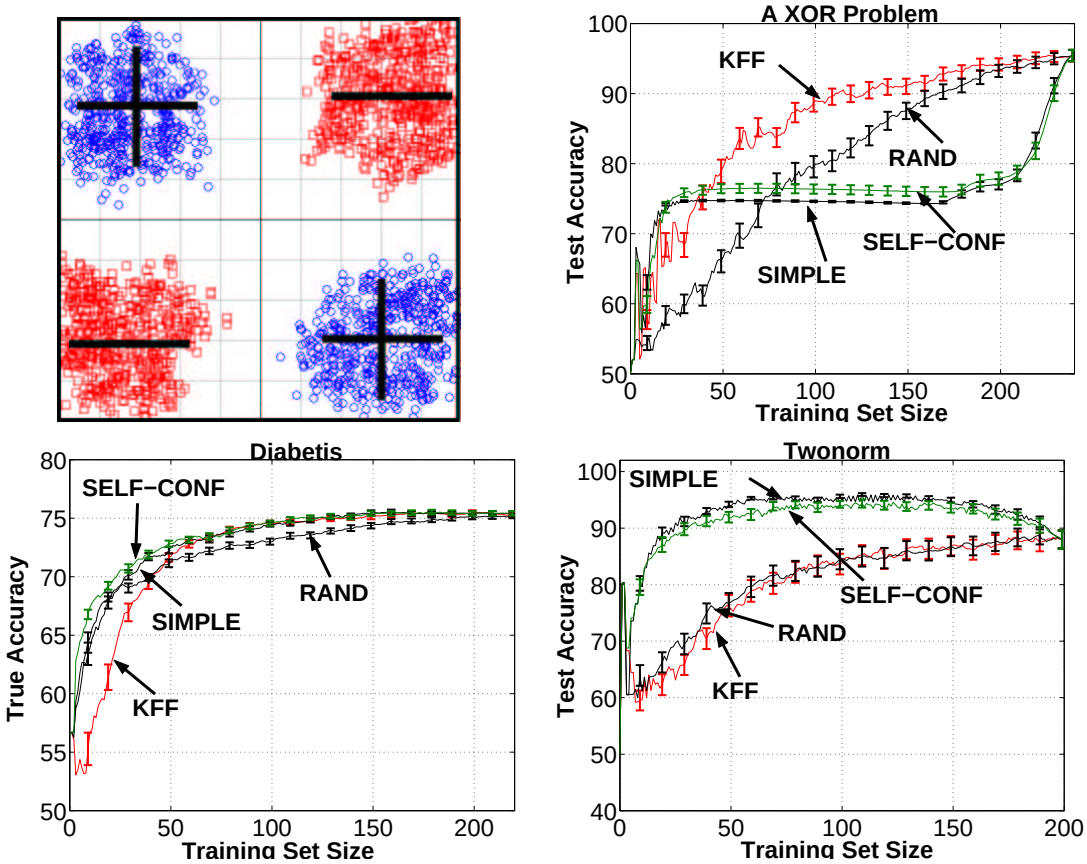


Figure 2: **Top Left**: 'XOR' problem consisting of 1000 points (250 in each "cluster"); 250 points form the pool and the rest form the test set. **Top Right**: Learning curves for SIMPLE, SELF-CONF, KFF and RAND for the 'XOR' problem where KFF exhibits superior performance. **Bottom left**: Learning curves for the same four learners on the 'Diabetis' data set in which SELF-CONF is superior. **Bottom Right**: Learning curves for the same four learners on the 'Twonorm' data set in which SIMPLE is superior. Each point on each learning curve represents an average over 100 folds (measured over a test set); each error bar represents one standard error of the mean. Error bars are diluted to enhance visibility.

5. The Multi-Armed Bandit (MAB) Problem

A major ingredient in our online learning approach to active learning is a known competitive algorithm for the multi-armed bandit problem. In this section we present this problem, its connection to online choice of active learners and the particular multi-armed bandit algorithm we use.

In the ***multi-armed bandit (MAB) problem***, a gambler must choose one of k non-identical slot machines to play in a sequence of trials. Each machine can yield rewards whose distribution is unknown to the gambler, and the gambler's goal is to maximize his total reward over the sequence. This classic problem represents a fundamental predicament whose essence is the tradeoff between

exploration and *exploitation*: Sticking with any single machine may prevent discovering a better machine; on the other hand, continually seeking a better machine will prevent achieving the best total reward.

We make the following straightforward analogy between the problem of online combining of an ensemble of active learners and the MAB problem. The k active-learning algorithms in our ensemble are the k slot machines. On each trial, choosing a query generated by one active-learning algorithm corresponds to choosing one slot machine. The true (generalization) accuracy achieved (by the combined algorithm) using the augmented training set (which includes the newly queried data point) corresponds to the gain achieved by the chosen machine. Of course, this rough analogy does not immediately provide a solution for combining active learners. In particular, one of the main obstacles in using MAB algorithms for combining active learners is how to define the reward of a query. The optimal reward might be the true accuracy achieved by the learner using the augmented training set, however this accuracy is unknown to us. We therefore have to estimate it in some way. The reward estimation technique we use is developed in Section 6. For the rest of this section we assume we have some (non-negative and bounded) reward function.

Most known MAB algorithms and their analyses assume that rewards are distributed according to certain statistical models. Typically, simple statistical models including independence and time invariance assumptions are taken (for example, i.i.d. Gaussian rewards). In our active-learning context (using, for example, the above analogy), overly simplistic statistical assumptions on reward distributions are likely to be violated. Therefore, it is particularly useful to use, in our active-learning context, the *adversarial* MAB results of Auer et al. (2002), which provide MAB algorithms that are guaranteed to extract a total gain close to that of the best slot machine (in hindsight) without *any* statistical assumptions. Two particular MAB algorithms developed by Auer et al. (2002) are potentially useful for implementing online choice of active learners. The first algorithm is called EXP3 (see Section 3 in Auer et al., 2002) and directly matches the above analogy. The second algorithm, called EXP4 (see Section 7 in Auer et al., 2002), appears to be more suitable for our purposes (see below). We first describe the EXP3 algorithm and then describe the EXP4 algorithm which we have chosen to use.

EXP3 is an algorithm for the standard MAB game, where n slot machines are played. On each trial t, one slot machine $i, i = 1,\ldots,n$, is chosen and played, yielding a reward $g_i(t)$, where $g_i(t)$ is non-negative and bounded. For a game consisting of T trials, define $G_{\text{MAX}} = \max_{1\le i\le n}\sum_{t=1}^{T} g_i(t)$, the expected reward of the best slot machine in hindsight. The goal of the online player in this game is to achieve a reward as close as possible to G_{MAX}. Let G_{EXP3} be the expected reward of this algorithm. The "regret" of EXP3 is defined to be $G_{\text{MAX}} - G_{\text{EXP3}}$. Given an upper bound $g \ge G_{\text{MAX}}$, it is guaranteed (Auer et al., 2002) that the regret of EXP3 is bounded above by

$$G_{\text{MAX}} - G_{\text{EXP3}} \le 2.63\sqrt{gn\ln n}.$$

This bound holds for any assignment of rewards and for any number of trials T. We also note that this bound holds for rewards in the range of $[0,1]$. In our algorithm we also used rewards in this range (see Section 7).

The problem of providing the upper bound $g \ge G_{\text{MAX}}$ is solved using a standard doubling algorithm that guesses a bound and runs the EXP3 algorithm until this bound is reached. Once reached, the guessed bound is doubled and EXP3 is restarted with the new bound. The guaranteed bound on the regret of this algorithm is

$$G_{\text{MAX}} - G_{\text{DBL-EXP3}} \le 10.5\sqrt{G_{\text{MAX}}n\ln n} + 13.8n + 2n\ln n,$$

where $G_{\text{DBL-EXP3}}$ is the expected reward of the doubling algorithm. Once again, this bound holds for any reward function and for any number of trials.

Combining active learners using EXP3 can be done in the following manner. Each one of the n active-learning algorithms in the ensemble is modelled as a slot machine. On each trial t, one active learner is chosen to provide the next query. The reward of this query is associated with the chosen learner.[17] In this modeling approach, the points are not at all considered by EXP3 and each active learner is a slot machine "black box". Thus, this approach does not directly utilize some useful information on the unlabeled points that may be provided by the active learners.

The other algorithm of Auer et al. (2002) called EXP4 is designed to deal with a more sophisticated MAB variant than the above (standard) MAB game. Here the goal is to combine and utilize a number k of *strategies* or *experts*, each giving "advice" on how to play the n slot machines. To employ EXP4 in our context, we associate the k experts with the ensemble of k active-learning algorithms. The slot machines are associated with the unlabeled instances in our pool $\mathcal{U}$. Using this approach for modeling active learning has a considerable benefit since now the choice of the next query is directly based on the opinions of all ensemble members.

Algorithm EXP4 operates as follows. On each trial t, each expert j, $j = 1,\ldots,k$, provides a weighting $\mathbf{b}^j(t) = (b_1^j(t),\ldots,b_n^j(t))$ with $\sum_i b_i^j(t) = 1$, where $b_i^j(t)$ represents the confidence (probability) of expert j for playing the ith machine, $i = 1,\ldots,n$, on trial t. Denoting the vector of rewards for the n machines on trial t by $\mathbf{g}(t) = (g_1(t),\ldots,g_n(t))$, where $g_i(t)$ is non-negative and bounded, the expected reward of expert j on trial t is $\mathbf{b}^j(t)\cdot\mathbf{g}(t)$. Note that in the MAB game only one reward from $\mathbf{g}(t)$ is revealed to the online player after the player chooses one machine in trial t. For a game consisting of T trials, define $G_{\text{MAX}} = \max_{1\le j\le k}\sum_{t=1}^{T}\mathbf{b}^j(t)\cdot\mathbf{g}(t)$, the expected reward of the best *expert* in hindsight. The goal of the online player in this game is to utilize the advice given by the experts so as to achieve reward as close as possible to G_{MAX}. Let G_{EXP4} be the expected reward of this algorithm. Given an upper bound $g \ge G_{\text{MAX}}$, the "regret" of EXP4, defined to be $G_{\text{MAX}} - G_{\text{EXP4}}$, is bounded above by

$$G_{\text{MAX}} - G_{\text{EXP4}} \le 2.63\sqrt{gn\ln k}.$$

This regret bound holds for any number of trials T, provided that one of the experts in the ensemble is the "uniform expert," which always provides the uniform confidence vector for n slot machines. The problem of providing the upper bound $g \ge G_{\text{MAX}}$ is solved by employing the same 'guess and double' technique used with EXP3. The guaranteed bound on the regret of this algorithm is

$$G_{\text{MAX}} - G_{\text{DBL-EXP4}} \le 10.5\sqrt{G_{\text{MAX}}n\ln k} + 13.8n + 2n\ln k,$$

where $G_{\text{DBL-EXP4}}$ is the expected reward of the doubling algorithm.

As mentioned above, we use EXP4 for active learning by modeling the k active-learning algorithms as k experts and the unlabeled pool points in $\mathcal{U}$ as the slot machines. Note that for the performance bound of EXP4 to hold we must include RAND in our pool of active learners, which corresponds to the "uniform expert". This modeling requires that expert advice vectors are probabilistic recommendations on points in $\mathcal{U}$. It is thus required that each algorithm in the ensemble will provide "rating" for the entire pool on each trial. In practice, the three algorithms we consider naturally provide such ratings, as discussed in Section 7.

17. The alternative possibility of modeling the pool samples as the slot machines clearly misses the 'exploitation' dimension of the problem since once we choose a sample, we cannot utilize it again. In addition, it is not clear how to utilize an ensemble of learners using this approach.

6. Classification Entropy Maximization

In order to utilize the MAB algorithm (EXP4, see Section 5) in our context, we need to receive, after each trial, the gain of the instance $x \in \mathcal{U}$ that was chosen to be queried (corresponding to one slot machine). While the ultimate reward in our context should be based on the *true* accuracy of the classifier resulting from training over $\mathcal{L} \cup \{(x,y)\}$ (where y is the label of x), this quantity is not available to us. At the outset it may appear that standard error (or accuracy) estimation methods could be useful. However, this is not entirely the case and, unless $\mathcal{L}$ (and $\mathcal{U}$) is sufficiently large, standard methods such as cross-validation or leave-one-out provide substantially biased estimates of an active learner's performance. This fact, which was already pointed out by Schohn and Cohn (2000), is a result of the biased sample acquired by the active learner. In order to progress quickly, the learner must focus on "hard" or more "informative" samples. Consider Figure 3 (left). The figure shows leave-one-out (LOO) estimates of four (active) learning algorithms: SIMPLE, SELF-CONF, KFF and RAND on the UCI 'Ringnorm' data set. For each training set size, each point on a curve is generated using the LOO estimate based on the currently available labeled set $\mathcal{L}$. In Figure 3 (middle) we see the "true accuracy" of these algorithms as estimated by an independent test set. Not only does LOO severely fail to estimate the true accuracy, it even fails to order the algorithms according to their relative success as active learners. This unfortunate behavior is quite typical of LOO on many data sets, and afflicts other standard error estimation techniques, including cross-validation. These techniques should, therefore, be avoided in general for receiving feedback on the (relative) progress of active learners, especially if one cares about the active learner's performance using a small number of labeled points.

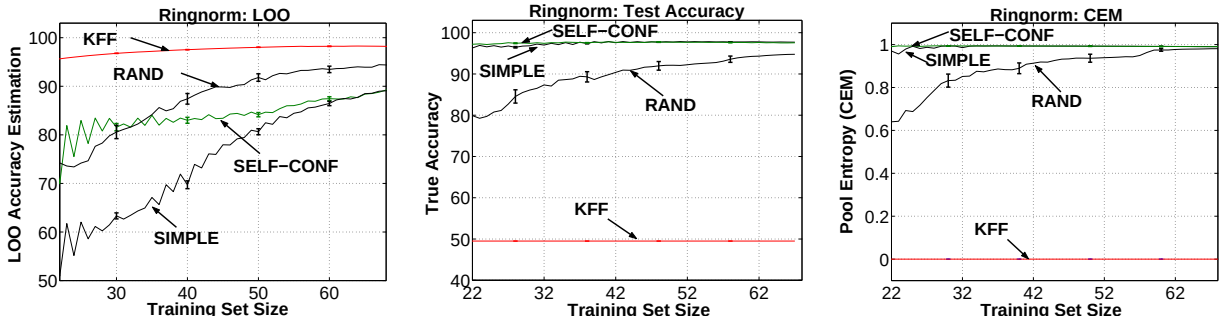


Figure 3: **Left**: LOO estimates of active-learning sessions of SIMPLE, SELF-CONF, RAND and KFF over the 'Ringnorm' data set; **Middle**: The "true" accuracy of these algorithms, as estimated using a test set; **Right**: CEM entropy scores of these algorithms. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

We propose the following semi-supervised estimator, which we call *Classification Entropy Maximization (CEM)*. Define the *CEM score* of a classifier with respect to an unlabeled set of points to be the binary entropy of the classification it induces on the unlabeled set. Specifically, if C is a binary classifier giving values in $\{\pm 1\}$, let $C^{+1}(\mathcal{U})$ and $C^{-1}(\mathcal{U})$ be the 'positively' and 'negatively' classified subsets of some unlabeled set $\mathcal{U}$, respectively, as determined by C. Then, the CEM score

of C (with respect to $\mathcal{U}$) is the binary entropy $H(\frac{|C^{+1}(\mathcal{U})|}{|\mathcal{U}|})$.[18] It is not difficult to see that the CEM score is larger if the division of the pool classification between classes is more balanced. Figure 3 (right) provides CEM curves for the four active learners discussed above. Clearly, the CEM measure orders the algorithms in accordance with their true accuracy as depicted in Figure 3 (middle), and moreover, if we ignore the scales, all CEM curves in this example appear surprisingly similar to the corresponding true accuracy curves. This behavior of CEM is typical in many of our empirical examinations, and somewhat surprisingly, CEM succeeds in correctly evaluating performance even when the positive and negative priors are not balanced. A more comprehensive discussion and analysis of the CEM criterion is provided later in Section 9.

Remark 1 *It is interesting to note that the CEM criterion is also useful for SVM model selection in (semi-)supervised learning settings where the (labeled) training set is small and some set of unlabeled points is available. In Appendix B we present an empirical comparison of CEM and LOO in this setting.*

7. Combining Active Learners Online

In this section we describe our master algorithm for combining active learners. The combination algorithm, called here for short COMB, is based on the EXP4 multi-armed bandit (MAB) algorithm discussed in Section 5 and on the CEM criterion presented in Section 6. In Figure 4 we provide an annotated pseudocode of COMB. The algorithm utilizes an ensemble of active-learning algorithms and tracks online the best algorithm in the ensemble. Many of the steps in this code are adapted from the EXP4 MAB algorithm of Auer et al. (2002) (in particular, steps 4,5,6,11,12). We refer the reader to Auer et al. (2002) for a more detailed exposition (and the proof of the performance guarantee) of EXP4. Here we elaborate on steps 1,2,3 and 10, which require further explanation (the remaining steps, 7,8 and 9, are self-explanatory).

In steps 1 and 2, we compute advice probability vectors of the active learners as required by the original EXP4 algorithm. Each algorithm in the ensemble provides (in Step 1) a scoring vector that "rates" each point in the pool $\mathcal{U}$. In practice, the three algorithms we consider naturally provide such ratings: SIMPLE uses the kernel distance from the decision hyperplane, SELF-CONF uses the expected loss and KFF uses the kernel distance from the current training set. We scale these scoring vectors (in Step 2) using a scaling parameter β and a Gibbs probability function $\exp\{-\beta x\}$. In all our experiments we used $\beta = 100$ (a sensitivity analysis of this parameter is provided in Section 8).

In Step 3, after producing (in Step 2) the advice probability vectors for the active learners, we project the pool $\mathcal{U}$ over high probability candidate instances. The projected pool is denoted $\mathcal{U}_e$. This projection is controlled by the parameter α and an instance x in $\mathcal{U}$ remains in $\mathcal{U}_e$ if at least one active learner assigns to x a probability mass greater than α. In all the experiments described below, we used $\alpha = 0.05$ (here again, a sensitivity analysis of this parameter is provided in Section 8).

In Step 6, the learner chooses one (unlabeled) point x_q from $\mathcal{U}_e$ as the next query. According to EXP4, this choice should be random according to the distribution computed in Step 5. In practice, in the experiments described in Section 8, we greedily picked the point with the largest probability

18. The binary entropy of a (Bernoulli) random variable with bias p is $H(p) = H(1-p) = -p\log(p) - (1-p)\log(1-p)$.

Algorithm COMB

Input: (i) A Pool $\mathcal{U} = \{x_1, \ldots, x_n\}$; (ii) An ensemble $\{\text{ALG}_j\}_{j=1}^k$ of k active learners;
(iii) An initial training set $\mathcal{L}_0$
Parameters: (i) A probability threshold α; (ii) A probability scaling factor β;
(iii) A bound g_{max} on the maximal reward
Init: Initialize expert weights: $w_j = 1$, $j = 1, \ldots, k$

For $t = 1, 2, \ldots$

1. Receive advice scoring vectors from ALG_j, $j = 1, \ldots, k$: $\mathbf{e}^j(t) = (e_1^j(t), \ldots, e_n^j(t))$. $e_i^j(t)$ is the score of the ith point in the pool. The scores are normalized to lie within [0,1].
2. For each ALG_j, $j = 1, \ldots, k$, we compute an advice probability vector $\mathbf{b}^j(t) = (b_1^j(t), \ldots, b_n^j(t))$ by scaling the advice scoring vectors. For each $\mathbf{b}^j(t)$, $j = 1, \ldots, k$, for each $b_i^j(t)$, $i = 1, \ldots, n$: $b_i^j(t) = (\exp\{-\beta(1 - e_i^j(t))\})/Z$, where Z normalizes $\mathbf{b}^j(t)$ to be a probability vector.
3. Extract from $\mathcal{U}$ an "effective pool" $\mathcal{U}_e$ by thresholding low probability points: For each point $x_i \in \mathcal{U}$ leave x_i in $\mathcal{U}_e$, iff $\max_j b_i^j \geq \alpha$. If $|\mathcal{U}_e| = 0$, reconstruct with $\alpha/2$, etc. Set $n_e = |\mathcal{U}_e|$.
4. Set $\gamma = \sqrt{\frac{n_e \ln k}{(e-1)g_{max}}}$.
5. Set $W = \sum_{j=1}^k w_j$ and for $i = 1, \ldots, n_e$, set $p_i = (1-\gamma)\sum_{j=1}^k w_j b_i^j(t)/W + \gamma/n_e$.
6. Randomly draw a point x_q from $\mathcal{U}_e$ according to $p_1, \ldots, p_{n_e}$.
7. Receive the label y_q of x_q from the teacher and update the training set and the pool: $\mathcal{L}_t = \mathcal{L}_{t-1} \cup \{(x_q, y_q)\}$; $U_{t+1} = U_t \setminus \{x_q\}$
8. Train a classifier C_t using $\mathcal{L}_t$.
9. Use C_t to classify all points in $\mathcal{U}$ and calculate $H_t = H_t(\frac{|C^{+1}(\mathcal{U})|}{|\mathcal{U}|})$, the entropy of the resulting partition $C^{+1}(\mathcal{U}), C^{-1}(\mathcal{U})$ (as in the CEM score; see Section 6).
10. Calculate the "reward utility" of x_q:
$r(x_q) = ((e^{H_t} - e^{H_{t-1}}) - (1-e))/(2e-2)$.
11. For $i = 1, \ldots, n$, set $\hat{r}_i(t) = r(x_q)/p_q$ if $i = q$ and $\hat{r}_i(t) = 0$ otherwise.
12. Reward/punish experts:
$w_j(t+1) = w_j(t)\exp(\mathbf{b}^j(t) \cdot \hat{r}(t)\gamma/n_e)$.

Figure 4: Algorithm COMB

(and in case of ties we randomly chose one of the points). This deterministic implementation is useful for reducing the additional variance introduced by these random choices.[19]

In Step 10 we calculate the "utility" of the last query. This utility is defined using the (convex) function e^x on the entropic reward calculated in Step 9 (that is, the CEM score discussed in

19. We also experimented with random query choices, as prescribed by EXP4. A slight advantage of the deterministic variant of the algorithm was observed.

Section 6). The utility function is essentially the difference

$$\Delta_t = e^{H_t} - e^{H_{t-1}},$$

where H_t is the entropy of the last partition of $\mathcal{U}$ generated using the last queried instance in the training set. H_{t-1} is the entropy of the partition of the same pool generated without using the last queried instance. Clearly, this function emphasizes entropy changes in the upper range of possible entropy values. The rationale behind this utility function is that it is substantially harder to improve CEM scores (and also accuracy) that are already large. The additional transformation $(\Delta_t - (1-e))/(2e-2)$ normalizes the utility to be in [0,1].

The reward bound parameter g_{max}, used for setting an optimal value for γ (Step 4) can be and is eliminated in all our experiments using a standard "guess and double" technique. In particular we operate the COMB algorithm in rounds $r = 1, 2, \ldots$, where in round r we set the reward limit to be $g_r = (n_e \ln k/(e-1))4^r$ and restart the COMB algorithm with $g_{max} = g_r$. The round continues until the maximal reward reached by one of the ensemble algorithms exceeds $g_r - n_e/\gamma_r$. For more details, the reader is referred to the discussion on the EXP3.1 algorithm in Section 4 of Auer et al. (2002).

8. Empirical Evaluation of the COMB Algorithm

We evaluated the performance of the COMB algorithm on the entire benchmark collection selected and used by Rätsch et al. (2001), consisting of 13 binary classification problems extracted from the UCI repository. For almost all problems, this collection includes fixed 100 folds each consisting of a fixed 60%/40% training/test partition.[20] The use of this set is particularly convenient as it allows for easier experimental replication. To this collection we also added our artificial 'XOR' problem (see Section 4.4).[21] Some essential characteristics of all these data sets appear on Table 1. For each problem we specify its size, its dimension, the bias (proportion of largest class), the maximal accuracy achieved using the entire pool as a training set, the (rounded up) number of instances required by the *worst* learner in the ensemble to achieve this maximal accuracy and, finally, the average fraction of support vectors (from the entire pool size) utilized by an SVM trained over the entire pool (this average is computed over 100 folds).

In all the experiments described below, each active learner is provided with two initial examples (one from each class) for each learning problem. These two examples were randomly chosen among all possible pairs. The same initial training set (pair) was given to all learners. All active-learning algorithms applied in our experiments used an SVM with RBF kernel as their learning component ($\mathcal{A}$). More particular implementation details, which are essential for replication, are given in Appendix A.

In Figure 5 we depict the learning curves of COMB and its ensemble members obtained on four data sets. Notice that for three of these data sets a different ensemble member is the winner. In the 'Image' data set it is SELF-CONF, in the 'XOR' data set it is KFF and in the 'Waveform' data set it is SIMPLE.[22] However, in all of these cases COMB tracks the winner. In the fourth data set

20. Two data sets in this collection ('Image' and 'Splice') include only 20 folds.

21. For this 'XOR' data set, consisting of 1000 points, we constructed 100 folds by randomly and uniformly splitting the data to 25%/75% training/test partitions.

22. In the 'Waveform' data set the accuracy decreases from a certain point. When there is noise in the data one may benefit by stopping early. See a discussion of this phenomenon by Schohn and Cohn (2000).

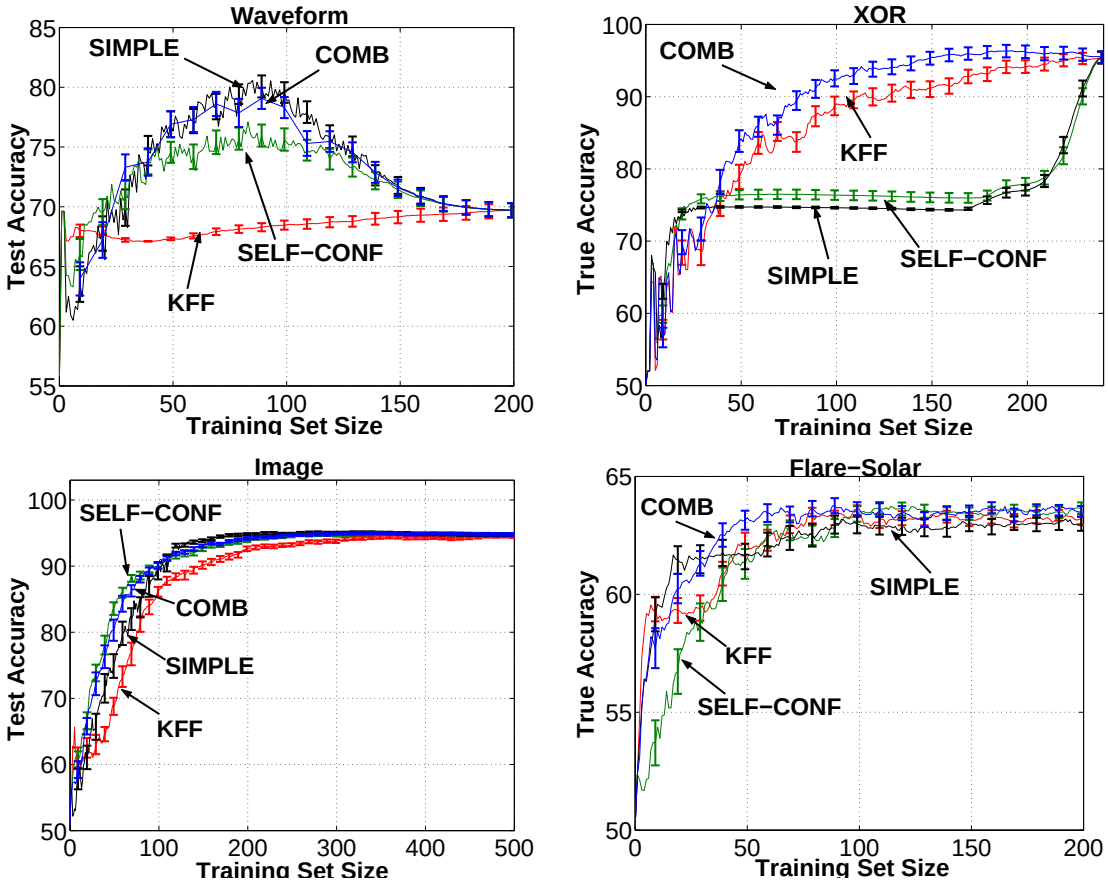


Figure 5: Learning curves of COMB and its ensemble members on four data sets. All estimates are averages over 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean. **Top Left**: 'Waveform' data set, where COMB performs almost as well as the winner SIMPLE. **Top Right**: 'XOR' data set, where COMB performs better than the winner KFF. **Bottom Left**: 'Image' data set, where COMB performs almost as well as the winner SELF-CONF. **Bottom Right**: 'Flare-Solar' data set, where COMB is the overall winner and significantly beats its ensemble members.

presented, the 'Flare-Solar' data set, COMB is the overall winner and significantly beats its three ensemble members.

Table 2 shows the average *deficiency* of COMB and its ensemble members for all these data sets. Recall the definition of deficiency of an active learner as given in Equation (1), where smaller values represent a larger active-learning *efficiency*. Each of these averages is calculated over the corresponding 100 folds. It is evident that none of the ensemble algorithms is consistently outdoing all sets (SELF-CONF, SIMPLE, and KFF win in 7, 4 and 3 cases, respectively). Nevertheless, SELF-CONF is the most dominant algorithm, winning on half of the data sets. In many cases KFF performs poorly and it is often inferior to RAND. Overall, this algorithm is significantly worse than

Data Set	Size	Dim	Bias	Max Acc. (%) (Sample Size)	SV proportion (%)
Banana	5300	2	0.551	88.22 (200)	14.3 ± 0.2
Breast-Cancer	277	9	0.707	73.60 (100)	45.8 ± 1.07
Diabetis	768	8	0.651	74.56 (220)	38.5 ± 1.11
Flare-Solar	1066	9	0.552	63.76 (200)	15.08 ± 0.4
German	1000	20	0.700	75.48 (330)	53.00 ± 0.56
Heart	270	13	0.555	82.33 (82)	41.29 ± 0.68
Image	2310	18	0.571	95.00 (500)	46.32 ± 1.17
Ringnorm	8300	20	0.549	97.96 (200)	79.0 ± 0.3
Splice	3175	60	0.519	85.86 (450)	74.1 ± 2.13
Thyroid	215	5	0.697	95.53 (69)	26.2 ± 0.8
Titanic	2201	3	0.676	74.14 (74)	17.09 ± 0.32
Twonorm	7400	20	0.500	95.94 (200)	87.45 ± 0.81
Waveform	5000	21	0.670	80.58 (200)	97.78 ± 0.55
XOR	3000	2	0.500	96.35 (240)	2.19 ± 0.01

Table 1: The data sets: some essential characteristics. For each problem we provide the size, the dimension, the bias (proportion of largest class), the maximal accuracy achieved using the entire pool, the (rounded up) number of instances required by the worst learner in the ensemble to achieve this maximal accuracy and the average fraction of the number of support vectors (from the pool size) utilized by an SVM trained over the entire pool (the average is computed over 100 folds).

Data Set	SIMPLE	KFF	SELF-CONF	COMB
Banana	1.13 ± 0.02	$\mathbf{0.73}* \pm 0.01$	$\mathbf{0.74} \pm 0.02$	$\mathbf{0.74} \pm 0.01$
Breast-Cancer	$\mathbf{1.06} \pm 0.02$	1.28 ± 0.03	$\mathbf{0.95}* \pm 0.03$	1.09 ± 0.01
Diabetis	$\mathbf{0.64} \pm 0.04$	1.07 ± 0.02	$\mathbf{0.48}* \pm 0.05$	0.82 ± 0.04
Flare-Solar	1.13 ± 0.01	$\mathbf{1.09} \pm 0.05$	1.39 ± 0.07	$\mathbf{0.79}* \pm 0.05$
German	0.71 ± 0.04	0.85 ± 0.01	$\mathbf{0.67} \pm 0.02$	$\mathbf{0.64}* \pm 0.02$
Heart	$\mathbf{0.57} \pm 0.03$	1.04 ± 0.01	$\mathbf{0.50}* \pm 0.03$	0.64 ± 0.02
Image	0.54 ± 0.02	0.76 ± 0.01	$\mathbf{0.45}* \pm 0.01$	$\mathbf{0.47} \pm 0.01$
Ringnorm	$\mathbf{0.34}* \pm 0.01$	4.70 ± 0.4	0.38 ± 0.01	$\mathbf{0.36} \pm 0.01$
Splice	$\mathbf{0.58} \pm 0.01$	2.54 ± 0.11	0.60 ± 0.01	$\mathbf{0.57}* \pm 0.01$
Thyroid	$\mathbf{0.55} \pm 0.01$	2.34 ± 0.09	$\mathbf{0.47}* \pm 0.01$	0.64 ± 0.03
Titanic	0.76 ± 0.04	0.92 ± 0.02	$\mathbf{0.68} \pm 0.06$	$\mathbf{0.65}* \pm 0.05$
Twonorm	$\mathbf{0.24}* \pm 0.01$	1.03 ± 0.01	0.32 ± 0.02	$\mathbf{0.26} \pm 0.01$
Waveform	$\mathbf{0.58}* \pm 0.05$	0.97 ± 0.01	0.66 ± 0.04	$\mathbf{0.60} \pm 0.05$
XOR	1.24 ± 0.07	$\mathbf{0.63} \pm 0.02$	1.19 ± 0.04	$\mathbf{0.47}* \pm 0.03$

Table 2: Average deficiency ($\pm$ standard error of the mean) achieved by COMB and its ensemble members. For each data set the winner appears in boldface and is marked with a star. The runner-up appears in boldface.

all the other algorithms. However, as noted above, KFF usually excels in "XOR-like" problems (for example, 'Banana', 'Flare-Solar' and 'XOR').

In 10 cases out of the 14 presented, COMB is either the winner or a close runner-up. In five cases out of these 10 ('Flare-Solar', 'German', 'Splice', 'Titanic' and 'XOR') COMB is the overall winner. A striking feature of COMB is that it does not suffer from the presence of KFF in the ensemble even in cases where this algorithm is significantly worse than RAND, and can clearly benefit from KFF in those cases where KFF excels.

In three cases ('Heart', 'Thyroid' and 'Breast-Cancer'), which are among the smallest data sets, COMB is not the winner and not even the runner-up, although even in these cases it is not far behind the winner. This behavior is reasonable because COMB needs a sufficient number of trials for exploration and then a sufficient number of trials for exploitation. There is only one case, the 'Diabetis' data set, in which COMB completely failed. A closer inspection of this case revealed that our entropy criterion failed in the online choosing of the best ensemble member.

We now turn to examine the possibility of using standard error estimation techniques instead of our CEM. In Table 3 we compare the deficiencies of COMB using (for computing the query rewards) both CEM and standard 10-fold cross validation (10-CV) on the training set. The 10-CV results are obtained by running the COMB algorithm such that in Step 9 of the pseudocode in Figure 4 we calculate the 10-CV of the current training set $\mathcal{L}_t$.[23] Denote this quantity by CV_t. Then in Step 10 we used the same utility function (applied on the CEM estimator) over the CV outcome, $r(x_q) = ((e^{CV_t} - e^{CV_{t-1}}) - (1-e))/(2e-2)$. The table indicates that CEM is more reliable than 10-CV as an online estimator for active learners' performance. 10-CV is very unreliable as it performs well in some cases and very poorly in others. In particular, CEM beats 10-CV on 12 out of 14 data sets, where in some data sets ('Ringnorm' and 'Heart' for example) the difference is quite large. The 10-CV estimator outperforms CEM on two data sets, 'Titanic' and 'Breast-Cancer', where in the first one the difference is small.

Data Set	COMB using CEM	COMB using 10-CV
Banana	$\mathbf{0.74} \pm 0.01$	0.78 ± 0.01
Breast-Cancer	1.09 ± 0.01	$\mathbf{0.99} \pm 0.05$
Diabetis	$\mathbf{0.82} \pm 0.04$	0.88 ± 0.04
Flare-Solar	$\mathbf{0.79} \pm 0.05$	0.90 ± 0.02
German	$\mathbf{0.64} \pm 0.02$	0.66 ± 0.03
Heart	$\mathbf{0.64} \pm 0.02$	0.93 ± 0.01
Image	$\mathbf{0.47} \pm 0.01$	0.48 ± 0.01
Ringnorm	$\mathbf{0.36} \pm 0.01$	0.68 ± 0.03
Splice	$\mathbf{0.57} \pm 0.01$	0.63 ± 0.01
Thyroid	$\mathbf{0.64} \pm 0.03$	0.74 ± 0.03
Titanic	0.65 ± 0.05	$\mathbf{0.61} \pm 0.05$
Twonorm	$\mathbf{0.26} \pm 0.01$	0.45 ± 0.01
Waveform	$\mathbf{0.60} \pm 0.05$	0.79 ± 0.06
XOR	$\mathbf{0.47} \pm 0.03$	0.53 ± 0.02

Table 3: COMB's *deficiency* when operated using CEM and 10-fold cross-validation (10-CV) for computing the query rewards. For each data set the winner appears in boldface.

As defined, the COMB algorithm has two parameters (see Figure 4): the probability threshold α, and the probability scaling factor β. The experimental results presented above were obtained using

23. For training sets of size smaller than 20, we used a leave-one-out (LOO) estimator.

the assignments $\alpha = 0.05$ and $\beta = 100$. These assignments were not optimized and in fact were a priori set to these values, which appeared "reasonable" to us. In the rest of this section we provide a sensitivity analysis of these parameters. This analysis indicates that better performance may be obtained by optimizing the β parameter.

In Table 4 we show a comparison of COMB's performance obtained with different values of the probability threshold parameter α (which determines the size of the "effective pool" computed in Step 3 of the pseudo-code in Figure 4). For the presentation here we selected the four data sets appearing in Figure 5. Recall that each of the first three data sets favors (using our "standard" $\alpha = 0.05$ value) a different ensemble member and in the fourth one (data set 'Flare-Solar'), COMB significantly beats its ensemble members. When applying COMB with $\alpha = 0.01, 0.05, 0.1$ to these four data sets, there is no significant difference in the deficiencies demonstrated by COMB (see Table 4). This result indicates that COMB is not overly sensitive to the choices of α values within this range.

Data Set	COMB with $\alpha = 0.05$ (used in this paper)	COMB with $\alpha = 0.01$	COMB with $\alpha = 0.1$
Flare-Solar	$\mathbf{0.79} \pm 0.05$	0.83 ± 0.02	0.80 ± 0.04
Image	0.47 ± 0.01	0.47 ± 0.01	$\mathbf{0.46} \pm 0.01$
Waveform	0.60 ± 0.05	$\mathbf{0.59} \pm 0.05$	0.61 ± 0.05
XOR	0.47 ± 0.03	$\mathbf{0.45} \pm 0.03$	0.48 ± 0.03

Table 4: COMB's *deficiency* with different values of the probability threshold parameter α (see Figure 4). For each data set the winner appears in boldface.

In Table 5 we show a comparison of COMB performance with three values of the probability scaling factor parameter β (see Step 2 in Figure 4). Here, again, we use the same four data sets presented in Table 4, to check the sensitivity of COMB with respect to this β parameter (recall that in the experiments above we use $\beta = 100$). Observing the deficiency of COMB operated with $\beta = 10$ and $\beta = 1000$, we see that the performance is dependent on this parameter. However, it is clear that the value we use in the experiments above ($\beta = 100$) was not optimized and the table indicates that $\beta = 10$ yields better results. Hence, here there is room for further improvements, which we have not pursued in this paper.

Data Set	COMB with $\beta = 100$ (used in this paper)	COMB with $\beta = 10$	COMB with $\beta = 1000$
Flare-Solar	0.79 ± 0.05	$\mathbf{0.68} \pm 0.08$	0.80 ± 0.04
Image	0.47 ± 0.01	0.48 ± 0.01	$\mathbf{0.45} \pm 0.01$
Waveform	$\mathbf{0.60} \pm 0.05$	$\mathbf{0.60} \pm 0.05$	0.63 ± 0.05
XOR	0.47 ± 0.03	$\mathbf{0.39} \pm 0.03$	0.67 ± 0.02

Table 5: COMB's *deficiency* with different values of β, the probability scaling factor parameter (see Figure 4). For each data set the winner appears in boldface.

9. On the CEM Criterion

While formal connections between CEM and generalization are currently unknown, the rather informal discussion in this section provides further insights into CEM and attempts to characterize conditions for its effectiveness.

A sequence of sets $S_1, S_2, \ldots$ is called an *inclusion sequence* if $S_1 \subset S_2 \subset, \cdots$. Consider an inclusion sequence of training sets. The sequence of training sets generated by an active learner is an inclusion sequence in which S_1 is the initial training set given to the learner. Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be any binary labeled set of samples where one of the classes (either $+1$ or -1) is a majority and its empirical proportion is r (that is, the size of the majority class over m is r). Consider any classifier C giving the label $C(x)$ for $x \in S$. We say that the classification $S_C = (x_1, C(x_1)), \ldots, (x_m, C(x_m))$ is *majority-biased* (with respect to S) if the majority class in S is also a majority class in S_C and its proportion in S_C is larger than or equal to r. Let $I = S_1 \subset \cdots \subset S_T$ be an inclusion sequence of labeled samples. We say that a *learning algorithm* ALG is *majority-biased* (with respect to I) if the classification of S_T by each of the classifiers $C_1, \ldots, C_T$ (induced by ALG) is majority-biased, where C_i is induced by ALG using S_i as a training set.

Our main (empirical) observation is that whenever the learning algorithm is majority biased with respect to the inclusion sequence of training sets (generated by the learner), CEM's growth rate corresponds to the growth rate of the true accuracy, in which case the CEM criterion can be used to compare online the performance of active learners. In other words, by comparing the growth of a learner's pool classification entropy, we can get a useful indication on the growth of the true accuracy. We next consider a couple of examples that demonstrate this behavior and then consider a negative example in which the majority-bias property does not hold and CEM fails. All our examples consider a setting in which the prior of the majority class is significantly larger than that of the other class.

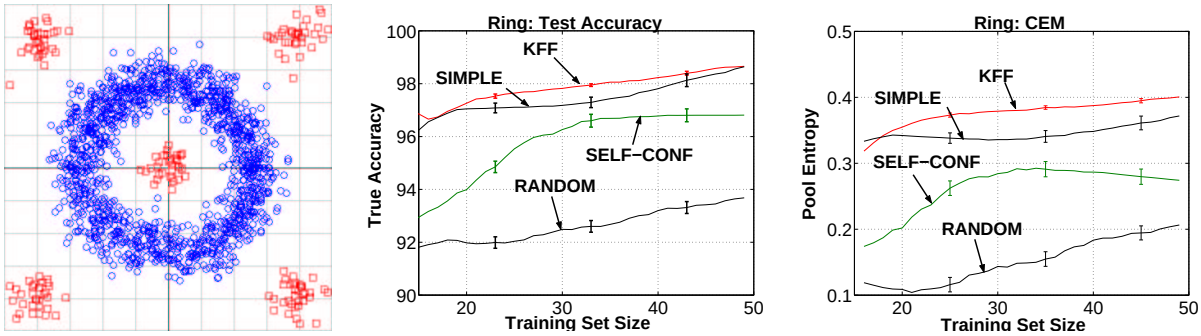


Figure 6: **Left**: A synthetic 'Ring' problem consisting of 3500 points, 90% (3150 circles) in the large 'ring' cluster, and 10% equally divided over the five small clusters (70 squares in each cluster); 1500 points were taken (uniformly at random) to form the pool and the rest form the test set. **Middle**: True accuracy curves of four active learners - KFF, SIMPLE, SELF-CONF and RAND on this 'Ring' data set as estimated on an independent test set. **Right**: Corresponding pool classification entropies (CEM values) of these four learners. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

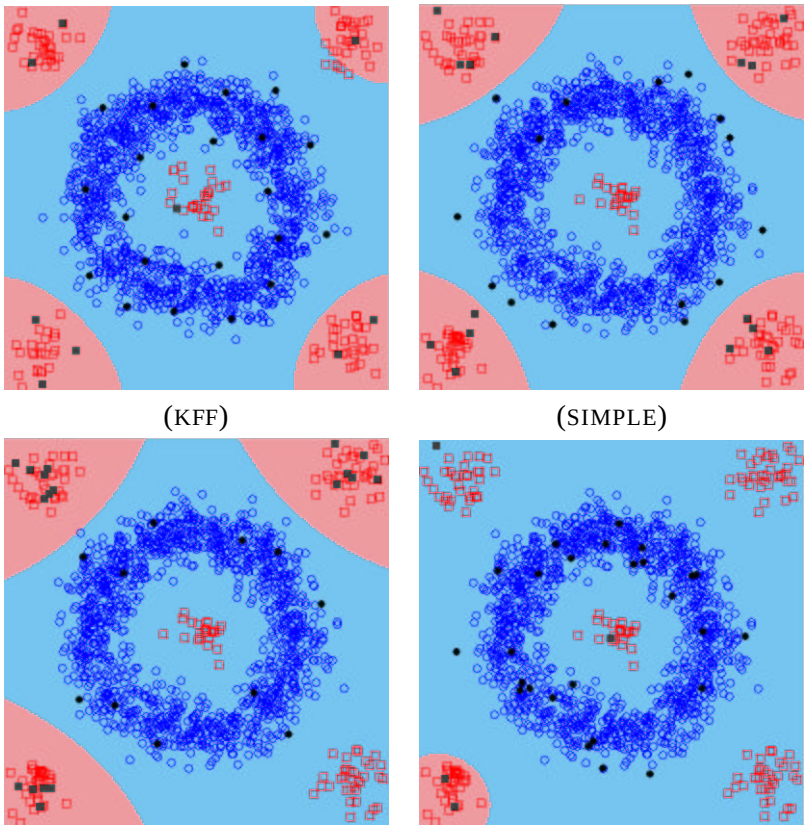


Figure 7: Decision boundaries generated by four active learners using 30 queries on the 'Ring' example of Figure 6 (left). The 30 queries are darker than the pool points.

Figure 6 (left) depicts a synthetic 'Ring' learning problem whose majority class is the 'ring' (blue circles) and its other (minority) class consists of the five small clusters (red squares). The class proportion in this data set is $r = 0.9$ for the 'ring' majority class. We ran the four active learners on this data set. Consider Figure 7 showing the pool classification (decision boundaries) and the training sets chosen by these four learners after 30 active-learning iterations. In this example all pool classifications are majority-biased. Intuitively, the progress of an active learner corresponds to how fast the learner discovers the small clusters and their boundaries. The better active learner, KFF, discovered all five red clusters while the second best learner, SIMPLE, has found four. SELF-CONF has found only three clusters, and RAND found two. We can see a clear correspondence between the pool classification entropy (CEM) and the true accuracy of the learners. Fast exploration of the minority class clusters corresponds to fast entropy maximization as well as fast learning (generalization).

The graphs in Figure 6 (middle) plot the true accuracy of the learners as estimated using an independent test set. The graphs in Figure 6 (right) represent the corresponding pool classification entropies (CEM values). The striking similarity between these curves is further exhibited in Table 6

giving the performance of each of the four active learners after 30 queries. Specifically, the table provides for each learner:

(i) The proportion of the majority class in the training subset (consisting of 30 samples). We call this the "training proportion".

(ii) The proportion of the majority class in the pool classification (generated by the classifier trained over the 30 samples). We call this the "pool proportion".

(iii) The CEM value (entropy) of the pool classification.

(iv) The "true" accuracy estimated on an independent test set.

As usual, all these numbers are averages over 100 random folds.

	RAND	SIMPLE	KFF	SELF-CONF
Training Proportion	86.38 ± 0.53	51.88 ± 0.24	66.06 ± 0.37	37.91 ± 0.66
Pool Proportion	97.46 ± 0.21	92.75 ± 0.20	92.05 ± 0.07	93.54 ± 0.24
Pool Classification Entropy (CEM)	0.14 ± 0.01	0.34 ± 0.01	0.38 ± 0.01	0.29 ± 0.01
True Accuracy	92.39 ± 0.23	97.15 ± 0.19	97.80 ± 0.05	96.09 ± 0.25

Table 6: Characterization of the state and progress of four active learners after querying 30 samples from the pool of the 'Ring' data set given in Figure 6 (left). Each entry provides the mean (over 100 random folds) and the standard error of this mean.

Although in this example the true proportion of the majority class is $r = 0.9$, some of the learners exhibited a significantly smaller training proportion (for example, SIMPLE sampled almost equally from both classes, and SELF-CONF even favored the minority class). The SVMs, which were induced using these minority-biased training sets, have still generated a *majority*-biased classification. In particular, all generated pool proportions are larger than 0.9. We note that this behavior is typical of SVM inducers but not of other learning algorithms.[24] Clearly, this example shows that the CEM criterion (as measured by the pool proportion) correctly ranks the true accuracy of these four learners.

Our next example is shown in Figure 8 (left). This example can be viewed as an "inverse" of the 'Ring' data set of Figure 6. This data set, called here the 'Satellites' data set, contains five clusters from the majority class (squares) and one small cluster from the minority class (circles). The class proportion in this data set is also $r = 0.9$.

As in the previous example, Figure 9 shows the decision boundaries obtained by the four active learners. Similarly, the learning curves in Figure 8 show the true accuracy (middle) and CEM values (right) and Table 7 is similar to Table 6 of the 'Ring' data set. As in the 'Ring' data set, all pool classifications are majority-biased. However, in contrast to the 'Ring' example, where good performance corresponds to quickly finding the minority clusters, in this example good performance should correspond to quickly finding the boundaries of the single minority cluster. The better active learner, now SIMPLE, has mapped more rapidly more areas of the minority cluster than the other

24. When running C4.5, 5-Nearest Neighbor and Bayes Point Machine (using a kernel perceptron for sampling) on these same samples, the pool proportions obtained are significantly minority-biased.

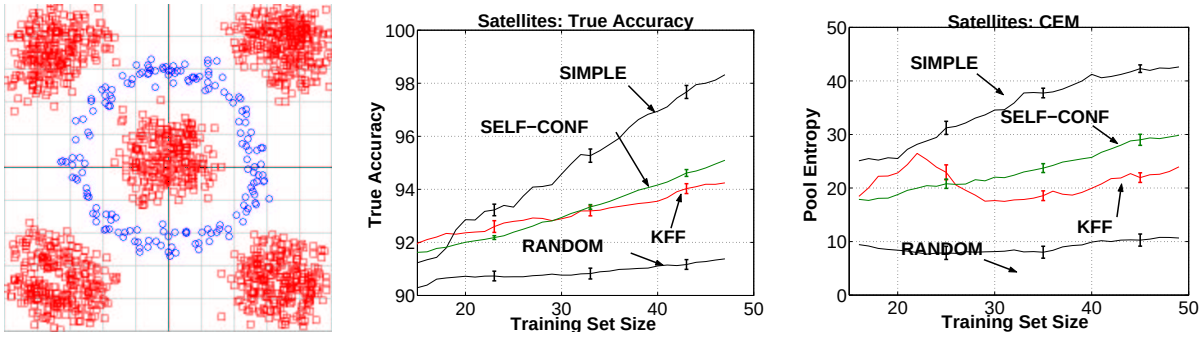


Figure 8: **Left**: The 'Satellites' data set consisting of 3500 points, 10% (350 circles) in the ring "cluster" and 90% equally divided between the five large round clusters (630 squares in each cluster); 1500 points were taken (randomly) to form the pool and the rest form the test set. **Middle**: True accuracy of four active learners - KFF, SIMPLE, SELF-CONF and RAND on this 'Satellites' data set as estimated using an independent test set. **Right**: Corresponding pool classification entropies of these four learners. All estimates are averages of 100 folds. Error bars (diluted to reduce clutter) represent one standard error of the mean.

Data Set 'Satellites'	RAND	SIMPLE	KFF	SELF-CONF
Training Proportion	87.23 ± 0.52	50.97 ± 0.29	74.97 ± 0.30	45.19 ± 0.06
Pool Proportion	98.33 ± 0.33	92.42 ± 0.28	96.67 ± 0.20	95.28 ± 0.17
Pool Classification Entropy (CEM)	0.08 ± 0.01	0.35 ± 0.01	0.18 ± 0.01	0.22 ± 0.01
True Accuracy	90.81 ± 0.19	94.18 ± 0.28	92.80 ± 0.19	92.83 ± 0.10

Table 7: Characterization of the state and progress of four active learners after querying 30 samples from the pool of the 'Satellites' data set of Figure 8 (left). Each entry provides the mean (over 100 random folds) and the standard error of the mean.

learners. Note that RAND treats the entire minority class points as noise. Once again we can see a clear correspondence between the pool classification entropy (CEM) and the true accuracy of the learners where now fast exploration of the minority cluster boundaries corresponds to fast entropy maximization as well as fast learning (generalization).

Here again, although the true proportion is 0.9, some of the learners exhibited a significantly smaller training proportion and still the SVMs, which were induced using these training sets, have generated a majority-biased classification.[25] This example also shows that the CEM criterion correctly identifies the true accuracy of these four learners (where the best learner is SIMPLE, followed by SELF-CONF, KFF and RAND).

Our last example is shown on the left-hand side of Figure 10, is referred to as 'Concentric-Rings'. This data set consists of one very large and one very small ring clusters with a small ring

25. In this example as well, the C4.5, 5-Nearest Neighbor and Bayes Point Machine learning algorithms were not majority-biased.

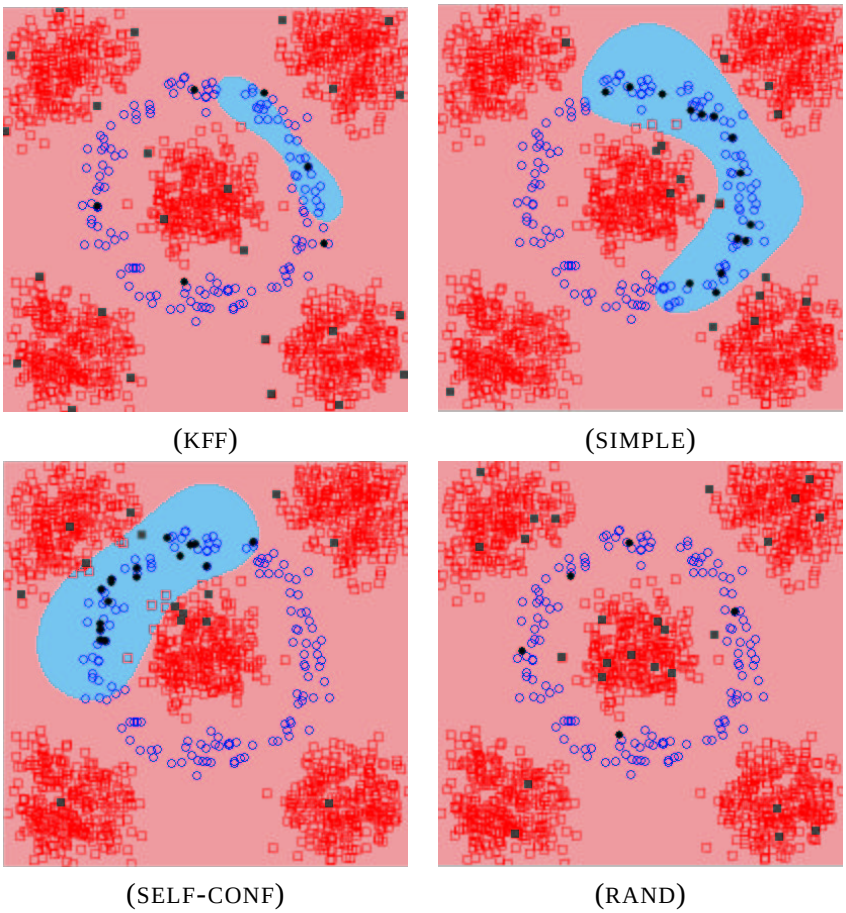


Figure 9: Decision boundaries generated by four active learners using 30 queries on the 'Satellites' example of Figure 8 (left). The 30 queries are darker than the pool points.

cluster between them. This example considers a case where the learning algorithm (SVM) is not majority-biased with respect to a particular inclusion sequence (generated by the SIMPLE querying function). In this example, the increase in the CEM criterion does not match the increase in true accuracy. This happens when the estimated pool entropy at some point is larger than the true entropy $H(r)$ so that the estimated entropy must eventually decrease to its true level while the generalization accuracy can simultaneously improve.

In Figure 10 (right) we observe SIMPLE's decision boundary after 20 queries. This boundary misclassifies the small central cluster and therefore favors the minority class. The pool classification entropy in this case is larger than its final entropy $H(0.9)$. Clearly, when the central cluster is discovered, the entropy will decrease but the true accuracy will increase. Note that the size of the central (isolated) cluster is related to its discovery rate; that is, larger clusters should be discovered faster by a good active learner. In the above example, even one point from this cluster will immediately change the pool classification to be majority-biased. In general, as motivated by this example,

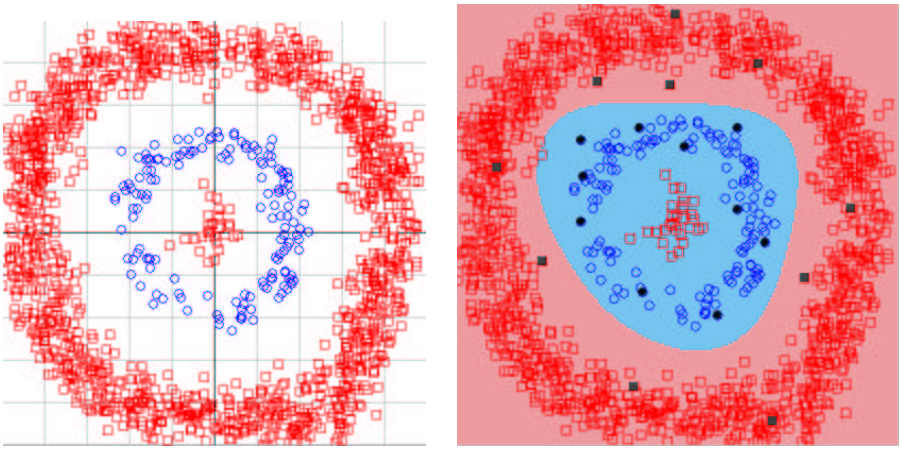


Figure 10: **Left**: The 'Concentric-Rings' problem consists of 3500 points, 10% (350 points) in the inner ring "cluster", 2% (70 points) in the small round cluster in the middle, and the rest in the big outer ring "cluster"; one class consists of all the points in the inner ring (consisting of circles) and the other class is the union of the small round middle cluster and the outer ring (both consisting of squares); 1500 points form the pool and the rest form the test set. **Right**: Decision boundary generated by SIMPLE using 20 queries on this problem. The 20 queries are darker than the pool points.

whenever a learning algorithm is *minority*-biased (with respect to some inclusion sequence) the CEM criterion will fail in the sense that the entropy will not correspond to the true accuracy.

Summarizing the above discussion and considering the experimental results of Section 8, we observe that the success of the CEM criterion depends on the properties of both the data set and the learning algorithm. It appears that data sets with isolated "clusters," whose total proportion is small, will more easily allow for the generation of inclusion sequences of training sets for which the learning algorithm will be *minority*-biased. The dependency of CEM on the learning algorithm can be (intuitively) tied to *algorithmic stability*. For example, if a learning algorithm is "stable" and it is majority-biased over a prefix of some inclusion sequence of training sets, then it is likely that the stability of the algorithm will prevent the generation of classifiers which are minority-biased. Thus, we speculate that the success of CEM in our context is tied to the algorithmic stability of SVMs (see, for example, Bousquet and Elisseeff, 2002, Kutin and Niyogi, 2002). On the other hand, our limited experiments with other, not so stable learning algorithms, such as C4.5, suggests that their CEM estimates are not accurate.

Let us now go back to the 'Concentric-Rings' data set where the SVM was not majority-biased on the inclusion sequence of training sets produced by SIMPLE. The reason for SIMPLE's minority-biased classification is that the training set does not include points from the very small isolated cluster belonging to the majority class, but this training set does include points from the minority class "surrounding" the small cluster. The consequence of this configuration is that the classifier will misclassify the majority class small cluster. This problem can be circumvented by including in the training set at least one point from the majority class small cluster. Formulating this setup

we can bound the probability that this configuration will occur due to a random choice of an initial training set of size k (instead of our "standard" choice of a training set of size two; see Section 2). Let n be the total number of points in the pool. Let n_{maj} be the number of points in a small cluster from the majority class. Let n_{min} be the number of points in a subset of points from the minority class surrounding that small cluster. The classifier (SVM) will misclassify the small cluster if the resulting training set does not include any point from that cluster and at the same time will include at least two points from the minority class subset (to induce two support vectors that will guarantee the wrong classification of the small cluster). Thus, using $(1-x)^z \leq e^{-zx}$, the probability ρ of choosing such a random training set satisfies

$$\rho \leq (\frac{n_{min}}{n})^2(1-\frac{n_{maj}}{n})^{k-2} \leq (\frac{n_{min}}{n})^2 \exp\{-(k-2)(\frac{n_{maj}}{n})\}. \tag{3}$$

Assuming that the bound (3) is smaller than δ we solve for k:

$$\begin{aligned}
& (\frac{n_{min}}{n})^2 \exp\left\{-(k-2)\frac{n_{maj}}{n}\right\} \leq \delta \\
\Leftrightarrow \quad & 2\ln\frac{n_{min}}{n} - (k-2)\frac{n_{maj}}{n} \leq \ln\delta \\
\Leftrightarrow \quad & n_{maj}(k-2) \geq 2n\ln\frac{n_{min}}{n} - n\ln\delta \\
\Leftrightarrow \quad & k \geq 2 + 2\frac{n}{n_{maj}}\ln\left(\frac{n_{min}}{n}\right) - \frac{n}{n_{maj}}\ln\delta \\
\Leftrightarrow \quad & k \geq 2 + \frac{n}{n_{maj}}\ln\left(\left(\frac{n_{min}}{n}\right)^2/\delta\right).
\end{aligned} \tag{4}$$

Thus, with probability at least $1-\delta$, if we choose a random initial training set of size k, the above "bad" configuration will not occur. In particular, if the argument of the 'ln' in (4) is smaller than 1, it is sufficient to take $k=2$. For example, taking $\delta = 0.01$, we have $(\frac{n_{min}}{n})^2/\delta = 1$ when n_{min} is 10% of the data.

In summary, the above discussion indicates that when sampling more points to be in the initial training set provided to an active learner (based on SVMs), the CEM criterion will not fail with high confidence.

Remark 2 *The CEM estimator can be derived using the Information Bottleneck framework (Tishby et al., 1999) as follows.*[26] *If X is a random variable representing the data and Y is another* target *variable, the information bottleneck method computes a partition T of X, while attempting to conserve as much as possible from the information X contains on Y. Formally, one seeks T such that the mutual information $I(T;Y)$ is maximized under some constraint on the magnitude of $I(X,T)$.*[27] *The CEM estimator can be derived by applying the information bottleneck principle with the target variable Y being the data X itself. In our context, T is always a binary classification (that is, a binary partition of X into two non-intersecting subsets), $T = (t^+, t^-)$ (with $X = t^+ \cup t^-$), which must be* consistent *with the current training set $\mathcal{L}$. Therefore, we would like to give higher "scores" for (consistent) partitions T that have higher information content $I(T,X)$ on the data. Consider a data set of size n. Assuming a uniform prior over the samples (that is, $p(x) = 1/n$) and noting that $p(t) = |t|/n$*

26. In fact, we discovered CEM using this framework.
27. In particular, in standard applications of the information bottleneck method, $I(X,T)$ is forced to be sufficiently small so as to achieve *compression*; see Tishby et al. (1999) for details.

and that $p(x|t) = 1/|t|$ if $x \in t$ (and $p(x|t) = 0$ otherwise), we have $p(x,t) = p(x|t)p(t) = 1/n$ if $x \in t$ and 0 otherwise. Thus, for any partition $T = (t^+, t^-)$,

$$\begin{aligned} I(T;X) &= \sum_{x\in X, t\in T} p(x,t)\log\frac{p(x,t)}{p(x)p(t)} \\ &= \sum_{x\in t^+}\frac{1}{n}\log\frac{n}{|t^+|} + \sum_{x\in t^-}\frac{1}{n}\log\frac{n}{|t^-|} \\ &= -\frac{|t^+|}{n}\log\frac{|t^+|}{n} - \frac{|t^-|}{n}\log\frac{|t^-|}{n} = H(T). \end{aligned}$$

10. Concluding Remarks

We presented an online algorithm that effectively combines an ensemble of active learners. The algorithm successfully utilizes elements from both statistical learning and online (adversarial) learning. Extensive empirical results strongly indicate that our algorithm can track the best algorithm in the ensemble on real world problems. Quite surprisingly, our algorithm can quite often outperform the best ensemble member. Practitioners can significantly benefit from our new algorithm in situations where not much is known about the classification problem at hand.

Some questions require further investigation. In our experience, the 'classification entropy maximization (CEM)' semi-supervised criterion for tracking active-learning progress outperforms standard error estimation techniques. Further studies of this overly simple but effective criterion may be revealing. It would also be interesting to examine alternative (semi-supervised) estimators. Farther improvements to our master algorithm may be achieved by developing MAB bounds which depend on the game duration. Such bounds can help in controlling the tradeoff between exploration and exploitation when using very small data sets. Finally, it would be interesting to extend our techniques to multi-valued classification problems (rather than binary) and to other learning tasks such as regression.

Acknowledgments

We thank Ron Meir and Ran Bachrach for useful discussions. We also thank Simon Tong for providing essential information for reproducing the results of Tong and Koller (2001). The work of R. El-Yaniv was partially supported by the Technion V.P.R. Fund for the Promotion of Sponsored Research.

Appendix A. Some Implementation Details

In this appendix we provide some particular implementation details which are essential for replication.

We operated all our SVMs using a kernel correction method discussed by Shaw-Taylor and Christianini (2002), which guarantees that the training set is linearly separable in kernel space as required by algorithms like SIMPLE. Specifically, this is done by modifying the kernel K so that for each training point x_i, the modified kernel K' is $K'(x_i,x_i) = K(x_i,x_i) + \lambda$ where λ is a positive

constant, and for all other arguments the kernel remains the same. In all the experiments described in this paper we took a fixed $\lambda = 2$.[28]

The RBF kernel has one parameter. In the SVM implementation we used (Chang and Lin, 2002) this parameter is denoted by γ. To obtain high performance it is crucial to use appropriate values of γ. This issue of model (and parameter) selection is not the main concern of our work. We therefore assume that all of our SVM based active learners have reasonably good parameters and all learners have the same parameters.[29]

We have normalized each feature to lie in [-1,1] as recommended by Chang and Lin (2002). We have also used a standard method to "eliminate" the bias term by increasing the input dimension by one with a fixed component.

In noisy settings, active learners such as SIMPLE tend to be very unstable at the early stages of the learning process. A description of this phenomenon and a proposed solution are described in Appendix C. The proposed solution is based on "buffering" labeled examples obtained from the learner and postponing the inclusion of buffered points in the training set. The buffer size we used in all our experiments is 3.

Appendix B. Semi-Supervised SVM Model Selection Using CEM

Here we briefly describe some numerical examples of using the CEM criterion of Section 6 as a model selection criterion for choosing the kernel parameters of an SVM (using the RBF kernel). Consider the following semi-supervised binary classification setting. We are given a small training set (for example, containing 20 instances) and a larger pool of unlabeled samples. Our goal is to to train an SVM classifier for the classification problem at hand. Clearly, bad parameter assignment for the SVM kernel will result in poor performance. The kernel parameters can of course be chosen using standard methods such as cross-validation or leave-one-out (LOO). Here we show that the CEM criterion can do slightly better than leave-one-out (and cross-validation) without computation time compromises. We emphasize that this appendix only concerns the CEM criterion and does not discuss active learning.

We applied CEM and LOO on the entire benchmark data set collection of Rätsch et al. (2001) as described in Table 1. Our experimental design is as follows. In all data sets we used an SVM inducer with an RBF kernel. The only relevant parameter for this kernel is γ, which determines the RBF kernel resolution.[30] We fixed a crude feasible set of γ values for all the data sets. This set is $\Gamma = \{0.01, 0.05, 0.1, 0.5, 1.0, 5.0\}$. Let F be a training fold partition (that is, one of the fixed 100 train/test folds in the original benchmark set) consisting of two parts: F_{train} and F_{test}. For each fold we performed the following procedure:

1. We randomly selected 20 instances from F_{train} and designated them as the (labeled) training part S, denoting the rest of the instances in F_{train} by U. The labels of instances in U were kept hidden from both LOO and CEM (and clearly, in all cases neither LOO nor CEM see any instance from F_{test}).

28. This value was crudely chosen to guarantee zero training error (without test error optimization).
29. For each learning problem different γ values were selected for each fold by randomly splitting the training set in half. One half was used as the pool for the active learner. The other half was used for choosing the value of γ out of a fixed grid of γ values using 10-fold cross validation.
30. Since we use the kernel correction "trick" mentioned above, our training sets are guaranteed to be linearly separable in feature space so there is no need to consider a soft margin cost parameter.

2. For each $\gamma \in \Gamma$ we applied LOO and CEM. For LOO this means training 20 classifiers, such that each classifier is trained on different 19 examples from S, tested on the other example and we count success percentage of these 20 classifiers. For CEM this means training one classifier over S and then computing the entropy of the resulting partition with respect to U.

3. LOO and CEM then choose their best candidate $\gamma \in \Gamma$. For LOO this means taking the parameter corresponding to the highest average precision, and for CEM, taking the parameter corresponding to the maximal entropy.

4. Two SVMs with the winning parameters (one for LOO and one for CEM) are then trained over S and the corresponding classifiers are tested on F_{test}.

In order to get a correct perspective on the performance of LOO and CEM we also computed the performance of the best and worst models in hindsight. Specifically, for each $\gamma \in \Gamma$ we computed the resulting accuracy over F_{test} of the corresponding SVM (which was trained with γ on S). The results of this procedure are given in Table 8. Each row of the table corresponds to one data set (among the 13) and the accuracy results for each of the methods is specified; that is, the SVM employing the best parameter value obtained for LOO and CEM. Note that each number in the table is an average over 100 folds and standard errors of the means are specified as well.

The first striking observation is that both LOO and CEM perform quite well. In particular, based on a tiny training set both estimators achieve performance quite close to the best possible. Second, it is evident that CEM outperforms LOO. Experiments we performed with other training set sizes show that the relative advantage of CEM increases as the training set size decreases. When the training set size increases CEM's advantage over LOO eventually disappears. For instance, when the sample size is 40, LOO becomes more reliable and slightly outperforms CEM. Similar results were obtained when instead of LOO we used k-fold cross-validation with "standard" values of k (for example, $k = 3, 4, 5$).

Appendix C. Stabling Active Learner Performance

As mentioned above (and following Tong and Koller, 2001) we use the kernel correction method of Shaw-Taylor and Christianini (2002). This correction guarantees linear separability in feature space, which guarantees the existence of the version space. However, the use of this correction, can introduce severe classification instability in early stages of the an active-learning process (for active learners such as SIMPLE) when learning noisy data sets. In this appendix we briefly present this problem and propose a simple solution based on "buffering".

This kernel correction of Shaw-Taylor and Christianini (2002) works by adding a constant to the diagonal of the kernel matrix, thus providing extra (additive) bias to the classification of the training points. One main effect of this correction is that all training points are correctly classified. This correction introduces discontinuities in the decision boundary.

Consider the example in Figure 11 (left) showing a 2D projection of the 'Twonorm' data set (see the details of this data set in Table 1). Figure 12 shows the pool classification and the training sets generated by the SIMPLE active learner after 4-8 iterations. The decision boundary changes drastically from one iteration to another. Note that the above kernel "correction" enables a correct classification of the training set using all the decision boundaries presented in the figure. A learning curve of SIMPLE (showing the true error) is given in Figure 11 (right). Clearly the true accuracy is extremely unstable.

Data set	Best Possible Accuracy	Worst Possible Accuracy	LOO Accuracy	CEM Accuracy
Banana	71.66±0.66	50.26+-0.51	**66.03±1.13**	65.98±0.97
Breast-Cancer	72.05±0.51	67.16+-0.85	**69.83±0.58**	69.69±0.60
Diabetis	68.44±0.36	63.26+-0.67	66.06±0.55	**67.39±0.34**
Flare-Solar	63.96±0.42	59.00+-0.60	62.04±0.52	**62.94±0.45**
German	70.43±0.38	68.23+-0.67	69.35±0.49	**69.48±0.40**
Heart	75.11±0.82	53.12+-0.62	71.59±0.11	**74.2 ±0.8**
Image	69.69±1.33	55.27+-1.04	64.51±1.71	**68.40±1.40**
Ringnorm	90.35±1.1	53.66+-0.76	88.97±1.21	**89.66±1.1**
Splice	55.56±1.48	49.40+-0.49	52.86±1.3	**55.46±1.44**
Thyroid	89.76±0.73	70.21+-0.61	85.73±0.94	**89.56±0.75**
Titanic	71.32±0.82	63.02+-1.08	**67.41±1.07**	64.65±1.1
Twonorm	88.09±1.09	54.11+-0.86	86.45±1.35	**88.01±1.08**
Waveform	76.73±0.73	65.56+-0.86	73.69±0.87	**76.58±0.72**
Averages	74.08	59.40	71.11	**72.46**

Table 8: Average accuracy (and its standard error) achieved by Classification Entropy Maximization (CEM) and Leave-one-out (LOO) estimators on the 13 UCI problems of Rätsch et al. (2001). For each data set, the winner (among CEM and LOO) appears in boldface. The accuracy of the best and worst possible models (in hindsight) are also given.

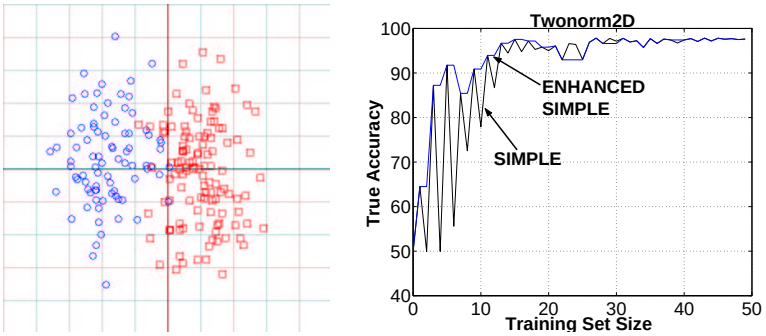


Figure 11: **Left**: Data Set 'Twonorm2D' that was created by projecting the UCI 'Twonorm' data set to two dimensions. This data set demonstrates the unstable general behavior of active learners during early stages of the learning session. **Right**: Learning curves of SIMPLE and enhanced SIMPLE (via buffering) on the 'Twonorm2D' data set.

We propose the following solution using the idea of "buffering", as well as our classification entropy estimator. One of the *symptoms* of the drastic decision boundary instability described above is a (drastic) change in the classification entropy. We observe that when the boundary shifts from the border between the two clusters to the middle of one of the clusters (iterations 5,7) the entropy decreases. Therefore, the solution we propose is to look at the change in entropy of the pool (and training set) classification in the end of each trial. As long as the entropy decreases, new examples are *buffered* and not added to the training set. Once the entropy has not decreased, all buffered

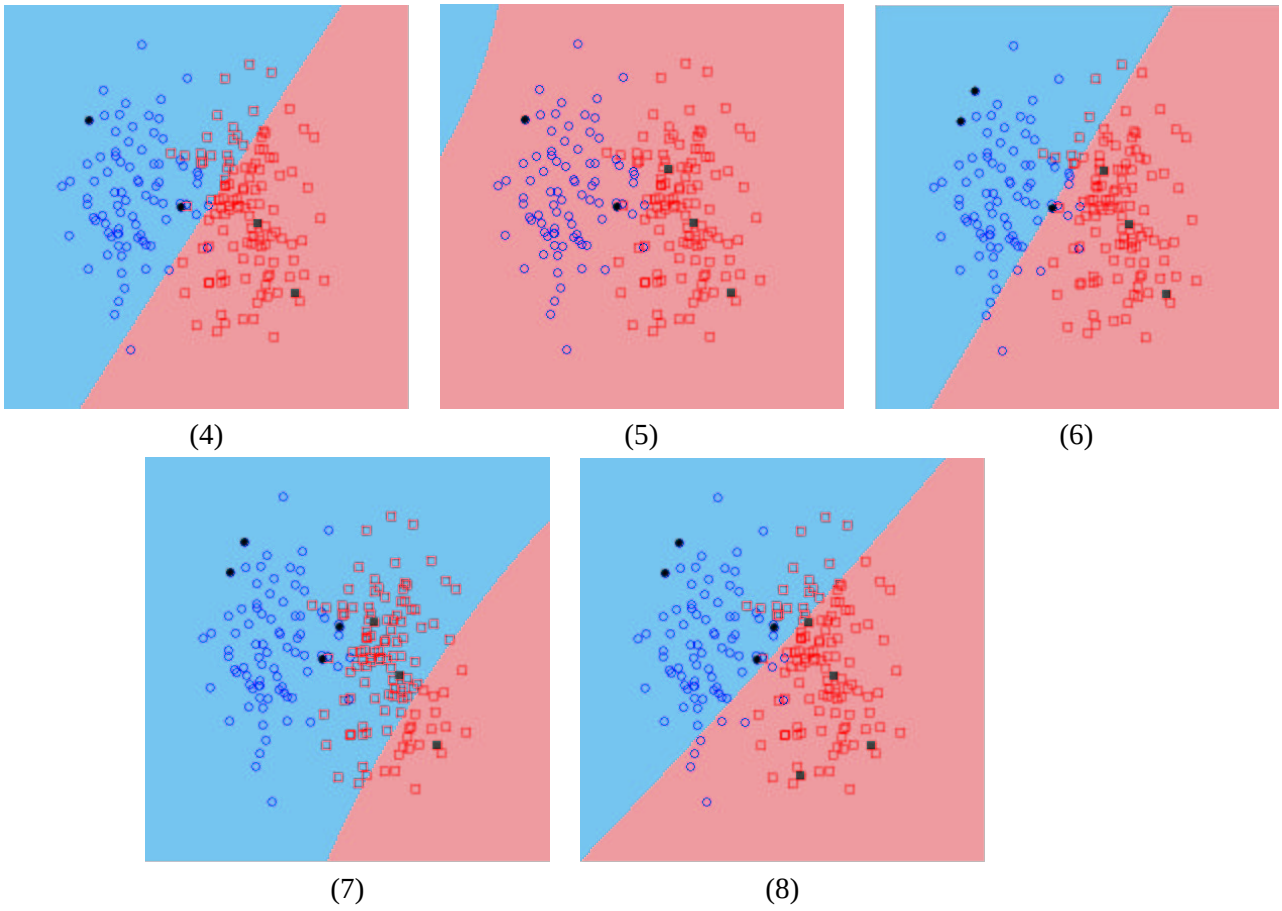


Figure 12: Pool classification and training sets (blackened points) generated by the SIMPLE active learner after 4-8 iterations on the 'Twonorm2D' data set.

points are added to the training set. If we want to keep a bounded buffer this solution introduces a new parameter; namely, the buffer size. In our experience a small buffer is sufficient and in all our experiments we used a buffer of three points.

This buffering approach provides an effective solution to the above problem. In Figure 11 (right) we see the learning curves of SIMPLE on the 'Twonorm2D' data set with and without this enhancement. It is evident that the learning curve of the original algorithm is very unstable. Adding the buffering enhancement smoothes the learning curve. Figure 13 shows the pool classification and the training sets generated by the enhanced SIMPLE active learner after 4-8 iterations. The decision boundary now does not change drastically from one iteration to another.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2(3):319–342, 1988.

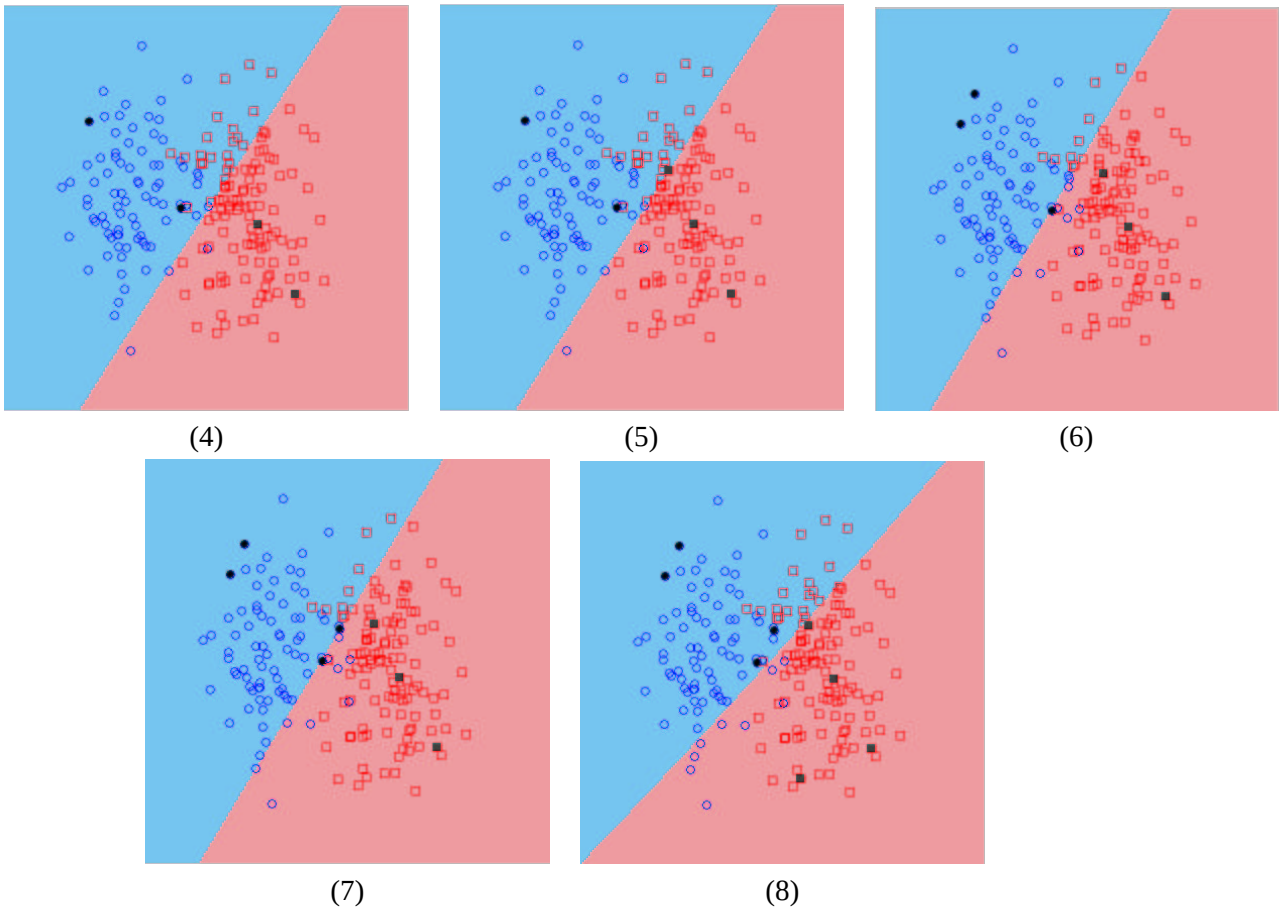


Figure 13: Pool classification and training sets (blackened points) generated by the ***enhanced*** SIMPLE active learner after 4-8 iterations on the 'Twonorm2D' data set.

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.

R. Bachrach, S. Fine, and E. Shamir. Query by committee, linear separation and random walks. In *Proceedings of Euro-COLT, 13th European Conference on Computational Learning Theory*, 1999.

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

C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of ICML-2000, 17th International Conference on Machine Learning*, pages 111–118, 2000.

G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Neural Information Processing Systems (NIPS)*, pages 409–415, 2000.

N. Ceza-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.

C. C. Chang and C. J. Lin. Libsvm: a library for support vector machines, 2002. URL `http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.pdf`.

D. A. Cohn, L. Atlas, and R. E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

I. Guyon, N. Matic, and V. Vapnik. Discovering informative patterns and data cleaning. In *Advances in Knowledge Discovery and Data Mining*, pages 181–203. 1996. URL `http://citeseer.nj.nec.com/guyon96discovering.html`.

R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001. URL `http://citeseer.nj.nec.com/herbrich01bayes.html`.

D. Hochbaum and D. Shmoys. A best possible heuristic for the K-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995. URL `http://citeseer.nj.nec.com/krogh95neural.html`.

S. Kutin and P. Niyogi. Almost-everywhere algorithmic stability and generalization error. Technical Report TR-2002-03, University of Chicago, 2002.

M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54(2):125–152, 2004.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992. URL `http://citeseer.nj.nec.com/47461.html`.

A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*. Morgan Kaufmann Publishers.

T. Mitchell. Generalization as search. *Artificial Intelligence*, 28:203–226, 1982.

G. Rätsch, T. Onoda, and K. R. Müller. Soft margins for Adaboost. *Machine Learning*, 42:287–320, 2001.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of ICML-2001, 18th International Conference on Machine Learning*, pages 441–448, 2001.

M. Saar-Tsechansky and F. Provost. Active learning for class probability estimation and ranking, 2002. URL `http://citeseer.nj.nec.com/tsechansky01active.html`.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *ICMLProceedings of ICML-2000, 17th International Conference on Machine Learning*, pages 839–846, 2000.

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

M. Seeger. Learning with labeled and unlabeled data, 2002. URL `http://citeseer.nj.nec.com/seeger01learning.html`.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Computational Learning Theory*, pages 287–294, 1992. URL `http://citeseer.nj.nec.com/seung92query.html`.

J. Shaw-Taylor and N. Christianini. Further results on the margin distribution. In *Proceedings of the 12th Annual ACM Conference on Computational Learning Theory (COLT)*, pages 278–285, 1999.

J. Shaw-Taylor and N. Christianini. On the generalization of soft margin algorithms. *IEEE Transactions on Information Theory*, 48(10):2721–2735, 2002.

N. Tishby, F. C. Pereira, and W. Bialek. Information bottleneck method. In *37-th Allerton Conference on Communication and Computation*, 1999.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

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

T. Zhang and F. Oles. A probability analysis on the value of unlabeled data for classification problems. In *International Joint Conference on Machine Learning*, pages 1191–1198, 2000.

Journal of Machine Learning Research 5 (2004) 293-323 Submitted 3/03; Revised 12/03; Published 4/04

A Compression Approach to Support Vector Model Selection

Ulrike von Luxburg ULRIKE.LUXBURG@TUEBINGEN.MPG.DE
Olivier Bousquet OLIVIER.BOUSQUET@TUEBINGEN.MPG.DE
Bernhard Schölkopf BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany

Editor: John Shawe-Taylor

Abstract

In this paper we investigate connections between statistical learning theory and data compression on the basis of support vector machine (SVM) model selection. Inspired by several generalization bounds we construct "compression coefficients" for SVMs which measure the amount by which the training labels can be compressed by a code built from the separating hyperplane. The main idea is to relate the coding precision to geometrical concepts such as the width of the margin or the shape of the data in the feature space. The so derived compression coefficients combine well known quantities such as the radius-margin term R^2/ρ^2, the eigenvalues of the kernel matrix, and the number of support vectors. To test whether they are useful in practice we ran model selection experiments on benchmark data sets. As a result we found that compression coefficients can fairly accurately predict the parameters for which the test error is minimized.

Keywords: Support vector machine, compression coefficient, minimum description length, model selection

1. Introduction

In classification one tries to learn the dependency of labels y on patterns x from a given training set $(x_i, y_i)_{i=1\ldots m}$. We are interested in the connections between two different methods to analyze this problem: a data compression approach and statistical learning theory.
The minimum description length (MDL) principle (cf. Barron et al., 1998, and references therein) states that, among a given set of hypotheses, one should choose the hypothesis that achieves the shortest description of the training data. Intuitively, this seems to be a reasonable choice: by Shannon's source coding theorem (cf. Cover and Thomas, 1991) we know that an efficient code is closely related to the data generating distribution. Moreover, an easy-to-describe hypothesis is less likely to overfit than a more complicated one.

There are several connections between the MDL principle and other learning methods. It can be shown that selecting the hypothesis with the highest posterior probability in a Bayesian setting is equivalent to choosing the hypothesis with the shortest code (cf. Hansen and Yu, 2001). For SVMs, the compression scheme approach of Floyd and Warmuth (1995), which describes the generalization of a learning algorithm by its ability to reduce the training set to a few important points, leads to a bound in terms of the number of support vectors. Combining this result with large margin bounds yields the sparse margin bound of Herbrich et al. (2000). In McAllester (1999), a PAC-Bayesian bound for learning algorithms was derived. For a given prior distribution P on the hypotheses

space, it bounds the generalization error essentially by the quantity $-\ln P(U)/m$, where U is the subset of hypotheses consistent with the training examples. Intuitively we can argue that, according to Shannon's theorem, $-\ln P(U)$ corresponds to the length of the shortest code for this subset.

There are statistical learning theory approaches which directly use compression arguments to bound the generalization ability of a classifier, for example the one of Vapnik (1998, sec. 6.2). In this framework, classifiers are used to construct codes that can transmit the labels of a set of training patterns. What those codes essentially do is to tell the receiver which classifier h from a given hypotheses class he should use to reconstruct the training labels from the training patterns (this is described in detail in Section 2). For such a code, the *compression coefficient* $C(h)$ is defined as

$$C(h) := \frac{\text{number of bits to code } y_1,\ldots,y_m \text{ using } h}{m}. \tag{1}$$

The denominator corresponds to the number of bits we need to transmit the uncompressed binary vector $(y_1,\ldots,y_m)$. The numerator tells us how many bits we need to transmit the same information using the code constructed with help of classifier h. Hence, the compression coefficient is a number between 0 and 1 which describes how efficient the code works. Intuitively, if the compression coefficient $C(h)$ is small, h is a "simple" hypothesis which we expect not to overfit too much and hence to have a small generalization error. This belief is supported by the following theorem, which bounds the risk $R(h)$ of a classifier h (i.e., the expected generalization error with respect to the 0-1-loss) in terms of the compression coefficient:

Theorem 1 (Section 6.2 in Vapnik 1998) *With probability at least $1-\eta$ over m random training points drawn iid according to the unknown distribution P, the risk $R(h)$ of classifier h is bounded by*

$$R(h) \leq 2\ln(2)C(h) - \frac{\ln(\eta)}{m}$$

simultaneously for all classifiers h in a finite hypotheses set.

This bound has the disadvantage that it is only valid in the restricted setting where the hypotheses space is finite and independent of the training data.

A different bound that directly works in the coding setting has recently been stated by Blum and Langford (2003). Their setting is slightly different from the one of Vapnik. For a given set of training and test points, the sender constructs a code σ that transmits the labels of both training and test points. R_{test} is then defined as the error which the code σ makes on the given test set.

Theorem 2 (Corollary 3 in Blum and Langford 2003) *With probability at least $1-\eta$ over m random training points and m random test points drawn iid according to the unknown distribution P, and for all codes σ which encode the training labels without error, the error $R_{test}(\sigma)$ on the test set satisfies*

$$R_{test}(\sigma) \leq C(\sigma) - \frac{\ln\eta}{m}.$$

The advantage of this bound is that the problem of data dependency does not occur. It is proved within the coding framework, without assuming a fixed hypotheses space. It is valid for all codes, as long as receiver and sender agree on how the code works before the sender knows the training data. In particular, the codes may depend on the training data in some predefined way, as it will be the case for the codes we are going to construct.

Inspired by the compression coefficient bounds we want to explore whether the connection between statistical learning theory and data compression is only of theoretical interest or whether it can also be exploited for practical purposes. The first part of the work (Section 2) will consist in using hyperplanes learned by SVMs to construct codes for the training labels. Those codes work by encoding the direction of the separating hyperplane. To make them efficient, we will use geometric concepts such as the size of the margin or the shape of the data in the feature space, and sparsity arguments. The main insight of this section is on how to transform those geometrical concepts into an actual code for labels. In the end we obtain compression coefficients that contain quantities already known to be meaningful in the statistical learning theory of SVMs, such as the radius-margin term R^2/ρ^2, the eigenvalues of the kernel matrix, and the number of support vectors. In the second part (Section 3) we then test the model selection performance of our compression coefficients on benchmark data sets. We find that the compression coefficients perform comparable or better than several standard bounds.

Now we want to establish some notation. We assume that the reader is familiar with some basic concepts concerning support vector machines such as a kernel, the margin, and support vectors. In the following, the training data will always consist of m pairs $(x_i, y_i)_{i=1,\ldots,m}$ of patterns with labels. The patterns are assumed to live in some Hilbert space (the feature space), and the labels have binary values ± 1. For a given kernel function k we denote the kernel matrix by $K := (k(x_i, x_j))_{i,j=1,\ldots,m}$. We will denote its eigenvalues by $\lambda_1, \ldots, \lambda_m$, where the λ_i are sorted in non-increasing order. Later on we will also consider the kernel matrix K_{SV} restricted to the span of the support vectors. To define it, let $\{x_{i_1}, \ldots, x_{i_s}\} \subset \{x_1, \ldots, x_m\}$ the set of support vectors, $SV := \{k \mid x_k \in \text{span}\{x_{i_1}, \ldots, x_{i_s}\}\}$ the indices of those training points which are in the subspace spanned by the support vectors. Then the kernel matrix restricted to the span of the support vectors is defined as $K_{SV} := (k(x_i, x_j))_{i,j \in SV}$. In the MDL literature, classifiers are called "hypotheses". In what follows, we use the word "hypothesis" synonymous to "classifier". A hypotheses space is then the space of all possible classifiers we can choose from. The sphere S_R^{d-1} is the surface of a ball with radius R in the space $\mathbb{R}^d$. The function log will always denote the logarithm to the base 2. Code lengths will often be given by some logarithmic term, for instance $\lceil \log m \rceil$. To keep the notations simple, we will omit the ceil brackets and simply write $\log m$.

2. Compression Coefficients for SVMs

The basic setup for the compression coefficient framework is the following. We are given m pairs $(x_i, y_i)_{i=1,\ldots,m}$ of training patterns with labels and assume that an imaginary sender and receiver both know the training *patterns*. It will be the task of the sender to transmit the *labels* of the training patterns to the receiver. This reflects the basic structure of a classification problem: we want to predict the labels y for given patterns x. That is, we want to learn something about $P(y|x)$. Sender and receiver are allowed to agree on the details of the code before transmission starts. Then the sender gets the training data and chooses a classifier that separates the training data. He transmits to the receiver which classifier he chose. The receiver can then apply this classifier to the training patterns and reconstruct all the labels.

To understand how this works let us consider a simple example. Before knowing the data, sender and receiver agree on a finite hypotheses space containing k hypotheses $h_1, \ldots, h_k$. The sender gets the training patterns and the labels, and the receiver is allowed to look at the training patterns only. Now the sender inspects the training data. For simplicity, let us first assume that one of the

hypotheses, say h_7, classifies all training points correctly. In this case the sender transmits "7" to the receiver. The receiver can now reconstruct the labels of the training patterns by classifying them according to hypothesis h_7. Now consider the case where there is no hypothesis that classifies all training patterns correctly. In this case, the receiver cannot reconstruct all labels without error if the sender only transmits the hypothesis. Additionally he has to know which of the training points are misclassified by this hypothesis. In our example, assume that hypothesis 7 misclassifies the training points 3, 14, and 20. The information the sender now transmits is "hypothesis: 7; misclassified points: 3, 14, 20". The receiver can then construct the labels of all patterns according to h_7 and flip the labels he obtained for patterns 3, 14, and 20. After this, he has labeled all training patterns correctly.

This example shows how a classification hypothesis can be used to transmit the labels of training points. In general, a finite, fixed hypothesis space as it was used in the example may not contain a good hypothesis for previously unseen training points and will result in long codes. To avoid this, the sender will have to adapt the hypothesis space to the training points and communicate this to the receiver.

One principle that can already be observed in the example above is the way training errors are handled. As above, the code will always consist of two parts: the first part which serves to describe the hypothesis, and the second part, which tells the receiver which of the training points are misclassified by this hypothesis.

In the following we want to investigate how SVMs can be used for coding the labels. The main part of those codes will consist in transmitting the direction of the separating hyperplane constructed by the SVM. We will always consider the simplified problem where the hyperplane goes through the origin. The hypotheses space consists of all possible directions the normal vector can take. It can be identified with the unit sphere in the feature space. In case of an infinite dimensional feature space, recall that by the representer theorem the solution of an SVM always lies in the subspace spanned by the training points. Thus the normal vector we want to code is a vector in a Hilbert space of dimension at most m (where m is the number of training points). The trick to determine the precision by which we have to code this vector is to interpret the margin in terms of coding precision: Suppose the data are (correctly) classified by a hyperplane with normal vector ω and margin ρ. A fact that often has been observed (e.g., Schölkopf and Smola, 2002, p. 194) is that in case of a large margin, small perturbations of the direction of the hyperplane will not change the classification result on the training points. In compression language this means that we do not have to code the direction of the hyperplane with high accuracy – the larger the margin, the less accurate we have to code. So we will adapt the precision by which we code this direction to the width of the margin. Suppose that all training patterns lie in a ball of radius R around the origin and that they are separated by a hyperplane H through the origin with normal vector ω and margin ρ. Then every "slightly rotated" hyperplane that still lies within the margin achieves the same classification result on all training points as the original hyperplane (cf. Figure 1a). Thus, instead of using the hyperplane with normal vector ω to separate the training points we could use any convenient vector v as normal vector – as long as the corresponding hyperplane still remains inside the margin. In this context, note that rotating a hyperplane by some angle α corresponds to rotating its normal vector by the same angle. We denote the set of normal vectors such that the corresponding hyperplanes still lie inside the margin the *rotation region* of ω. The region in which the corresponding hyperplanes lie will be called the *rotation region of* H (cf. Figures 1a and b).

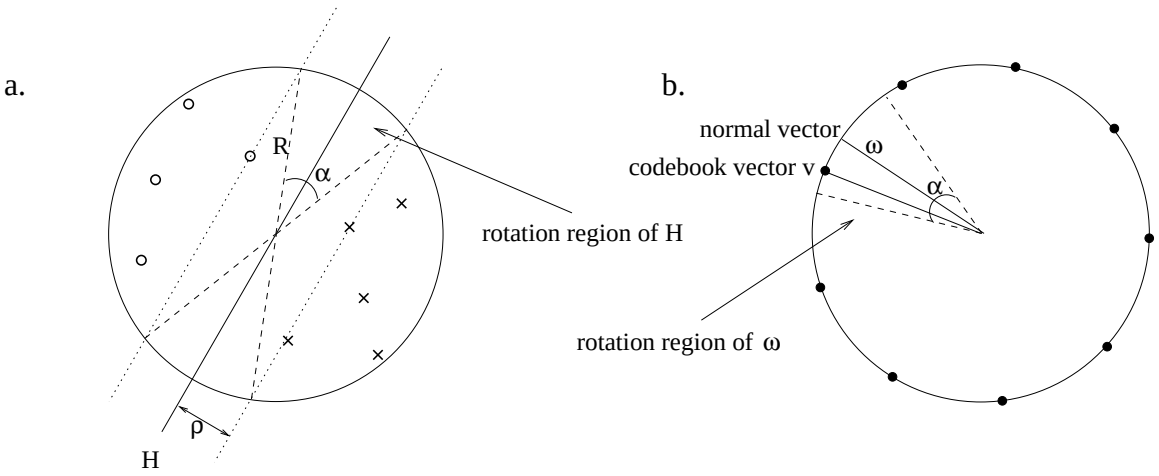


Figure 1: (a) The training points are separated by hyperplane H with margin ρ. If we change the direction of H such that the new hyperplane still lies inside the rotation region determined by the margin (dashed lines), the new hyperplane obtains the same classification result on the training points as H. (b) The hyperplanes in the rotation region of H (indicated by the dashed lines) correspond to normal vectors inside the rotation region of ω. The black points indicate the positions of equidistant codebook vectors on the sphere. The distance between those vectors has to be chosen so small that in each cone of angle α there is at least one codebook vector. In this example, vector v is the codebook vector closest to normal vector ω, and by construction it lies inside the rotation region of ω.

To code the direction of ω, we will construct a discrete set of "codebook vectors" on the sphere. An arbitrary vector will be coded by choosing the closest codebook vector. From the preceding discussion we can see that the set of codebook vectors has to be constructed in such a way that the closest codebook vector for every possible normal vector ω is inside the rotation region of ω (cf. Figure 1b). An equivalent formulation is to construct a set of points on the surface of the sphere such that the balls of radius ρ centered at those points cover the sphere. The minimal number of balls we need to achieve this is called the covering number of the sphere.

Proposition 3 (Covering numbers of spheres) *The number n_d of balls of radius ρ which are required to cover the sphere S_R^{d-1} of radius R in d-dimensional Euclidean space ($d \geq 2$) satisfies*

$$\left(\frac{R}{\rho}\right)^{d-1} \leq n_d \leq 2\left\lceil\frac{R\pi}{\rho}\right\rceil^{d-1}.$$

The constant in the upper bound can be improved, but as we will be interested in log-covering numbers later, this does not make much difference in our application.

Proof We prove the upper bound by induction. For $d = 2$, the sphere is a circle which can be covered with $\lceil 2R\pi/\rho\rceil \leq 2\lceil R\pi/\rho\rceil$ balls. Now assume the proposition is true for the sphere S_R^{d-1}. To construct a ρ-covering on S_R^d we first cover the cylinder $S_R^{d-1} \times [-R\pi/2, R\pi/2]$ with a grid of $\tilde{n}_{d+1} := n_d \cdot \lceil R\pi/\rho\rceil$ points. This grid is a ρ-cover of the cylinder. The grid is then mapped on the

sphere such that the one edge of the cylinder is mapped on the north pole, the other edge on the south pole, and the 'equator' of the cylinder is mapped to the equator of the sphere. As the distances between the grid points do not increase by this mapping, the projected points form a ρ-cover of the sphere S_R^d. By the induction assumption, the number of points in this ρ-cover satisfies

$$n_{d+1} \leq \tilde{n}_{d+1} = n_d \cdot \lceil R\pi/\rho \rceil \leq 2 \lceil R\pi/\rho \rceil^{d-1} \cdot \lceil R\pi/\rho \rceil = 2 \lceil R\pi/\rho \rceil^d .$$

We construct a lower bound on the covering number by dividing the surface area of the whole sphere by the area of the part of the surface covered by one single covering ball. The area of this part is smaller than the whole surface of the small ball. So we get a lower bound by dividing the surface area of S_R^{d-1} by the surface area of S_ρ^{d-1}. As the surface area of a sphere with radius R is R^{d-1} times the surface area of the unit sphere we get $(R/\rho)^{d-1}$ as lower bound for the covering number. ■

Now we can explain how the sender will encode the direction of the separating hyperplane. Before getting the data, sender and receiver agree on a procedure on how to determine the centers of a covering of a unit sphere, given the number of balls to use for this covering, and on a way of enumerating these centers in some order. Furthermore, they agree on which kernel the sender will use for his SVM. Both sender and receiver get to see the training patterns, the sender also gets the training labels. Now the sender trains a (soft margin) SVM on the training data to obtain a hyperplane that separates the training patterns with some margin ρ (maybe with some errors). Then he computes the number n of balls of radius ρ one needs to cover the unit sphere in the feature space according to Proposition 3. He constructs such a covering according to the procedure he and the receiver agreed on. The centers of the covering balls form his set of codebook vectors. The sender enumerates the set of codebook vectors in the predefined way from 1 to n. Then he chooses a codebook vector which lies inside the rotation region of the normal vector (this is always possible by construction). We denote its index $i_n \in \{1,\ldots,n\}$. Now he transmits the total number n of codebook vectors and the index i_n of the one he chose. The receiver now constructs the same set of codebook vectors according to the common procedure, enumerates them in the same predefined way as the sender and picks vector i_n. This is the normal vector of the hyperplane he was looking for, and he can now use the corresponding hyperplane to classify the training patterns. In the codes below, we refer to the pair (n, i_n) as *position of the codebook vector*.

When we count how many bits the sender needs to transmit the two numbers n and i_n we have to keep in mind that to decode, the receiver has to know which parts of the binary string belong to n and i_n, respectively. The number n of codebook vectors is given as in Proposition 3, but as it depends on the margin it cannot be bounded independent from the training data. So the sender cannot use a fixed number of bits to encode n. Instead we use a trick described in Cover and Thomas (1991, p. 149): To build a code for the number n, we take the binary representation of n and duplicate every bit. To mark the end of the code we use the string 01. As an example, n = 27 with the binary representation 11011 will be coded as 111100111101. So the receiver knows that the code of n is finished when he comes upon a pair of nonequal bits. We now apply this trick recursively: to code n, we first have to code the length $\log n$ of its binary code and then send the actual bits of the code of n. But instead of coding $\log n$ with the duplication code explained above, we can also first transmit the length $\log\log n$ of the code of $\log n$ and then transmit the code of $\log n$, and so on. At some point we stop this recursive procedure and code the last remaining number with the duplication code described above. This procedure of coding n needs $\log^*(n) := \log n + \log\log n + \ldots$ bits, where

the sum continues until the last positive term (cf. p. 150 in Cover and Thomas, 1991). Having transmitted n, we can send i_n with $\log n$ bits, because i_n is a number between 1 and n and the sender now already knows n. All in all, the sender needs $\log^* n + \log n$ bits to transmit the position of the codebook vector.

The second part of the code deals with transmitting which of the training patterns are misclassified by the hyperplane corresponding to the chosen codebook vector. We have to be careful how we define the misclassified training points in case of a soft margin SVM. For a soft margin SVM it is allowed that some training points lie inside the margin. For those training points which end up inside the rotation region of the hyperplane, our rotation argument breaks down. It cannot be guaranteed that when the receiver uses the hyperplane corresponding to the codebook vector, the points inside the rotation region are classified in the same way as the sender classified them with the original hyperplane. Thus the sender has to transmit which points are inside the rotation region, and he also has to send their labels. All other points can be treated more easily. The sender has to transmit which of the points outside the rotation region were misclassified. The receiver knows that those points will also be misclassified by his hyperplane, and he can flip their labels to get them right. Below we will refer to the part consisting of the information on the points inside the rotation region and on the misclassified points outside the rotation region as *misclassification information*.

The number r of points inside the rotation region is a number between 0 and m, thus we can transmit its binary representation using $\log m$ bits. After transmitting r, the receiver knows how many training points lie inside the region, but not which of them. There are $\binom{m}{r}$ possibilities which of the training points are the points inside the rotation region. Before transmission, sender and receiver agree on an ordering on those possibilities. Now the sender can transmit the index i_r of the one that is the true one. As this is a number between 1 and $\binom{m}{r}$, and as the receiver at this point already knows r, this can be encoded with $\log \binom{m}{r}$ bits. Next the sender can use r bits to send the labels of the r points inside the rotation region. Finally the sender has to transmit the number l of misclassified training points outside the rotation region. It is a number between 1 and $m-r$, thus we can use $\log(m-r)$ bits for this. To transmit which of the vectors are the misclassified ones, we send the index i_l with $\log \binom{m-r}{l}$ bits. All together, we need $\log m + \log \binom{m}{r} + r + \log(m-r) + \log \binom{m-r}{l}$ bits to transmit the misclassification information. For simplicity, we bound this quantity from above by $(r+l+2)\log m + r$.

Now we can formulate our first code:

Code 1 *Sender and receiver agree on the training patterns, a fixed kernel, and on a procedure for choosing the positions of t balls to cover the unit sphere in a d-dimensional Euclidean space. Now the sender trains an SVM with the fixed kernel on the training patterns, determines the size of the margin ρ and the number n of balls he needs to cover the sphere up to the necessary accuracy according to Proposition 3. Furthermore, he determines which of the training patterns lie inside the rotation region and which patterns outside this region are misclassified by the SVM solution. Now he transmits*

- *the position of the codebook vector (*$\log^* n + \log n$ *bits)*
- *the misclassification information (* $(r+l+2)\log m + r$ *bits)*

To decode this information, the receiver constructs a covering of the sphere in the feature space with $t := n$ balls according to the common procedure, determines the used codebook vector i, and

constructs a hyperplane using this vector as normal vector. He classifies all training patterns according to this hyperplane, labels the points inside the rotation region as transmitted by the sender, and flips the labels of the misclassified training points outside the rotation region.

As defined in Equation (1), the compression coefficient of a code is given by the number of bits it needs to transmit all its information, divided by the number m of labels that were transmitted. Hence, according to our computations above, the compression coefficient of Code 1 is given by

$$C_1 = \frac{1}{m}\left(\log^* n + \log n + (r+l+2)\log m + r\right)$$

with $n = 2\lceil R\pi/\rho \rceil^{d-1}$ according to Proposition 3.

Now we want to refine this code in several aspects. In the construction above we worked with the smallest sphere which contains all the training points. But in practice, the shape of the data in the feature space is typically ellipsoid rather than spherical. This means that large parts of the sphere we used above are actually empty, and thus the code we constructed is not very efficient. Now we want to take into account the shape of the data in the feature space to construct a shorter code. In this setting it will turn out to be convenient to choose the hypotheses space to have the same shape as the data space. The reason for this is the following: When using the rotation argument from above, we observe that in the ellipsoid situation the maximal rotation angle of the hyperplane induced by some fixed margin ρ depends on the actual direction of the hyperplane (cf. Figure 2). This means that the sets of points on a *spherical hypotheses space* which classify the training data in the same way as the hyperplane also have different sizes, depending on the direction of the hyperplane. To construct an optimal set of codebook vectors on the spherical hypotheses space we thus had to cover the sphere with balls of different sizes. Instead of this spherical hypotheses space now consider a hypotheses space which has the same ellipsoid form as the data space. In this case, the sets of vectors which correspond to directions inside the rotation regions can be represented by balls of equal sizes, centered on the surface of the ellipsoid (cf. Figure 2).

Now we want to determine the shape of the ellipsoid containing the data points in the feature space. The lengths of the principal axes of this ellipse can be described in terms of the eigenvalues of the kernel matrix:

Proposition 4 (Shape of the data ellipse) *For given training patterns $(x_i)_{i=1,\ldots,m}$ and kernel k, let $\lambda_1, \ldots, \lambda_d$ be the eigenvalues of the kernel matrix $K = (k(x_i, x_j))_{i,j=1,\ldots,m}$. Then all training patterns are contained in an ellipse with principal axes of lengths $\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_d}$ in the feature space.*

Proof The trick of the proof is to interpret the eigenvectors of the kernel matrix, who originally live in $\mathbb{R}^d$, as vectors in the feature space. Let $H_m := \mathrm{span}\{\delta_{x_i} | i = 1, \ldots, m\}$ the subspace of the feature space spanned by the training examples. It is endowed with the scalar product $\langle \delta_{x_i}, \delta_{x_j} \rangle_K = k(x_i, x_j)$. Let $(e_i)_{i=1,\ldots,m}$ the canonical basis of $\mathbb{R}^m$ and $\langle \cdot, \cdot \rangle_m$ the Euclidean scalar product. Define the mapping $T : \mathbb{R}^m \to H_m$, $e_i \mapsto \delta_{x_i}$. For $u = \sum_{i=1}^m u_i e_i$, $v = \sum_{j=1}^m v_j e_j \in \mathbb{R}^m$ we have

$$\langle Tu, Tv \rangle_K = \langle \sum_{i=1}^m u_i \delta_{x_i}, \sum_{j=1}^m v_j \delta_{x_j} \rangle_K = \sum_{i,j=1}^m u_i v_j \langle \delta_{x_i}, \delta_{x_j} \rangle_K = \sum_{i,j=1}^m u_i v_j k(x_i, x_j) = u' K v \,. \tag{2}$$

Let $v_1, \ldots, v_d \in \mathbb{R}^m$ be the normalized eigenvectors of the matrix K corresponding to the eigenvalues $\lambda_1, \ldots, \lambda_d$, i.e., $Kv_i = \lambda_i v_i$ and $\langle v_i, v_j \rangle_m = \delta_{ij}$. From Equation (2) we can deduce

$$\langle Tv_i, Tv_j \rangle_K = v_i' K v_j = v_i' \lambda_j v_j = \lambda_j \langle v_i, v_j \rangle_m = \lambda_i \delta_{ij},$$

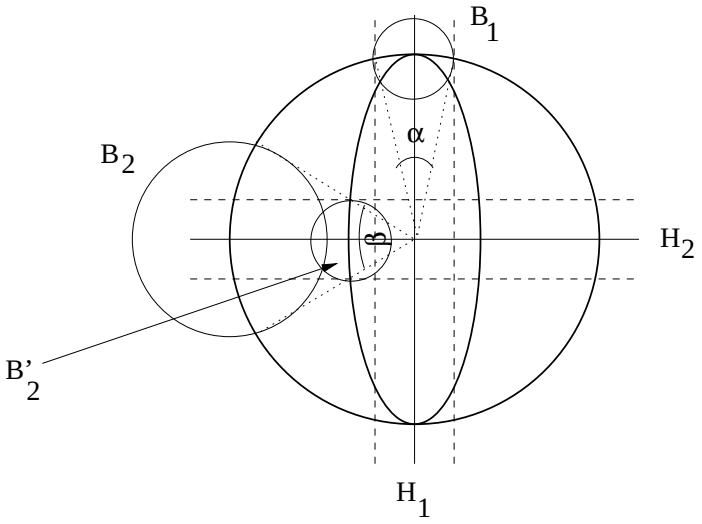


Figure 2: In this figure, the ellipse represents the data domain and the large circle represents a spherical hypotheses space. Consider two hyperplanes H_1 and H_2 with equal margin. The balls B_1 resp. B_2 indicate the sets of hypotheses that yield the same classification result as H_1 resp. H_2. The sizes of those balls depend on the direction of the hyperplane with respect to the ellipse. In the example shown, B_1 is smaller than B_2, hence H_1 must be coded with higher accuracy than H_2. Note that if we use the ellipse itself as hypotheses space, we have to consider the balls B_1 and B_2' which are centered on the surface of the ellipse and have equal size.

in particular $\|Tv_i\|_K = \sqrt{\lambda_i}$. Furthermore we have

$$\langle \delta_{x_i}, Tv_j \rangle_K = \langle \delta_{x_i}, \sum_{l=1}^{m} (v_j)_l \delta_{x_l} \rangle_K = \sum_{l=1}^{m} (v_j)_l k(x_i, x_l) = (Kv_j)_i = (\lambda_j v_j)_i = \lambda_j (v_j)_i.$$

Altogether we can now see that in the feature space, all data points δ_{x_i} lie in the ellipse whose principal axes have direction Tv_j and length $\sqrt{\lambda_j}$ because the ellipse equation is satisfied:

$$\sum_{j=1}^{d} \left(\frac{\langle \delta_{x_i}, \frac{Tv_j}{\|Tv_j\|_K} \rangle_K}{\sqrt{\lambda_j}} \right)^2 = \sum_j ((v_j)_i)^2 \leq 1.$$

Here the last equality follows from the fact that $\sum_j ((v_j)_i)^2$ is the Euclidean norm of a row vector of the orthonormal matrix containing the eigenvectors $(v_1, ..., v_d)$. ■

Now that we know the shape of the ellipse, we have to find out how many balls of radius ρ we need to cover its surface. As surfaces of ellipses in high dimensions are complicated to deal with, we simplify our calculation. Instead of covering the surface of the ellipse, we will cover the ellipse completely. This means that we use one extra dimension (volume instead of area), but in

high dimensional spaces this does not make much difference, especially if some of the axes are very small. Computing a rough bound on the covering numbers of an ellipse is easy:

Proposition 5 (Covering numbers of ellipses) *The number n of balls of radius ρ which are required to cover a d-dimensional ellipse with principal axes $c_1, \ldots, c_d$ satisfies*

$$\prod_{i=1}^{d} \frac{c_i}{\rho} \leq n \leq \prod_{i=1}^{d} \left\lceil \frac{2c_i}{\rho} \right\rceil .$$

Proof The smallest parallelepiped containing the ellipse has side lengths $2c_1, \ldots, 2c_d$ and can be covered with a grid of $\prod_{i=1}^{d} \lceil 2c_i/\rho \rceil$ balls of radius ρ. This gives an upper bound on the covering number.

To obtain a lower bound we divide the volume of the ellipse by the volume of one single ball. Let v_d be the volume of a d-dimensional unit ball. Then the volume of a d-dimensional ball of radius ρ is $\rho^d v_d$ and the volume of an ellipse with axes $c_1, \ldots, c_d$ is given by $v_d \prod_{i=1}^{d} c_d$. So we need at least $\prod_{i=1}^{d} (c_i/\rho)$ balls. ■

Now we can formulate the refined code:

Code 2 *This code works analogously to Code 1, the only difference is that sender and receiver work with a covering of the data ellipse instead of a covering of the enclosing sphere.*

The compression coefficient of Code 2 is

$$C_2 = \frac{1}{m} \left(\log^* n + \log n + (r + l + 2) \log m + r \right),$$

with $n = \prod_{i=1}^{d} \lceil 2\sqrt{\lambda_i}/\rho \rceil$ according to Propositions 4 and 5.

It is interesting to notice that the main complexity term in the compression coefficient is the logarithm of the number of balls needed to cover the region of interest of the hypotheses space. So the shape of the bounds we obtain is very similar to classical bounds based on covering numbers in statistical learning theory. This is not so surprising since we explicitly approximate our hypotheses space by covers, but there is a somewhat deeper connection. Indeed, when we construct our code, we consider all normal vectors in a certain region as equivalent with respect to the labeling they give of the data. This means that we define a metric on the set of possible normal vectors which is related to the induced Hamming distance on the data (that is the natural distance in the "coordinate projections" of our function class on the data). Hence, when we adapt the size of the balls to the direction in hypotheses space (in Figure 2), we actually say that Hamming distance 1 on the data translates into a certain radius. Hence, we are led to build covers in this induced distance which is exactly the distance which is used in classical covering number bounds for classification. So the compression approach gives another motivation, of information theoretic flavor, for considering that the right measure of the capacity of a function class is the metric entropy of its coordinate projections.

Both compression coefficients we derived so far implicitly depend on the dimension d of the feature space in which we code the hyperplane. Above we always used $d = m$ as the solution of an SVM always lives in the subspace spanned by the training examples. But as the solution even lies in the subspace spanned by the support vectors, an easy dimension reduction can be achieved

by working in this subspace. The procedure then works as follows: the sender trains the SVM and determines the support vectors and the margin ρ. The ellipse that we have to consider now is the ellipse determined by the kernel matrix K_{SV} restricted to the linear span of the support vectors (cf. notations at the end of Section 1). The reason for this is that we are only interested in what happens if we slightly change the direction of the normal vector *within* the subspace spanned by the support vectors.

To let the receiver know the subspace he is working in, the sender has to transmit which of the training patterns are support vectors. This part of the code will be called *support vector information*. As the number s of support vectors is between 0 and m, the sender first codes s with $\log m$ bits and then the index i_s of the actual support vectors among the $\binom{m}{s}$ possibilities with $\log\binom{m}{s}$ bits. So the support vector information can be coded with $\log m + \log\binom{m}{s} \leq (s+1)\log m$ bits. After submitting the information about the support vectors, the code proceeds analogously to Code 2.

Code 3 *Sender and receiver agree on the training patterns, the kernel, and the procedure of covering an ellipsoid. After training an SVM, the sender transmits*

- *the support vector information (*$(s+1)\log m$ *bits),*
- *the position of the codebook vector (*$\log^* n + \log n$ *bits),*
- *the misclassification information (*$(r+l+2)\log m + r$ *bits).*

To decode, the receiver constructs the hypotheses space consisting of the data ellipse projected on the subspace spanned by the support vectors. He covers this ellipse with n balls and chooses the vector representing the normal vector of the hyperplane. Then he labels the training patterns by first projecting them into the subspace and then classifying them according to the hyperplane. Finally, he deals with the misclassified training points as in the codes before.

The compression coefficient of Code 3 is given as

$$C_3 = \frac{1}{m}\left(\log^* n + \log n + (r+l+s+3)\log m + r\right),$$

with $n \leq \prod_{i=1}^{s} \lceil 2\sqrt{\gamma_i}/\rho \rceil$ according to Propositions 4 and 5. Here γ_i denote the eigenvalues of the restricted kernel matrix K_{SV}.

A further dimension reduction can be obtained with the following idea: It has been empirically observed that on most data sets the axes of the data ellipse decrease fast for large dimensions. Once the axis in one direction is very small, we want to discard this dimension by projecting in a lower dimensional subspace using kernel principal component decomposition (cf. Schölkopf and Smola, 2002). A projection P will be allowed if the image $P(\omega)$ of the normal vector ω is still within the rotation region induced by the margin ρ. In this case we construct codebook vectors for $P(\omega)$ in the lower dimensional subspace. We have to make sure that the vector representing $P(\omega)$ is still contained in the original rotation region.

In more detail, this approach works as follows: First we train the SVM and get the normal vector ω and margin ρ. For convenience, we now normalize ω to length R by $\omega_0 := \frac{\omega}{||\omega||}R$. After normalizing, we know that a vector v still lies inside the rotation region if $\|\omega_0 - v\| \leq \rho$. Now we perform a kernel PCA of the training data in the feature space. For $d_P \in \{1,\ldots,m\}$ let P be

the projection on the subspace spanned by the first d_P eigenvectors. To determine whether we are allowed to perform projection d_P we have to check whether $\|P(\omega_0) - \omega_0\| \leq \rho$. If not, the hyperplane corresponding to $P(\omega_0)$ is not within the rotation region any more, and we are not allowed to make this projection. Otherwise, we are still within the rotation region after projecting ω_0, so we can discard the last $m - d_P$ dimensions. In this case, we call P a *valid* projection. We then can encode the projected normal vector $P(\omega_0)$ in an d_P-dimensional subspace. As $P(\omega_0)$ is not in the center of the rotation region any more, we have to code its direction more precisely now. We have to ensure that the codebook vector v for $P(\omega_0)$ still is in the original rotation region, that is $\|v - \omega_0\| \leq \rho$. Define

$$c_P := \frac{1}{\rho}\|\omega_0 - P(\omega_0)\|$$

(note that for a valid projection, $c_P \in [0,1]$), and choose the radius r of the covering balls as $r = \rho\sqrt{1-c_P^2}$. Then we have

$$\|\omega_0 - v\|^2 \leq \|\omega_0 - P(\omega_0)\|^2 + \|P(\omega_0) - v\|^2 \leq c_P^2\rho^2 + (1-c_P^2)\rho^2 = \rho^2 \,.$$

Thus the codebook vector v is still within the allowed distance of ω_0.

All in all our procedure now works as follows: The sender trains an SVM. From now on he works in the subspace spanned by the support vectors only. In this subspace, he performs the PCA and determines the smallest d_P such that the projection on the subspace of d_P dimensions is a valid projection.The principal axes of the ellipse in the subspace are now given by the first d_P eigenvalues of the restricted kernel matrix K_{SV}, and we have to construct a covering of this ellipse with covering radius $r = \rho\sqrt{1-c_P^2}$. Then the code proceeds as before. The number d_P will be called the *projection information* and can be encoded with $\log m$ bits.

Code 4 *Sender and receiver agree on the training patterns, the kernel, the procedure of covering ellipsoids, and on how to perform kernel PCA. The sender trains the SVM and chooses a valid projection on some subspace. He transmits*

- *the support vector information (*$(s+1)\log m$ *bits),*
- *the projection information (*$\log m$ *bits),*
- *the position of the codebook vector (*$\log^* n + \log n$ *bits),*
- *the misclassification information (*$(r+l+2)\log m + r$ *bits).*

To decode, the receiver constructs the hypotheses space consisting of the ellipse in the subspace spanned by the support vectors. Then he performs a PCA in this subspace and projects the hypotheses space on the subspace spanned by the first d_P principal components. He covers the remaining ellipse with n balls and continues as in the codes before.

The compression coefficient of this code is given by

$$C_4 = \frac{1}{m}\left(\log^* n + \log n + (r+l+s+4)\log m + r\right),$$

with $n \leq \prod_{i=1}^{d_P} \lceil 2\sqrt{\gamma_i}/(\rho\sqrt{1-c_P^2}\,)\rceil$, c_P as described above, and γ_i the eigenvalues of the restricted kernel matrix K_{SV}.

So far we always considered codes which use the direction of the hyperplane as hypothesis. A totally different approach is to reduce the data by transmitting support vectors and their labels. Then the receiver can train his own SVM on the support vectors and will get the same result as the sender. Note that in this case, we not only have to transmit which of the vectors are support vectors as in the other codes, but also the labels of the support vectors. On the other hand we have the advantage that we do not have to treat the points inside the rotation region separately as they are support vectors anyway. The misclassification information only consists in the misclassified points which are not support vectors. This simple code works as follows:

Code 5 *Sender and receiver agree on training patterns and a kernel. The sender sends*

- *the support vector information (*$(s+1)\log m$ *bits),*
- *the labels of the support vectors (s bits),*
- *the information on the misclassified points outside the rotation region (*$(l+1)\log m$ *bits).*

To decode this information, the receiver trains an SVM with the support vectors as training set. Then he computes the classification result of this SVM for the remaining training patterns and flips the labels of the misclassified non-support vector training points.

This code has a compression coefficient of

$$C_5 = \frac{1}{m}\left((s+l+2)\log m + s\right).$$

3. Experiments

To test the utility of the derived compression coefficients for applications, we ran model selection experiments on different artificial and real world data sets. We used all data sets in the benchmark data set repository compiled and explained in detail in Rätsch et al. (2001). The data sets are available at http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm. Most of the data sets in this repository are preprocessed versions of data sets originating from the UCI, Delve, or STATLOG repositories. In particular, all data sets are normalized. The data sets are called banana (5300 points), breast cancer (277 points), diabetis (768 points), flare-solar (1066 points), german (1000 points), heart (2700 points), image (2310 points), ringnorm (7400 points), splice (3175 points), thyroid (215 points), titanic (2201 points), twonorm (7400 points), waveform (5100 points). To be consistent with earlier versions of this manuscript we also used the data sets abalone (4177 points), Wisconsin breast cancer (683 points) from the UCI repository, and the US postal handwritten digits data set (9298 points). In all experiments, we first permuted the whole data set and divided it into as many disjoint training subsets of sample size $m = 100$ or 500 as possible. We centered each training subset in the feature space as described in Section 14.2. of Schölkopf and Smola (2002) and then used it to train soft margin SVMs with Gaussian kernels. The test error was computed on the training subset's complement.

For different choices of the soft margin parameter $C \in [10^0, ..., 10^5]$ and the kernel width $\sigma \in [10^{-2}, ..., 10^3]$ we computed the compression coefficients and chose the parameters where the compression coefficients were minimal. Note that as we centered the data in the feature space, the radius R can be approximately computed as the maximum distance of the centered training points to

the origin. We compared the test errors corresponding to the chosen parameters to the ones obtained by model selection criteria from different generalization bounds. To state those bounds we denote the empirical risk of a classifier by R_{emp} and the true risk of a classifier by R_{true}. Note that the definition of the risks varies slightly for the different bounds, for example with respect to the used loss function. We refer to the cited papers for details. The bounds we consider are the following:

- The radius-margin bound of Vapnik (1998). We here cite the version stated in Bartlett and Shawe-Taylor (1999): with probability at least $1-\delta$,

$$R_{true} \leq R_{emp} + \sqrt{\frac{c}{m}\left(\frac{R^2}{\rho^2}\log^2 m - \log\delta\right)}.$$

  The quantity we compute in the experiments is $R_{emp} + \sqrt{(R^2 \log m)/(\rho^2 m)}$.

- The rescaled radius-margin bound of Chapelle and Vapnik (2000). It uses the shape of the training data in the feature space to refine the classical radius margin bound. In this case, the quantity R^2/ρ^2 in the above bound is replaced by

$$\sum_{k=1}^{m} \lambda_k^2 \max_{i=1,\ldots,m} A_{ik}^2 \Big(\sum_{j=1,\ldots,m} A_{jk} y_j \alpha_j \Big)^2,$$

  where A is the matrix of the normalized eigenvectors of the kernel matrix K, λ_k are the eigenvalues of the kernel matrix, and α the coefficients of the SVM solution.

- The trace bound of Bartlett and Mendelson (2001) which contains the eigenvalues of the kernel matrix: with probability at least $1-\delta$,

$$R_{true} \leq R_{emp} + \mathit{Rademacher} + \sqrt{\frac{8\ln(2/\delta)}{m}},$$

  where the Rademacher complexity is given by $\mathit{Rademacher} \leq \sum_{i=1}^{m} \sqrt{\lambda_i/m}$. The quantity we compute in the experiments is $R_{emp} + \mathit{Rademacher}$.

- The compression scheme bound of Floyd and Warmuth (1995) using the sparsity of the SVM solution: with probability at least $1-\delta$,

$$R_{true} \leq \frac{1}{m-s}\left(\ln\binom{m}{s} + \ln\frac{m^2}{\delta}\right),$$

  where s is the number of support vectors. In the experiments we computed the quantity $\ln\binom{m}{s}/(m-s)$. Note that if $s = m$ this bound is infinite. We omitted those cases from the plots.

- The sparse margin bound of Herbrich et al. (2000) which uses the size of the margin and the sparsity of the solution: with probability at least $1-\delta$,

$$R_{true} \leq \frac{1}{m-s}\left(\kappa \ln\frac{em}{\kappa} + \ln\frac{m^2}{\delta}\right),$$

  where $\kappa = \min(\lceil \frac{R^2}{\rho^2} + 1 \rceil, d+1)$. For our experiments we compute the quantity $\frac{1}{m-s}\left(\kappa \ln\frac{em}{\kappa}\right)$. In case $s = m$ this bound is infinite and we omit those cases from the plots.

- The span estimate of Vapnik and Chapelle (2000), cf. also Opper and Winther (2000). This bound is different from all the other bounds as it estimates the leave-one-out error of the classifier. To achieve this, it bounds for each support vector how much the solution would change if this particular support vector were removed from the training set.

All experimental results shown below were obtained with training set size $m = 100$; the results for $m = 500$ are comparable. The interested reader can study those results, as well as many more plots which we cannot show here because they would use too much space, on the webpage http://www.kyb.tuebingen.mpg.de/bs/people/ule.

The goal of our first experiment is to use the compression coefficients to select the kernel width σ. In Figure 3 we study which of the five compression coefficients achieves the best results on this task. The plots in this figure were obtained as follows. For each training set, each parameter C and each compression coefficient we chose the kernel width σ for which the compression coefficient was minimal. Then we evaluated the test error for the chosen parameters on the test set and computed the mean over all runs on the different training sets. Plotted are the means of these test errors versus parameter C, as well as the means of the minimal true test errors.

We can observe that for nearly all data sets, the compression coefficients C_2, C_3, and C_4 yield better results than the simple coefficients C_1 and C_5. This can be explained by the fact that C_1 and C_5 only use one part of information (the size of the margin or the number of support vectors, respectively), while the other coefficients combine several parts of information (margin, shape of data, number of support vectors). When we compare coefficients C_3 and C_4 we observe that they have nearly identical values. This indicates that the gain we obtain by projecting into a smaller subspace using kernel PCA is outweighed by the additional information we have to transmit about this projection. As C_3 is simpler to compute, we thus prefer C_3 to C_4. From now on we want to evaluate the results of the most promising compression coefficients C_2 and C_3.

In Figure 4 we compare compression coefficients C_2 and C_3 to all the other model selection criteria. The plots were obtained in the same way as the ones above. We see that for most data sets, the performance of C_2 is rather good, and in most cases it is better than C_3. It performs nearly always comparable or better than the standard bounds, in particular it is nearly always better than the widely-used radius margin bound. Among all bounds, C_2 and the span bound achieve the best results. Comparing those two bounds shows that no one is superior to the other one: C_2 is better than the span bound on five data sets (abalone, banana, diabetis, german, usps), the span bound is better than C_2 on five data sets (image, ringnorm, splice, thyroid, waveform), and they achieve similar results on six data sets (breast-cancer, flare-solar, heart, titanic, twonorm, wisconsin).

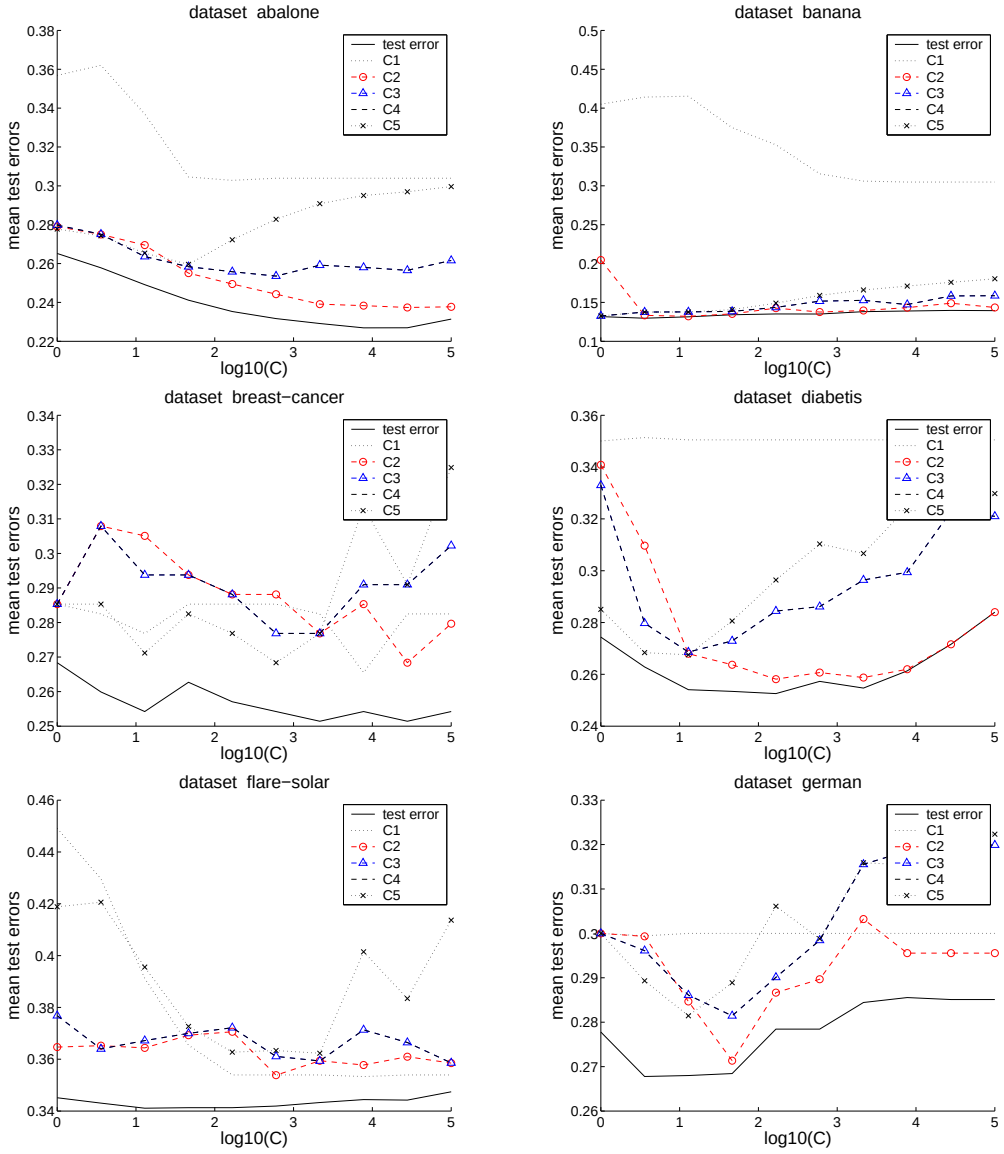


Figure 3: Comparison among the compression coefficients. For each training set, each soft margin parameter C and each compression coefficient we chose the kernel width σ for which the compression coefficient was minimal and evaluated the test error for the chosen parameters. Plotted are the mean values of the test errors over the different training sets, as well as the means of the true minimal test errors (this figure is continued on the next page).

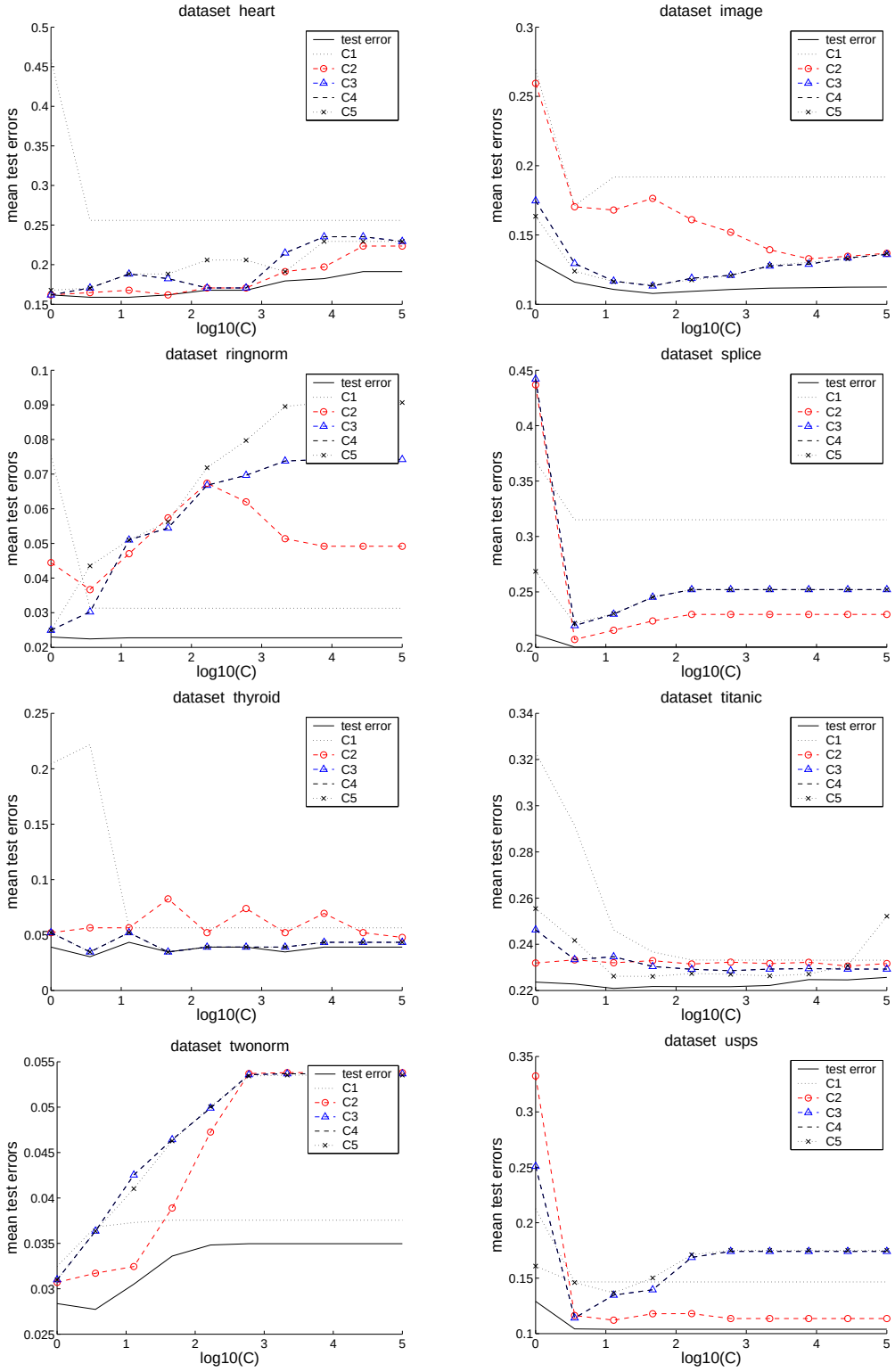


Figure 3, continued

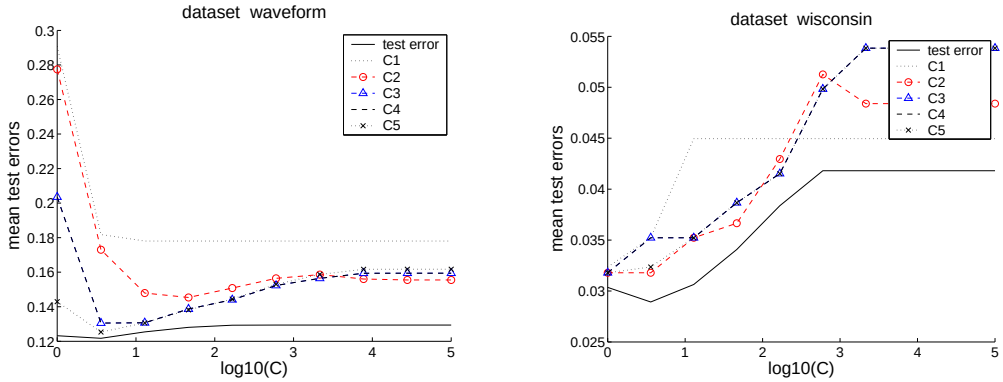


Figure 3, continued

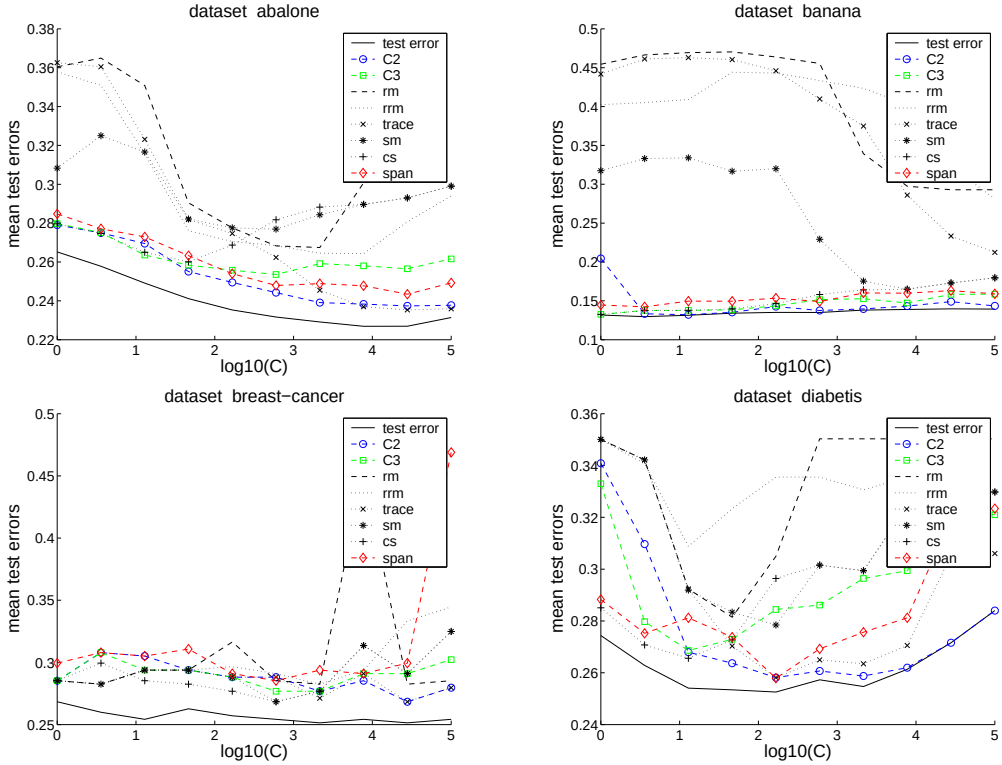


Figure 4: Comparison between C_2, C_3, and the other bounds. For each training set, each soft margin parameter C and each bound we chose the kernel width σ for which the bound was minimal and evalutated the test error for the chosen parameters. Plotted are the mean values of the test errors over the different training sets. In the legend we use the abbreviations rm = radius margin bound, rrm = rescaled radius margin bound, sm = sparse margin bound, and cs = compression scheme bound (continued on the next page).

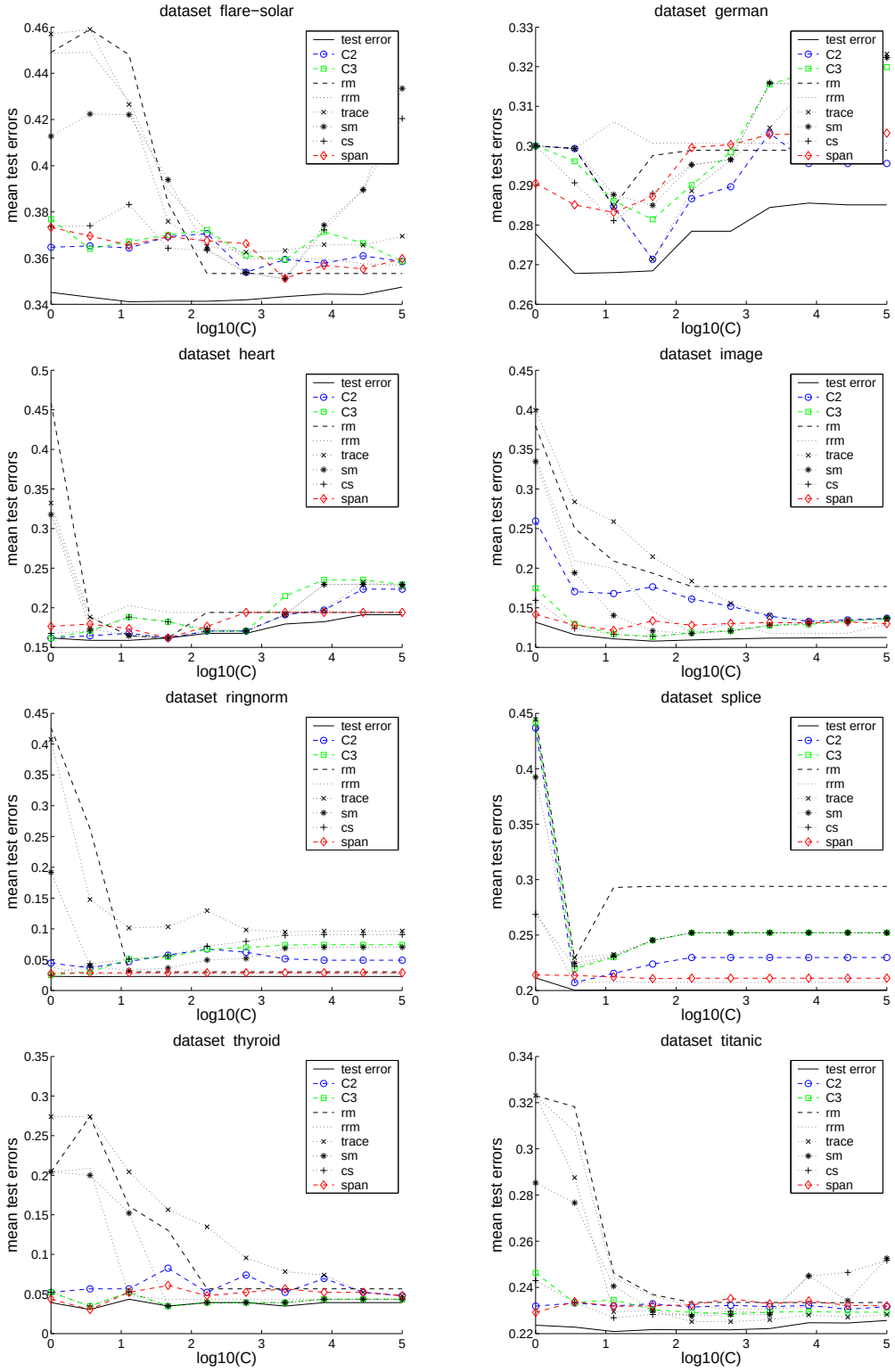


Figure 4, continued

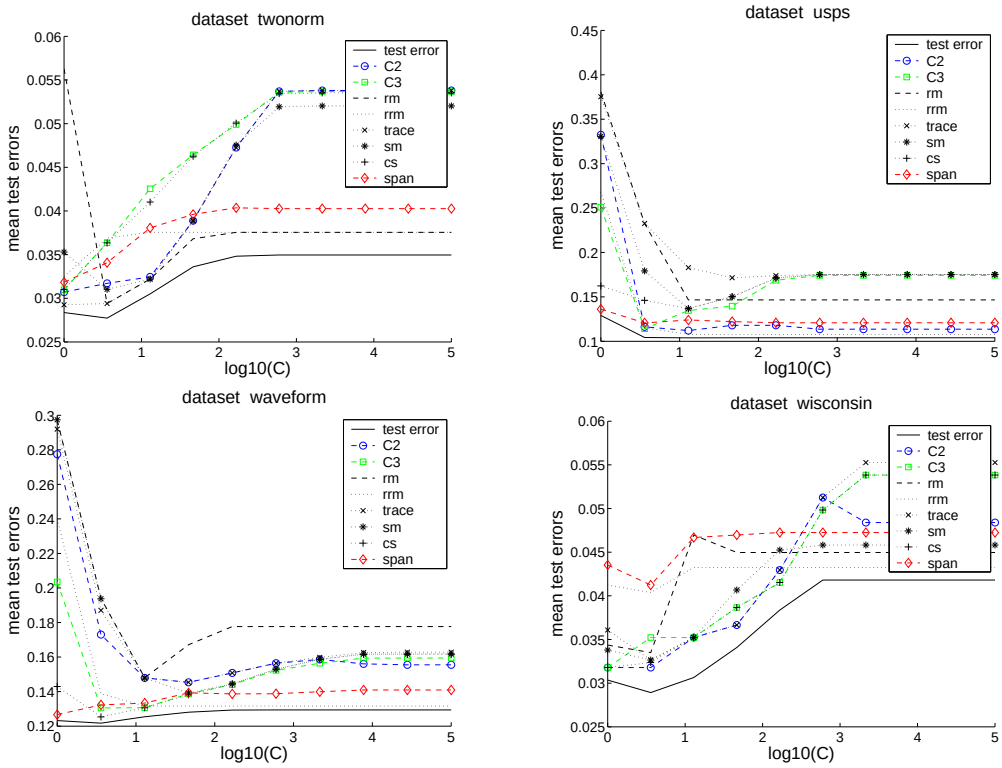


Figure 4, continued

The goal of the next experiment was not only to select the kernel width σ, but to find the best values for the kernel width σ and the soft margin parameter C simultaneously. Its results can be seen in Table 1, which was produced as follows. For each training set we chose the parameters σ and C for which the respective bounds were minimal, and evaluated the test error for the chosen parameters. Then we computed the means over the different training runs. The first column contains the mean value of the minimal test error. The other columns contain the offset by which the test errors selected by the different bounds are worse than the optimal test error. On most data sets, the radius margin bound, the rescaled radius margin bound, and the sparse margin bound perform rather poorly. Often their results are worse than those of the other bounds by one order of magnitude. Among the other bounds, C_2 and the span bound are the two superior bounds. Between those two bounds, there is a tendency towards the span bound in this experiment: C_2 beats the span bound on 6 data sets, the span bound beats C_2 on 10 data sets. Thus C_2 does not perform as good as the span bound, but it gets close.

To explain the good performance of the compression coefficients we now want to analyze their properties in more detail. As the compression coefficients are a sum of several terms it is a natural question which of the terms has the largest influence on the actual value of the sum. To answer this question we look at Figure 5, where we plotted the different parts of C_3: the term $(s+1)\log m$ cor-

data set	test error	C_2	C_3	span	rm	rrm	trace	sm	cs
abalone	0.224	0.017	0.036	0.029	0.137	0.134	0.012	0.084	0.072
banana	0.124	0.021	0.024	0.020	0.347	0.278	0.141	0.202	0.047
breast-cancer	0.251	0.062	0.065	0.209	0.034	0.034	0.028	0.034	0.042
diabetis	0.247	0.012	0.049	0.025	0.103	0.103	0.059	0.103	0.080
flare-solar	0.339	0.026	0.027	0.020	0.120	0.110	0.030	0.101	0.101
german	0.263	0.037	0.042	0.026	0.037	0.037	0.060	0.037	0.044
heart	0.156	0.015	0.015	0.021	0.303	0.168	0.071	0.159	0.053
image	0.105	0.074	0.028	0.023	0.275	0.235	0.031	0.183	0.028
ringnorm	0.021	0.054	0.052	0.007	0.405	0.017	0.075	0.178	0.068
splice	0.198	0.039	0.053	0.016	0.249	0.035	0.054	0.084	0.053
thyroid	0.026	0.030	0.017	0.026	0.178	0.178	0.022	0.178	0.017
titanic	0.220	0.010	0.010	0.012	0.103	0.103	0.008	0.065	0.041
twonorm	0.027	0.016	0.026	0.007	0.029	0.006	0.027	0.009	0.026
usps	0.103	0.075	0.071	0.020	0.278	0.141	0.072	0.190	0.072
waveform	0.118	0.047	0.040	0.019	0.179	0.120	0.045	0.179	0.042
wisconsin	0.028	0.007	0.011	0.015	0.006	0.013	0.027	0.005	0.008

Table 1: Model selection results for selecting the kernel width σ and the soft margin parameter C simultaneously. For each training set and each bound we chose the parameters σ and C for which the bound was minimal, and evaluated the test error for the chosen parameters. Shown are the mean values of the test errors over the different training sets. The first column contains the value of the test error. The other columns contain the offset by which the test errors achieved by the different bounds are worse than the optimal test error.

responding to the support vector information, the term $\log^* n + \log n$ corresponding to the position of the codebook vector, the term $(r+1)\log m + r$ corresponding to the points inside the rotation region, and the term $(l+1)\log m$ corresponding to the information on the misclassified points outside the rotation region. For each fixed soft margin parameter C we chose the kernel width σ where the value of the compression coefficient C_3 is minimal. For this value of σ, we plotted the means of the different terms. Here we only show the plots for compression coefficient C_3 (as C_3 has more different terms than C_2) on the first six data sets (in alphabetical order). The plots on the other data sets, as well as the plots for C_2, are very similar to the ones we show. We find that all terms (except the term "misclass. inside" which is negligible) are of the same order of magnitude, with no term consistently dominating the other ones. This behavior is attractive as it shows that all different parts of information have substantial influence on the value of the compression coefficient.

Finally we want to study the shapes of the curves of the different bounds. As we use the values of the bounds to predict the qualitative behavior of the test error it is important that the shapes of the bounds' curves are similar to the shape of the test error curve. In Figure 6 we plot those shapes for the compression coefficients C_2, C_3, and the other bounds versus the kernel width σ. We show the plots for every second value of C (to cover the whole range of values of C we used) and for the first six data sets (in alphabetical order). First of all we can observe that the value of the compression coefficient is often larger than 1. This is also the case for several of the other bounds, and is due to the fact that most bounds only yield nontrivial results for very large sample sizes. This need not be a problem for applications as we use the bounds to predict the qualitative behavior of the test error, not the quantitative one. Secondly, the compression scheme and the sparse margin bound suffer from the fact that they only attain finite values when the number of support vectors is smaller than the number of training vectors. Among all bounds, only C_2, C_3 and the span bound seem to be able to predict the shape of the test error curve.

The main conclusions we can draw from all experimental results is that in all three tasks (predicting the shape of the test error curve, choosing parameter σ, choosing parameters σ and C) the span bound and compression coefficient C_2 have the best performance among all bounds, where none of the two bounds is clearly superior to the other one. The latter fact is also remarkable for the following reason. All considered bounds apart from the span bound use the capacity of the model class to bound the expected risk of the classifier. The span bound on the other hand is a clever way of computing an upper bound on the leave-one-out error of the classifier, which is known to be an almost unbiased estimator of the true risk. Thus the methods by which those bounds are derived are intrinsically different. Our results now show that the bounds derived by studying the size of the model class can achieve results in practice that are comparable to using the state of the art span bound.

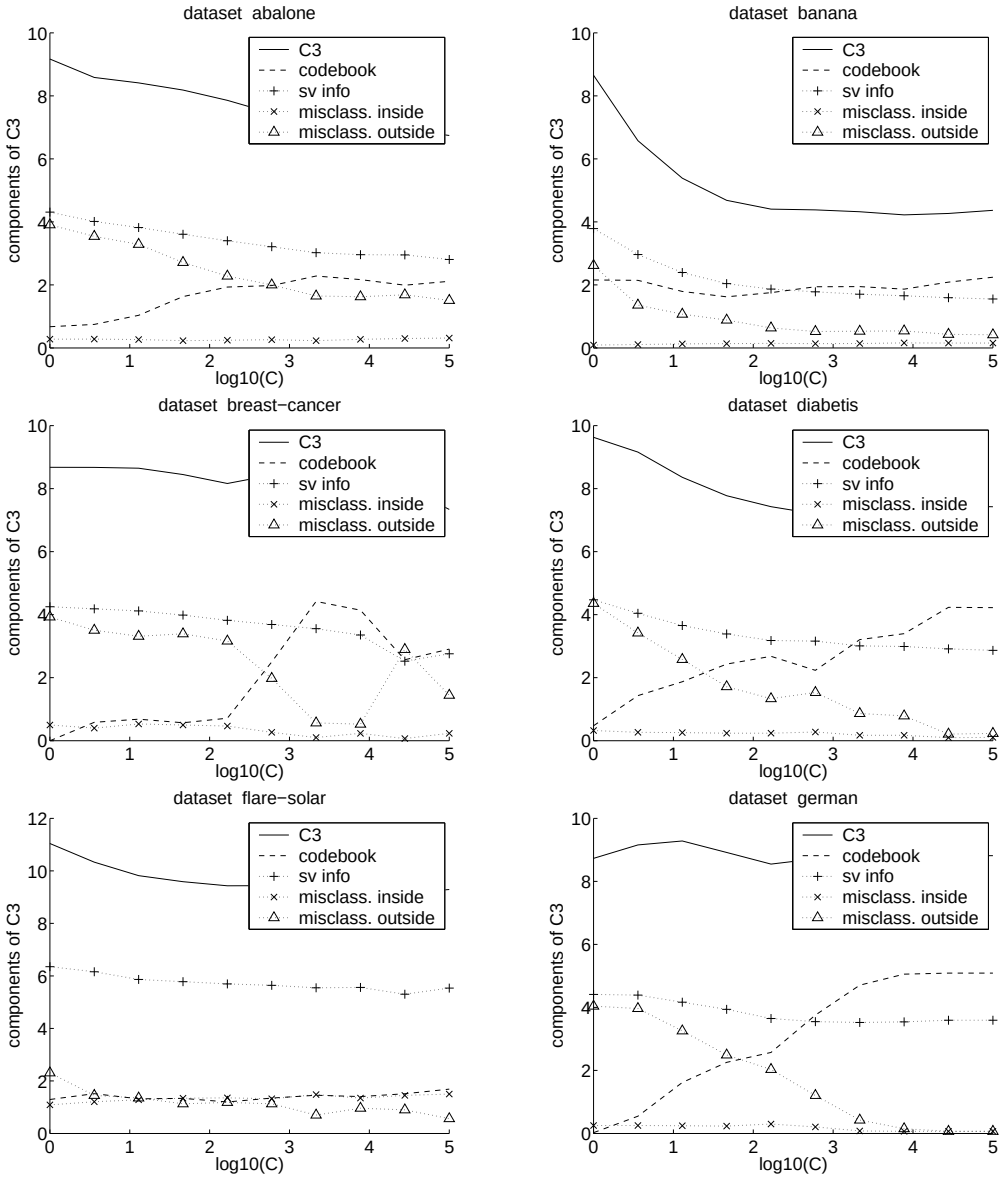


Figure 5. Here we study the relationship between the different components of C_3. We plot the lengths of the codes for the support vector information ("sv info"), the position of the codebook vector ("codebook"), the information on the points inside the rotation region ("misclass. inside"), and the information about the misclassified points outside the rotation region ("misclass. outside").

4. Conclusions

We derived five compression coefficients for SVMs which combine information on the geometry of the training data in the feature space with information about geometry and sparsity of the classifier. In our model selection experiments it turned out that the compression coefficients can be readily used to predict the parameters where the test error is small. Our favorite compression coefficient is

C_2 because it is easy to compute and yields good results in the experiments. The results it achieves are comparable to those of the state of the art span bound. The theoretical justification for using compression coefficients are the generalization bounds we cited in Section 2. They were proved in an abstract coding theoretic setting. We now derived methods to apply these bounds in practical applications. This shows that the connection between information theory and learning can be exploited in every-day machine learning applications.

Acknowledgements

We thank the anonymous reviewers for comments that significantly improved the quality of the manuscript.

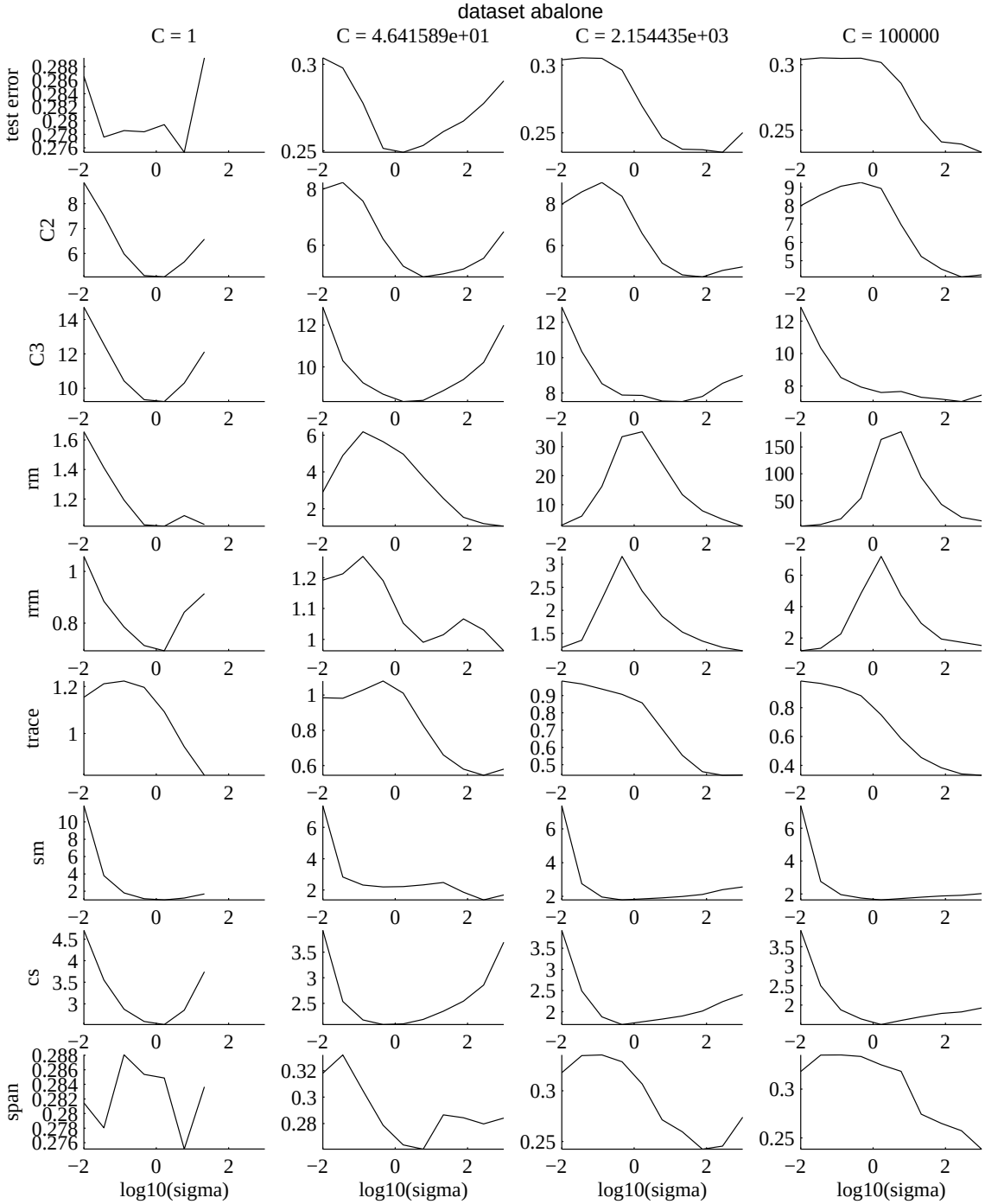


Figure 6: Shapes of curves. Plotted are the mean values of the bounds themselves over the different training runs versus the kernel width σ, for fixed values of the soft margin parameter C.

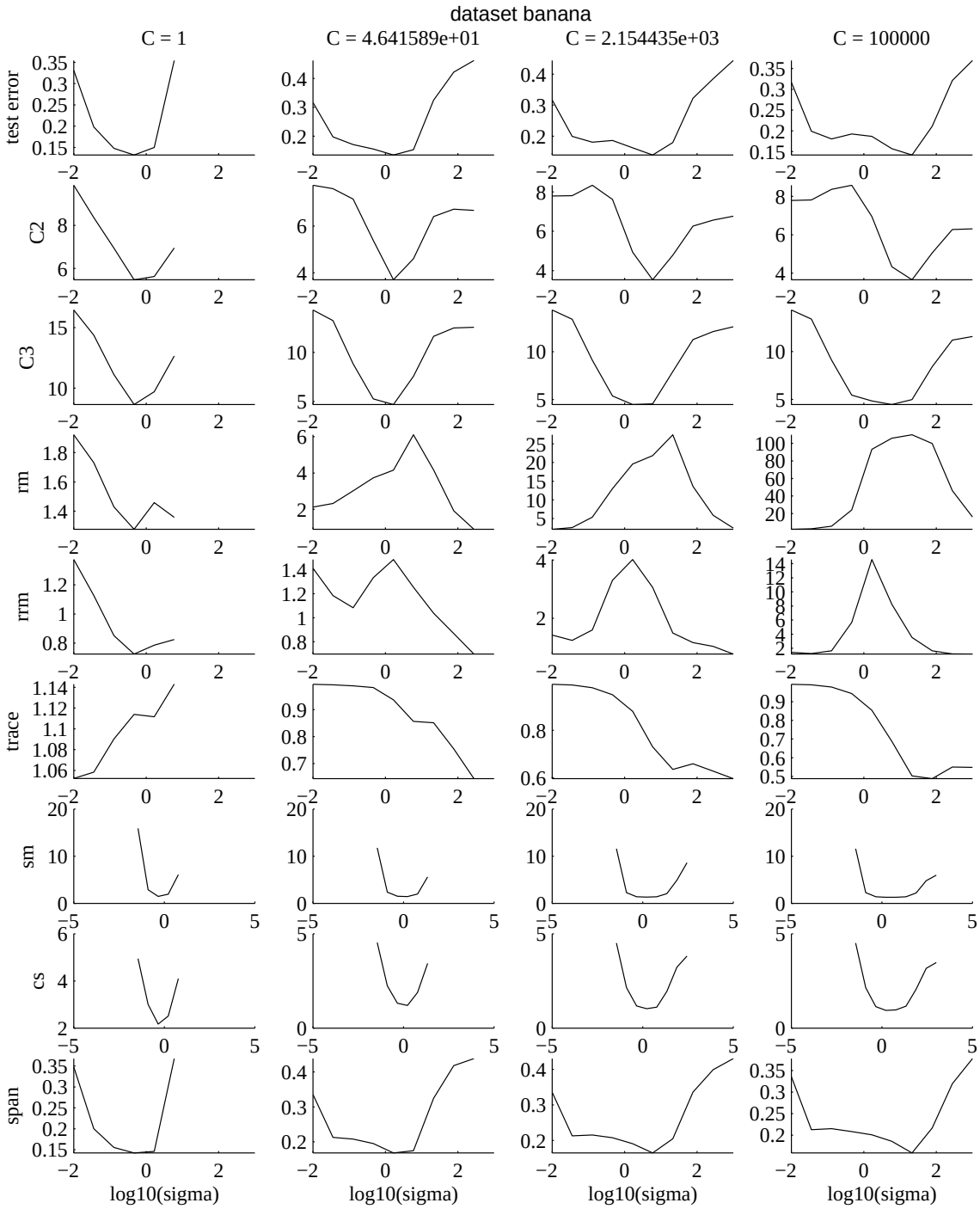


Figure 6, continued

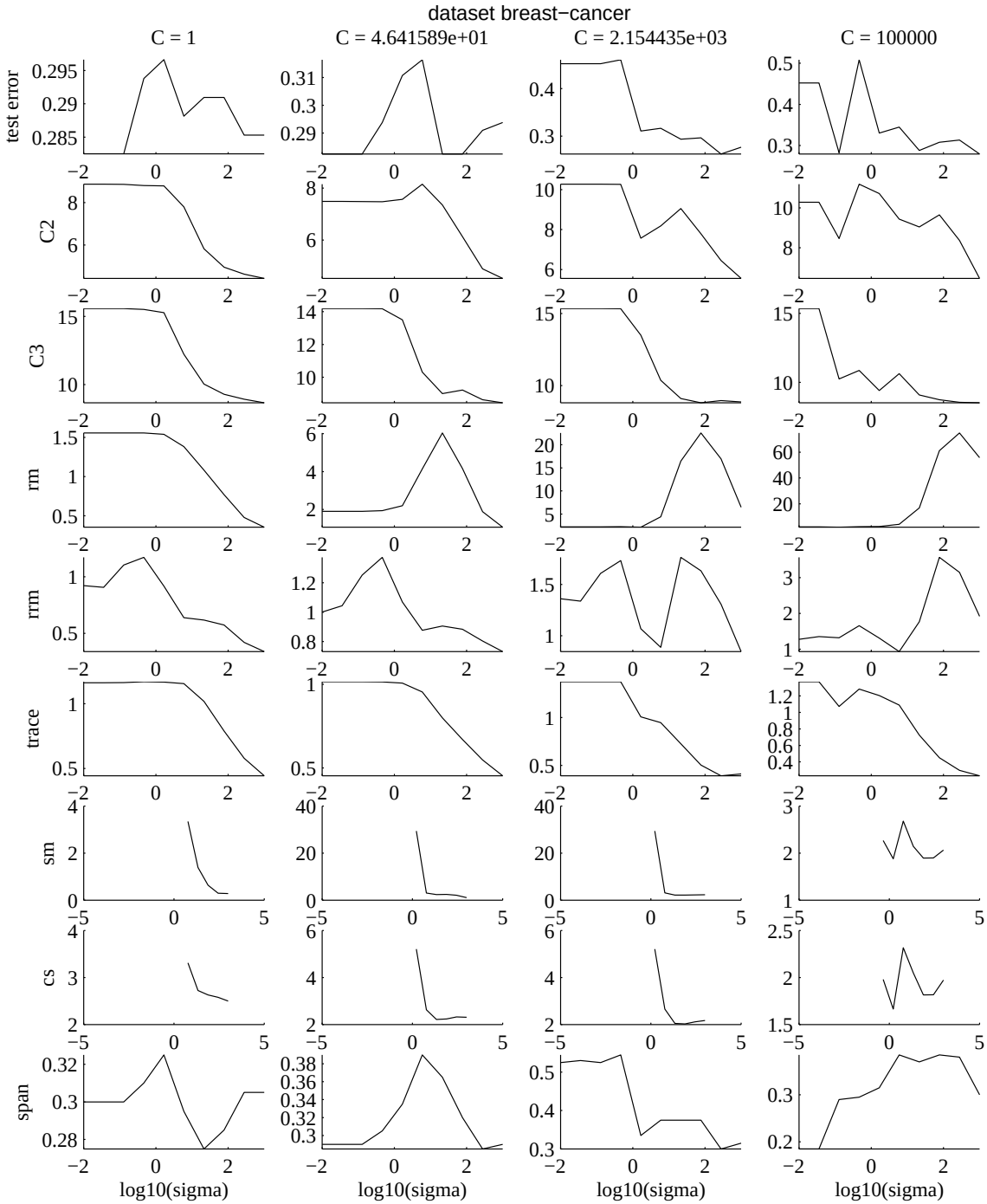


Figure 6, continued

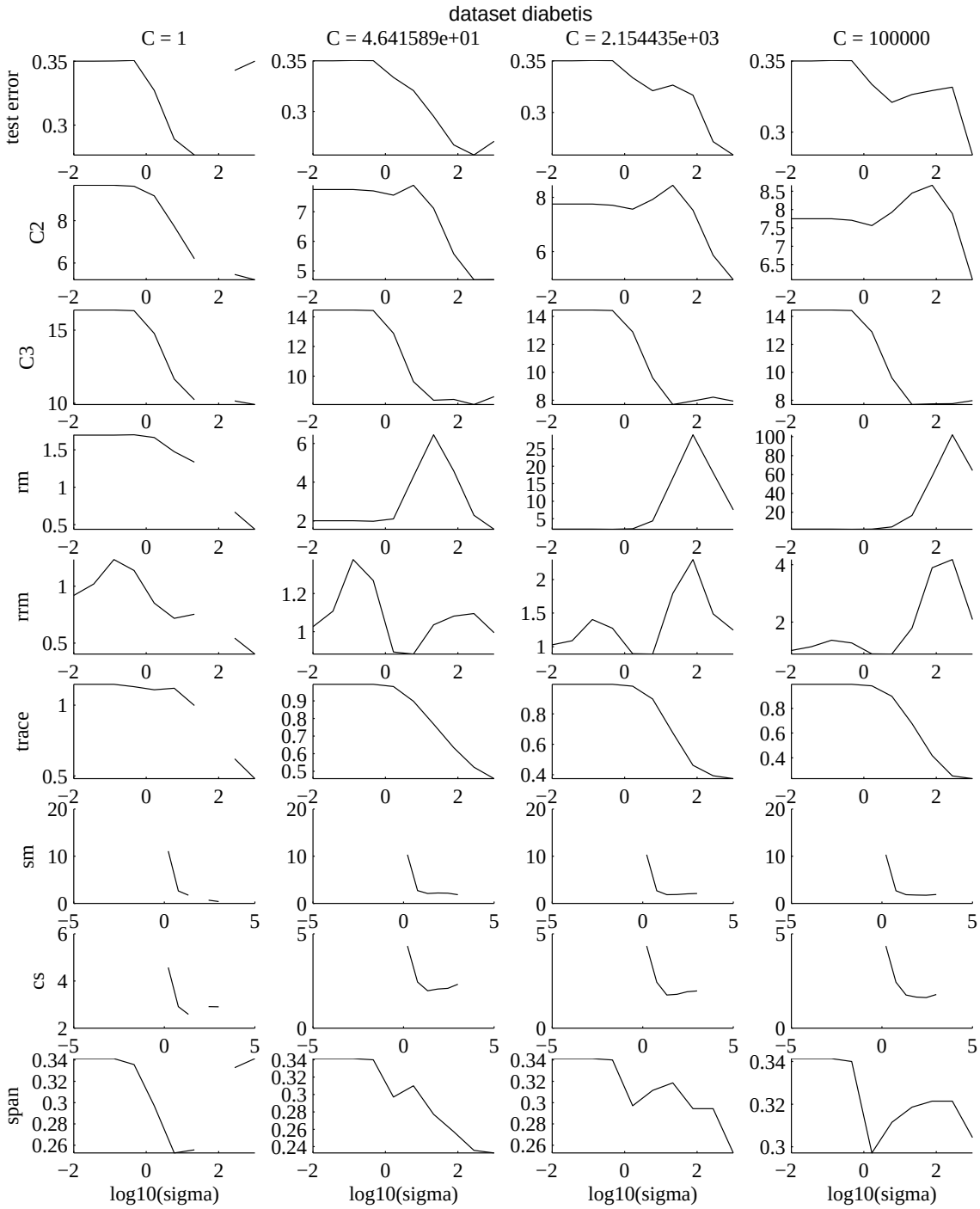


Figure 6, continued

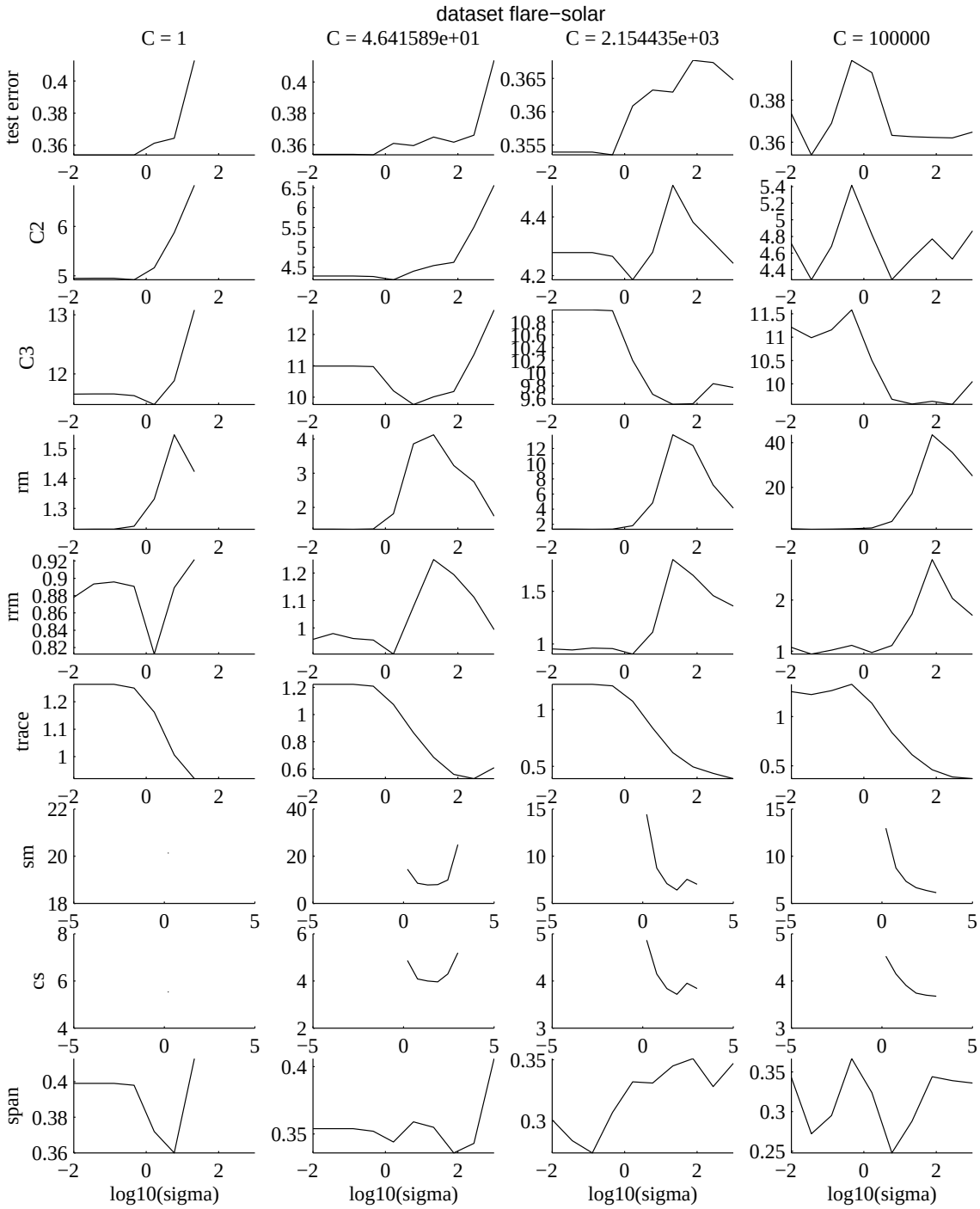


Figure 6, continued

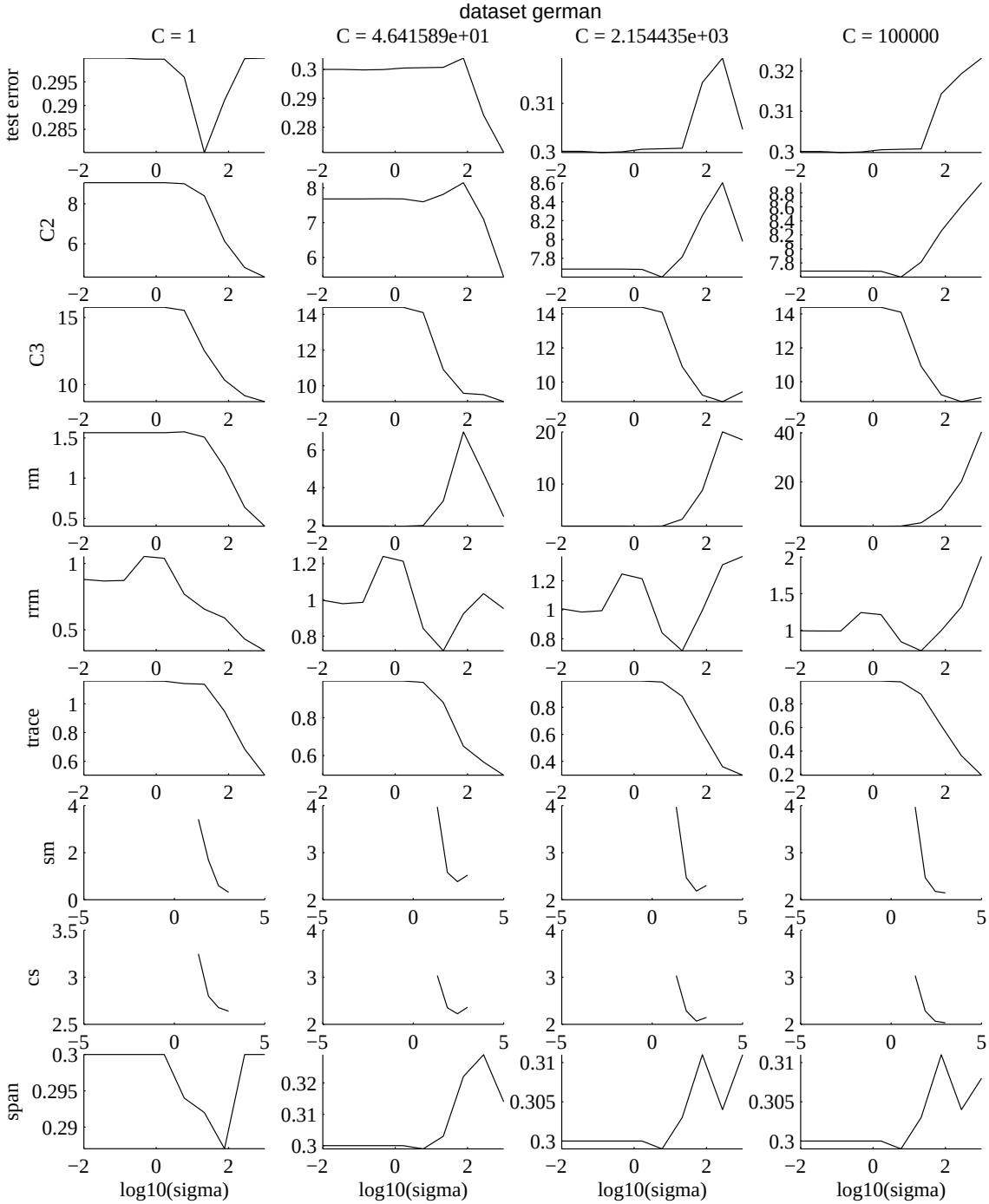


Figure 6, continued

References

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743 – 2760, 1998.

P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. In D. Helmbold and B. Williamson, editors, *Proceedings of the 14th annual conference on Computational Learning Theory*, pages 273–288, 2001.

P. L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 43–54. MIT Press, 1999.

A. Blum and J. Langford. PAC-MDL bounds. In B. Schölkopf and M.K. Warmuth, editors, *Learning Theory and Kernel Machines*, pages 344–357. 16th Annual Conference on Learning Theory, Springer, 2003.

O. Chapelle and V. Vapnik. Model selection for support vector machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

S. Floyd and M. K. Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

R. Herbrich, T. Graepel, and J. Shawe-Taylor. Sparsity vs. large margins for linear classifiers. In N. Cesa-Bianchi and S. Goldman, editors, *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 304–308. Morgan Kaufmann, 2000.

D. McAllester. Some PAC–Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

M. Opper and O. Winther. Gaussian processes and SVM: Mean field and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 311–326. MIT Press, 2000.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3): 287–320, March 2001.

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

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000.

Journal of Machine Learning Research 5 (2004) 325-360 Submitted 8/02; Revised 12/03; Published 4/04

A Geometric Approach to Multi-Criterion Reinforcement Learning

Shie Mannor SHIE@MIT.EDU
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Nahum Shimkin SHIMKIN@EE.TECHNION.AC.IL
Department of Electrical Engineering
Technion, Israel Institute of Technology
Haifa 32000, Israel

Editor: Sridhar Mahadevan

Abstract

We consider the problem of reinforcement learning in a controlled Markov environment with multiple objective functions of the long-term average reward type. The environment is initially unknown, and furthermore may be affected by the actions of other agents, actions that are observed but cannot be predicted beforehand. We capture this situation using a stochastic game model, where the learning agent is facing an adversary whose policy is arbitrary and unknown, and where the reward function is vector-valued. State recurrence conditions are imposed throughout. In our basic problem formulation, a desired target set is specified in the vector reward space, and the objective of the learning agent is to *approach* the target set, in the sense that the long-term average reward vector will belong to this set. We devise appropriate learning algorithms, that essentially use multiple reinforcement learning algorithms for the standard scalar reward problem, which are combined using the geometric insight from the theory of approachability for vector-valued stochastic games. We then address the more general and optimization-related problem, where a nested class of possible target sets is prescribed, and the goal of the learning agent is to approach the smallest possible target set (which will generally depend on the unknown system parameters). A particular case which falls into this framework is that of stochastic games with average reward constraints, and further specialization provides a reinforcement learning algorithm for constrained Markov decision processes. Some basic examples are provided to illustrate these results.

1. Introduction

Agents that operate in the real world often need to consider several performance criteria simultaneously. In this paper we address the problem of RL (Reinforcement Learning) in a dynamic environment, where the controlling agent's goals are formulated in terms of multiple objective functions, each one corresponding to a long-term average reward functional. Furthermore, we deviate from the strictly Markovian environment model by allowing the presence of additional agents whose policies may be arbitrary. Our goal then is to formulate performance objectives that are meaningful in such a setting, and develop learning algorithms that attain these objectives.

Multi-criterion decision making is a well established research area with a wide range of solution concepts and methods (for an overview see, for example, Steuer, 1986; Ehrgott and Gandibleux,

2002). Most basic to multi-criterion optimization is the concept of efficient (or Pareto optimal) solutions, namely those solutions that cannot be improved upon in all coordinates (with strict improvement is at least one coordinate) by another solution. Unfortunately efficient solutions are essentially non-unique, and various methods have been developed to single out one "optimal" solution" — for example, by forming appropriate scalar combinations of the different objective functions. Another formulation of multi-criterion optimization which leads to a well-defined solution is the constrained optimization problem, where one criterion is optimized subject to explicit constraints on the others. In the context of Markov decision problems (MDPs), several papers have developed dynamic programming algorithms to compute efficient solutions (see White, 1982; Henig, 1983; Carraway et al., 1990), while constrained MDPs have received more extensive attention — see Altman (1999) and references therein. Constrained Stochastic Games were considered by Shimkin (1994) and Altman and Shwartz (2000). Learning schemes for constrained MDPs were devised by Pozniyak et al. (1999), based on the theory of stochastic learning automata. A Q-learning approach to constrained MDPs was considered by Gábor et al. (1998).

In the present paper, we shall use the notion of a *target set* as the basic concept which specifies the controller's goals with respect to its multiple objective functions. This set specifies the range in which the vector of objective functions is required to reside. More precisely, since each objective function is in the form of a long-range average reward, the controller's objective is to ensure that the average-reward vector converges to the given target set. Note that in its basic form the target set concept does not correspond to an optimization problem but rather to constraint satisfaction. However, the optimization aspect may be added by looking for target sets that are minimal in some sense; indeed, we will employ this approach in the latter part of the paper to obtain a learning algorithm for constrained stochastic games (and, in particular, constrained MDPs).

As the target set concept is basic to our problem formulation, it is worth taking some time to highlight its relevance and utility in dynamic decision problems. In various engineering applications, performance requirements are given in the form of bounds on variables of interest, that need to hold under varying system conditions (parameters) and external disturbances. A temperature regulator, for example, may be required to keep the temperature with a certain range (say $\pm 1^o$C) of the nominal operating point. The corresponding target set is then simply the interval $[\underline{T}, \overline{T}]$. In communication networks which provide quality of service (QoS) guarantees, such as certain service classes of ATM networks, a user may be given guarantees in terms of a Minimal Cell Rate (MCR), which lower bounds the available bandwidth, and a Cell Loss Ratio (CLR) that upper bounds the allowed fraction of lost cells (e.g., ATM Forum Technical Committee, 1999). The controller's goal is to keep BW $\geq$ MCR and LR $\leq$ CLR$_0$, where BW is the available bandwidth and LR is the fraction of lost cells. This translates to a target set T of the form $T = [\text{MCR}, \infty) \times [0, \text{CLR}]$, which resides in the two-dimensional objective space $\mathbb{R}^2$. A more elaborate service guarantee may ensure a certain loss ratio up to a given cell rate, and allow for a higher one when the cell rate is higher; the corresponding target set can still be easily constructed in $\mathbb{R}^2$, although it will no longer be rectangular due to the coupling between the two criterions. We note that regulating mechanisms are equally prevalent in biological systems, where such quantities as temperature, pressure and concentration need to be maintained within certain bounds.

We observe, in passing, that goal setting and their attainment play a central role in prominent descriptive models of human decision making. Simon's theory of *satisfying* decisions suggests that actual decision makers are rarely optimizers, but rather accept "good enough" alternatives that meet previously determined aspiration levels (Simon, 1996). Kahaneman and Tversky's prospect theory

(Kahaneman and Tversky, 1979) evaluates gain or loss relative to a preset reference point, which can also be interpreted as an aspiration level for the decision maker.

A second challenging ingredient in our problem formulation is that of the arbitrarily varying environment. The decision problem is considered from the viewpoint of a particular controlling agent, whose environment may be affected by other, noncooperative, agents. We make no assumptions regarding the choices of these other agents, and allow them to choose their actions arbitrarily. Such agents may also represent non-stationary moves of Nature, or account for non-Markovian dynamics over a partially-observed state. For modelling simplicity we collect all the additional agents as a single one, the *adversary*, whose actions may affect both the state transitions and obtained rewards. The model then reduces to a two-person stochastic game with vector-valued rewards. The adversary is free to choose its actions according to any control policy, subject only to causality limitations (both agents are assumed to fully observe the state and action sequences). Accordingly, we shall require that the controlling agent will achieve the required goal (convergence of the average reward vector to the target set) *for any policy of the adversary*.

The proposed problem formulation is inspired by the theory of approachability, introduced by Blackwell (1956) in the context of repeated matrix games with vector payoffs. This theory provides geometric conditions, and matching control policies, for a set in the reward space to be *approachable* by a player, in the sense that the average reward vector approaches this target set for any policy of the adversary. Essentially, Blackwell approaching strategies are based on reviewing the current average reward vector, and whenever it is outside the target set, the reward is "steered" in the direction of the target set. More precisely, the controlling agent computes the direction of closest distance from current reward vector to the target set, and plays his maximin policy in the matrix game obtained by projecting the vector-valued reward function along this direction. Among its many extensions and applications, approachability theory has been extended to stochastic (Markov) games by Shimkin and Shwartz (1993) under appropriate state recurrence conditions; these conditions are enforced throughout the present paper. The relevant results are reviewed in Section 3.1 and some new results are provided in Section 3.2. In this paper we add the learning aspect, and consider the problem of learning approaching policies on-line. Accordingly, we refer below to the controlling agent as the *learning* agent.

It will be useful at this point to illustrate some of the geometric ideas behind Blackwell's approaching policies via a simple example. Note that this example is strictly illustrative. Consider first a scalar temperature regulation problem, where the performance objective of interest is to regulate the *long-term average* temperature, and guarantee that it is in some prescribed interval $[\underline{T}, \overline{T}]$. The agent may activate a cooler or a heater at will, although when activated it must remain so for a certain period of time. When the cooler is activated the temperature drops to a range $[a,b]$, with $b < \underline{T}$, and when the heater is on the temperature reaches a range $[c,d]$, with $c > \overline{T}$. The precise temperature reached in the allocated range varies due to environmental influence. Given this uncertainty, the simplest way to regulate the average temperature would be to keep note of the current average temperature (over the time interval between the initial and present time), and then activate the cooler whenever the current average increases above $\overline{T}$, and the heater whenever it decreases below $\underline{T}$: see Figure 1 for an illustration. In between these levels, any (reasonable) activation rule may be used. It should be noted that the proposed policy is not stationary in the usual sense, since it depends on the current *average* temperature, which is not part of a standard definition of a state for this system. We also note the similarity with the common "bang-bang" (or threshold) feedback policy for temperature regulation: the goal there is of course to regulate the *instantaneous* temper-

ature, which is accordingly used as the feedback variable. It is not possible (due to the assumed relationships between the temperature intervals) to maintain the instantaneous temperature in the specified interval.

Figure 1: The single dimensional temperature regulation example. If the temperature is higher than $\overline{T}$ the control policy is to cool, and if the temperature is lower than $\underline{T}$ the control policy is to heat.

This one-dimensional example already contains one of the key elements of the approaching policies. Namely, the average reward vector is "steered" in the direction of the target set by using direction dependent sub-policies. While each sub-policy may itself be stationary, the overall approaching policy is not. Except for the special case when the adversary is trivial (so that the model reduces to a single-controller one), the use of such non-stationary policies is essential, in the sense that target sets that are approachable with general polices need not be so with stationary policies.

Consider next a multi-objective version of the temperature regulation problem. The controller's first objective, as before, is to maintain the average temperature in a certain range. One can consider other criteria of interest such as the average humidity, frequency of switching between policies, average energy consumption and so on. This problem may be characterized as a multi-criterion problem, in which the objective of the controller is to have the average reward in some target set. Suppose that the additional variable of interest is the average humidity. In a controlled environment such as a greenhouse, the allowed level of humidity depends on the temperature. An illustrative target set is shown in Figure 2. A steering policy now relies on more elaborated geometry. In place of the two directions (left/right) of the one-dimensional case, we now face a continuous range of possible directions, each associated with a possibly different steering policy. For the purpose of the proposed learning algorithm, it will be useful to consider only a finite number of steering directions (and associated policies). We will show that this can always be done, with negligible effect on the attainable performance.

The analytical basis for this work relies on three elements: the theory of approachability for vector-valued dynamic games; RL algorithms for (scalar) average reward problems; and stochastic game models. As approachability was already introduced above, we next briefly address the other two topics.

Reinforcement Learning has emerged in the last decade as a unifying discipline for learning and adaptive control. Comprehensive overviews may be found in Bertsekas and Tsitsiklis (1995); Sutton and Barto (1998), and Kaelbling et al. (1996). It should be mentioned that reinforcement learning has come to encompass a wide range of learning methods, which surpasses the restricted sense of adapting actions directly in response to a (scalar) reinforcement signal; our method belongs in this wider class. RL for average reward Markov Decision Processes (MDPs) was suggested by Schwartz (1993) and Mahadevan (1996) and later analyzed in Abounadi et al. (2002). The analysis for the average reward criterion is considerably more difficult than the discounted case, as the dynamic programming operator is no longer a contraction. Several methods exist for average reward RL,

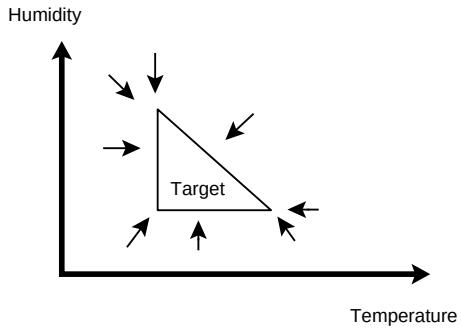


Figure 2: The two dimensional temperature-humidity example. The steering directions, which point to the closest point in the set, are denoted by arrows.

including Q-learning (Abounadi et al., 2002), the E^3 algorithm (Kearns and Singh, 1998), and actor-critic schemes (Sutton and Barto, 1998). RL for multi-objective problems was considered, to the best of our knowledge, only by Gábor et al. (1998). This paper considers discounted constrained MDPs, that is, the objective is to find a maximizing policy for the discounted reward problem, given that some additional discounted constraints are satisfied. The algorithm of Gábor et al. (1998) searches for the best *deterministic* policy. In general, however, the optimal policy for constrained MDPs is not deterministic (e.g., Altman, 1999).

Stochastic Games (SGs) present a flexible model of dynamic conflict situations, with extensive theory and various applications in economics, operations research, and more. For recent overviews see Filar and Vrieze (1996) and Mertens (2002). In particular, existence of a state-independent value and stationary optimal policies is established for average-reward stochastic games under appropriate recurrence conditions, which are assumed in this paper. Stochastic games provide a natural generalization of the single-controller Markov decision problem to the multi-agent setting, and are highly relevant to various RL applications, starting with board games. The naive learning approach ignores possible non-stationarity of the opponent and simply applies standard RL algorithms as used for MDPs. This approach has shown some notable success (Baxter et al., 1998), but provides no performance guarantees. The seminal work of Tesauro (1996) is a notable example of an algorithm that learns under the assumption that the opponent play is stationary and adversarial, but provides no performance guarantees. Learning algorithms for zero-sum stochastic games with discounted rewards have been introduced by Littman (1994) and shown to converge to the max-min optimal policies (Littman and Szepesvári, 1999). Corresponding algorithms for the average reward problem have emerged only recently, and include model-based learning (Brafman and Tennenholtz, 2002), and Q-learning (Mannor and Shimkin, 2002).

Our main results are as follows. We present two learning algorithms for approaching a prescribed target set. The first (Multiple Directions Reinforcement Learning) algorithm is based on discretizing the set of possible steering directions, using a (dense) grid of direction vectors, and maintaining a learning algorithm for the stochastic games induced by each grid direction; the (scalar) reward function for each game is obtained by projecting the vector reward function on the corresponding direction. The second (Current Direction Reinforcement Learning) algorithm focuses on a single direction at a time, which is frozen for a long enough period so that both learning and steering

are assured. Both algorithms are efficient, in the sense that they approach (to required accuracy) any target set that is approachable by steering policies when the model is known.

Equipped with the two basic algorithms, we consider the important extension to variable target sets. Here the target set to be approached is not fixed *a-priori*. Indeed, the target set of interest may depend on the model parameters, which are initially unknown. Our starting point here will be a *nested* family of target sets, with the goal of the learning agent naturally defined as approaching the smallest possible set in this family (note that approaching a smaller set automatically implies that any larger set in the nested family is also approached). We present a suitable learning algorithm which combines the basic algorithms with an on-line estimate of the minimally approachable target set. We will then show that this framework can be specialized to give a solution to the important problems of constrained SGs and MDPs.

We shall also briefly consider an alternative algorithm for approaching a (fixed) target set, which is based on the adversarial multi-armed bandit problem (e.g., Auer et al., 2002). This algorithm is suitable for the case where there is only a small set of sub-strategies that the controlling agent may alternate between. The idea here is to treat each of the possible strategies as an arm, and make sure that after enough rounds the reward is as high as that of the best arm (policy). Again, we show that the algorithm attains the same performance as in the known model case.

The paper is organized as follows. In Section 2 we describe the stochastic game model and formulate the control objective. In Section 3 we recall basic results from approachability theory, and prove that it suffices to use finitely many steering directions in order to approach a target set with a given precision. We also discuss RL for scalar-valued SGs and define two notions of optimality that will be used in the sequel. The next two sections describe the Multiple Directions RL algorithm and the Current Direction RL algorithm, while the bandit-based algorithm is described in Section 6. The extension to variable target sets is discussed in Section 7, which also discusses constrained problems. In Section 8 we briefly discuss the specialization of the previous results to (single-controller) MDPs with multiple objectives. Two examples are described in Section 9, followed by concluding remarks. An appendix contains a refined algorithm and its convergence proof.

2. Multi-Criterion Stochastic Games

In this section we present the multi-criterion SG model. We describe the learning agent's objective and define a related scalar projected games which will be useful in the sequel.

2.1 Model Definition

We consider a two-person average reward SG, with a vector-valued reward function. We refer to the players as P1 (the learning agent) and P2 (the arbitrary adversary). The game is defined by the following elements:

1. $\mathcal{S}$, a finite state space.
2. $\mathcal{A}$ and $\mathcal{B}$, finite sets of actions for P1 and P2, respectively. To streamline our notation it is assumed that P1 and P2 have the same action sets available in each state.
3. $P = P(s'|s,a,b)$, the state transition function.
4. $m : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}^k$, a vector-valued reward function.

At each time epoch $n \geq 0$, both players observe the current state s_n, and then P1 and P2 simultaneously choose actions a_n and b_n, respectively. As a result P1 receives the reward vector $m_n = m(s_n, a_n, b_n)$ and the next state is determined according to the transition probability $P(\cdot|s_n, a_n, b_n)$. More generally, we allow the actual reward m_n to be random, in which case $m(s_n, a_n, b_n)$ denotes its mean and we assume the reward is bounded.[1] We further assume that both players observe the previous rewards and actions (however, in some of the learning algorithms below, the assumption that P1 observes P2's action may be relaxed). A policy $\pi \in \Pi$ for P1 is a mapping which assigns to each possible observed history a mixed action in $\Delta(\mathcal{A})$, namely a probability vector over P1's action set $\mathcal{A}$. A policy $\sigma \in \Sigma$ for P2 is defined similarly. A policy of either player is called *stationary* if the mixed action it prescribes depends only on the current state s_n. The set of stationary policies of P1 is denoted by F, and that of P2 by G. Let $\hat{m}_n$ denote the average reward by time n:

$$\hat{m}_n \stackrel{\triangle}{=} \frac{1}{n} \sum_{\tau=0}^{n-1} m_\tau .$$

Note that we do not define a reward for P2. Since we allow P2 to use an arbitrary policy, such reward function, even if it exists, is irrelevant to our formulation.

The following recurrence assumption will be employed. Let state s^* denote a specific reference state to which a return is guaranteed. We define the renewal time of state s^* as

$$\tau \stackrel{\triangle}{=} \min\{n > 0 : s_n = s^*\} . \tag{1}$$

Assumption 1 (Recurrence) *There exist a state $s^* \in \mathcal{S}$ and a finite constant N such that*

$$\mathbb{E}^s_{\pi,\sigma}(\tau^2) < N \quad \text{for all } \pi \in \Pi, \sigma \in \Sigma \text{ and } s \in \mathcal{S},$$

where $\mathbb{E}^s_{\pi\sigma}$ is the expectation operator when starting from state $s_0 = s$ with policies π and σ for P1 and P2, respectively.

For finite state and action spaces this assumption is satisfied if state s^* is accessible from all other states under any pair of stationary deterministic policies (Shimkin and Shwartz, 1993). We note that Assumption 1 holds if the game is irreducible (as defined in Hoffman and Karp, 1966) or ergodic (as defined in Kearns and Singh, 1998).

Remark 1 *Assumption 1 may be relaxed in a similar manner to Mannor and Shimkin (2000). There, it is assumed that state s^* is* reachable*, that is, either player has a policy that guarantees a return to s^*. The approachability result that is obtained under this related assumption is somewhat more complicated, and we adhere here to Assumption 1 for simplicity.*

Remark 2 *In the absence of any recurrence conditions, tight approachability conditions and corresponding policies are extremely hard to obtain. This may be attributed to the dependence of the minimax value of such games (with scalar reward) on the initial state. Still, non-tight sufficient conditions and corresponding learning algorithms may be obtained by requiring the approachability conditions below to hold for all initial states. Again, we shall not pursue this direction here.*

1. Bounded reward is required essentially only for the SPRL algorithm from Section 6. This assumption can be relaxed for the algorithms of Sections 5,4, and 7 to requiring bounded second moments. The result in the appendix is also proved using the weaker assumption of bounded second moment.

2.2 P1's Objective

P1's task is to approach a target set T, namely to ensure convergence of the average reward vector to this set irrespectively of P2's actions. Formally, let $T \subset \mathbb{R}^k$ denote the target set. In the following, d is the Euclidean distance in $\mathbb{R}^k$. The set-to-point distance between a point x and a set T is $d(x,T) \stackrel{\triangle}{=} \inf_{y\in T} d(x,y)$. We let $P^s_{\pi,\sigma}$ denote the probability measure on the states and rewards sequences when P1 plays the policy π, P2 plays policy σ, and the initial state is s.

Definition 3 *A policy π^* of P1* approaches *a set $T \subset \mathbb{R}^k$ (from initial state s) if*

$$\lim_{n\to\infty} d(\hat{m}_n, T) = 0 \quad P^s_{\pi^*,\sigma}\text{-a.s., for every } \sigma \in \Sigma.$$

A policy $\sigma^* \in \Sigma$ of P2 excludes *a set T (from initial state s) if for some $\delta > 0$,*

$$\liminf_{n\to\infty} d(\hat{m}_n, T) > \delta \quad P^s_{\pi,\sigma^*}\text{-a.s. for every } \pi \in \Pi,$$

where a.s. stands for almost surely.

The policy π^* (σ^*) will be called an approaching (excluding) policy for P1 (P2). A set is approachable if there exists an approaching policy from all states. Noting that approaching a set and its topological closure are the same, we shall henceforth suppose that the set T is closed.

Remark 4 *The original definition of approachability by Blackwell (1956) required uniformity of convergence with respect to P2's policy. Some of our results do hold uniformly with respect to P2's policy and some do not. Specifically, the rate of convergence in the algorithms described in Sections 5 and 6 is uniform with respect to P2's policy. The convergence rate of the algorithm described in Section 4 is not uniform unless further assumptions are made. The issue of uniformity of convergence was not pursued here.*

Remark 5 *While the main focus in this paper is on the long-term average criterion, the target set concept does allow some consideration of instantaneous quantities. For example, recall the temperature regulation problems presented in the Introduction. Suppose that the instantaneous temperature (*in addition *to its average) is required to stay in some different range $[T_{\min}, T_{\max}]$. We can add a component to the reward vector, which indicates deviation from the required range: i.e., it takes the value '1' if the instantaneous temperature is outside this range, and '0' otherwise. We can now define the target set so that the long-term average of this component is required to approach 0, thereby assuring that the temperature stays in the required range save for a negligible fraction of times. Thus, both instantaneous quantities and their time averages can be considered in a unified framework.*

2.3 The Projected Scalar Game

Let u be a unit vector in the reward space $\mathbb{R}^k$. We often consider the *projected game in direction u* as the zero-sum stochastic game with the same dynamic as above, and *scalar* rewards $r_n \stackrel{\triangle}{=} m_n \cdot u$. Here "$\cdot$" stands for the standard inner product in $\mathbb{R}^k$. Denote this game by $\Gamma_s(u)$, where s is the initial state. The scalar stochastic game $\Gamma_s(u)$, has a *value*, denoted $\mathrm{v}\Gamma_s(u)$, if

$$\begin{aligned} \mathrm{v}\Gamma_s(u) &= \sup_{\pi}\inf_{\sigma}\liminf_{n\to\infty} \mathbb{E}^s_{\pi\sigma}(\hat{m}_n \cdot u) \\ &= \inf_{\sigma}\sup_{\pi}\limsup_{n\to\infty} \mathbb{E}^s_{\pi\sigma}(\hat{m}_n \cdot u)\,. \end{aligned} \tag{2}$$

The value is known to exist for any game with finite state and action spaces (Mertens and Neyman, 1981). Furthermore, under Assumption 1 the value is independent of the initial state and can be achieved in stationary policies (Filar and Vrieze, 1996). We henceforth simply write $\Gamma(u)$ for the average reward zero-sum game which reward is $r_n = m_n \cdot u$, and denote its value by $\mathrm{v}\Gamma(u)$.

3. Preliminaries

In this section we recall the basic results of approachability theory for SGs. In Subsection 3.2 we extend approachability theory by considering approximate approachability. A target set is said to be *approximately approachable* if for every $\varepsilon > 0$ there exists a policy such that the distance between the average vector-valued reward vector and the target set is asymptotically less than ε. We prove that it suffices to consider finitely many steering directions in order to approximately approach a target set. We discuss relevant types of optimality of RL algorithms in scalar games in Subsection 3.3.

3.1 Approachability for Stochastic Games

We recall the basic results from Shimkin and Shwartz (1993) regarding approachability for known stochastic games, which generalize Blackwell's conditions for repeated matrix games. Let

$$\phi(\pi,\sigma) \stackrel{\triangle}{=} \frac{\mathbb{E}^{s^*}_{\pi,\sigma}(\sum_{n=0}^{\tau-1} m_n)}{\mathbb{E}^{s^*}_{\pi,\sigma}(\tau)}$$

denote the *average per-cycle* reward vector, which is the expected total reward over the cycle that starts and ends in the reference state, divided by the expected duration of that cycle. For any $x \notin T$, denote by C_x a closest point in T to x, and let u_x be the unit vector in the direction of $C_x - x$, which points from x to the goal set T. The following theorem requires, geometrically, that there exists a policy $\pi(x)$ such that the set of all possible (vector-valued) expected rewards is on the other side of the hyperplane supported by C_x in direction u_x. See Figure 3 for an illustration, the polygon represents the set of all possible rewards when $\pi(x)$ is used.

Theorem 6 (Shimkin and Shwartz, 1993) *Let Assumption 1 hold. Suppose that for every point $x \notin T$ there exists a policy $\pi(x)$ such that*

$$(\phi(\pi(x),\sigma) - C_x) \cdot u_x \geq 0\,, \quad \forall \sigma \in \Sigma\,. \tag{3}$$

Then T is approachable by P1. An approaching policy is given as follows: If $s_n = s^*$ and $\hat{m}_n \notin T$, play $\pi(\hat{m}_n)$ until the next visit to state s^*; otherwise, play arbitrarily until the next return to s^*.

Intuitively, the condition in Equation (3) means that P1 can ensure, irrespectively of P2's policy, that the average per-cycle reward will be on the other side (relative to x) of the hyperplane that passes through C_x and is perpendicular to the line segment that points from x to C_x. Note that this hyperplane actually separates T and x when T is convex. We shall refer to the direction u_x as the *steering direction* from point x, and to the policy $\pi(x)$ as the *directional steering policy* from x. The approaching policy uses the following rule: between successive visits to the reference state, a fixed (possibly stationary) policy is used. When in the reference state, the current average reward vector $\hat{m}_n$ is inspected. If this vector is not in T, then the steering policy that satisfies Equation (3) with

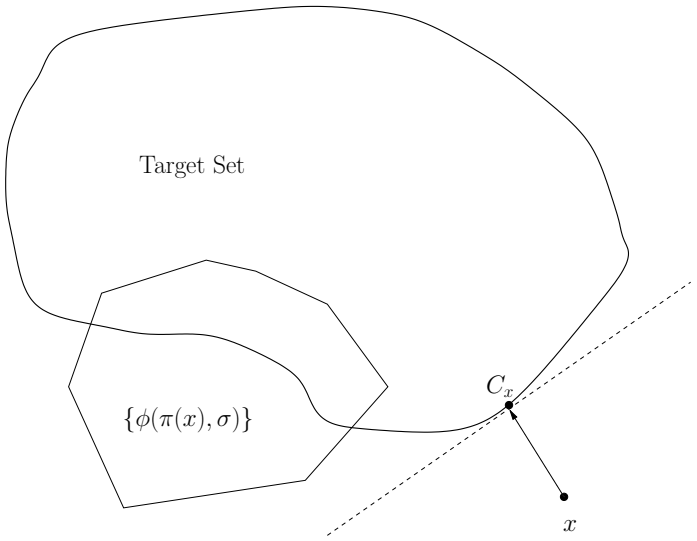


Figure 3: An illustration of the approachability condition. The location of the current average reward is x. The closest point to x in the target set is C_x. The polygon represents the set of all possible expected reward when $\pi(x)$ is played. It can be observed that $\pi(x)$ ensures that P1's reward will be on the other side of the hyperplane perpendicular to the line segment between C_x and x.

$x = \hat{m}_n$ is selected for the next cycle. As a consequence the average reward is "steered" towards T, and eventually converges to it.

Recalling the definition of the projected game in direction u and its value $\mathrm{v}\Gamma(u)$, the condition in Equation (3) may be equivalently stated as $\mathrm{v}\Gamma(u_x) \geq C_x \cdot u_x$. Furthermore, the policy $\pi(x)$ can always be chosen as the stationary policy which is optimal for P1 in the projected game $\Gamma(u_x)$. In particular, the directional steering policy $\pi(x)$ depends only on the corresponding steering direction u_x.

For *convex* target sets, the condition of the last theorem turns out to be both sufficient and necessary. Moreover, this condition may be expressed in a simpler form, which may be considered as a generalization of the minimax theorem for scalar games (Blackwell, 1956). Given a stationary policy $g \in G$ for P2, let $\Phi(F,g) \stackrel{\triangle}{=} \mathrm{co}(\{\phi(f,g)\}_{f\in F})$, where co is the convex hull operator. The Euclidean unit sphere in $\mathbb{R}^k$ is denoted by $\mathbb{B}^k$.

Theorem 7 (Shimkin and Shwartz, 1993) *Let Assumption 1 hold. Let T be a closed convex set in $\mathbb{R}^k$.*

(i) T is approachable if and only if $\Phi(F,g) \cap T \neq \emptyset$ for every stationary policy $g \in G$.

(ii) If T is not approachable then it is excludable by P2. In fact, any stationary policy g that violates (i) is an excluding policy.

(iii) T is approachable if and only if $\mathrm{v}\Gamma(u) \geq \inf_{m\in T} u \cdot m$ for every $u \in \mathbb{B}^k$.

3.2 Approachability with a Finite Set of Steering Directions

Standard approaching policies, as outlined above, may involve an infinite number of steering directions. The corresponding set of directional steering policies may turn out to be infinite as well. For the purpose of our learning scheme, we shall require an approaching policy which relies on a *finite* set of steering directions and policies. In this section we establish appropriate extensions of the results surveyed above. We prove that using a finitely many steering directions to approach a target set can indeed be done, possibly requiring to slightly expand the target set. Specifically, we establish that if the condition in Equation (3) is satisfied for the set T, then for any $\varepsilon > 0$ there exists a finite partition $\{U_1, U_2, \ldots, U_J\}$ of the unit sphere $\mathbb{B}^k$, and a corresponding set $\{\pi_1, \pi_2, \ldots, \pi_J\}$ of policies for P1, such that the required separation condition in Equation (3) may be satisfied within ε precision for any $u \in U_j$ by using the policy π_j. In the following, let M be an upper bound on the magnitude of the expected one-stage reward vector, so that $\|m(s,a,b)\| < M$ for all (s,a,b) ($\|\cdot\|$ denotes the Euclidean norm).

Proposition 8 *Let Assumption 1 hold and suppose that the target set $T \subset \mathbb{R}^k$ satisfies the condition in Equation (3). Fix $\varepsilon > 0$. Let $U = \{u_1, u_2, \ldots, u_J\}$ be a set of unit vectors which are the centers of open balls of radius $\varepsilon/2M$ that cover the unit sphere $\mathbb{B}^k$. Let π_j be the (stationary) policy which is optimal in the projected game $\Gamma(u_j)$. Then for any $x \notin T$, if the unit vector u_x is closest to u_j in U, then*

$$(\phi(\pi_j, \sigma) - C_x) \cdot u_x \geq -\varepsilon, \quad \forall \sigma \in \Sigma. \tag{4}$$

The cardinality J may be chosen so that $J \leq C(2M/\varepsilon)^k$, where C is a constant that depends only on the dimension k.

Proof Let $U = \{u_1, \ldots, u_J\}$ be the prescribed set of unit vectors. A standard sphere packing argument shows that J need not be larger than the stated upper bound (see, e.g., Anthony and Bartlett, 1999). Let π_j be an optimal policy for the projected game $\Gamma(u_j)$. By assumption, Equation (3) holds for any $x \notin T$ and associated unit vector u_x. Let u_j be the closest vector to u_x in U. Rewriting Equation (3) we have that

$$\begin{aligned} u_x \cdot C_x &\leq \inf_\sigma \{\phi(\pi(x), \sigma) \cdot u_x\} \\ &\leq \sup_\pi \inf_\sigma \{\phi(\pi, \sigma) \cdot u_x\} \\ &= \sup_\pi \inf_\sigma \left\{\phi(\pi, \sigma) \cdot (u_j + (u_x - u_j))\right\}. \end{aligned}$$

By construction, $\|u_j - u_x\| < \varepsilon/2M$ so that for every pair $\sigma \in \Sigma$ and $\pi \in \Pi$ we have that $\|\phi(\pi, \sigma) \cdot (u_i - u_x)\| \leq \varepsilon/2$.

$$\begin{aligned} u_x \cdot C_x &\leq \sup_\pi \inf_\sigma \left\{\phi(\pi, \sigma) \cdot u_j\right\} + \varepsilon/2 \\ &= \inf_\sigma \left\{\phi(\pi_j, \sigma) \cdot u_j\right\} + \varepsilon/2 \\ &\leq \inf_\sigma \left\{\phi(\pi_j, \sigma) \cdot u_x\right\} + \varepsilon. \end{aligned}$$

The optimality of π_j in $\Gamma(u_j)$ was used in the first equality and the bound on $\|u_j - u_x\|$ was used again in the last inequality. ∎

For a set $T \subset \mathbb{R}^k$ we define its ε-expansion, T^ε, as $T^\varepsilon \stackrel{\triangle}{=} \{x \in \mathbb{R}^k : d(x,T) \leq \varepsilon\}$. It is now easy to verify that Equation (4) suffices for approaching T^ε, which leads to the following result.

Theorem 9 *Let Assumption 1 hold and suppose that T satisfies Equation (3). For any $\varepsilon > 0$ let $\{u_1, \ldots, u_J\}$ be the unit vectors in Proposition 8, with corresponding directional steering polices $\{\pi_1, \pi_2, \ldots, \pi_J\}$. Then the following policy approaches T^ε, the ε-expansion of T: If $s_n = s^*$ and $\hat{m}_n \notin T^\varepsilon$, then choose j so that $u_{\hat{m}_n}$ is closest to u_j (in Euclidean norm) and play π_j until the next visit to state s^*; otherwise, play arbitrarily.*

Proof We aim to show that the stated policy satisfies the requirements of Theorem 6 with respect to T^ε. That is, for every $x \notin T^\varepsilon$ the policy π_j which is optimal for the u_j which is closest to x satisfies

$$(\phi(\pi_j, \sigma) - C_x^\varepsilon) \cdot u_x^\varepsilon \geq 0, \quad \forall \sigma \in \Sigma, \tag{5}$$

where C_x^ε is the closest point in T^ε to x and u_x^ε is the unit vector from x to C_x^ε. We first claim that $C_x = C_x^\varepsilon + \varepsilon u_x^\varepsilon$, and hence $u_x = u_x^\varepsilon$. Indeed, if this is not true then there is a point z in T such that $d(z,x) < d(C_x^\varepsilon + \varepsilon u_x^\varepsilon, x) = d(C_x^\varepsilon, x) + \varepsilon$. Let u' denote the unit vector from x to z, we have that $z^\varepsilon = z - \varepsilon u'$ is in T^ε. It follows that $d(z^\varepsilon, x) = d(z,x) - \varepsilon$, combining this with the above, we get that $d(z^\varepsilon, x) < d(C_x^\varepsilon, x)$ which contradicts the assumption that C_x^ε is a closest point to x in T^ε.

Next, observe that by Proposition 8 we have that

$$(\phi(\pi_j, \sigma) - C_x) \cdot u_x \geq -\varepsilon, \quad \forall \sigma \in \Sigma,$$

but since $C_x = C_x^\varepsilon + \varepsilon u_x^\varepsilon$ and $u_x = u_x^\varepsilon$, (5) follows. ∎

Remark 10 *It follows immediately from the last theorem that the set T itself (rather than its ε-expansion) is approachable with a finite number of steering directions if $T^{-\varepsilon}$, the ε shrinkage of T satisfies Equation (3). Equivalently, T is required to satisfy Equation (3) with the 0 on the right-hand side replaced by ε.*

For convex target sets the sufficient condition Equation (3) for approachability is also necessary. This implies that every ε-expansion is approachable using a finite number of directions as per Theorem 9. The following corollary is an immediate result of the above theorem and Theorem 7.

Corollary 11 *Let Assumption 1 hold. A convex target set T is approachable if and only if for every $\varepsilon > 0$ the set T^ε can be approached by considering a finite number of directions as in Theorem 9.*

3.3 Reinforcement Learning in Scalar Games

In this subsection we review RL for zero-sum stochastic games with scalar rewards. We define two notions of approximate optimality in these games that will be useful in the sequel. The first type of optimality is optimality of the expected reward in finite time and the second type is asymptotic optimality of the learned policy. In both cases, the reference level for optimal performance is the value of the game. The first type of optimality requires that the average reward is not much less than the value of the game. The second type of optimality requires that the learned policy asymptotically achieves an expected reward which is not much worse than the value of the game. We note that a more ambitious goal may be to consider the Bayes envelope as in Mannor and Shimkin (2003) instead of the value, but this is not pursued here.

Consider a scalar zero-sum stochastic game. Denote the scalar reward at time n by r_n and the value by v (which is defined as in Equation (2)). Similarly to the vector-valued case, $\hat{r}_n$ denotes the average reward by time n. Our first notion of optimality is in expectation, after finitely many epochs.

Definition 12 *A learning algorithm π is* ε-optimal in expectation *from state s if there exists a deterministic finite time $N(\varepsilon)$ such that the expected average reward by time $N(\varepsilon)$ satisfies*

$$\mathbb{E}^{s}_{\pi,\sigma}\hat{r}_{N(\varepsilon)} \geq v - \varepsilon, \tag{6}$$

for every policy σ of P2.

There are several algorithms that are ε-optimal in expectation. For example, by carefully choosing the parameters of the R–MAX algorithm of Brafman and Tennenholtz (2002) the optimality requirement in Equation (6) is satisfied. Indeed, Brafman and Tennenholtz (2002) suggested a model-based learning policy π of P1 that satisfies that for every $\varepsilon, \delta > 0$ there exists a time $N(\varepsilon, \delta)$ such that the expected average reward satisfies at time $N(\varepsilon, \delta)$ with probability at least $1 - \delta$

$$\mathbb{E}_{\pi,\sigma}\hat{r}_{N(\varepsilon,\delta)} \geq v - \varepsilon,$$

for every policy σ of P2. By proper choice of parameters, the R–MAX algorithm yields an ε-optimal in expectation algorithm. This can be done by running the R–MAX algorithm with $\varepsilon' = \varepsilon/2$ and $\delta' = \varepsilon/2M$, where M is a bound on the expected immediate reward. Another algorithm that is ε-optimal in expectation is the Q-learning algorithm of Mannor and Shimkin (2002). An additional assumption that is required there is that P2 plays every action infinitely often (the analysis follows Kearns and Singh, 1999).

The next definition concerns the optimality of the learned policy and is of asymptotic nature. It implies that the learned policy of P1 is almost optimal in the limit. Formally, a policy of P1 is a mapping from all possible histories to the set of mixed actions of P1. Let π_n denote the policy of P1 from time n onwards (in the definition below this policy depends on the entire history up to time n, and the time axis is moved back n steps so that π_n starts at $t = 0$).

Definition 13 *A learning policy π of P1 is* asymptotically ε-optimal *in a scalar game if, for every policy σ of P2, π_n satisfies*

$$\liminf_{n\to\infty} \frac{\mathbb{E}^{s^*}_{\pi_n,\sigma} \sum_{t=0}^{\tau-1} r_t}{\mathbb{E}^{s^*}_{\pi_n,\sigma} \tau} \geq v - \varepsilon \quad P_{\pi,\sigma}\text{-a.s.} \tag{7}$$

The renewal time τ is as defined in Equation (1).

It should be noted that the requirement in Equation (7) is on the policy π_n and not on the actual reward obtained. Intuitively, the condition means that after enough time P1 cannot be surprised anymore, namely no policy of P2 can decrease his (per cycle) payoff below the value of the game.

RL for average reward zero-sum stochastic games was devised along similar lines as RL for average reward Markov decision processes. A Q-learning based algorithm which combines the ideas of Littman (1994); Littman and Szepesvári (1999) with those of Abounadi et al. (2002) was devised by Mannor and Shimkin (2002). An additional assumption that was needed for the convergence analysis there is that all actions of both players are used infinitely often in order to satisfy Equation (7). A model-based algorithm that is ε-optimal asymptotically can be suggested. By assuming that all actions are played infinitely often a model of the stochastic game may be calculated and an ε-optimal policy played, similarly to the R-MAX algorithm of Brafman and Tennenholtz (2002). It should be noted that P2 may refrain from playing certain actions. Consequently, P1 may never observe these actions and never learn the exact model of the game. This implies that unless we

impose some restrictions on P2's policy, the policy learned by P1 may be suboptimal. However, P1 can only benefit if P2 does not play certain actions, since in that case P2's play is constrained. The topic of learning to play optimally in SGs when no restrictions are made on P2 calls for further study.

In the special case of MDPs there are several algorithms that are ε-optimal in expectation (e.g., Kearns and Singh, 1998, 1999). Algorithms that are asymptotically optimal include ε-greedy Q-learning (Singh et al., 2000) and actor-critic (Konda and Tsitsiklis, 2001).

4. The Multiple Directions Algorithm

In this section we introduce and prove the convergence of the MDRL (Multiple-Directions Reinforcement Learning) algorithm. We consider the model of Section 2, but here we assume that P1, the learning agent, does not know the model parameters, namely the state transition probabilities and reward functions. A policy of P1 that does not rely on prior knowledge of these parameters will be referred to as a *learning* policy.

The proposed learning algorithm relies on the construction of the previous section of approaching policies with a finite number of steering directions. The main idea is to apply a (scalar) learning algorithm for each of the projected games $\Gamma(u_j)$ corresponding to these directions. Recall that each such game is a standard zero-sum stochastic game with average reward. The required learning algorithm for game $\Gamma(u)$ should secure an average reward that is not less than the value $v\Gamma(u)$ of that game.

We now describe the MDRL algorithm that approximately approaches any target set T that satisfies Equation (3). The parameters of the algorithm are ε and M, where ε is the approximation level and M is a known bound on the norm of the expected reward per step. The goal of the algorithm is to approach T^ε, the ε expansion of T. There are J learning algorithms that are run in parallel, denoted by $\pi_1, \ldots \pi_J$. Each of the J algorithms is responsible for learning to steer the reward in a different direction. Each of the J algorithms is assumed to be an asymptotically $\varepsilon/2$-optimal learning algorithm. The MDRL is described in Figure 4 and is given here as a meta-algorithm (the scalar RL algorithms π_i are not specified).

The algorithm is based on the idea of changing the control policy on arrivals to the reference state s^*. When arriving to s^*, the learning agent checks if the average reward vector is outside the set T^ε. In that case, it switches to an appropriate policy that is intended to "steer" the average reward vector towards the target set. The steering policy (π_j) is chosen so that the steered direction (u_j) is close to the exact direction towards T. The intuition behind the algorithm and the proof is that instead of steering in the current direction into T, a "close" direction, quantized out of a finite set is used. By employing the same quantized directions often enough, the decision maker eventually learns how to steer in those directions. Once the steering policy in each of the quantized directions is learned, the decision maker can almost steer in the desired direction at every return to s^*, by choosing a close enough quantized steering direction.

Theorem 14 *Suppose that Assumption 1 holds and that the target set satisfies Equation (3). Then the MDRL algorithm approaches T^ε.*

Proof Using the construction of Theorem 9 there are finitely many directions $u_1, \ldots, u_J$ that induce J projected SGs. At every return to state s^*, one of the J projected games is played and algorithm π_i is run. If the i-th game is not played infinitely often then there exists τ_i after which the i-th

Input: Target set T; desired approximation level ε; a bound M on the expected one step reward; a recurrent state s^*.
0. Divide the unit vector ball $\mathbb{B}^k$ as in Proposition 8 for approaching the $\varepsilon/2$ expansion of T. Let $u_1,\ldots u_J$ be the chosen directions there ($J = C(4M/\varepsilon)^k$). Initialize J different $\varepsilon/2$-optimal asymptotically scalar algorithms, $\pi_1,\ldots,\pi_J$. Set $n = 0$. 1. If $s_n \neq s^*$ play arbitrarily until $s_n = s^*$. 2. ($s_n = s^*$) If $\hat{m}_n \in T^\varepsilon$ goto step 1. Else let $i = \arg\min_{1\leq i\leq J} \|u_i - u_{\hat{m}_n}\|_2$. 3. While $s_n \neq s^*$ play according to π_i, the reward π_i receives is $u_i \cdot m_n$. 4. When $s_n = s^*$ goto step 2.

Figure 4: The MDRL algorithm.

algorithm is not played. Let τ^1 be the maximum of all the times τ_i for i-s that were played finitely often. Let $I \subseteq \{1,\ldots,J\}$ be the indices of algorithms that are played infinitely often. Since each π_i is eventually almost optimal it follows that after some finite time $\tau'(\varepsilon/2)_i$ each of the algorithms almost achieves the maximal reward, that is for $t > \tau'(\varepsilon/2)_i$ (7) holds with $\varepsilon/2$ instead of ε. Let $\tau^2 = \max_{i\in I} \tau'(\varepsilon/2)_i$. Let $\tau^3 = \max\{\tau^1,\tau^2\}$. From time τ^3 onward P1's play is $\varepsilon/2$ optimal. Thus, we can apply Theorem 9 so that T^ε is approached starting from τ^3. Fix $\delta > 0$. Let τ^4 be a large enough time such that the effect of the first τ^3 time epochs is not more δ in norm, i.e., $|\frac{1}{\tau^4}\sum_{n=0}^{\tau^3-1} m_n| < \delta$. Since δ is arbitrary, by Theorem 9 the result follows. ∎

Remark 15 *In the algorithm below, when π_j is not played, its learning pauses and the process history during that time is ignored. Note however that some "off-policy" algorithms (such as Q-learning) can learn the optimal policy even while playing a different policy. In that case, a more efficient version of the MDRL may be suggested, in which learning is performed by all learning policies π_j continuously and concurrently.*

5. The Current Direction Learning Algorithm

In this section we introduce and prove the convergence of the CDRL (Current Direction Reinforcement Learning) algorithm. P1's goal is the same as in Section 4 - to approach a given target set T. P1 does not know the model parameters and aims to ensure the convergence of the average reward vector to this set irrespectively of P2's actions.

Instead of using several fixed steering directions while employing a scalar learning algorithm for each of these directions separately, a single learning algorithm is used with the reward function changing over time. The idea is to learn to play a scalar game whose reward is the projected average reward on the direction towards the target set. Once a direction is fixed, the policy is to steer in that direction towards the target set for a while. After a long enough time, the direction is replaced by the new steering direction into the target set, and a new scalar learning algorithm is initiated.

We now describe the CDRL algorithm for approaching any target set T that satisfies Equation (3). The CDRL is described in Figure 5 and is given here as a meta-algorithm. The algorithm is based on the idea of changing the control policy after a long enough time. Whenever a switch occurs the new policy is intended to "steer" the average reward vector towards the target set.

Input: Target set T; Required precision ε; ε-optimal learning algorithm; $N(\varepsilon)$ from Definition 12.
0. Set $\hat{m}_0$ arbitrarily; $n=0$.
1. Repeat forever:
2. If $\hat{m}_n \in T^\varepsilon$ play arbitrarily for $N(\varepsilon)$ steps; $n := n+N(\varepsilon)$.
3. Else let $u = u_{\hat{m}_n}$.
4. Play an ε-optimal in expectation learning algorithm with reward at time n given by $r_n = u \cdot m_n$ and parameter ε. Play the algorithm for $N(\varepsilon)$ time epochs; $n := n+N(\varepsilon)$.

Figure 5: The CDRL algorithm.

Theorem 16 *Fix $\varepsilon > 0$. Suppose that for every $u \in \mathbb{B}^k$ there exists an ε-optimal in expectation algorithm for each projected SG, $\Gamma(u)$. Assume that the target set satisfies Equation (3). Then T^ε is approached using the CDRL algorithm.*

Proof (outline) The proof follows similarly to the more general Theorem 26 that is provided in the appendix. Consider the time epochs $\tau_i = iN(\varepsilon)$, $i = 1,2,\ldots$. If $\hat{m}_{\tau_i} \notin T^\varepsilon$ it follows that

$$\mathbb{E}\left(\left(\frac{1}{N(\varepsilon)} \sum_{n=\tau_i}^{\tau_{i+1}-1} (m_n - C_{\hat{m}_{\tau_i}}) \right) \cdot u_{\hat{m}_{\tau_i}} \middle| F_{\tau_i} \right) \geq -\varepsilon ,$$

where F_t is the sigma algebra generated until time t. Consequently,

$$\mathbb{E}\left(\left(\frac{1}{N(\varepsilon)} \sum_{n=\tau_i}^{\tau_{i+1}-1} (m_n - C^\varepsilon_{\hat{m}_{\tau_i}}) \right) \cdot u_{\hat{m}_{\tau_i}} \middle| F_{\tau_i} \right) \geq 0 ,$$

where C^ε_x is the closest point to x in T^ε. A similar argument to that of Theorem 26 can be used to show that $d(\hat{m}_{\tau_i}, T^\varepsilon) \to 0$ (a.s.). The result for all t follows by the boundedness of the immediate reward. ∎

Remark 17 *The CDRL algorithm was described above in extreme generality. There are several possible refinements and improvements which depend on the specific scalar algorithm used and on the domain of the problem. Specifically, knowledge is not assumed to be preserved from one activation of the scalar algorithm to the other. One may be interested to transfer knowledge between activations. This is especially natural for model-based algorithm in the spirit of Brafman and Tennenholtz (2002), where estimation of the model is performed.*

Remark 18 *A variant of the CDRL algorithm in which the precision level ε is decreased with time is described in the Appendix. It is shown there that ε can be slowly reduced to 0 so that the target set T itself, rather than its ε-expansion, is approached.*

6. Policy Mixing

Up to now we did not impose any restrictions on the directional steering policies that P1 can employ. Suppose now that P1 may choose between a finite and given set of policies $\pi_1, \ldots, \pi_J$. For simplicity

we may restrict the policy changes to occur in renewal epochs to some reference state s^*. To motivate this problem, one may consider an agent who has a few pre-programmed control policies. These policies may have been found useful for different tasks in this environment or are just intelligent guesses of reasonable policies. The agent wants to mix the policies in such a way so that the target set is approached. The idea is to mix between the prescribed policies so that when steering in a direction u the best policy among the J available policies is used. To choose the best such policy, a Multi-Armed Bandit (MAB) algorithm is employed. We describe an algorithm in the spirit of CDRL which we call SPRL (Specified Policies Reinforcement Learning).

If P1 knew the model parameters, P1 could choose the policy that is best in Equation (3) in the current steering direction (out of the set of given policies). In that case it follows from Theorem 6 that the set T is approachable if for every point $x \notin T$ there exists a policy π_i, $1 \leq i \leq J$, such that

$$(\phi(\pi_i, \sigma) - C_x) \cdot u_x \geq 0, \quad \forall \sigma \in \Sigma. \tag{8}$$

In that case we say that T is approachable with $\pi_1, \ldots, \pi_J$.

We set out to show that if T is approachable with $\pi_1, \ldots, \pi_J$ when the model is known then it is approachable with $\pi_1, \ldots, \pi_J$ when the model is unknown. The following algorithm resembles the CDRL algorithm in that every once in a while P1 changes the steering direction to best approach T. In order to steer in the current direction we use appropriate algorithms for the adversarial MAB problem, such as the Exp3.1 algorithm of Auer et al. (2002), as a policy selection mechanism. Recall that in the adversarial MAB problem a decision maker is repeatedly required to decide between J arms. When choosing arm i at time epoch n a reward of r_n^i is obtained. The decision maker tries to make sure that its cumulative reward is close to the one obtained at the best arm. We assume that a MAB arm selection algorithm, π, is defined by the following operations:

1. $Init_{\mathrm{MAB}}(J)$ - Initialize the parameters of MAB with J arms.
2. $GetArm_{\mathrm{MAB}}()$ - Returns the arm a that MAB wants to sample next.
3. $Update_{\mathrm{MAB}}(i, r)$ - Informs MAB the latest reward r of arm i.

The dynamics of every algorithm for the MAB problem are described as follows. First, each MAB algorithm is initialized. At every time epoch n, $GetArm_{\mathrm{MAB}}$ is used for picking an arm i_n and a reward $r_n = r_n^{i_n}$ is obtained. $Update_{\mathrm{MAB}}(i_n, r_n)$ is used to update the algorithm with the knowledge of the obtained reward. In the adversarial MAB formulation, there are two players - P1 and P2. P1 chooses which arm to play and simultaneously (not knowing P1's choice) P2 chooses how much reward to give to P1 for every possible choice of P1. The basic requirement from a MAB algorithm is that for every $\varepsilon > 0$ there exists $N(\varepsilon)$ such that

$$\mathbb{E}_{\pi,\sigma} \frac{1}{N(\varepsilon)} \sum_{n=0}^{N(\varepsilon)-1} r_n \geq \max_{1 \leq j \leq J} \sum_{n=0}^{N(\varepsilon)-1} r_n^j - \varepsilon \quad \text{for every policy } \sigma \text{ of P2,} \tag{9}$$

where the expectation is taken over both player's policies, as the resulting policy (for both players) may be stochastic. Equation (9) means that the expected average reward by time $N(\varepsilon)$ is almost as high as the average reward that would have been obtained if P1 would have known the reward of each arm in advance. Note that it is assumed that P2 decides on P1's reward for every action (arm) chosen at every time epoch. The Exp3.1 algorithm satisfies Equation (9). Moreover, $N(\varepsilon)$ depends only on ε, J and a bound on the immediate reward.

To state the SPRL algorithm we shall require the concept of ε-*mixing time* (see Brafman and Tennenholtz, 2002). The ε-mixing time K is a (large enough) number so that for every $1 \leq i \leq J$, every policy σ of P2, and every initial state s,

$$\mathbb{E}^{s}_{\pi_i,\sigma}\frac{1}{K}\sum_{n=0}^{K-1} r_n \geq \inf_{\sigma'}\frac{\mathbb{E}^{s^*}_{\pi_i,\sigma'}\sum_{t=0}^{\tau-1} r_t}{\mathbb{E}^{s^*}_{\pi_i,\sigma'}\tau} - \varepsilon,$$

where τ is the stopping time defined in Equation (1). Having an ε-mixing time implies that the expected reward starting from any initial state is at least as high (up to ε) as the worst case average reward starting from s^*. It follows by Lemma 3.4 from Shimkin and Shwartz (1993) that when Assumption 1 holds then there exists K_0 such that the expected return time to s^* from every state and every pair of policies is bounded by some K_0. By choosing $K \geq K_0 M/\varepsilon$ the above condition is satisfied (M is a bound on the expected one stage reward). The basic requirement of the SPRL algorithm is that K, or an upper bound on it, is provided.

Input: Required precision ε; set of policies $\pi_1,\ldots,\pi_J$; $\varepsilon/2$-mixing time K; a recurrent state s^*.
0. $n=0$; play arbitrarily until $s_n = s^*$.
1. Repeat forever
2. If $\hat{m}_n \in T^\varepsilon$ play arbitrarily for K steps.
3. Else let $u = u_{\hat{m}_n}$.
4. $Init_{MAB}(J)$.
5. Repeat $N(\varepsilon/2)$ times:
6. $i = GetArm_{MAB}$.
7. Play π_i for K epochs, let the total reward over the K epochs be m_{tot}, set $r = m_{tot}\cdot u$; $n := n + N(\varepsilon/2)$.
8. $Update_{MAB}(i,r)$.

Figure 6: The SPRL algorithm - switching between a given small set of policies.

Theorem 19 *Suppose that T is approachable with $\pi_1,\ldots\pi_J$ when the model is known. Then the SPRL algorithm described in Figure 6 approaches T^ε.*

Proof The proof is based on showing that by using the SPRL algorithm Equation (3) holds for the expanded target set T^ε. Consider the reward in the epochs $\tau_i = iKN$ $(i = 1,2,\ldots)$, for $N = N(\varepsilon/2)$. We claim that the vector valued reward at those points satisfies Blackwell's condition. By using the SPRL algorithm it follows that either $\hat{m}_{\tau_i} \in T^\varepsilon$ or that

$$\mathbb{E}\left(\left(\frac{\sum_{n=\tau_i}^{\tau_{i+1}-1} m_n}{KN} - C_{\hat{m}_{\tau_i}}\right)\cdot u_{\hat{m}_{\tau_i}}\middle| F_{\tau_i}\right) \geq -\varepsilon \quad \text{a.s.}, \tag{10}$$

where $C_{\hat{m}_{\tau_i}}$ is the closest point to $\hat{m}_{\tau_i}$ in T and F_n is the sigma algebra up to time n. Fix i and let $r_n \stackrel{\triangle}{=} (m_n - C_{\hat{m}_{\tau_i}})\cdot u_{\hat{m}_{\tau_i}}$ for $\tau_i \leq n \leq \tau_{i+1}-1$. By our choice of playing an $\varepsilon/2$ MAB algorithm it

follows that

$$\mathbb{E}^{s_{\tau_i}}_{\pi,\sigma}\left(\frac{1}{KN}\sum_{n=\tau_i}^{\tau_{i+1}-1} r_n\right) = \mathbb{E}^{s_{\tau_i}}_{\pi,\sigma}\left(\frac{1}{N}\sum_{\ell=0}^{N-1}\frac{1}{K}\sum_{n=\tau_i+\ell K}^{\tau_i+(\ell+1)K-1} r_n\right)$$
$$\geq \max_{1\leq j\leq J}\frac{1}{N}\sum_{\ell=0}^{N-1}\frac{1}{K}\mathbb{E}^{s_{\tau_i+\ell K}}_{\pi_j,\sigma}\left(\sum_{n=\tau_i+\ell K}^{\tau_i+(\ell+1)K-1} r_n\right) - \varepsilon/2\,. \tag{11}$$

Since K is an $\varepsilon/2$-mixing time we have that for every $1 \leq j \leq J$ and $1 \leq \ell < N$,

$$\mathbb{E}^{s_{\tau_i+\ell K}}_{\pi_j,\sigma}\frac{1}{K}\sum_{n=\tau_i+\ell K}^{\tau_i+(\ell+1)K-1} r_n \geq \inf_{\sigma'}\frac{\mathbb{E}^{s^*}_{\pi_j,\sigma'}\sum_{t=0}^{\tau-1} r_t}{\mathbb{E}^{s^*}_{\pi_i,\sigma'}\tau} - \varepsilon/2\,.$$

Since this holds for every $1 \leq j \leq J$, we substitute this in Equation (11) to obtain

$$\mathbb{E}^{s_{\tau_i}}_{\pi,\sigma}\left(\frac{1}{KN}\sum_{n=\tau_i}^{\tau_{i+1}-1} r_n\right) \geq \max_{1\leq j\leq J}\inf_{\sigma'}\frac{\mathbb{E}^{s^*}_{\pi_j,\sigma'}\sum_{t=0}^{\tau-1} r_t}{\mathbb{E}^{s^*}_{\pi_i,\sigma'}\tau} - \varepsilon\,.$$

Using the definitions of r_n and $C_{\hat{m}_{\tau_i}}$, Equation (10) follows. The rest of the proof follows almost exactly as Theorem 16 and is therefore omitted. ∎

Several comments are in order.

Remark 20 *We note that even if π_i are learning policies rather than a-priori fixed ones the algorithm still works. The only thing that matters in this case is that the strategies converge after some finite stochastic time. This suggests the following algorithm. Learn the optimal policy for several directions, and use the SPRL algorithm for mixing the learned strategies.*

Remark 21 *It is possible to obtain Theorem 19 with $\varepsilon = 0$. This can be done by a limiting procedure similar to Theorem 26 with the additional complication that both the ε-mixing time and the time required by the MAB algorithm increase. It can be shown that for certain MAB algorithms $N(\varepsilon) \sim \frac{1}{\varepsilon^2}$ and that the behavior of the mixing time as a function of ε is $K(\varepsilon) \sim \frac{1}{\varepsilon}$. The total time is approximately $O(\frac{1}{\varepsilon^3})$, and the technique of Theorem 26 may be employed.*

7. On-line Selection of Target Sets

In many problems of interest the target set that we wish to approach may itself depend on the unknown model parameters, so that it cannot be explicitly prescribed *a-priori* to the learning agent. The simplest and most common instance of this situation is the simple maximization of a scalar objective function: this corresponds to a target set of the form $[a,\infty)$, with a required to be as large as possible. We generalize this situation to the multi-objective case by considering a *nested* family of possible target sets, and assume that the agent's task it to approach the minimal target set, namely the smallest possible one for the actual model at hand. The major difficulty in applying the steering approach now is that the required steering direction is not known, as it may be different for each set in the given family. We address this difficulty by maintaining an estimate of the smallest set that can be approached, which is used as a target for steering, and updating this estimate according to observations. In the latter part of this section we apply these results to solve the problem of learning optimal policies in constrained stochastic games. Throughout this section we assume that all target sets are convex.

7.1 Nested Target Sets

Consider an increasing class of convex sets $\{T_\alpha\}_{\alpha\in\mathcal{L}}$, indexed by the ordered set of indices $\mathcal{L}$ such that $\alpha<\beta$ implies $T_\alpha\subseteq T_\beta$. For simplicity, we may consider $\mathcal{L}=\mathbb{R}$ and assume that $\{T_\alpha\}$ is continuous.[2] The objective is to approach the smallest possible set in this class, that is to approach the smallest possible T_α. Recall that the third part of Theorem 7 claims that a convex target set is approachable if and only if for every $u\in\mathbb{B}^k$, $v\Gamma(u)\geq\inf_{m\in T}u\cdot m$. Consider a direction u, and let $v(u)=v\Gamma(u)$ denote the value of the game in this direction. Let $\alpha(u,v(u))$ denote the smallest (or infimal) value α for which $v(u)\geq\inf_{m\in T_\alpha}u\cdot m$. (Note that we define $\alpha(u,v(u))$ for any value $v(u)$, since below we shall use an estimate to $v\Gamma(u)$ rather than the exact value). Then, according to Theorem 7 the set T_α may be approachable only if $\alpha\geq\alpha(u,v\Gamma(\alpha))$, and the smallest approachable set is

$$T^*=\bigcap_{u\in\mathbb{B}^k}T_{\alpha(u,v\Gamma(u))}=T_{\alpha^*},$$

where $\alpha^*=\sup_{u\in\mathbb{B}}\alpha(u,v\Gamma(u))$. See Figure 7 for a geometric illustration of the problem.

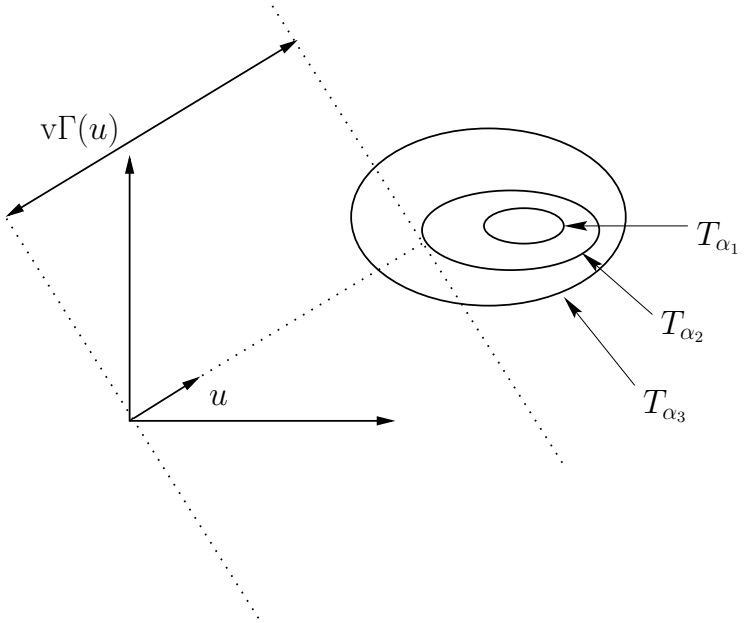


Figure 7: A two dimensional Illustration of nested sets. In the illustration $\alpha_1<\alpha_2<\alpha_3$ so that $T_{\alpha_1}\subset T_{\alpha_2}\subset T_{\alpha_3}$. For the direction u the value of the projected game $\Gamma(u)$ is such that $\alpha(u,v\Gamma(u))=\alpha_2$. As a result a target T_α set may be approachable if $\alpha\geq\alpha_2$. In this case T_{α_3} may be approachable, but T_{α_1} is not approachable.

We now suggest a modification of the MDRL algorithm for approximately approaching T^*. Recall that the MDRL algorithm is based on switching between a grid of J steering directions $u_1,\ldots,u_J$. For each direction u_j an asymptotically ε-optimal learning algorithm π_j is applied. In

2. Continuity here is induced by the metric, implying that the distance between sets is small when the indices are close. Formally, we require that if $\alpha_k\to\alpha$ then $\sup_{x\in T_\alpha\cup T_{\alpha_k}}\inf_{y\in T_\alpha\cap T_{\alpha_k}}d(x,y)\to 0$.

the original MDRL algorithm some directions may be encountered finitely many times so that the learned directional steering policy may be suboptimal. Since T is not prescribed and should be estimated on-line, we will impose exploration in each of the J steering directions. Fix a direction u_j. Let $\tilde{\mathrm{v}}(u_j)_n$ be the estimate by time n of $\mathrm{v}\Gamma(u_j)$. We require that the estimate is not too pessimistic. That is,

$$\liminf_{n\to\infty} \tilde{\mathrm{v}}(u_j)_n \geq \mathrm{v}\Gamma(u_j) - \varepsilon \quad P_{\pi_j,\sigma} - \text{a.s}, \tag{12}$$

where π_j is the learning policy when steering in direction u_j, σ is P2's policy, and Equation (12) holds for all σ. We also require that for every π_j,

$$\lim_{n\to\infty} \tilde{\mathrm{v}}(u_j)_n \quad \text{exists } P_{\pi_j,\sigma} - \text{a.s.} \tag{13}$$

This requirement is not essential, but simplifies the analysis and is fulfilled by all the learning algorithms we are aware of. We also assume that the estimate of the value is not too optimistic and matches P2's policy in the following sense:

$$\liminf_{n\to\infty} \left(\frac{\mathbb{E}^{s^*}_{\pi_n,\sigma} \sum_{t=0}^{\tau-1} r_t}{\mathbb{E}^{s^*}_{\pi_n,\sigma} \tau} - \tilde{\mathrm{v}}(u_j)_n \right) \geq -\varepsilon \quad P_{\pi_j,\sigma}\text{-a.s}, \tag{14}$$

for every policy σ of P2 (τ is the renewal time from Equation (1)). Intuitively, Equation (14) means that P1's policy guarantees an expected reward which is not much worse than the estimated value $\tilde{\mathrm{v}}(u_j)_n$. Without Equation (14), the target set may not be estimated correctly, and the steering direction may be wrong. All three assumptions Equation (12)-(14) are satisfied by Q-learning for average reward games of Mannor and Shimkin (2002) under mild additional assumptions on P2's play (the assumption there was that every action is either played a non vanishing fraction of the times or finitely many times, but the fraction of times may not alternate between 0 and something bigger than 0). The above requirements may also be seen to be satisfied by model-based learning algorithms under the assumption that all actions by P2 are played infinitely often. We note that $\tilde{\mathrm{v}}(u_j)_n$ in Equation (12) may be larger than the true value $\mathrm{v}\Gamma(u_j)$. The reason for that is that $\tilde{\mathrm{v}}(u_j)$ is an estimate which is based on the observations. If P2 refrains from playing certain actions (which are worst-case in $\Gamma(u_j)$) then P1 cannot, and need not, take these action into consideration.

In Figure 8 we describe a modified version of the MDRL algorithm. Essentially we add to the MDRL algorithm a small probability of exploration on each return to the reference state s^*. Exploration is made uniformly over the J available directions. For each direction u_i (all directions are chosen comparatively often due to the exploration) an index is maintained for the value of the projected game. This index represents the current estimate of $\alpha(u, \mathrm{v}\Gamma(u))$. The goal is to approach an "almost" minimal set. The steps of the algorithm are very similar to the MDRL algorithm from Figure 4. The differences are the persistent exploration that is performed continually (step 2 of the algorithm) and the fact that the estimated target set replaces the true (unknown) target set. Exploration is made when either $\hat{m}_n$ is inside the estimated target set (and therefore the decision maker does not need to steer in any direction), or with some small probability when returning to s^*. The latter exploration is needed in order to guarantee that the estimate of the target set is eventually consistent. The estimate of the target set is based on estimating the index that each of the J projected games and taking the maximal index to be the current estimate of the index of the target set. We wish to approach the set T_α with minimal index that is approachable. The estimate of the index by

time n is denoted by $indmax_n$ and equals

$$indmax_n \stackrel{\triangle}{=} \max_{1\le i\le J} \alpha(u_i, \tilde{v}(u_i)_n).$$

We take the maximal index (over the J available directions), because if $\alpha < \alpha(u_i, \tilde{v}(u_i)_n)$ for some i it implies that, according to the current estimate, T_α is not approachable for $\alpha < \alpha(u_i, \tilde{v}(u_i)_n)$. The estimate of the target set by time n is

$$\tilde{T}_n = T_{indmax_n}. \tag{15}$$

For every $\varepsilon > 0$ define the maximal index $indmax(\varepsilon) \stackrel{\triangle}{=} \max_{1\le i\le J} \alpha(u_i, v\Gamma(u_i) - \varepsilon)$. The target set P1 tries to approach is $T_{indmax(\varepsilon)}$. The modified MDRL algorithm from Figure 8 approaches the set $T^{\varepsilon}_{indmax(\varepsilon)}$, which is the ε-expansion of $T_{indmax(\varepsilon)}$.

Input: Desired approximation level ε; a set class $\{T_\alpha\}_{\alpha\in\mathcal{L}}$; a bound M on the expected one step reward; a bound N on the expected renewal time to s^*.
0. Divide the unit vector ball $\mathbb{B}^k$ as in Proposition 8 for approaching the $\varepsilon/3$ expansion of any target set. Let $U = u_1, \ldots, u_J$ be the chosen directions. Initialize J different asymptotically $\varepsilon/3$-optimal scalar algorithms, $\pi_1, \ldots, \pi_J$ that satisfy Equation (14). 1. Choose a random direction $u_i \in U$ and play π_i until $s_n = s^*$. 2. ($s_n = s^*$) If $\hat{m}_n \in \tilde{T}_n$ goto step 1. Else with probability $\delta = \varepsilon/(3MN)$ goto 1. 3. Let $u = \frac{\hat{m}_n - C(\tilde{T}_n, \hat{m}_n)}{\|\hat{m}_n - C(\tilde{T}_n)\|}$. ($C(T,x)$ denotes the closest point in T to x.) 4. Let $i = \operatorname{argmin}_{1\le i\le J} \|u - u_i\|$. 5. While $s_n \neq s^*$ play according to π_i, the reward π_i receives is $u_i \cdot m_n$. 6. When $s_n = s^*$ goto step 2.

Figure 8: The modified MDRL for unprescribed target sets.

The following Theorem claims that $T^{\varepsilon}_{indmax(\varepsilon)}$ is approached using the modified MDRL. Since we assume that the class $\{T_\alpha\}$ is continuous in α, by taking ε small enough we can practically get as close as we wish to T.

Theorem 22 *Suppose that Assumption 1 holds and that the target set satisfies Equation (3). Further assume that each scalar learning algorithm satisfies Eqs. (12), (13), and (14), where Equation (12) and Equation (14) are satisfied with $\varepsilon/3$ instead of ε. Then $T^{\varepsilon}_{indmax(\varepsilon)}$ is approached using the modified MDRL algorithm from Figure 8.*

Proof (outline) Exploration is persistent in all directions. By the definition of the algorithm, exploration cannot cause more that $\varepsilon/3$ drift away from the target set. Since each algorithm satisfies Equation (14) and since the estimate $\tilde{v}(u_j)_n$ converges to a limit by Equation (13) it follows that from some (random) time on P2's expected reward when steering in direction j is up to $\varepsilon/3$ as high as $\lim_{n\to\infty} \tilde{v}(u_j)_n$. By Equation (12) and Equation (13) the limit of $\tilde{v}(u_j)_n$ is not less than $v\Gamma(u_j) - \varepsilon/3$. It follows that from some (random) time on the estimate of $\tilde{T}_n$ almost does not change (more specifically, we assume that Equation (13) holds so that $\tilde{v}(u_j)_n$ converges. From some random time on,

all the J estimates $\tilde{v}(u_j)_n$ converge. By the continuity of T_α in α, it follows that the set $\tilde{T}_n$ converges to some, random, $\tilde{T}_\infty$). After that time - the same analysis as in Theorem 14 may be performed. ∎

There is a special case in which there is no need to perform the exploration for estimating $\tilde{v}(u_i)_n$. This is the case where the sets in the target set class are balls around a target point. Namely, if there is some point x_0 such that for every α in $\mathcal{L}$ there is a number γ such that T_α is a ball with radius γ around x_0. The steering directions is always towards x_0 and is the same for all the candidate target sets. Consequently, the MDRL algorithm does not require any changes, and exploration is not required.

7.2 Constrained Stochastic Games

A specific case of interest is the case of constrained optimization which has been extensively explored for MDPs (see Altman, 1999, and references therein). In this setup the decision maker obtains in every time step both reward and additional variables. The objective of the decision maker is to maximize its average reward $\hat{r}_n$ given that some additional average variables $\hat{c}_n^1,\ldots,\hat{c}_n^k$ satisfy the constraints $\hat{c}_n^i \leq \bar{c}, i = 1,\ldots,k$. The policy of P2 remains unconstrained and arbitrary. The immediate reward vector in this game is comprised of one coordinate which is the actual scalar reward and the remaining k coordinates that are the constraints. The maximization of the reward is performed on a sample path, so the average reward should be optimal almost surely. For zero-sum stochastic games where P1 tries to maximize the average reward subject to the constraints it was shown by Shimkin (1994) that under Assumption 1, and provided that the constraints can be satisfied by some policy, there exists a value v. This implies that P1 can guarantee that $\liminf_{n\to\infty} \hat{r}_n \geq \mathrm{v}$ and $\limsup_{n\to\infty} \hat{c}_n \leq \bar{c}$ (with the last inequality a vector inequality) almost surely. It was also shown there that P1 cannot guarantee a reward higher than v while satisfying the constraints.

We now show that the constrained optimization problem may be solved by the framework of the nested target sets. Let the candidate set class $\{T_\alpha\}$ be defined in the reward-constraints space:

$$T_\alpha = \{(r, c^1 \ldots, c^k) : r \geq -\alpha, c^1 \leq \bar{c}, \ldots, c^k \leq \bar{c}\} \subseteq \mathrm{IR}^{k+1}.$$

The α belongs to the index set $\mathcal{L} = \mathrm{IR}$. It obviously holds that if $\alpha < \beta$ then $T_\alpha \subset T_\beta$ and that T_α is convex for all α. Continuity of the class $\{T_\alpha\}$ holds as well. Using the MDRL algorithm with persistent exploration we can find the minimal set to be approached. Since we only use a finite grid of directions as in the MDRL algorithm, the guaranteed reward is $\mathrm{v} - \varepsilon$, with constrains satisfied up to ε. But since ε is arbitrary we can effectively get as close as we want to the value. We summarize the results in the following proposition.

Proposition 23 *Assume that some policy satisfies* $\limsup_{n\to\infty} \hat{c}_n \leq \bar{c}$. *Suppose that the modified MDRL algorithm is used for finding the optimal policy in a constrained SG, and that the assumptions of Theorem 22 are satisfied. Then* $\liminf_{n\to\infty} \hat{r}_n \geq \mathrm{v} - \varepsilon$ *and* $\limsup_{n\to\infty} \hat{c}_n \leq \bar{c} + \varepsilon e$ *where the last inequality is in vector notations, and e is a vector of ones.*

We now describe more explicitly the estimation of the target set for the constrained optimization problem. We denote the first coordinate of a unit vector in IB^{k+1} by u^r and the rest of the k coordinates by u^c. Suppose that at time n we are given an estimate $indmax_n$ of the value v. By the geometry of the problem it follows that for any point (r,c), the closest point in $\tilde{T}_n$ is $(r', c_1', c_2', \ldots, c_k')$ where $r' = \max\{r, indmax_n\}$ and $c_i' = \min\{c_i, \bar{c}_i\}$. Because of the special structure of the target set

and its convexity, the set of directions which may possibly be steering directions (in the $k+1$ unit ball) is given by

$$SD = \{(u^r, c^1, \ldots, c^k) \in \mathbb{B}^{k+1} : u^r \geq 0, c^1 \leq 0, \ldots, c^k \leq 0\}. \tag{16}$$

To observe that this is the set of possible steering directions, we first note that $u^r \geq 0$ since there is no point in trying to reduce the reward. Indeed, it is easily seen that otherwise $\inf_{m \in T_\alpha} u \cdot m = -\infty$ for all α, so that $\alpha(u, \mathrm{v}(u)) = -\infty$. Second, if a constraint is satisfied then the steering direction will not attempt to violate it. It therefore suffices to consider steering directions, $u_1, \ldots, u_J$, that belong to *SD*. We now show how to estimate the index of the target set given current estimates of the value for each of the projected games.

Proposition 24 *Given directions $u_1, \ldots, u_J$ and estimates $\tilde{\mathrm{v}}(u_j)$, $1 \leq j \leq J$, the estimated index of the target set $\tilde{T}_n$ is given by*

$$indmax_n = - \min_{1 \leq j \leq J, u_j^r > 0} \frac{\tilde{\mathrm{v}}(u_j)_n - u_j^c \cdot (\bar{c}^1, \ldots, \bar{c}^k)}{u_j^r},$$

Proof By the geometry of the problem each direction u_j and corresponding estimate of the value $\tilde{\mathrm{v}}(u_j)$ define a "half space" which must intersect with the estimated target set (by Theorem 7). This half space is given by: $\{(r,c) \in \mathbb{R} \times \mathbb{R}^k : (r,c) \cdot (u_j^r, u_j^c) \geq \tilde{\mathrm{v}}(u_j)\}$. The target set is of the form $\{(r,c) : r \geq r_{max}(u), c \leq \bar{c}\}$, where $r_{max}(u)$ the maximal scalar reward which can be achieved by a specific steering direction u while satisfying the constraints, for an illustration see Figure 9. It follows that r_{max} is in fact the index α in the construction of the candidate target set class. Since $u_r \geq 0$ and $u_c \leq 0$ it follows that

$$(u^r, u^c) \cdot (r_{max}(u), \bar{c}^1, \ldots, \bar{c}^k) = \tilde{\mathrm{v}}(u),$$

so that (assuming $u^r \neq 0$)

$$r_{max}(u) = \frac{\tilde{\mathrm{v}}(u) - u^c \cdot (\bar{c}^1, \ldots, \bar{c}^k)}{u^r}.$$

Overall, the maximal reward that can be achieved is

$$r_{max} = \inf_{\{u \in \mathbb{B}^{k+1} : u^r > 0, u^c \leq 0\}} r_{max}(u). \tag{17}$$

The result follows by considering finitely many directions and recalling that the index is minus r_{max}. ∎

The last two propositions imply that we can solve constrained SGs, and in particular, constrained MDPs. To the best of our knowledge this is the first RL algorithm that solves constrained SGs. Regarding constrained MDPs, we note that one can envision a model-based algorithm (in the spirit of the E^3 algorithm of Kearns and Singh, 1998), however such an algorithm would work only for small problems. Our algorithm may be applied in a model-free setup, as long as underlying model-free scalar RL algorithms can be used. We also comment that the Q-learning style algorithm of Gábor et al. (1998) searches for the optimal deterministic policy, while the optimal one may be stochastic. An example for the solution of constrained MDPs is provided in Section 9.2.

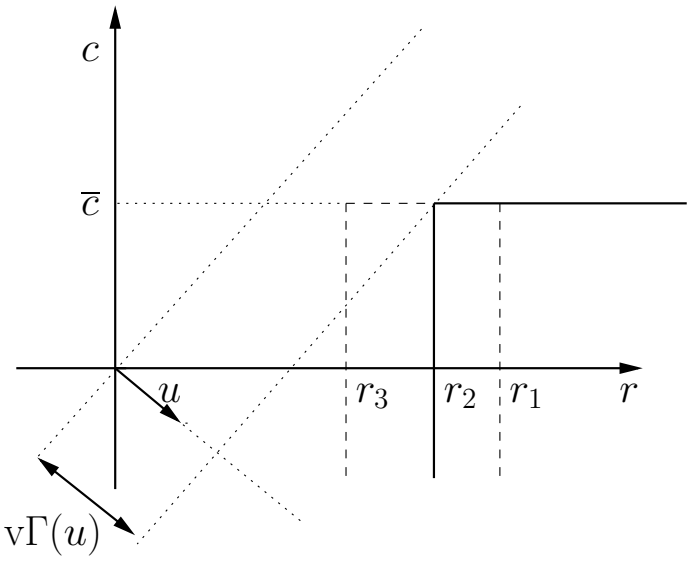


Figure 9: A two dimensional Illustration of finding the value for a constrained game. The *x* axis is the reward and the *y* axis is the constraint. The value of the game projected on direction u is $v\Gamma(u)$ which implies that the achievable reward is at most r_2, and possibly less. An average reward of r_1 is not achievable since the set $\{(r,c) : r \geq r_1, c \leq \overline{c}\}$ does not satisfy Theorem 6 for the direction u.

Remark 25 *For general target sets it is possible to devise a similar algorithm to the above where the target set is estimated on-line. For example, one can estimate the game parameters and derive a candidate for the target set based on that estimate. We note that as long as the estimate is consistent (i.e., the final estimate is the actual set to be approached) then schemes which are based on MDRL or CDRL will approach the target set. For example, one may consider approaching the convex Bayes envelope (Mannor and Shimkin, 2003) to arrive at a optimal on-line algorithm for stochastic games. For convex sets there is a particularly interesting possibility of learning the support function (Rockafellar, 1970). Recall that a convex set T can be written as*

$$T = \bigcap_{u \in \mathbb{B}^k} \{x : u \cdot x \geq f_T(u)\},$$

where the function $f_T(u) : \mathbb{B}^k \to \mathbb{R}$ is the support function *of T, which is defined as $f_T(u) \stackrel{\triangle}{=} \inf_{y \in T} u \cdot y$. The steering direction can be characterized as the maximizer of $f_T(u) - u \cdot x$ (Mannor, 2002). Now, if $f_T(u)$ can be learned somehow then instead of the true unknown steering direction an approximate steering direction which the maximizer of the $\tilde{f}_T(u) - u \cdot x$, where $\tilde{f}_T$ is the estimate of f_T, can be used. This approach may prove useful when the support function has a simple representation.*

8. Multi-Criterion MDPs

In this section we briefly discuss multi-criterion MDPs. These models may be regarded as a special case of the stochastic game model that was considered so far, with P2 eliminated from the problem. We shall thus only outline the main distinctive features of the MDP case.

Both the MDRL algorithm from Section 4 and the CDRL algorithm from Section 5 remain essentially the same. Their constituent scalar learning algorithms are now learning algorithms for average-reward MDPs. These algorithms are generally simpler than algorithms for the game problem. Examples of optimal or ε-optimal algorithms are Q-Learning with persistent exploration (Bertsekas and Tsitsiklis, 1995), actor-critic schemes (Sutton and Barto, 1998), the E^3 algorithm (Kearns and Singh, 1998), and others. Additionally, it should be noted that the steering policies learned and used within the MDRL algorithm may now be *deterministic* stationary policies, which simplifies the implementation of this algorithm.

Working with a finite number of specified policies as in Section 6 is particularly simple. In this case all that P1 needs to do is to estimate the average reward vector that each policy yields. P1 can attain every point in the convex hull of the average per policy reward vectors by mixing the strategies appropriately. In the absence of an adversary, the problem of approaching a set in a known model becomes much simpler. In this case, if a set is approachable then it may be approached using a stationary (possibly mixed) policy. Indeed, any point in the feasible set of state-action frequencies may be achieved by such a stationary policy (Derman, 1970). Thus, alternative learning schemes that do not rely on the steering approach may be applicable to this problem.

9. Examples

Two brief examples will be used to illustrate the above framework and algorithms. In Subsection 9.1 we show what sample paths of the MDRL algorithm look like for the two dimensional temperature-humidity problem from the introduction. In Subsection 9.2 we solve on-line a queuing problem which is modelled as a constrained MDP.

9.1 The Two-Dimensional Temperature-Humidity Problem

Recall the humidity-temperature example from the introduction. Suppose that the system is modelled in such a way that first P1 chooses a temperature-humidity curve by, say, deciding if to activate heater/cooler or humidifier/dehumidifier a-priori. Then Nature (modelled as P2) chooses the exact location on the temperature-humidity curve by choosing the weather that affect the conditions in the greenhouse. Essentially, P1 chooses the set where the immediate rewards lie in and P2 chooses the exact location of the reward in that set. In Figure 10(a) we show the target set T and three different temperature-humidity curves (these curves are representative curves). Geometrically, P1 chooses the curves where the reward belongs to (each defined by a certain policy of P1 - f_0, f_1, f_2), and P2 chooses the exact location. Additional zero mean with unit variance noise was added to each reward entry. It so happens that there is no stationary policy for P1 which guarantees that the average reward is in T, regardless of P2's actions. We implemented an MDRL algorithm with a "dense" grid of only nine directions. In each direction a version of Littman's Q-learning (Littman, 1994), adapted for average cost games, was used (Mannor and Shimkin, 2002). P2's policy was adversarial, and it tried to get the average reward away from the target set. Three sample paths of the average reward for 100,000 epochs that were generated by the MDRL algorithm are shown in

Figure 10(b). Each sample path starts at 'S' and finishes at 'E'. It can be seen that the learning algorithm pushes towards the target set so that the path is mostly on the edge of the target set. Note that in this example a small number of directions was quite enough for approaching the target set. Since the convergence was so fast it made no sense to use the CDRL algorithm, though we would expect it to work well in this case.

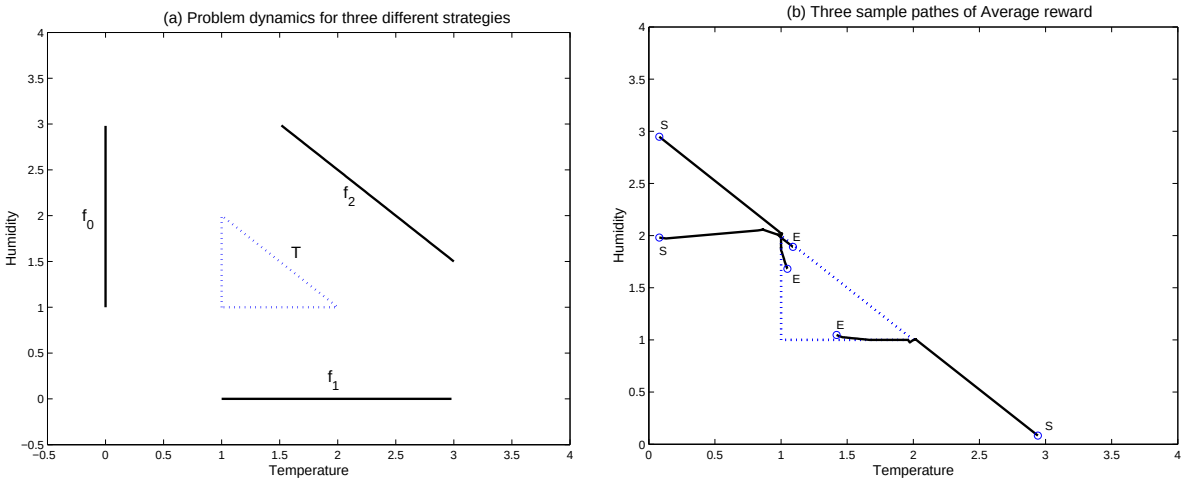


Figure 10: **(a)** Greenhouse problem dynamics. **(b)** Three sample paths from 'S' to 'E'.

9.2 Constrained Optimization of a Queue

In this section we consider the model of a queue with varying service rate. Suppose that there is a queue of buffer length L. Messages arrive to the queue at a rate of μ. P1 controls the service rate which means that if the queue is not empty then a message leaves the queue with probability a_n. The state of the system is the number of messages in the queue $\mathcal{S} = \{0, 1, 2, \ldots, L\}$. The state transition probability satisfies

$$P(s_{n+1}|s_n, a) = \begin{cases} (1-\mu)a & \text{if } L \geq s_n \geq 1 \text{ and } s_{n+1} = s_n - 1, \\ \mu a + (1-\mu)(1-a) & \text{if } L > s_n \geq 1 \text{ and } s_{n+1} = s_n, \\ \mu + (1-\mu)(1-a) & \text{if } s_{n+1} = s_n = L, \\ \mu(1-a) & L - 1 \geq \text{if } s_n \geq 0 \text{ and } s_{n+1} = s_n + 1, \\ 1 - \mu(1-a) & \text{if } s_{n+1} = s_n = 0, \\ 0 & \text{otherwise.} \end{cases}$$

The reward of P1 is a function of the number of vacant places in the queue. For simplicity we assumed that $r_n = c(L - s_n)$ where c is a positive constant. P1's objective is to maximize the reward it obtains. This is equivalent to minimizing the average queue length. P1 may affect the state transitions through change of the service rate $a_n \in \{a_L, a_H\}$. The service constraint of P1 is to have the average service rate $\hat{a}_n$ lower than some pre-specified constant c_0. The optimization problem we have is

$$\max \hat{r}_n$$

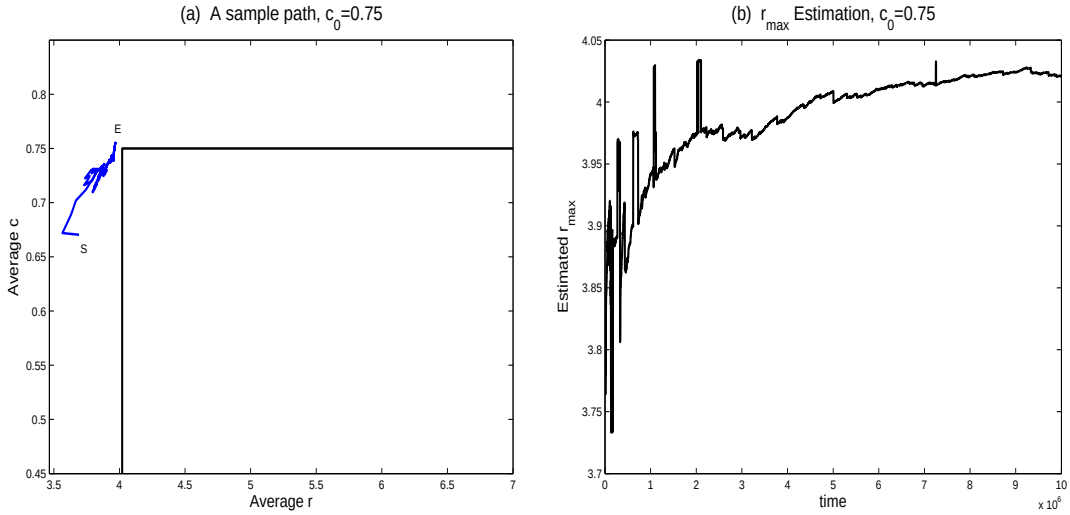


Figure 11: A sample path (a) and r_{max} estimation (b) for a run of the modified MDRL for the constrained queue problem, $c_0 = 0.75$. Note that r_{max} converged to 4.02.

$$\text{s.t. } \hat{a}_n \leq c_0 \,.$$

Note that the maximum should be attained per sample path and may be achieved by a stationary policy (e.g., Altman, 1999). Moreover, it can be shown that there exists an optimal policy with at most one randomization. We used the modified MDRL algorithm for solving the above problem for different values of c_0. The parameters of the problems were $\mu = 0.5$, $L = 9$, $c = 0.5$, $\delta = 0.01$ (exploration rate in stage 2 of the modified MDRL algorithm from Figure 8), $a_L = 0.5$, $a_H = 0.8$, and the number of steering directions was 10. For three values of c_0 we show in Figures 11-13 the sample path (average reward and average constraint) on the left side and the estimation of r_{max} over time as per Equation (17) on the right side. The final estimate of the target set is also drawn for each sample path. The average vector-valued reward typically converges to the corner of the target set and pretty much stays there with random fluctuations caused by the exploration. The estimation of r_{max} is quite stable and tends to converge. As expected, the reward increases as the constraint becomes less stringent.

10. Conclusion

We have presented learning algorithms that address the problem of multi-objective decision making in a dynamic environment, which may contain arbitrarily varying elements, focusing on the long-term average reward vector. The proposed algorithms learn to approach a prescribed target set in multi-dimensional performance space, provided this set satisfies a certain geometric condition with respect to the dynamic game model. For convex sets, this sufficient condition is also necessary. The algorithms we suggested essentially rely on the theory of approachability for stochastic games, and are based on the idea of steering the average reward vector towards the target set. We then extended the algorithms to the case of a target set which is not prescribed but rather may depend on

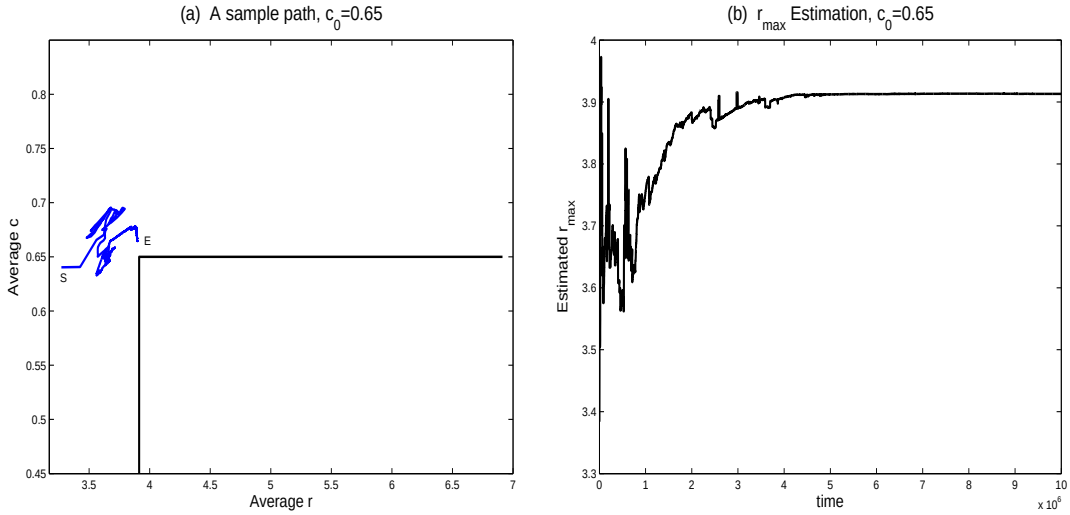


Figure 12: A sample path (a) and r_{max} estimation (b) for a run of the modified MDRL for the constrained queue problem, $c_0 = 0.65$. Note that r_{max} converged to 3.92.

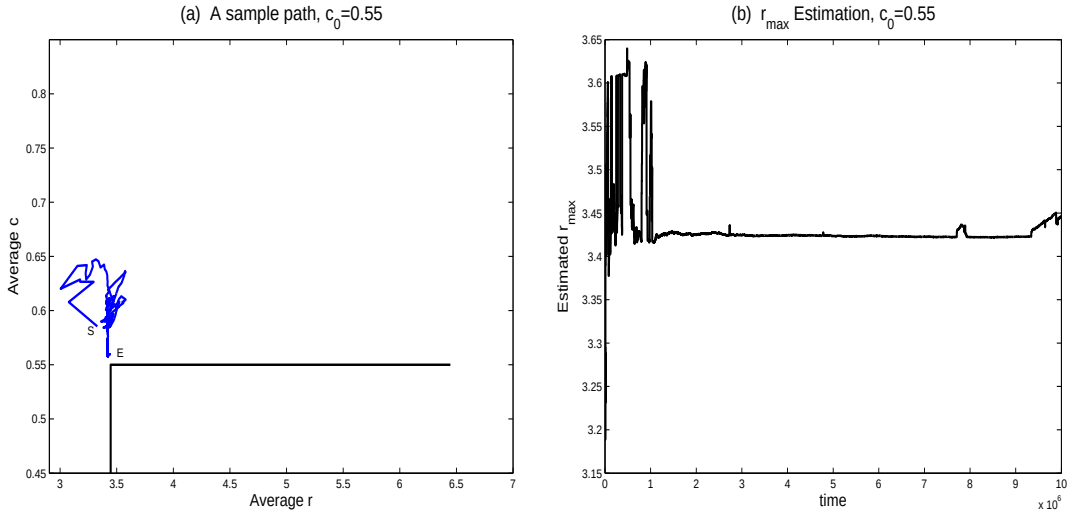


Figure 13: A sample path (a) and r_{max} estimation (b) for a run of the modified MDRL for the constrained queue problem, $c_0 = 0.55$. Note that r_{max} converged to 3.45.

the parameters of the game. This was done by requiring an order relation between candidate target sets or by assuming that the set can be learned consistently. Specifically, we showed that optimal learning policies in constrained MDPs and constrained games may be learned using the proposed schemes. It should be noted that the proposed schemes are in fact algorithmic frameworks, or *meta-*

algorithms, in which one can plug-in different scalar learning algorithms, provided they satisfy the required convergence properties. All results are developed under the single-state recurrence assumption (Assumption 1), with possible relaxations outlined in Remarks 1 and 2.

Several issues call for further study. First, the algorithmic framework we presented can undoubtedly be improved in terms of convergence speed and memory complexity. For example, a reduction of the number of steering directions used in the MDRL algorithm is of practical interest for both convergence speed (less things to learn) and memory requirement (less policies to store in memory). In some cases, especially when the requirements embodied by the target set T are not stringent, this number may be quite small compared to the conservative estimate used above. A possible refinement of the MDRL algorithm is to eliminate directions that are not required, perhaps doing so on-line. Second, specific algorithms may be devised for the special problems of constrained MDPs and constrained SGs which are simpler and with improved convergence rates. Third, the performance criteria considered here were solely based on the long-term average reward. It should be of practical interest to extend this framework to an appropriately windowed or discounted criterion, that gives more emphasis to *recent* rewards. Finally, it should be challenging to apply the proposed framework and algorithms to realistic learning problems. An interesting application class is related to communication networks, with their multiple QoS requirements and averaged performance criteria. We note that the average reward is the criterion of interest is such applications, and that the high rate of data makes our algorithms particularly useful (see Brown, 2001). Such applications would undoubtedly require further attention to issues of computational complexity and convergence rates.

Acknowledgments

This research was supported by the fund for the promotion of research at the Technion. We are indebted to three anonymous referees for thoughtful remarks and for significantly improving the presentation of this work. S.M. would like to thank Yishay Mansour, Eyal Even-Dar, and Ishai Menache for helpful discussions.

Appendix A.

In this appendix we discuss a generalization of the CDRL algorithm and prove that it approaches T rather than the ε expansion of T. Recall that in the CDRL algorithm (see Figure 5) an accuracy parameter ε was provided. The algorithm progressed in blocks where in each block a direction is fixed for a duration of $N(\varepsilon)$. In this $N(\varepsilon)$ duration the vector-valued reward is steered towards the target set T using an ε-optimal in expectation learning algorithm. The following algorithm is a modification of the CDRL algorithm where we slowly take ε to 0. As before, we denote by $N(\varepsilon)$ the time required for the learning algorithm to achieve an ε optimal reward in expectation (see Definition 12).

Theorem 26 *Suppose that for every $\varepsilon > 0$ there exists an ε-optimal in expectation algorithm for each projected SG. Suppose further that $N(\varepsilon)$ has polynomial dependence in $1/\varepsilon$, and assume that the target set satisfies Equation (3). Then T is approached using the modified CDRL algorithm.*

Proof We note that we cannot use Theorem 6 or any of its extensions from Mannor and Shimkin (2000) since a stopping time (in which steering strategies are switched) which is uniformly bounded is not provided. We therefore use a direct argument. It suffices to prove that T^η is approached for

Input: Target set T; ε-optimal learning algorithm with polynomial dependency N.
0. Set $\hat{m}_0$ arbitrarily; $i = 1$; $n = 0$.
1. Repeat forever:
2. Set $\varepsilon_i = \frac{1}{i}$.
3. If $\hat{m}_n \in T$ play arbitrarily for $N(\varepsilon_i)$ steps; $n := n + N(\varepsilon_i)$.
4. Else let $u_i = u_{\hat{m}_n}$.
5. Play an ε_i-optimal in expectation learning algorithm with reward at time n given by $r_n = u_i \cdot m_n$ and parameter ε_i. Play the algorithm for $N(\varepsilon_i)$ time epochs; $n := n + N(\varepsilon_i)$.
6. Set $i = i + 1$.

Figure 14: The modified CDRL algorithm.

every $\eta > 0$ (Mannor and Shimkin, 2000). Fix $\eta > 0$.
Let $\tau_0 = 0$ and define $\tau_i = \tau_{i-1} + N(\varepsilon_i)$ for $i > 0$. Let the polynomial dependence of the execution time of each stage of the algorithm be denoted by $N(\varepsilon_i) = Ci^\ell$. It follows that $\tau_i = C\sum_{j=1}^{i} j^\ell = O(i^{\ell+1})$.
By the definition of the algorithm either $\hat{m}_{\tau_i} \in T^\eta$ or each algorithm produces an average reward which is higher than the value of the projected game (minus ε). This implies that the expected progress in direction u_i is positive:

$$\frac{\mathbb{E}(\sum_{t=\tau_i}^{\tau_{i+1}-1} m_t \,|F_{\tau_i})}{\tau_{i+1}-\tau_i} \cdot u_i \geq \mathrm{v}\Gamma(u_i) - \varepsilon_i \,, \tag{18}$$

where F_t is the sigma algebra generated by all observations up to time t. Let i be large enough so that $\varepsilon_i < \eta$. Let d_i denote the distance from $\hat{m}_{\tau_i}$ to T^η, i.e., $d_i = d(\hat{m}_{\tau_i}, T^\eta)$. We begin by showing that $d_i \to 0$ almost surely, and then show that $d(\hat{m}_n, T^\eta) \to 0$ for all n.

Consider the square of the distance, d_i^2. Let y_i denote the closest point in T^η to $\hat{m}_{\tau_i}$, for an illustration see Figure 15. Equation (18) implies that for i large enough (so that $\varepsilon_i < \eta$),

$$\mathbb{E}\Big(\frac{(\sum_{t=\tau_i}^{\tau_{i+1}-1} m_t \,|F_{\tau_i})}{\tau_{i+1}-\tau_i} - y_i\Big) \cdot u_i \geq 0. \tag{19}$$

It follows that

$$\begin{aligned}
\mathbb{E}(d_{i+1}^2|F_{\tau_i}) &= \mathbb{E}(\|\hat{m}_{\tau_{i+1}} - y_{i+1}\|_2^2 \,|F_{\tau_i}) \\
&\leq \mathbb{E}(\|\hat{m}_{\tau_{i+1}} - y_i\|_2^2 \,|F_{\tau_i}) \\
&= \mathbb{E}(\|\hat{m}_{\tau_{i+1}} - \hat{m}_{\tau_i} + \hat{m}_{\tau_i} - y_i\|_2^2 \,|F_{\tau_i}) \\
&= \|\hat{m}_{\tau_i} - y_i\|_2^2 + \mathbb{E}(\|\hat{m}_{\tau_{i+1}} - \hat{m}_{\tau_i}\|_2^2 \,|F_{\tau_i}) + \\
&\qquad \mathbb{E}(2(\hat{m}_{\tau_i} - y_i) \cdot (\hat{m}_{\tau_{i+1}} - \hat{m}_{\tau_i}) \,|F_{\tau_i}),
\end{aligned} \tag{20}$$

where the last equality is just the result of simple algebraic manipulation. The first element in Equation (20) is simply d_i^2. To bound the second element in Equation (20) note that

$$\Big\|\hat{m}_{\tau_{i+1}} - \hat{m}_{\tau_i}\Big\|_2^2 = \Big\|\frac{1}{\tau_{i+1}} \sum_{t=\tau_i}^{\tau_{i+1}-1} m_t + (\frac{1}{\tau_{i+1}} - \frac{1}{\tau_i}) \sum_{t=0}^{\tau_i - 1} m_t\Big\|_2^2$$

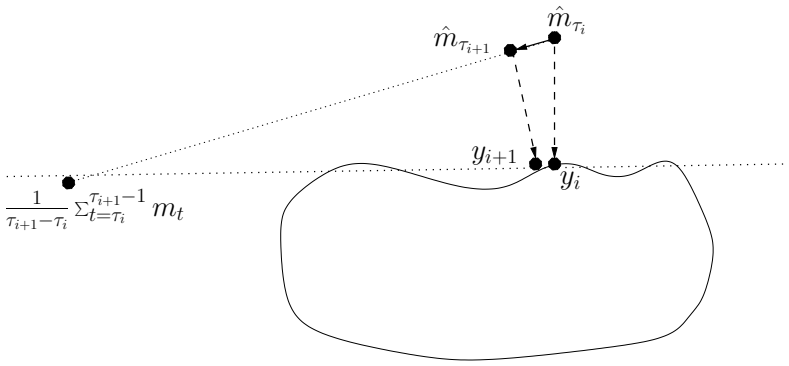


Figure 15: Illustration of one step of the algorithm. The closest point in T to $\hat{m}_{\tau_i}$ is y_i and similarly the closest point to $\hat{m}_{\tau_{i+1}}$ is y_{i+1}. Between "switching times" the reward obtained by P1 lies on the other side of the hyperplane perpendicular to the segment $(y_i, \hat{m}_{\tau_i})$.

$$\begin{aligned} &\leq \frac{C_1}{(i+1)^{2\ell+2}} \Big\| \sum_{t=\tau_i}^{\tau_{i+1}-1} m_t \Big\|_2^2 + C_2 \Big(\frac{i^{\ell+1} - (i+1)^{\ell+1}}{(i+1)^{\ell+1} i^{\ell+1}} i^{\ell+1} \Big)^2 \Big\| \hat{m}_{\tau_i} \Big\|_2^2 \\ &\leq C_3 \frac{1}{i^2} \Big(\frac{1}{\tau_{i+1} - \tau_i} \Big)^2 \Big\| \sum_{t=\tau_i}^{\tau_{i+1}-1} m_t \Big\|_2^2 + C_4 \frac{1}{i^2} \Big\| \hat{m}_{\tau_i} \Big\|_2^2, \end{aligned}$$

where we used the fact that $\|a+b\|_2^2 \leq 2\|a\|_2^2 + 2\|b\|_2^2$ for the first inequality and C_1, C_2, C_3, C_4 are constants independents of i. Taking the expectation conditioned on F_{τ_i} we can bound $\mathbb{E}(\|\hat{m}_{\tau_{i+1}} - \hat{m}_{\tau_i}\|_2^2 | F_{\tau_i})$ by $\frac{C}{i^2}$. Note that the second term is deterministic (conditioned on F_{τ_i} and vanishes from some point on.

The third element of Equation(20) is more tricky. Since

$$\begin{aligned} \hat{m}_{\tau_{i+1}} - \hat{m}_{\tau_i} &= \frac{1}{\tau_{i+1}} \sum_{t=0}^{\tau_{i+1}-1} m_t - \frac{1}{\tau_i} \sum_{t=0}^{\tau_i - 1} m_t \\ &= (\frac{1}{\tau_{i+1}} - \frac{1}{\tau_i}) \sum_{t=0}^{\tau_i - 1} m_t + \frac{1}{\tau_{i+1}} \sum_{t=\tau_i}^{\tau_{i+1}-1} m_t \\ &= \frac{\tau_i - \tau_{i+1}}{\tau_{i+1}} \hat{m}_{\tau_i} + \frac{1}{\tau_{i+1}} \sum_{t=\tau_i}^{\tau_{i+1}-1} m_t, \end{aligned}$$

we have that

$$\begin{aligned} (\hat{m}_{\tau_i} - y_i) \cdot (\hat{m}_{\tau_{i+1}} - \hat{m}_{\tau_i}) &= (\hat{m}_{\tau_i} - y_i) \cdot (\hat{m}_{\tau_{i+1}} - \frac{\tau_i - \tau_{i+1}}{\tau_{i+1}} y_i + \frac{\tau_i - \tau_{i+1}}{\tau_{i+1}} y_i - \hat{m}_{\tau_i}) \\ &= (\hat{m}_{\tau_i} - y_i) \cdot \Big(\frac{\tau_i - \tau_{i+1}}{\tau_{i+1}} \hat{m}_{\tau_i} - \frac{\tau_i - \tau_{i+1}}{\tau_{i+1}} y_i + \\ &\qquad \frac{\tau_i - \tau_{i+1}}{\tau_{i+1}} y_i + \frac{1}{\tau_{i+1}} \sum_{t=\tau_i}^{\tau_{i+1}-1} m_t \Big) \\ &= \frac{\tau_i - \tau_{i+1}}{\tau_{i+1}} (\hat{m}_{\tau_i} - y_i) \cdot (\hat{m}_{\tau_i} - y_i) \end{aligned}$$

$$+\frac{\tau_i-\tau_{i+1}}{\tau_{i+1}}(\hat{m}_{\tau_i}-y_i)\cdot(\frac{1}{\tau_{i+1}-\tau_i}\sum_{t=\tau_i}^{\tau_{i+1}-1}m_t-y_i).$$

Since $\tau_i = Ci^{\ell+1}$ the fraction $\frac{\tau_{i+1}-\tau_i}{\tau_{i+1}} = O(1/i)$. The first term is therefore approximately $-d_i^2\frac{C}{i}$. We get the following inequality:

$$\mathbb{E}(d_{i+1}^2|F_{\tau_i}) \le (1-\frac{C'}{i})d_i^2+\frac{C''}{i^2}+C'''\mathbb{E}\left(2\frac{(\hat{m}_{\tau_i}-y_i)\cdot(\sum_{t=\tau_i}^{\tau_{i+1}-1}m_t-y_i)}{i+1}\Big|F_{\tau_i}\right), \tag{21}$$

where C', C'', and C''' are positive constants. Now according to Equation (19) the last term in (21) is negative in expectation. We therefore have that $\mathbb{E}(d_{i+1}^2|F_{\tau_i}) \le (1-\frac{C_1}{i})d_i^2+\frac{C_2}{i^2}$. Using Lemma 27 we have that $d_i^2 \to 0$ almost surely. To conclude the proof one should show that $d(\hat{m}_n, T^\eta) \to 0$ for all n. Let i denote the maximal τ_i smaller than n. By the triangle inequality,

$$d(\hat{m}_n, T^\eta) \le d(\hat{m}_{\tau_i}, T^\eta) + \|\hat{m}_{\tau_i}-\hat{m}_n\|_2 .$$

The first term converges to 0 by the above. To bound the second term note that

$$\begin{aligned}
\|\hat{m}_{\tau_i}-\hat{m}_n\|_2 &= \Big\|\frac{1}{n}\sum_{t=0}^{n-1}m_t-\frac{1}{\tau_i}\sum_{t=0}^{\tau_i-1}m_t\Big\|_2 \\
&= \Big\|\frac{1}{n}\sum_{t=\tau_i}^{n-1}m_t+\frac{\tau_i-n}{n\tau_i}\sum_{t=0}^{\tau_i-1}m_t\Big\|_2 \\
&\le \frac{1}{n}\Big\|\sum_{t=\tau_i}^{n-1}m_t\Big\|_2+\frac{n-\tau_i}{n\tau_i}\Big\|\sum_{t=0}^{\tau_i-1}m_t\Big\|_2 \\
&= \frac{n-\tau_i}{n}\frac{1}{n-\tau_i}\Big\|\sum_{t=\tau_i}^{n-1}m_t\Big\|_2+\frac{n-\tau_i}{n}\frac{1}{\tau_i}\Big\|\sum_{t=0}^{\tau_i-1}m_t\Big\|_2,
\end{aligned} \tag{22}$$

where the inequality is due to the triangle inequality. By our construction of τ_i it follows that $\lim_{n\to\infty}\frac{n-\tau_i}{n}=0$. Since the random vector reward m_t has finite expectation and bounded second moment (uniformly in t, since there are finitely many states and actions) it follows that $\frac{1}{\tau_i}\|\sum_{t=0}^{\tau_i-1}m_t\|_2$ is asymptotically contained in some ball of finite radius with probability one.[3] Consequently, the second term in Equation (22) converges to 0 almost surely. The first term in Equation (22) converges to 0 as well by a standard probabilistic argument.[4] ∎

Lemma 27 *Assume e_t is a non-negative random variable, measurable according to the sigma algebra F_t ($F_t \subset F_{t+1}$) and that*

$$\mathbb{E}(e_{t+1}|F_t) \le (1-d_t)e_t + cd_t^2. \tag{23}$$

Further assume that $\sum_{t=1}^{\infty}d_t=\infty$, $d_t \ge 0$, and that $d_t \to 0$. Then $e_t \to 0$ P-a.s.

3. To see that, take the centralized version of that random variable which converges to 0, almost surely by an appropriate version of the strong law of large numbers. Note that the center may not converge, but is guaranteed to be in some bounded region.
4. The result follows by Chebyshev's inequality on the centralized version of $\sum_{t=\tau_i}^{n-1}m_t$ and the Borel-Cantelli Lemma, noticing that $\sum_{t=\tau_i}^{n-1}m_t$ is multiplied by a deterministic series that converges to 0.

Proof First note that by taking the expectation of Equation (23) we get

$$\mathbb{E}e_{t+1} \leq (1-d_t)\mathbb{E}e_t + cd_t^2.$$

According to Bertsekas and Tsitsiklis (1995, page 117) it follows that $\mathbb{E}e_t \to 0$. Since e_t is non-negative it suffices to show that e_t converges. Fix $\varepsilon > 0$, let

$$V_t^\varepsilon \stackrel{\triangle}{=} \max\{\varepsilon, e_t\}.$$

Since $d_t \to 0$ there exists $T(\varepsilon)$ such that $cd_t < \varepsilon$ for $t > T$. Restrict attention to $t > T(\varepsilon)$. If $e_t < \varepsilon$ then

$$\mathbb{E}(V_{t+1}^\varepsilon | F_t) \leq (1-d_t)\varepsilon + cd_t^2 \leq \varepsilon \leq V_t^\varepsilon.$$

If $e_t > \varepsilon$ we have

$$\mathbb{E}(V_{t+1}^\varepsilon | F_t) \leq (1-d_t)e_t + d_t e_t \leq V_t^\varepsilon.$$

V_t^ε is a super-martingale, by a standard convergence argument we get $V_t^\varepsilon \to V_\infty^\varepsilon$.

By definition $V_t^\varepsilon \geq \varepsilon$ and therefore $\mathbb{E}V_t^\varepsilon \geq \varepsilon$. Since $\mathbb{E}[\max(X,Y)] \leq \mathbb{E}X + \mathbb{E}Y$ it follows that $\mathbb{E}V_t^\varepsilon \leq \mathbb{E}e_t + \varepsilon$. So that $\mathbb{E}V_\infty^\varepsilon = \varepsilon$. Now we have a positive random variable, with expectation ε which is not smaller than ε with probability 1. It follows that $V_\infty^\varepsilon = \varepsilon$.

To summarize, we have shown that for every $\varepsilon > 0$ with probability 1,

$$\limsup_{t\to\infty} e_t \leq \limsup_{t\to\infty} V_t^\varepsilon = \lim_{t\to\infty} V_t^\varepsilon = \varepsilon.$$

Since ε is arbitrary and e_t non-negative it follows that $e_t \to 0$ almost surely. ∎

References

J. Abounadi, D. Bertsekas, and V. Borkar. Stochastic approximation for non-expansive maps: Application to Q-learning algorithms. *SIAM Journal on Control and Optimization*, 41:1–22, 2002.

E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

E. Altman and A. Shwartz. Constrained Markov games: Nash equilibria. In V. Gaitsgory, J. Filar, and K. Mizukami, editors, *Annals of the International Society of Dynamic Games*, volume 5, pages 303–323. Birkhauser, 2000.

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

ATM Forum Technical Committee. Traffic management specification version 4.1. www.atmforum.org, March 1999.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002.

J. Baxter, A. Tridgell, and L. Weaver. TDLeaf(λ): Combining temporal difference learning with game-tree search. In *Proceedings of the 9th Australian Conference on Neural Networks (ACNN-98)*, 1998. URL `http://cs.anu.edu.au/~Lex.Weaver/pub_sem/publications/ACNN98.pdf`.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1995.

D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

R. I. Brafman and M. Tennenholtz. R-MAX, a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

T. X. Brown. Switch packet arbitration via queue-learning. In *Advances in Neural Information Processing Systems 14*, pages 1337–1344, 2001.

R. L. Carraway, T. L. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44:95–104, 1990.

C. Derman. *Finite state Markovian decision processes*. Academic Press, 1970.

M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization. State of the Art Annotated Bibliographic Surveys*. Kluwer, 2002.

J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer Verlag, 1996.

Z. Gábor, Z. Kalmár, and C. Szepesvári. Multi-criteria reinforcement learning. In *Proc. of the 15th Int. Conf. on Machine Learning*, pages 197–205. Morgan Kaufmann, 1998.

M. I. Henig. Vector-valued dynamic programming. *SIAM Journal on Control and Optimization*, 21 (3):490–499, 1983.

A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Science*, 12(5): 359–370, 1966.

L. P. Kaelbling, M. Littman, and A. W. Moore. Reinforcement learning - a survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.

D. Kahaneman and A. Tversky. Prospect theory: an analysis of decision under risk. *Econometrica*, 47:263–291, 1979.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *Proc. of the 15th Int. Conf. on Machine Learning*, pages 260–268. Morgan Kaufmann, 1998.

M. Kearns and S. Singh. Finite-sample convergence rates for Q-learning and indirect algorithms. In *Neural Information Processing Systems 11*, pages 996–1002. Morgan Kaufmann, 1999.

V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. Submitted to the SIAM Journal on Control and Optimization, an earlier version appeared in NIPS 1999, February 2001.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In Morgan Kaufman, editor, *Eleventh International Conference on Machine Learning*, pages 157–163, 1994.

M. L. Littman and C. Szepesvári. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Computation*, 11(8):2017–2059, 1999.

S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1):159–196, 1996.

S. Mannor. *Reinforcement Learning and Adaptation in Competitive Dynamic Environments*. PhD thesis, Department of Electrical Engineering, Technion, April 2002.

S. Mannor and N. Shimkin. Generalized approachability results for stochastic games with a single communicating state. Technical report EE- 1263, Faculty of Electrical Engineering, Technion, Israel, October 2000. Appeared in ORP3, 2001.

S. Mannor and N. Shimkin. Reinforcement learning for average reward zero-sum games. Technical report EE- 1316, Faculty of Electrical Engineering, Technion, Israel, May 2002.

S. Mannor and N. Shimkin. The empirical Bayes envelope and regret minimization in competitive Markov decision processes. *Mathematics of Operations Research*, 28(2):327–345, May 2003.

J. F. Mertens. Stochastic games. In Robert J. Aumann and Sergiu Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3, chapter 47. Elsevier Science Publishers, 2002.

J. F. Mertens and A. Neyman. Stochastic games. *International Journal of Game Theory*, 10(2): 53–66, 1981.

A. S. Pozniyak, K. Najim, and E. Gomez-Ramirez. *Self Learning Control of Finite Markov Chains*. Marcel Decker, 1999.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 298–305. Morgan Kaufmann, 1993.

N. Shimkin. Stochastic games with average cost constraints. In T. Basar and A. Haurie, editors, *Advances in Dynamic Games and Applications*, pages 219–230. Birkhauser, 1994.

N. Shimkin and A. Shwartz. Guaranteed performance regions in Markovian systems with competing decision makers. *IEEE Transactions on Automatic Control*, 38(1):84–95, January 1993.

H. A. Simon. *The Sciences of the Artificial*. MIT Press, 1996.

S. Singh, T. Jaakkola, M. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. Wiley, 1986.

R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, 1998.

G. J. Tesauro. TD-gammon, a self-teaching backgammon program, achieves a master-level play. *Neural Computation*, 6:215–219, 1996.

D. White. Multi-objective infinite-horizon discounted markov decision processes. *Journal of Mathematical Analysis and Applications*, 89:639–647, 1982.

Journal of Machine Learning Research 5 (2004) 361-397 Submitted 9/02; Published 4/04

RCV1: A New Benchmark Collection for Text Categorization Research

David D. Lewis LYRL2004@DAVIDDLEWIS.COM
Ornarose, Inc. and David D. Lewis Consulting
858 West Armitage Avenue, #296
Chicago, IL 60614, USA

Yiming Yang YIMING@CS.CMU.EDU
Language Technologies Institute & Computer Science Department
Carnegie Mellon University
Newell Simon Hall 3612D, LTI
5000 Forbes Avenue
Pittsburgh, PA 15213, USA

Tony G. Rose TGR@TONYROSE.NET
Cancer Research UK
Advanced Computation Laboratory
44 Lincoln's Inn Fields
London WC2A 3PX, UK

Fan Li HUSTLF@CS.CMU.EDU
Language Technologies Institute
Carnegie Mellon University
Newell Simon Hall 3612D, LTI
5000 Forbes Avenue
Pittsburgh, PA 15213, USA

Editor: Thomas G. Dietterich

Abstract

Reuters Corpus Volume I (RCV1) is an archive of over 800,000 manually categorized newswire stories recently made available by Reuters, Ltd. for research purposes. Use of this data for research on text categorization requires a detailed understanding of the real world constraints under which the data was produced. Drawing on interviews with Reuters personnel and access to Reuters documentation, we describe the coding policy and quality control procedures used in producing the RCV1 data, the intended semantics of the hierarchical category taxonomies, and the corrections necessary to remove errorful data. We refer to the original data as RCV1-v1, and the corrected data as RCV1-v2. We benchmark several widely used supervised learning methods on RCV1-v2, illustrating the collection's properties, suggesting new directions for research, and providing baseline results for future studies. We make available detailed, per-category experimental results, as well as corrected versions of the category assignments and taxonomy structures, via online appendices.

Keywords: applications, automated indexing, controlled vocabulary indexing, effectiveness measures, evaluation, feature selection, *k*-NN, methodology, multiclass, multilabel, nearest neighbor, news articles, operational systems, Rocchio, SCut, SCutFBR, support vector machines, SVMs, term weighting, test collection, text classification, thresholding

1. Introduction

Text categorization is the automated assignment of natural language texts to predefined categories based on their content. It is a supporting technology in several information processing tasks, including controlled vocabulary indexing, routing and packaging of news and other text streams, content filtering (spam, pornography, etc.), information security, help desk automation, and others. Closely related technology is applicable to other classification tasks on text, including classification with respect to personalized or emerging classes (alerting systems, topic detection and tracking), non-content based classes (author identification, language identification), and to mixtures of text with other data (multimedia and cross-media indexing, text mining).

Research interest in text categorization has been growing in machine learning, information retrieval, computational linguistics, and other fields. This partly reflects the importance of text categorization as an application area for machine learning, but also results from the availability of *text categorization test collections* (Lewis, Schapire, Callan, and Papka, 1996; Lewis, 1997; Yang, 1999; Sebastiani, 2002). These are collections of documents to which human indexers have assigned categories from a predefined set. Test collections enable researchers to test ideas without hiring indexers, and (ideally) to objectively compare results with published studies.

Existing text categorization test collections suffer from one or more of the following weaknesses: few documents, lack of the full document text, inconsistent or incomplete category assignments, peculiar textual properties, and/or limited availability. These difficulties are exacerbated by a lack of documentation on how the collections were produced and on the nature of their category systems. The problem has been particularly severe for researchers interested in hierarchical text categorization who, due to the lack of good collections and good documentation, have often been forced to impose their own hierarchies on categories (Koller and Sahami, 1997; Weigend, Wiener and Pedersen, 1999).

Even if current collections were perfect, however, there would be an ongoing need for new ones. Just as machine learning algorithms can overfit by tuning a classifier's parameters to the accidental properties of a training set, a research community can overfit by refining algorithms that have already done well on the existing data sets. Only by periodically testing algorithms on new test collections can progress be verified.

A data set recently made available, Reuters Corpus Volume 1 (RCV1) (Rose, Stevenson and Whitehead, 2003), has the potential to address many of the above weaknesses. It consists of over 800,000 newswire stories that have been manually coded using three category sets. However, RCV1 as distributed is simply a collection of newswire stories, not a test collection. It includes known errors in category assignment, provides lists of category descriptions that are not consistent with the categories assigned to articles, and lacks essential documentation on the intended semantics of category assignment.

This paper attempts to provide the necessary documentation, and to describe how to eliminate miscodings where possible. We begin in Section 2 by describing the operational setting in which RCV1 was produced, with particular attention to the categories and how they were assigned. Besides being crucial to understanding the semantics of the category assignments, the insight into operational text categorization may be of independent interest. Section 3 examines the implications of the production process for the use of RCV1 in research, and Section 4 summarizes the changes we recommend to produce a better test collection, which we call RCV1-v2. (We refer to the original data as RCV1-v1.)

Appendix	Description
1	Valid Topic categories
2	Original Topics hierarchy
3	Expanded Topics hierarchy
4	Valid Industry categories
5	Best-guess Industries hierarchy
6	Valid Region categories
7	IDs of RCV1-v2 documents
8	RCV1-v2 Topic assignments
9	RCV1-v2 Industry assignments
10	RCV1-v2 Region assignments
11	SMART stopword list
12	Tokenized RCV1-v2 data
13	Vectorized RCV1-v2 data (LYRL2004 training/test split)
14	Term dictionary for vectorized data
15	Contingency tables for experimental results
16	RBB Topics list
17	RBB Industries list
18	RBB Regions list

Table 1: List of online appendices accompanying this paper. They provide data sets used in or produced by the experiments, as well as additional information on the RCV1 collection, and are explained later in the paper.

Sections 2 to 4 are based on Reuters documentation, interviews with Reuters personnel, and statistical analysis of the documents and categories. To complement this analysis, we provide benchmark results on RCV1-v2 for well-known supervised learning approaches to text categorization. These results provide future users with a standard for comparison, as well as reassurance that the tasks posed by the corrected collection are neither trivial nor impossible. Section 5 gives the design of our experiments, Sections 6 & 7 discuss the algorithms and text representation, and Section 8 presents the benchmark results and observations. We end with some thoughts on research directions the new collection may support.

Several online appendices accompany this paper, and are listed in Table 1.

2. Coding the RCV1 Data

> *Apart from the terrible memories this stirs up for me personally (coding stories through the night etc.), I can't find fault with your account.*
>
> – Reuters editor commenting on a draft of this section.

The RCV1 data was produced in an operational setting at Reuters, Ltd., under procedures that have since been superceded. Only later was use of the data in research contemplated. Information that in a research setting would have been retained was therefore not recorded. In particular,

no formal specification remains of the coding practices at the time the RCV1 data was produced. However, by combining related documentation and interviews with Reuters personnel we believe we have largely reconstructed those aspects of coding relevant to text categorization research.

2.1 The Documents

Reuters is the largest international text and television news agency. Its editorial division produces some 11,000 stories a day in 23 languages. Stories are both distributed in real time and made available via online databases and other archival products.

RCV1 is drawn from one of those online databases. It was intended to consist of all and only English language stories produced by Reuters journalists between August 20, 1996, and August 19, 1997. The data is available on two CD-ROMs and has been formatted in XML.[1] Both the archiving process and later preparation of the XML dataset involved substantial verification and validation of the content, attempts to remove spurious or duplicated documents, normalization of dateline and byline formats, addition of copyright statements, and so on.

The stories cover the range of content typical of a large English language international newswire. They vary from a few hundred to several thousand words in length. Figure 1 shows an example story (with some simplification of the markup for brevity).

2.2 The Categories

To aid retrieval from database products such as Reuters Business Briefing (RBB), category codes from three sets (Topics, Industries, and Regions) were assigned to stories. The code sets were originally designed to meet customer requirements for access to corporate/business information, with the main focus on company coding and associated topics. With the introduction of the RBB product the focus broadened to the end user in large corporations, banks, financial services, consultancy, marketing, advertising and PR firms.

2.2.1 TOPIC CODES

Topic codes were assigned to capture the major subjects of a story. They were organized in four hierarchical groups: CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT (Markets). This code set provides a good example of how controlled vocabulary schemes represent a particular perspective on a data set. The RCV1 articles span a broad range of content, but the code set only emphasizes distinctions relevant to Reuters' customers. For instance, there are three different Topic codes for corporate ownership changes, but all of science and technology is a single category (GSCI).

2.2.2 INDUSTRY CODES

Industry codes were assigned based on types of businesses discussed in the story. They were grouped in 10 subhierarchies, such as I2 (METALS AND MINERALS) and I5 (CONSTRUCTION). The Industry codes make up the largest of the three code sets, supporting many fine distinctions.

1. Further formatting details are available at *http://about.reuters.com/researchandstandards/corpus/*.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<newsitem itemid="2330" id="root" date="1996-08-20" xml:lang="en">
<title>USA: Tylan stock jumps; weighs sale of company.</title>
<headline>Tylan stock jumps; weighs sale of company.</headline>
<dateline>SAN DIEGO</dateline>
<text>
<p>The stock of Tylan General Inc. jumped Tuesday after the maker of
process-management equipment said it is exploring the sale of the
company and added that it has already received some inquiries from
potential buyers.</p>
<p>Tylan was up $2.50 to $12.75 in early trading on the Nasdaq market.</p>
<p>The company said it has set up a committee of directors to oversee
the sale and that Goldman, Sachs &amp;amp; Co. has been retained as its
financial adviser.</p>
</text>
<copyright>(c) Reuters Limited 1996</copyright>
<metadata>
<codes class="bip:countries:1.0">
  <code code="USA"> </code>
</codes>
<codes class="bip:industries:1.0">
  <code code="I34420"> </code>
</codes>
<codes class="bip:topics:1.0">
  <code code="C15">  </code>
  <code code="C152"> </code>
  <code code="C18">  </code>
  <code code="C181"> </code>
  <code code="CCAT"> </code>
</codes>
<dc element="dc.publisher" value="Reuters Holdings Plc"/>
<dc element="dc.date.published" value="1996-08-20"/>
<dc element="dc.source" value="Reuters"/>
<dc element="dc.creator.location" value="SAN DIEGO"/>
<dc element="dc.creator.location.country.name" value="USA"/>
<dc element="dc.source" value="Reuters"/>
</metadata>
</newsitem>
```

Figure 1: An example Reuters Corpus Volume 1 document.

2.2.3 REGION CODES

Region codes included both geographic locations and economic/political groupings. No hierarchical taxonomy was defined.

2.3 Coding Policy

Explicit policies on code assignment presumedly increase consistency and usefulness of coding, though coming up with precise policies is difficult (Lancaster, 1998, pp. 30-32). Reuters' guidance for coding included two broad policies, among others. We have named these policies for convenience, though they were not so named by Reuters:

1. Minimum Code Policy: Each story was required to have at least one Topic code and one Region code.

2. Hierarchy Policy: Coding was to assign the most specific appropriate codes from the Topic and Industry sets, as well as (usually automatically) all ancestors of those codes. In contrast to some coding systems, there was no limit on the number of codes with the same parent that could be applied.

These policies were (imperfectly) implemented by a combination of manual and automated means during coding, as discussed below and in Section 3.3.

2.4 The Coding Process

During the years 1996 and 1997, the period from which the corpus is drawn, Reuters produced just over 800,000 English language news stories per year. Coding was a substantial undertaking. At one point Reuters employed 90 people to handle the coding of 5.5 million English language stories per year. However, this figure includes both English language stories produced by Reuters journalists and ones obtained from other sources, and included additional code sets not present in the RCV1 data. Therefore, the exact effort devoted to documents and codes of the sort represented in RCV1 is unclear, though one estimate is around 12 person-years (Rose, Stevenson and Whitehead, 2003).

Coding of Reuters-produced stories was accomplished in three stages: autocoding, manual editing, and manual correction.

2.4.1 AUTOCODING

Stories first passed through a rule-based text categorization system known as *TIS* (Topic Identification System), a descendant of the system originally developed for Reuters by Carnegie Group (Hayes and Weinstein, 1990). Most codes had at least one rule that could assign them, but automated coding was not attempted for some codes believed to be beyond the capability of the technology. Two of the codes perceived to be difficult were GODD (human interest) and GOBIT (obituaries). It is interesting to note that these two categories proved in our experiments to be two of the most difficult to assign automatically.

In addition to their text, some stories entering the system already had codes, from a different code set (the "Editorial codes"), that had been manually assigned by journalists. Some simple "source processing" rules were used that mapped these codes to equivalent codes in the final code set. For example, a story with the Editorial code SPO (Sport) would automatically be assigned

the final code GSPO. Other source processing rules triggered on other parts of the markup, for instance assigning any story whose slug (a brief line of identifying information on a newswire story) contained the string "BC-PRESS DIGEST" to the most general news code (GCAT).

Finally, as discussed in Section 3, some Topic, Industry, and Region codes were assigned on the basis of other codes of the same or different type, to enforce the Hierarchy Policy or capture other relationships.

2.4.2 MANUAL EDITING

The output of TIS was automatically checked for compliance with the Minimum Code policy. If so, the story was sent to a holding queue. If not, the story was first sent to a human editor. This editor would assign the codes they felt applied, while ensuring the story got at least one Topic and one Region code. Editors could also delete or change automatically assigned codes. Editors occasionally fixed errors in the formatting of the story during this phase, but their primary responsibility was correction of coding. The edited story then went to the holding queue for final review.

2.4.3 MANUAL CORRECTION IN THE HOLDING QUEUE

Every six hours, the holding queue was reviewed by editors, who had the opportunity to correct mistakes in coding. Once stories passed through the holding queue, they were batched up and loaded into the database in blocks.

2.5 Coding Quality

Human coding is inevitably a subjective process. Studies have shown considerable variation in interindexer consistency rates for different data sets (Cleverdon, 1991). The process described above was an attempt to achieve high consistency and correctness for the Reuters codes. Stories were sampled periodically and feedback given to coders on how to improve their accuracy. The consistency of coders with each other and with standards was evaluated from samples and found to be high, though we were not able to obtain quantitative data from these evaluations for publication.

Table 2 provides some additional evidence of consistency of the coding. It shows, for the year 1997, how many stories had autocoding that failed the Minimum Code test and thus underwent manual editing, as well as how many had at least one code corrected in the holding queue. Note that RCV1 contains stories spanning parts of 1996 and 1997, so the number of stories in the corpus is not the same as the number of stories in Table 2.

A total of 312,140 stories had autocoding that failed the Minimum Code test and were thus manually edited. All of these stories were also reviewed by a second editor in the holding queue, but only 23,289 or 13.4% had codes changed by that second editor. In contrast, 334,975 (66.2%) of the 505,720 stories whose autocoding passed the Minimum Code test were changed in the holding queue. In other words, a manually edited coding was much less likely to be overridden in the holding queue than a coding assigned by the automated system.

It should be noted that an annotation let editors reviewing the holding queue know which stories had been manually edited, and this undoubtedly influenced their choice of stories to correct. Table 2 therefore cannot be considered an objective measure of interindexer consistency. However, it provides some additional evidence that the different human coders were mostly in agreement on the meaning of the codes. Rose, Stevenson and Whitehead (2003) present additional data on corrections by editors.

		Manually Corrected	
		No	Yes
Manually Edited	No	170,745	334,975
	Yes	288,851	23,289

Table 2: Number of stories produced by Reuters in 1997 that received manual editing and/or manual correction.

2.6 The Evolution of Coding at Reuters

It should be mentioned that the above approach, based on TIS and manual correction, has since been superceded at Reuters. The rule-based approach of TIS had several drawbacks:

- Creating rules required specialized knowledge, thus slowing down the addition of new codes and the adaptation of rules to changes in the input.
- The rules did not provide an indication of the confidence in their output. There was thus no way to focus editorial correction on the most uncertain cases, nor any way of detecting (except by violation of coding policy) that new types of stories were appearing that would suggest changes or additions to the code set.

Reuters now uses a machine learning approach for text categorization. Classifiers are induced from large amounts of training data, with a feedback loop to trigger the involvement of human editors (based on autocoding confidence scores) and analysis tools to indicate when new training data/categories may be required.

3. RCV1 and Text Categorization Research

A test collection is more than a corpus. In this section we consider how the production and character of RCV1 impact its use for text categorization research. In Section 4 we go on to describe how to correct errors in the raw RCV1 data (which we call RCV1-v1) to produce a text categorization test collection (which we call RCV1-v2). Therefore here we present statistics for both versions of the data, indicating when they are different.

3.1 Documents

RCV1 contains 35 times as many documents (806,791 for RCV1-v1, and 804,414 for RCV1-v2) as the popular Reuters-21578 collection and its variants (Lewis, 1992, 1997), and 60 times as many with reliable coding. Indeed, the only widely available text categorization test collection of comparable size is OHSUMED (Hersh, Buckley, Leone, and Hickman, 1994; Lewis, Schapire, Callan, and Papka, 1996; Yang and Pedersen, 1997; Yang, 1999) at 348,566 documents. While useful, OHSUMED has disadvantages: it does not contain the full text of documents, its medical language is hard for non-experts to understand, and its category hierarchy (MeSH) is huge and structurally complex.

RCV1 is also “cleaner” than previous collections. Stories appear one to a file, and have unique document IDs. IDs range from 2286 to 810597 for RCV1-v1, and 2286 to 810596 for RCV1-v2.

There are gaps in the range of IDs in the original RCV1-v1, and additional gaps (due to deleted documents) in RCV1-v2. Regrettably, the ID order does not correspond to chronological order of the stories, even at the level of days. Fortunately, the documents do have time stamps (in the <newsitem> element), and chronological order at the level of days can be determined from those. The time stamps do not give a time of day since the stories were taken from an archival database, not from the original stream sent out over the newswire.

XML formatting of both text and metadata in RCV1 simplifies use of the data. The fact that the stories are from an archival database means fewer brief alerts (the infamous *"blah, blah, blah"* stories of Reuters-21578), corrections to previous stories, and other oddities. RCV1 contains all or almost all stories of a particular type from an interval of one year. For temporal studies, this is a major advantage over Reuters-21578, which had bursty coverage of a fraction of a year.

The processes that produced the archival database and, later, the research corpus, were inevitably imperfect. Khmelev and Teahan (2003) discuss a number of anomalies in the corpus, including the presence of approximately 400 foreign language documents. They also emphasize the presence of duplicate and near-duplicate articles. Some of these simply reflect the fact that very similar stories do occasionally appear, particularly ones containing financial data. In other cases multiple drafts of the same story were retained. Some simple accidents undoubtedly occurred as well.

We found between 2,500 and 30,000 documents that could be considered duplicates of some other document, depending on the definition of duplication. Our analysis is consistent with that of Teahan and Kmelev, who found 27,754 duplicate or substantially overlapping documents in their analysis.

Whether the number of duplicates, foreign language documents, and other anomalies present in RCV1 is problematic depends on the questions a researcher is using RCV1 to study. We believe the number of such problems is sufficiently small, or sufficiently similar to levels seen in operational settings, that they can be ignored for most purposes.

3.2 Categories

RCV1 documents are categorized with respect to three controlled vocabularies: *Topics*, *Industries*, and *Regions*. In this section, we discuss the three RCV1 category sets and their implications for text categorization experiments. In particular, we describe our interpretation of the hierarchical structure for each code set, something that is not made clear in the documentation on the RCV1 CD-ROMs.

3.2.1 TOPIC CODES

The file *topic_codes.txt* on the RCV1 CD-ROMs lists 126 Topic codes. However, some of these codes were not actually used by editors at the time the RCV1 data was categorized. Various evidence, including Reuters documentation on an alternate version of the Topics hierarchy, suggests that these codes were not used:

1POL, 2ECO, 3SPO, 4GEN, 6INS, 7RSK, 8YDB, 9BNX, ADS10, BRP11, ENT12, PRB13, BNW14, G11, G111, G112, G113, G12, G13, G131, G14, GEDU, MEUR.

This leaves 103 Topic codes we believe were actually available for coding, and which we therefore recommend be used in text categorization experiments. We provide a list of valid Topic codes as Online Appendix 1. As it happens, all of these 103 codes occur at least once in both the RCV1-v1 and RCV1-v2 datasets. Their corpus frequencies span five orders of magnitude, from 5 occurrences

for GMIL (MILLENNIUM ISSUES), to 374,316 occurrences (381,327 in RCV1-v2) for CCAT (CORPORATE/INDUSTRIAL). Note that some Topic category frequencies are higher in RCV1-v2 than in RCV1-v1, despite RCV1-v1 having fewer documents, because RCV1-v2 fills in missing hierarchical expansions of Topic categories (Section 4).

The code symbols inserted in articles to indicate their membership in categories were chosen so that related categories would have related codes. This "morphological structure" of the codes reflects two distinct needs:

1. Defining a hierarchy to support automated assignment of more general codes on the basis of (manual or automated) assignment of more specific codes (Section 3.3).

2. Imposing an alphanumeric sort order that grouped related codes, aiding manual lookup.

For instance, the code C311 (DOMESTIC MARKETS) is a child in the hierarchy of its truncation, code C31 (MARKETS/MARKETING). Code C311 also appears near related codes, such as C32 (ADVERTISING/PROMOTION), in an alphanumeric listing.

The original hierarchy used for automated assignment can be reconstructed as follows:

1. Treat the codes CCAT, ECAT, GCAT, and MCAT as actually being the corresponding single letters C, E, G, and M.

2. To find the parent of a code, remove the minimal suffix such that the result is another code. The codes C, E, G, and M have as parent the root of the tree.

However, there are other versions of the hierarchy that might be of interest. In particular, one could introduce an additional level of the hierarchy corresponding to the high level numeric groupings that aided lookup. This can be done by first adding to the hierarchy the artificial codes C1-C4, E1-E7, G1, and M1, and then following the above procedure. Taking into account this new hierarchy level might (or might not) improve the effectiveness of hierarchy-based algorithms when assigning the original 103 categories. (We doubt it is interesting to actually assign the 13 artificial codes to documents or to measure classifiers' accuracy at assigning them.)

Online Appendix 2 specifies the original version of the hierarchy. It contains a total of 104 nodes: 103 for assignable Topic codes and 1 root node. Online Appendix 3 specifies a hierarchy that includes the two-character truncations as a new intermediate layer. It contains a total of 117 nodes: 103 for assignable Topic codes, 13 nodes in the new non-assignable intermediate layer, and 1 root node.

Editors were able to assign any of the 103 Topic codes to a story, not just codes at leaf nodes of the hierarchy. They were instructed to use the most specific code applicable to a particular aspect of a story, a common indexing principle (Lancaster, 1998, pp. 28-30). Codes at internal nodes of the hierarchy thus acted much like named "Other" categories, implicitly forming a contrast set with their child codes.

However, in the RCV1 data a non-leaf code may be present not because it was directly found to be applicable, but because it was the ancestor of a code found to be applicable. We call this "Other + expansion" semantics, to distinguish it from pure "Other" semantics. We discuss the implications of this for research use of RCV1 in Section 3.3.

3.2.2 INDUSTRY CODES

The file *industry_codes.txt* on the RCV1 CD-ROMs lists a total of 870 codes. Most do not appear in the documents and at first glance they appear confusing and redundant. As discussed below, only 354 of these codes appear to have been available for use by the coders. We therefore recommend that only these 354 codes (which we list in Online Appendix 4) be used in experiments.

Of the 354 valid Industry codes, 350 have at least one occurrence in the corpus (in both RCV1-v1 and RCV1-v2). Nonzero category frequencies range from two for I5020030 (RESERVOIR CONSTRUCTION) and I5020050 (SEA DEFENCE CONSTRUCTION) to 34,788 (34,775 in RCV1-v2) for I81402 (COMMERCIAL BANKING). In contrast to Topic and Region codes, Industry codes were not required to be assigned. Only a subset of documents (351,812 for RCV1-v1 and 351,761 for RCV1-v2) have them.

The Industry codes incorporate many fine-grained distinctions in subject matter. (For instance, there are five variations on the real estate industry.) They may therefore provide a test of the ability of text categorization systems to distinguish small differences in content.

As with Topics, the Industry code symbols encode both a hierarchy and a numeric sort order. The hierarchy was used for automated assignment of ancestor categories, though these automated assignments were imperfectly preserved in RCV1 (Section 3.5.2). In addition, some use of relationships between codes for companies (not present in the RCV1 CD-ROMs) and codes for Industries was used during automated assignment of Industries.

Several anomalies of the morphology of the Industry code symbols, and in the way the codes were used, make the relationships among codes hard to discern. We first discuss these anomalies, and then how to deal with them for experimental purposes.

Anomaly 1: The legacy editing interface used by coders required Industry code symbols to be either six or eight characters, regardless of hierarchy position. For instance, here is a subset of the codes in the form that editors apparently conceived of them (we indent the codes to indicate hierarchical structure):

```
I8                  FINANCIAL AND BUSINESS SERVICES
    I82             INSURANCE
        I82001      COMPOSITE INSURANCE
        I82002      LIFE INSURANCE
        I82003      NON-LIFE INSURANCE
            I8200316 MOTOR INSURANCE
            I8200318 REINSURANCE
```

However, the editing interface required that the codes be padded to six or eight characters with trailing digits. The trailing digits are usually (but not always) 0's. Thus the above codes are present in *industry_codes.txt* in this form:

```
I80000              FINANCIAL AND BUSINESS SERVICES
    I82000          INSURANCE
        I82001      COMPOSITE INSURANCE
        I82002      LIFE INSURANCE
        I82003      NON-LIFE INSURANCE
            I8200316 MOTOR INSURANCE
            I8200318 REINSURANCE
```

The 6- or 8-character padded versions of the codes are the ones found in the RCV1 documents. We refer to these as "padded" codes, and the raw versions (which more directly encode the hierarchy) as "unpadded" codes.

Anomaly 2: The hierarchical expansion software apparently required a code list containing codes in both unpadded and padded forms, and the intermediate forms as well. So the file *industry_codes.txt* actually contains:

```
I8                  FINANCIAL AND BUSINESS SERVICES
I80                 FINANCIAL AND BUSINESS SERVICES
I800                FINANCIAL AND BUSINESS SERVICES
I8000               FINANCIAL AND BUSINESS SERVICES
I80000              FINANCIAL AND BUSINESS SERVICES
    I82             INSURANCE
    I820            INSURANCE
    I8200           INSURANCE
    I82000          INSURANCE
        I82001      COMPOSITE INSURANCE
        I82002      LIFE INSURANCE
        I82003      NON-LIFE INSURANCE
            I8200316 MOTOR INSURANCE
            I8200318 REINSURANCE
```

Anomaly 3: There are nine 7-character codes (such as I815011) in *industry_codes.txt*. Codes with seven characters were purely a navigational aid to editors in searching the code set. These 7-character codes were not assigned to documents either by editors or during hierarchical expansion.

Anomaly 4: There are nine codes labeled TEMPORARY, eight with 6 characters and one with five characters. There are also two codes labeled DUMMY CODE (I9999 and I99999). These appear to be placeholders where new, meaningful codes (or navigational aids) might have been added but weren't. These codes were not assigned to documents either by editors or during hierarchical expansion.

Anomaly 5: The top level of codes, I0 through I9 in unpadded form (I00000 through I90000 in padded form), were apparently not allowed to be assigned to documents.

Anomaly 6: The code I50000 *was* assigned to documents. It is a 6-character padding of unpadded code I500 (GENERAL CONSTRUCTION AND DEMOLITION), not a padding of disallowed unpadded code I5 (CONSTRUCTION).

Anomaly 7: There are six cases (excluding TEMPORARY and DUMMY codes) where two or more distinct 6- or 8-character padded codes have the same name in *industry_codes.txt*. Each of these cases appears to have a different interpretation, as described next.

Anomaly 7a: The unpadded code I161 (ELECTRICITY PRODUCTION) has two padded forms, I16100 and I16101, listed in *industry_codes.txt*. The code I16100 is assigned to many documents, but I16101 to none. Other documentation suggests I16101 should not be considered an assignable code, and that the children of I16101 should instead be considered children of I16100.

Anomaly 7b: The padded codes I22400 and I22470 both have the name NON FERROUS METALS. Other documentation suggests the original name for I2247 (padded to I22470) was OTHER NON FERROUS METALS and that it is a child of I224 (padded to I22400). Both I22400 and I22470 are assigned to documents, so both should be viewed as assignable.

Anomaly 7c: The padded codes I45500 (HOUSEHOLD TEXTILES) and I64700 (HOUSEHOLD TEXTILES) are distinct codes with the same name. The I455 version is in the subhierarchy for I4 (PROCESSING INDUSTRIES), while I647 is in the subhierarchy for I6 (DISTRIBUTION, HOTELS AND CATERING). Both should be viewed as assignable, and both are in fact assigned to documents.

Anomaly 7d: The 6-character code I47521 (TRADE JOURNAL PUBLISHING) has a single child code, I4752105 (TRADE JOURNAL PUBLISHING). There are no occurrences of I47521 on the corpus, but several occurrences of I4752105. Other documentation also suggests that I4752105 is the unpadded version of the code, while I47521 was not available for use.

Anomaly 7e: The codes I64000 and I65000 have the name RETAIL DISTRIBUTION. At one point these apparently referred to "RETAIL - GENERAL" (I64000) and "RETAIL - SPECIALIST" (I65000). Later the two were merged, and it appears that for the RCV1 data they should be considered to be the same code. The code I64000 is assigned to documents, while I65000 is not, so the children of I65000 should instead be considered children of I64000, and I65000 ignored.

Anomaly 7f: Similarly to Anomaly 7a, the unpadded code I974 (TELEVISION AND RADIO) has two 6-character paddings: I97400 and I97411. The code I97400 is assigned to many documents, while I97411 is assigned to none. Other documentation also suggests I97411 was not available for use. I97411 should be considered unavailable, and its children should be considered children of I97400.

Anomaly 8: The padded code I16300 has the name "ALTERNATIVE ENERGY" which is slightly different than the name ("ALTERNATIVE ENERGY PRODUCTION") for the apparent unpadded version of it (I163). Other documentation suggests there is not meant to be a distinction between these, so we rename I16300 to "ALTERNATIVE ENERGY PRODUCTION".

Given these anomalies, we believe the set of Industry codes that were available to be assigned to documents are those from *industry_codes.txt* that satisfy these criteria:

- Have six or eight characters (i.e., five or seven digits)
- Are not named DUMMY or TEMPORARY
- Are not of the form Ix0000, except for I50000
- Are not any of I16101, I47521, I65000, or I97411.

There are 354 such Industry codes, of which 350 appear in the corpus (both RCV1-v1 and RCV1-v2). The four available codes that do not appear in any document (I32753, I3302018, I841 padded to I84100, and I84802) are leaf nodes of the hierarchy. They have narrow enough meanings that there plausibly was no RCV1 document to which they were applicable. We provide a list of these 354 Industry codes as Online Appendix 4.

Reproducing the hierarchical structure in which the codes were embedded is more difficult. In producing our best guess at the hierarchy, we made use both of documentation (of uncertain vintage) from Reuters and of the *UK Standard Industrial Classification of Economic Activities (UK SIC(92))* (Great Britain Office for National Statistics, 1997, 2002), since it is known that some version of the UK SIC was consulted by Reuters personnel during design of the Industries codes. One of our informants also suggested that some codes from a set defined by the International Press Telecommunications Council (*http://www.iptc.org/*) may have been used as well, but we have not been able to determine which codes these were.

We also had to choose what kinds of codes to include in the hierarchy. We decided to omit TEMPORARY, DUMMY, and 7-character codes, as well as other codes that weren't available to editors. The only exception to requiring that codes have been assignable was that we included the unassignable second level codes I0 through I9.

Online Appendix 5 contains our hierarchy. It has 365 nodes: one root, the 10 second level codes I0 through I9, and the 354 assignable codes. As part of the hierarchy file, we include the name of each node. We rename I22470 to OTHER NON FERROUS METALS, I45500 to HOUSEHOLD TEXTILES PROCESSING, and I64700 to HOUSEHOLD TEXTILES DISTRIBUTION, so that all valid codes have a unique name.

3.2.3 REGION CODES

The file *region_codes.txt* on the RCV1 CD-ROMs contains 366 geographic codes, of which 296 occur at least once in the corpus. The Reuters documentation we could obtain suggests that all 366 of these codes were available to Reuters editors, and so are appropriate to use in experiments. We provide a list of these 366 valid Region codes as Online Appendix 6. Nonzero class frequencies span the range from one (for 10 codes in RCV1-v1 and eight codes in RCV1-v2) to 266,239 (265,625 in RCV-v2) for USA.

In addition, three codes with a total of four occurrences are present in the RCV1 articles but not in the file *region_codes.txt*, bringing the total number of Region codes actually present in RCV1-v1 articles to 299. These codes are CZ - CANAL ZONE (one occurrence), CZECH - CZECHOSLOVAKIA (two occurrences), and GDR - EAST GERMANY (one occurrence). These codes appear to be errors, so in producing RCV1-v2 relevance judgment files we replaced them by what appear to be the corresponding correct codes from *region_codes.txt*: PANA (PANAMA), CZREP (CZECH REPUBLIC), and GFR (GERMANY).

While no formal category hierarchy is provided with the RCV1 data, some Reuters personnel did view the Region codes as falling into three informal groups: Countries, Regional Groupings, and Economic Groupings. Other personnel viewed the latter two groups as not being clearly distinct. We did not find documentation defining the groupings, and so do not include a hierarchy or grouping of Region categories in our online appendices.

Hierarchies or networks of Region categories could be defined based on geographic, economic, political, or other criteria. Indeed, one Reuters informant has indicated that there was automatic assignment of some country codes based on company codes (not present in RCV1), and automated assignment of some regional or economic grouping codes (such as GSEVEN) based on country codes of member countries. We have not investigated this issue.

Whether assigning RCV1 Region codes is a good test of text categorization capability as opposed to named entity recognition capability (Grishman and Sundheim, 1995), is debatable. It is clear, however, that assigning Region codes is not *solely* a named entity task. There are many stories that mention the United States, for instance, that are not assigned to the USA code, and there are Region codes which are not named entities, such as WORLD and DEVGCO (DEVELOPING COUNTRIES).

3.2.4 RBB FILES

Just as the final version of this paper was being submitted, Reuters gave permission to publicly release some of the documentation we used in the above analysis. We therefore include, as Online

Appendices 16, 17, and 18, the RBB lists of Topics, Industries, and Region codes. RBB refers to the Reuters Business Briefing archival database offering (Section 2.2).

The RBB files present code sets that are related to the codes appearing in the RCV1 documents, and to the codes specified in the CD-ROM files *industry_codes.txt*, *topic_codes.txt*, and *region_codes.txt*. None of the corresponding sets of codes is exactly identical to any of the others, however, and the time period during which any particular set of codes was in use is not clear. We have slightly edited the Topics and Industries RBB files to fix some inconsistencies in code names, and to add descriptions for two codes missing from the RBB data, to make the resulting files more consistent with the RCV1 data. (Note that the Industries files also contains the RBB descriptions for the intermediate non-code nodes I0 through I9.) We have not edited the Regions RBB file, since it has significant differences from the RCV1 data.

Despite these differences, the RBB files should prove a useful supplement to the CD-ROM files, particularly since the RBB files give more extensive descriptions of some categories.

3.3 Coding Policy

Coding policies specify certain requirements for how coding should be done, beyond an editors' judgment of which codes capture the content of a particular text. As mentioned in Section 2.3, at least two coding policies, which we call the Hierarchy Policy and the Minimum Code Policy, were used by Reuters during the period the data in RCV1 was produced. We discuss here their implications for the use of RCV1 as a test categorization test collection.

3.3.1 IMPLICATIONS FOR CORPUS PROPERTIES

The Hierarchy Policy required that when a Topic or Industry code was assigned to an article, all the codes which were ancestors of it in the Topic code hierarchy should be assigned as well. (The application of this policy in producing the data that became RCV1 was imperfect, as discussed in Section 3.5.) Adding ancestor codes creates some very high frequency codes (CCAT is assigned to 46% of the corpus), as well as strong, partially deterministic, dependencies between hierarchically related codes.

The Minimum Code Policy required that articles get at least one Region code and one Topic code. This policy probably did not greatly affect the codes assigned, since the code sets themselves were designed to cover the likely content of the newswire. However, unlike the Hierarchy Policy, the Minimum Code Policy did require human coders to change their behavior: in cases where they might otherwise decide that no code applies, they were forced to choose some assignment. From a statistical standpoint, the Minimum Coding Policy introduces a weak dependence among all codes in a set.

3.3.2 IMPLICATIONS FOR ALGORITHM DESIGN

If one knows that the correct categorization of a document obeys coding policies, it is natural to attempt to modify a text categorization algorithm so its output obeys those policies. Whether doing this will actually improve the effectiveness of a given system is, however, less clear.

The obvious approach to implementing the Hierarchy Policy is to run a categorizer as usual, and then add the ancestors of all assigned categories if not already present. This runs the risk, however, of adding a high level category which was rejected by a well-trained classifier, on the basis of a low level category assigned by a less well-trained classifier.

How easy (and desirable) it is to implement the Minimum Code Policy varies with the text categorization method. For instance, a common strategy in text categorization is to create a separate binary classifier for each category. This approach is likely to assign no categories to some documents, and so would sometimes violate the Minimum Code Policy.

3.3.3 IMPLICATIONS FOR EVALUATION

When testing algorithms on a corpus produced using a particular coding policy, should one disallow outputs that violate that policy? This is often done when testing algorithms for multiclass (1-of-*k*) categorization: only algorithms that assign exactly one category for each test document are allowed. In an operational setting the data model, software interfaces, or other constraints might require strict adherence to coding policy.

On the other hand, if we view the system's output as something which will be reviewed and corrected by a human editor, a more relaxed approach may be appropriate. Rather than forbidding outputs that violate coding policy, one can instead measure the effort that would be required to correct these policy violations, along with correcting any other errorful assignments.

One way to measure the effort that would be required to correct errors is simply to compute the usual microaveraged or macroaveraged effectiveness measures from binary contingency tables for the categories. This is the approach we adopt in reporting benchmark results in Section 8.

3.4 Was Each Document Manually Coded?

There are two somewhat conflicting worries that one might have about the RCV1 corpus. One is that a portion of the corpus might have been missed during coding, as was the case with Reuters-21578 (Lewis, 1997). Conversely, one might worry that the use of autocoding (Section 2.4.1) means that achieving good effectiveness on RCV1 is an exercise in rediscovering the (possibly simple and uninteresting) rules used by the automated categorizer.

We believe neither worry is justified. Reuters procedures assured that each story was coded automatically, and then had those codes checked by at least one, and sometimes two human editors. Further, a simple check of the raw RCV1-v1 corpus shows no documents that are totally lacking in codes, though some are missing one or another type of obligatory code (Section 3.5.2).

On the second question, we note that for each document a human editor always made the final decision on the codes to be assigned. Indeed, Table 2 shows that on average 79% of stories had at least one autocoding decision overruled. This argues that, despite the use of automated coding, RCV1 can be considered a manually categorized test collection.

We believe that the only code whose automated assignment was not checked in this process was GMIL, for millennium-related stories. This was automatically assigned, possibly without manual checking, sometime after the period the documents were originally archived. There may have been a very small number of other such codes, but we have not found evidence for this.

3.5 Coding Errors

The Reuters-supplied interindexer consistency data presented in Section 2.5 suggests low levels of disagreements between indexers, and low levels of simple errors. However, there are also ways to study interindexer consistency directly on the collection. We investigate two such methods below, as well as discussing a more fundamental difficulty with the concept of coding errors.

3.5.1 DETECTING CODING ERRORS USING DUPLICATE DOCUMENTS

One way to detect coding errors is to take advantage of documents which are duplicates of each other and so presumedly should have the same codes assigned. Using a substring-based measure, Khmelev and Teahan (2003) found a total of 27,754 identical or highly similar documents in RCV1-v1. They observed that 52.3% of such documents had the same set of Topics, 80.1% had the same set of Industries, and 86.8% had the same set of Regions. They suggest that the percentage of matching Topics is worrisomely low.

We have done a similar study which suggests less cause for concern. We identified the 14,347 documents in RCV1-v2 whose `<headline>` and `<text>` elements are identical to those of another document (ignoring variations in whitespace). We then computed classification effectiveness for each category based on treating all copies of a document as supplying fractional relevance judgments for that document. For instance, if there were three copies of a document, each would be evaluated against the other two, with each of the other two contributing a relevance judgment with weight 0.5. We computed the $F_{1.0}$ measure for each category that appeared at least once in the 14,347 duplicated documents, and took the macroaverage of these values. (See Section 5.3 for this measure.)

The resulting macroaveraged $F_{1.0}$ values were 0.69 for Topics (with 102 out of 103 possible categories being observed in the duplicates), 0.57 for Industries (262 of 354 categories observed), and 0.74 for Regions (206 of 366 categories observed). These values are all higher than the best macroaveraged $F_{1.0}$ values seen in our experiments (Section 8) on categories with at least one positive test example (0.61 for Topics, 0.27 for Industries, and 0.47 for Regions). So even if duplicate documents gave an accurate measure of the limitations on interindexer agreement, we are not reaching this limit.

Further, we suspect that the duplicated documents have a higher proportion of errorful assignments than do nonduplicated documents. A surprisingly high proportion of duplicated documents have category assignments that are a superset of the assignments of one of their duplicates. This is most clear in cases where there were exactly two documents with the same `<headline>` and `<text>`. There were 6,271 such pairs. Of these, 4,182 had some difference in their Topics assignments, and in 1,840 of these cases one set of assignments is a superset of the other. For Regions, 967 pairs have some difference, and 801 of these have a superset relationship. And for Industries, 1,500 pairs have some difference, and 1,328 of these have a superset relationship.

The proportion of superset relationships seems higher than would be expected for independent indexings of the documents, though a precise statistical model is hard to pose. One hypothesis is that the duplicate documents are present precisely because one editor was correcting an assignment produced by a previous editor (or by the automated coder). While there was an attempt to remove duplicated stories before archiving, this was not done perfectly, so both the corrected and uncorrected versions may have been archived. If this was the case, then the disagreement rate seen among duplicated stories will be much higher than for independent indexings of stories in general.

3.5.2 DETECTING CODING ERRORS BY VIOLATIONS OF CODING POLICIES

Another approach to identifying coding errors comes from knowledge of Reuters coding policies. There are 2,377 documents (0.29% of RCV1-v1) which violate the Minimum Code Policy by having either no Topic codes (2,364 documents) or no Region codes (13 documents). There are 14,786 documents (1.8% of RCV1-v1) which violate the Hierarchy Policy on Topic codes, i.e., an ancestor of some assigned Topic code is missing. Of the 103 Topic codes that were used for the RCV1 data,

21 have at least one child in the Topic hierarchy. Each of these 21 codes is missing from at least one document to which the Hierarchy Policy says it should have been assigned. A total of 25,402 occurrences of these 21 codes are missing in RCV1-v1.

With respect to Industry codes, application of the Hierarchy Policy was also imperfect:

- The immediate parent of an 8-character code was automatically added to the document in most cases, but these cases were missed:
  - Some 8-character codes with one or more appearances in the corpus had (assuming we have inferred the hierarchy correctly) an immediate parent code that is not the 6-character truncation of the 8-character code. These 8-character codes are (with parent shown in parentheses): I1610107 (I16100), I1610109 (I16100), I4752105 (I47520), I9741102 (I97400), I9741105 (I97400), I9741109 (I97400), I9741110 (I97400), and I9741112 (I97400). Three parents account for these cases (I16100, I47520, I97400) and they are assigned in only 7.1% to 46.6% of documents containing the child code, depending on the particular child category. This contrasts with essentially 100% assignment of parent codes which were 6-character truncations of 8-character codes.
  - A single document containing two children of I01001 is missing I01001 itself. This appears to be a simple error. By contrast, all 12,782 other occurrences of children of I01001 are in documents that also contain I01001.
- No grandparents or higher level ancestors of 8-character codes appear to have been automatically added, nor any ancestors of 6-character codes. The few cases where both a code and one of these other ancestors are assigned to a document appear to result from a manual editorial decision to assign both.

These violations result from some combination of human error, glitches in the hierarchical expansion software, and/or omissions of some codes from the archival data when producing RCV1. Some errors appear to have resulted from manual additions of codes after hierarchical expansion had already been run.

In Section 4 we propose an approach to correcting these errors, where possible, before using the corpus for experimental purposes.

3.5.3 ERRORFUL CODES AND PLAUSIBLE CODES

While Reuters did compute measures of consistency between indexers working independently, as well as traditional effectiveness measures for categorization software, these were not necessarily the most important measures for them. When evaluating vendors for eventual selection of a new automated categorization system (Section 2.6), Reuters used a measure based on the rate at which a human expert actively disagreed with the coding choice made for the document. The idea is that there are some codes that plausibly might be assigned or might not be assigned.

In our experience, this is not an unusual stance for users of text classification to take. It suggests, unfortunately, that we should really consider the codes present in many corpora (including RCV1) to be those found necessary by the indexer, plus some (but not all) of those found plausible but not necessary. How this ambiguity should best be handled in text classification evaluations is an open question.

4. RCV1-v2: A New Text Categorization Test Collection

Since all evidence suggests that violations of the Hierarchy Policy and Minimum Coding Policy are simple errors, removing these violations where possible will produce more accurate results in any classification experiments made using RCV1. In this section we describe the procedures necessary to remove these errors. We call the resulting corrected text categorization test collection RCV1-v2 (for version 2), while referring to the uncorrected original version as RCV1-v1.

The following corrections convert RCV1-v1 to RCV1-v2:

1. Remove from the corpus the 13 documents that violate the Minimum Code Policy due to missing all Region codes, and the 2,364 documents that violate the policy due to missing all Topics. This leaves a total of 804,414 documents. Online Appendix 7 provides a list of the IDs of the 804,414 documents in RCV1-v2.

2. For each Topic code present in a document, add all missing ancestors of the code. This adds 25,402 Topic code assignments.

3. Replace the four errorful occurrences of Region codes, as described in Section 3.2.3.

We applied these corrections to the corpus before producing the results reported in Section 8.

We decided not to try to correct violations of the Hierarchy Policy for Industry codes. One reason is that we are unsure of the exact Industry hierarchy at the time the RCV1 data was produced. In addition, it is not clear that the coding resulting from an expansion would actually be superior for research purposes. There are three classes of codes to consider:

- Leaf codes (i.e., all 8-character codes and some 6-character codes). Their assignments would not be affected under any hierarchical expansion scheme.

- Non-leaf 6-character codes with 8-character children. All but 4 of these 6-character codes are assigned in 100% of the cases one or more of their children are present. One (I01001) is missing from only one of the 12,783 documents that contain one or more of its children. The three remaining codes (I16100, I47520, and I97400) are assigned to a fraction of the documents to which their children are assigned. These are exactly the three codes where the unpadded code for the parent is not a truncation of the unpadded code for one or more of its child codes. We do not know if the assignments of these codes which are present in the corpus represent a partially successful automated assignment or, conversely, an intended omission of automated assignment in combination with manual decisions to assign the codes in certain cases. If we modified the corpus by assigning these codes when their children are present, it is unclear whether we would be respecting the intended semantics, or washing it out.

- Non-leaf 6-character codes that only have 6-character children. There seems to have been little or no assignment of these codes based on expansion of children. Occurrences of these codes appear to correspond to a manual judgment that this code is appropriate. Automated expansion would swamp these manual judgments with large numbers of expansion-based assignments (up to 100-fold more), producing an arguably less interesting classification task.

We therefore decided not to attempt hierarchical expansion of Industry codes. This means that some non-leaf Industry categories (6-character codes with 8-character children) have "Other + expansion" semantics, some (I16100, I47520, and I97400) have unclear semantics, and the rest apparently have pure "Other" semantics (Section 3.2.1).

4.1 Availability of RCV1-v2 Data

Online Appendix 7 gives the complete list of RCV1-v2 document IDs. The complete sets of corrected RCV1-v2 category assignments are provided in Online Appendices 8, 9, and 10. In addition, two versions of the complete set of RCV1-v2 documents in vector form are provided as Online Appendices (see Section 7).

5. Benchmarking the Collection: Methods

An important part of the value of a machine learning data set is the availability of published benchmark results. Among other things, good benchmark results serve to ensure that apparently superior new methods are not being compared to artificially low baselines. We therefore ran three of the most popular supervised learning approaches on the RCV1-v2 data, both to provide such a benchmark, and as a check that our corrections to the data did not introduce any new anomalies.

5.1 Training/Test Split

We split the RCV1-v2 documents chronologically into a training set (articles published from August 20, 1996 to August 31, 1996; document IDs 2286 to 26150) and test set (September 1, 1996 to August 19, 1997; document IDs 26151 to 810596). The result is a split of the 804,414 RCV1-v2 documents into 23,149 training documents and 781,265 test documents. We call this the *LYRL2004 split*. (Notice that ID order does not always correspond to chronological order in either RCV1-v1 or RCV1-v2, so chronological splits in general should be based on the date tag in the `<newsitem>` element, not on IDs.)

The chronological boundary we used is the same used in the TREC-10/2001 filtering track (Robertson and Soboroff, 2002). However the TREC-10/2001 filtering track used the raw RCV1 data (806,791 uncorrected RCV1-v1 documents split into 23,307 training documents and 783,484 test documents) and raw category labels, so the TREC results are not comparable with ours.

A chronological split, rather than a random one, is realistic since the majority of operational text categorization tasks require training on currently available material, and then applying the system to material that is received later. A chronological split also reduces the tendency of duplicate and near-duplicate documents to inflate measured effectiveness. The chronological breakpoint we chose has the advantage of giving almost all Topic categories two or more training examples, while still retaining most of a complete year as test data.

5.2 Categories

We provide benchmark data on all categories that evidence indicates were available to Reuters indexers, even those with few or no positive examples. There are 103 Topic categories, 101 with one or more positive training examples on our training set. All 103 (including all the 101, obviously) have one or more positive test examples on our test set. There are 354 Industry categories, 313 with positive training examples, and 350 (including all of the 313) with positive test examples. And there are 366 Region categories, 228 with positive training examples, and 296 (including all of the 228) with positive test examples. (All counts are the same for RCV1-v1, if the four invalid assignments of three invalid Region categories in RCV1-v1 are ignored.)

5.3 Effectiveness Measures

We measure the effectiveness of a text classifier on a single category with the F_β measure (van Rijsbergen, 1972, 1979; Lewis, 1995):

$$F_\beta = \frac{(\beta^2+1)A}{(\beta^2+1)A+B+\beta^2 C},$$

where A is the number of documents a system correctly assigns to the category (true positives), B is the number of documents a system incorrectly assigns to the category (false positives), and C is the number of documents that belong to the category but which the system does not assign to the category (false negatives). We report values for $\beta = 1.0$, which corresponds to the harmonic mean of recall and precision:

$$F_{1.0} = \frac{2A}{2A+B+C} = \frac{2RP}{R+P},$$

where R is recall, i.e., A/(A+C), and P is precision, i.e., A/(A+B).

The F-measure as presented above is undefined when $A = B = C = 0$. The experiments reported here treat $F_{1.0}$ as equal to 0.0 in this case, though a strong argument could be made for a value of 1.0 instead, or possibly other values (Lewis, 1995).

To measure effectiveness across a set of categories we use both the *macroaverage* (unweighted mean of effectiveness across all categories) and the *microaverage* (effectiveness computed from the sum of per-category contingency tables) (Lewis, 1991; Tague, 1981).

6. Benchmarking the Collection: Training Algorithms

We benchmarked three supervised learning approaches that have been widely studied in text categorization experiments: support vector machines (SVMs) (Joachims, 1998), weighted *k*-Nearest Neighbor (*k*-NN) (Yang and Liu, 1999), and Rocchio-style algorithms (Ittner, Lewis, and Ahn, 1995; Yang, Ault, Pierce, and Lattimer, 2000; Ault and Yang, 2002). We describe these core supervised learning algorithms below, as well as the supervised threshold setting and feature selection procedures used with some of them.

6.1 SVM

SVM algorithms find a linear decision surface (hyperplane) with maximum margin between it and the positive and the negative training examples for a class (Joachims, 1998). SVMs using non-linear kernel functions are also possible, but have not shown a significant advantage in past text categorization studies, and are not investigated here.

SVMs have outperformed competing approaches in a number of recent text categorization studies, but there has been some suggestion that they choose a poor decision threshold when the numbers of positive and negative examples are very different, as they are for low frequency categories in random or systematic samples of documents (Zhang and Oles, 2001). We therefore used in our baselines two SVM variants that adjust for category frequency:

- SVM.1: A single SVM classifier was trained for each category. SVM training used the *SVM_Light* (Joachims, 1998, 1999, 2002) package, version 3.50. All parameters were left

at default values. This meant, in particular, that we used a linear kernel (by leaving *-t* unspecified), equal weighting of all examples whether positive or negative (by leaving *-j* unspecified), and set the tradeoff *C* between training error and margin to the reciprocal of the average Euclidean norm of training examples (by leaving *-c* unspecified). Since we were using cosine-normalized training examples, leaving *-c* unspecified meant *C* was set approximately to 1.0. *SVM_Light* was used to produce scoring models, but the *SVM_Light* thresholds were replaced with ones chosen by the SCutFBR.1 algorithm (Section 6.4).

- SVM.2: In this approach (Lewis, 2002), *SVM_Light*, version 3.50, was run multiple times for each category, once for each of these settings of its *-j* parameter: 0.1, 0.2, 0.4, 0.6, 0.8, 0.9, 1.0, 1.25, 1.5, 2.0, 3.0, 4.0, 6.0, 8.0, 10.0, and 15.0. The *-j* parameter controls the relative weighting of positive to negative examples in choosing an SVM classifier, and thus provides a way to compensate for unbalanced classes. Leave-one-out cross-validation (LOO) (turned on by *SVM_Light*'s *-x 1* parameter) was used to compute a training set contingency table corresponding to each setting of *-j*. All other *SVM_Light* parameters were left at their default values.

  For each category, the $F_{1.0}$ value for each setting of *-j* was computed from its LOO contingency table. The *-j* setting giving the highest LOO-estimated $F_{1.0}$ for a category was selected for that category. (In case of ties, the value of *-j* closest to 1.0 was used, with *-j* values less than 1.0 replaced by their reciprocals when computing closeness. If a value had been tied only with its reciprocal for best and closest, we planned to choose the value greater than 1.0, but this situation did not arise.)

  As expected, the algorithm tended to choose values of *-j* that gave additional weight to positive examples. The value 0.8 was chosen once for *-j*, 1.0 was chosen five times, 1.25 was chosen eight times, 1.5 was chosen 11 times, 2.0 was chosen 27 times, 3.0 was chosen 27 times, 4.0 was chosen 17 times, 6.0 was chosen four times, and 8.0 was chosen once.

  A final classifier was trained on all training data for the category using the chosen setting of *-j*. The threshold chosen by *SVM_Light* based on the selected setting of *-j* was used as is for that category (SCutFBR.1 was not used). Due to its expense, SVM.2 was tried only for Topic categories.

  SVM.2 was the top-ranked approach in the batch filtering and routing tasks in the TREC-10 evaluation (Robertson and Soboroff, 2002).

6.1.1 PARAMETER TUNING

The SVM.1 approach had one free parameter, the value of *fbr* in the SCutFBR.1 threshold setting algorithm (Section 6.4). We compared the values 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8 for *fbr* using five-fold cross-validation on the training set and picked the best value for each category set (Topics, Industries, Regions) and effectiveness measure (microaveraged $F_{1.0}$ and macroaveraged $F_{1.0}$). (Note this five-fold cross-validation loop called the SCutFBR.1 procedure, which in turn used its own five-fold cross-validation internally.) The final classifiers for each category in a category set were then trained using all training data and the chosen *fbr* value for their category set and effectiveness measure.

The SVM.2 algorithm itself incorporated tuning of its only free parameter (*-j*) so no outside tuning was needed. Given the robustness of SVMs to high dimensional feature sets, no feature selection was used with either of the SVM algorithms.

6.2 *k*-NN

Weighted *k*-NN (*k-nearest neighbor*) classifiers have been consistently strong performers in text categorization evaluations (Yang, 1999; Yang and Liu, 1999). The variant we used here chooses, as neighbors of a test document, the *k* training documents that have the highest dot product with the test document. Then, for each category, the dot products of the neighbors belonging to that category are summed to produce the score of the category for the document. That is, the score of category c_j with respect to test document $\vec{x}$ (a vector of term weights) is

$$s(c_j, \vec{x}) = \sum_{\vec{d} \in R_k(\vec{x})} \cos(\vec{x}, \vec{d})\, I(\vec{d}, c_j),$$

where $\vec{d}$ is a training document; $R_k(\vec{x})$ is the set consisting of the *k* training documents nearest to $\vec{x}$; and $I(\vec{d}, c_j)$ is indicator function whose value is 1.0 if $\vec{d}$ is a member of category c_j, and 0.0 otherwise. Since $\vec{x}$ and $\vec{d}$ were normalized to have Euclidean norm of 1.0, their dot product is equal to the cosine of the angle between them, so we write the dot product as $\cos(\vec{x}, \vec{d})$. The resulting score is then compared to the category threshold to determine whether or not to assign the category to the test document. Thresholds were chosen by SCutFBR.1 (Section 6.4).

The *k*-NN method is more sensitive to nonrelevant features than SVMs are, so the vectors used with it first had feature selection applied (Section 6.5).

6.2.1 PARAMETER TUNING

The *k*-NN algorithm had three free parameters, *fbr*, *k* (neighborhood size), and the feature set size. Five-fold cross-validation on the training set was used to select values for these parameters for each category set and effectiveness measure. The following values were tried:

- *fbr* : 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
- *k* : 1, 3, 5, 10, 20, 40, 60, 80, 100, 130, 160, 200, 400, 600, 800
- Feature set size : 50, 100, 200, 400, 600, 800, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 12000, 14000, 16000, 20000, 25000, 30000, 47152

However, not all combinations of values were tried. Instead parameter values were first initialized to defaults: 0.3 for *fbr*, 50 for *k*, and the number of terms with nonzero values in the training set (47,152 terms) for feature set size. Then one parameter value at a time was optimized while holding the others fixed: first *k* (holding default *fbr* and feature set size fixed), then feature set size (holding the chosen *k* and default *fbr* fixed), and finally *fbr* (holding the chosen *k* and chosen feature set size fixed). Table 3 shows the *k*-NN parameter values chosen by this cross-validation process.

6.3 Rocchio-Style Prototype Classifier

The Rocchio method was developed for query expansion using relevance feedback in text retrieval (Rocchio, 1971; Salton and Buckley, 1990). Applied to text classification, it computes a *prototype*

Category Set	Effectiveness Measure	Parameters: Features selected	Parameters: Neighborhood size (k)	Parameters: *fbr*
Topics	Micro F1	8000	100	0.5
	Macro F1	8000	100	0.1
Industries	Micro F1	10000	10	0.4
	Macro F1	10000	10	0.1
Regions	Micro F1	10000	10	0.5
	Macro F1	10000	100	0.1

Table 3: Parameters chosen by cross-validation for our weighted k-NN algorithm, for each of the six combinations of category set and averaged effectiveness measure.

vector for each category as a weighted average of positive and negative training examples (Ittner, Lewis, and Ahn, 1995).

Our Rocchio prototype for category c_j was

$$\vec{p}_j(\gamma) = \frac{1}{|\mathcal{D}(c_j)|} \sum_{\vec{d}_i \in \mathcal{D}(c_j)} \vec{d}_i - \gamma \frac{1}{|\mathcal{D}_n(\overline{c_j})|} \sum_{\vec{d}_i \in \mathcal{D}(\overline{c_j})} \vec{d}_i,$$

where $\vec{d}_i$ is a training document; $\mathcal{D}(c_j)$ and $\mathcal{D}(\overline{c_j})$ are, respectively, the set of positive and negative training examples for category c_j; and γ is the weight of the negative centroid.

Many enhancements have been proposed to the original Rocchio algorithm (Schapire, Singer and Singhal, 1998; Ault and Yang, 2002). We used only the following ones:

1. As with k-NN, we do an initial feature selection for each category set and averaged effectiveness measure, using the χ^2_max criterion (Section 6.5).

2. We then do a further feature selection on a per-category basis by zeroing out all but the p_{max} largest nonzero coefficients in the Rocchio vector. This keeps all positive coefficients before any negative ones. It is uncommon, but possible, for negative coefficients to remain in the Rocchio vector after this procedure.

3. The Rocchio algorithm produces a scoring model only. We choose a threshold for this model using the SCutFBR.1 algorithm (Section 6.4).

6.3.1 PARAMETER TUNING

Our modified Rocchio algorithm had four free parameters, *fbr*, γ, p_{max}, and the feature set size. However, preliminary experiments on the training data for Topics showed that the choice of p_{max} had little impact on effectiveness. A value of 3000 for p_{max} was found to be best in the Topics run, and so was used in all runs for all category sets and effectiveness measures. Five-fold cross-validation on the training data was used to select values for the other three parameters for each category set and effectiveness measure. The following values were tried:

Category Set	Effectiveness Measure	Parameters: Initial features selected	Parameters: Features retained in model (p_{max})	Parameters: Nonrelevant centroid weight (γ)	Parameters: *fbr*
Topics	Micro F1	5000	3000	1	0.3
	Macro F1	5000	3000	1	0.2
Industries	Micro F1	10000	3000	2	0.4
	Macro F1	10000	3000	6	0.1
Regions	Micro F1	10000	3000	2	0.5
	Macro F1	10000	3000	2	0.5

Table 4: Parameters chosen by cross-validation for our modified Rocchio algorithm, for each of the six combinations of category set and averaged effectiveness measure. The value of p_{max} was chosen in an initial run on Topics, and then used for all combinations.

- *fbr* : 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8
- γ : 50, 20, 15, 10, 8, 6, 4, 3, 2, 1, 0, -1, -2
- Feature set size : 50, 100, 200, 400, 600, 800, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 12000, 14000, 16000, 20000, 25000, 30000, 47152

As with *k*-NN, we first initialized the three parameters to default values (0.3 for *fbr*, 1 for γ, and 47,152 for feature set size), and then optimized one parameter at a time: first γ, then feature set size, and finally *fbr*. The selected parameter values are shown in Table 4.

6.4 Supervised Threshold Setting

Each of our algorithms produces, for each category, a model that assigns scores to documents. To use these models for classification, we use the SCut strategy (Yang, 2001), i.e., simply associating a threshold value with each category, and assigning the category to a document when the score for that category exceeds the threshold. Other category assignment strategies (Yang, 2001) besides SCut were evaluated on the training data, but Scut was consistently superior so only it was used to produce classifiers evaluated on the test data.

The SVM.2 algorithm incorporates its own method for choosing a threshold to be used in the SCut approach. The other core training algorithms (SVM.1, *k*-NN, and Rocchio) were used to train scoring models. Thresholds for those scoring models were found by wrapping the core training algorithm within Yang's *SCutFBR.1* algorithm (Yang, 2001). The *.1* refers to the rank of the validation document whose score becomes the threshold if the cross-validated threshold gives poor estimated effectiveness (see below). The core SCutFBR algorithm can be used with other fallback ranks as well (Yang, 2001).

SCutFBR.1 uses five-fold cross-validation with random assignment of documents to folds (i.e., no balancing of positive and negative examples). In each fold, a scoring model was trained on four-fifths of the data and its threshold was tuned on the remaining one-fifth. If the tuned threshold gave

an $F_{1.0}$ value less than a specified minimum value *fbr*, then that threshold was replaced by the score of the top-ranked validation document. The final threshold for the category is the average of the thresholds across the five folds.

6.5 Supervised Feature Selection

Our text representation approach produced a set of 47,236 features (stemmed words), of which 47,152 occurred in one or more training set documents and so potentially could be included in classifiers (Section 7). Two of the algorithms studied, *k*-NN and Rocchio, are known to be significantly hampered by irrelevant features. Feature selection based on labeled data was used with these two algorithms. A separate feature set was chosen for each combination of algorithm (*k*-NN or Rocchio), category set (Topics, Industries, or Regions), and effectiveness measure (microaveraged $F_{1.0}$ or macroaveraged $F_{1.0}$).

The feature set for a combination was chosen by first ranking the 47,152 features by their χ^2_max score (Yang and Pedersen, 1997; Rogati and Yang, 2002) with respect to the category set. To compute this score, we first separately compute the χ^2 statistic (Altman, 1991, Section 10.7) for the feature with respect to each category in the category set:

$$\chi^2 = \frac{n(ad-bc)}{(a+b)(a+c)(b+d)(c+d)},$$

where n is the total number of examples used in calculating the statistic, a is the number of examples with both the feature and the category, b is the number of examples with the feature and not the category, c is the number of examples with the category and not the feature, and d is the number of examples with neither the feature nor the category.

The feature's χ^2_max score is the maximum value of the χ^2 statistic across all categories in the category set. Note that the use of χ^2_max feature selection means that training data from all categories in a category set influences the set of features used with each individual category in the set.

The χ^2_max score produces a ranking of all features from best to worst. To choose a feature set, we then had to choose a size for the feature set to know how far down that ranking to go. This was done by evaluating each of 23 corresponding feature sets using five-fold cross-validation on the training data, and picking the best (Sections 6.2.1 and 6.3.1).

7. Benchmarking the Collection: Text Representation

The same sets of training and test document feature vectors were provided to each algorithm. The feature vector for a document was produced from the concatenation of text in the `<headline>` and `<text>` XML elements. It is important to note that the `<title>` element of RCV1 documents contains a "country code" string that was semi-automatically inserted, possibly based on the Region codes. The `<title>` element should therefore not be used in experiments predicting category membership. (The `<headline>` element was added by Reuters during the production of the RCV1 corpus. It contains the same text as the `<title>` element, but strips out the country code.)

Text was reduced to lower case characters, after which we applied tokenization, punctuation removal and stemming, stop word removal, term weighting, feature selection, and length normalization, as described below.

We defined tokens to be maximal sequences of nonblank characters. Tokens consisting purely of digits were discarded, as were words found on the stop word list from the SMART system (Salton, 1971). The list is found at *ftp://ftp.cs.cornell.edu/pub/smart/english.stop*, and we also include it as Online Appendix 11.

The remaining tokens were stemmed with our own implementation of the Porter stemmer (Porter, 1980). Our implementation did considerable punctuation removal as well as stemming. As the author of the Porter stemmer has discussed (Porter, 2003) it is very rare for two implementations of the Porter stemmer to behave identically. To enable reproducing our results, we therefore provide our stemmed tokens in Online Appendix 12, as discussed at the end of this section.

Document vectors based on the stemmed output were then created, with each coordinate of the vectors corresponding to a unique term (stemmed word). Only terms which occurred in one or more of the 23,307 RCV1-v1 documents falling before our chronological breakpoint were used in producing vectors. Terms which had their only occurrences in post-breakpoint documents (i.e., our test documents) did not affect vector formation in any way. In particular, they were not taken into account during cosine normalization (below).

We had intended to use only our training set (the 23,149 pre-breakpoint RCV1-v2 documents) for feature definition, not all 23,307 pre-breakpoint RCV1-v1 documents. The effect of accidentally including stems from these 158 additional documents in document vectors is that a few features whose value is 0.0 on all of our training documents are nonetheless allowed to have nonzero values in test documents. (Except for words in these 158 documents, words that show up in the testset, but not the training set, do not participate in any vectors.) These additional features will occasionally have nonzero values on test documents, thus slightly impacting (through cosine normalization) the value of the other features. The impact on overall results should be negligible. No label information from these 158 mistaken documents was used.

The number of unique terms present in the 23,307 pre-breakpoint RCV1-v1 documents was 47,236. Of these, only 47,219 occur in RCV1-v2 training and/or test documents, so 47,219 is size of the complete feature set for RCV1-v2. Of these 47,219 terms, only 47,152 have one or more occurrences in the RCV1-v2 training set, and so were available to be included in classifiers. Average document length for RCV1-v2 documents with our text representation is 123.9 terms, and average number of unique terms in a document is 75.7.

The weight of a term in a vector was computed using Cornell *ltc* term weighting (Buckley, Salton, and Allan, 1994), a form of $TF \times idf$ weighting. This gives term t in document d an initial weight of

$$w_d(t) = (1 + \log_e n(t,d)) \times \log_e(|\mathcal{D}|/n(t)),$$

where $n(t)$ is the number of documents that contain t, $n(t,d)$ is the number of occurrences of term t in document d, and $|\mathcal{D}|$ is the number of documents used in computing the inverse document frequency weights (*idf* weights).

The *idf* weights used were computed from all 23,307 RCV1-v1 documents which fall before our chronological breakpoint, not just the 23,149 RCV1-v2 documents we used for training. Again, while unintentional, this a legitimate use of additional unlabeled data. Only the document text from the additional documents was used, not their codes. The resulting *idf* values are in most cases almost identical to the intended ones.

For the k-NN and Rocchio algorithms (but not SVM.1 and SVM.2) we then applied feature selection to the vectors, as described in Section 6.5. This implicitly replaced $w_d(t)$ with $w'_d(t)$, where

$w'_d(t)$ was equal to $w_d(t)$ for features chosen by feature selection, but equal to 0.0 for nonselected features.

Finally, *ltc* weighting handles differences in document length by cosine normalizing the feature vectors (normalizing them to have a Euclidean norm of 1.0). These resulting final weights were

$$w''_d(t) = \frac{w'_d(t)}{\sqrt{\sum_u w'_d(u) \times w'_d(u)}}.$$

Since cosine normalization was done after feature selection, both the set of nonzero feature values, and the feature values themselves, differ among the runs.

Despite being fairly straightforward by IR standards, we recognize that the above preprocessing would be nontrivial to replicate exactly. We therefore have made the exact data used in our experiments available in two forms.

Online Appendix 12 contains documents that have been tokenized, stopworded, and stemmed. Online Appendix 13 contains documents in final vector form, i.e., as *ltc* weighted vectors. No feature selection has been done for these vectors, i.e., they are as used for SVM training and testing. (And thus are different from those used with *k*-NN and Rocchio, since those had feature selection applied.) Online Appendix 13 uses numeric term IDs rather than the string form of words. Online Appendix 14 gives the mapping between the numeric term IDs and string forms.

The vectors in Online Appendix 13 are based on terms that occurred in our pre-breakpoint documents, and so *should only be used in experiments based on the same training/test split as in this paper.* In contrast, the tokenized representations in Online Appendix 12 contain all nonstopwords, and so can be used with any training/test split. Online Appendix 12 will be preferable for most purposes.

Reuters has agreed (Rose, 2002; Whitehead, 2002) to our distribution of these token and vector files without a license agreement. We nevertheless strongly encourage all users of these files to license the official RCV1 CD-ROMs (see the Acknowledgments section at the end of this paper for details).

8. Benchmarking the Collection: Results

Tables 5 and 6 give microaveraged and macroaveraged values of $F_{1.0}$ for the four classification methods, three category sets, and three subsets of each category set. The results largely confirm past studies: SVMs are dominant, weighted *k*-NN is competitive, and the Rocchio-style algorithm is a plausible but lagging straw man. The choice of averaging, category set, and effectiveness measure affects absolute scores but rarely the ordering of approaches.

We provide not only the averaged data of Tables 5 and 6, but also the full testset contingency tables for each category as Online Appendix 15. This allows computing alternate effectiveness measures for our classifiers, recognizing of course that the classifiers were trained to optimize $F_{1.0}$.

For example, one might feel that only leaf nodes of the Topic hierarchy should be used for evaluation, since assignments of internal nodes are partially based on automated expansion of leaf assignments. Averaged effectiveness measures using only leaf categories could be computed from our contingency tables.

Category Set	Subset	SVM.1	SVM.2	*k*-NN	Rocchio
Topics	1+ train (101)	0.816	0.810	0.765	0.693
	1+ test (103)	0.816	0.810	0.765	0.693
	all (103)	0.816	0.810	0.765	0.693
Industries	1+ train (313)	0.513	–	0.396	0.384
	1+ test (350)	0.512	–	0.396	0.384
	all (354)	0.512	–	0.395	0.384
Regions	1+ train (228)	0.874	–	0.792	0.794
	1+ test (296)	0.873	–	0.791	0.793
	all (366)	0.873	–	0.791	0.793

Table 5: Effectiveness (microaveraged $F_{1.0}$) of classifiers trained with four supervised learning algorithms, with parameter settings chosen to optimize microaveraged $F_{1.0}$ on cross-validation folds of the training set. Classifiers were trained on our RCV1-v2 training set (23,149 documents) and tested on our RCV1-v2 test set (781,265 documents). The computationally expensive SVM.2 algorithm was run only on Topics. Separate microaverages are presented for categories with one or more training set positive examples (all of which also have one or more test set positive examples), categories with one or more test set positive examples but no training set positive examples, and all categories. The number of categories in each subset is shown in parentheses.

Category Set	Subset	SVM.1	SVM.2	*k*-NN	Rocchio
Topics	1+ train (101)	0.619	0.557	0.560	0.504
	1+ test (103)	0.607	0.546	0.549	0.495
	all (103)	0.607	0.546	0.549	0.495
Industries	1+ train (313)	0.297	–	0.235	0.170
	1+ test (350)	0.266	–	0.210	0.152
	all (354)	0.263	–	0.208	0.151
Regions	1+ train (228)	0.601	–	0.588	0.572
	1+ test (296)	0.463	–	0.453	0.441
	all (366)	0.375	–	0.366	0.356

Table 6: Effectiveness (macroaveraged $F_{1.0}$) of classifiers trained with four supervised learning algorithms, with parameter settings chosen to optimize macroaveraged $F_{1.0}$ on cross-validation folds of the training set. Other details are as in Table 5.

8.1 Microaveraging vs. Macroaveraging

Microaveraged measures are dominated by high frequency categories. For RCV1, this effect varies among the category sets. For Topics, hierarchical expansion inflates the frequency of all non-leaf categories. Non-leaf categories account for only 20% (21/103) of Topic categories but 79% (2,071,530 / 2,606,875) of all Topic code assignments. The four top level Topic categories (CCAT, ECAT, GCAT, and MCAT) alone account for 36% (945,334 / 2,606,875) of all Topic assignments. Thus, microaveraged scores for Topics largely measure effectiveness at broad, perhaps less interesting, content distinctions. In contrast, hierarchical expansion for Industry categories affected only a few categories, and Regions underwent no hierarchical expansion at all. The most frequent (and thus dominant) categories for Industries and Regions are not necessarily the semantically broadest categories.

Macroaveraging, on the other hand, gives equal weight to each category, and thus is dominated by effectiveness on low frequency categories. For Topics and Industries this is largely the leaf categories in each taxonomy, thus categories with narrow meanings. For Regions, narrowness of meaning is less the issue than degree to which the particular geographic entity is covered in the news. Macroaveraged effectiveness for Regions is dominated by categories corresponding to countries that are discussed only infrequently in international news.

8.2 Averaging Over Categories with No Positive Examples

Past research has varied in how categories with no training or test examples are handled in measuring text categorization effectiveness. We include averages over all categories, over categories with at least one positive training example, and over categories with at least one positive training examples and at least one positive test example. (For our training/test split of RCV1-v2 there were no categories that had one or more positive training examples but zero positive test examples.) Each average is useful for different purposes:

- *Averaging over all categories:* This best reflects the operational task. Such an average is also the most appropriate for comparisons with knowledge-based and string-matching approaches, since these can be used even on categories with no positive training examples.
- *Averaging over categories with one or more positive test examples:* This factors out the impact of choosing an arbitrary value (0.0 in our case) for $F_{1.0}$ when there are no positive test examples (Section 5.3). This impact can occasionally be large. For instance, macroaveraged effectiveness figures for Regions on RCV1-v2 are strongly affected by whether categories with no positive test examples are included in the average (Table 6).
- *Averaging over categories with one or more positive training examples:* This is appropriate when the primary goal is research on supervised learning methods.

8.3 Effectiveness on Individual Categories

Past text categorization research arguably has overemphasized average effectiveness. This was partly a necessity. With the widely used ModApte split of Reuters-21578, the median frequency Topic category (of 135 Topic categories defined on that collection) has only three test set occurrences, so averaging was necessary to produce effectiveness figures that were at all accurate.

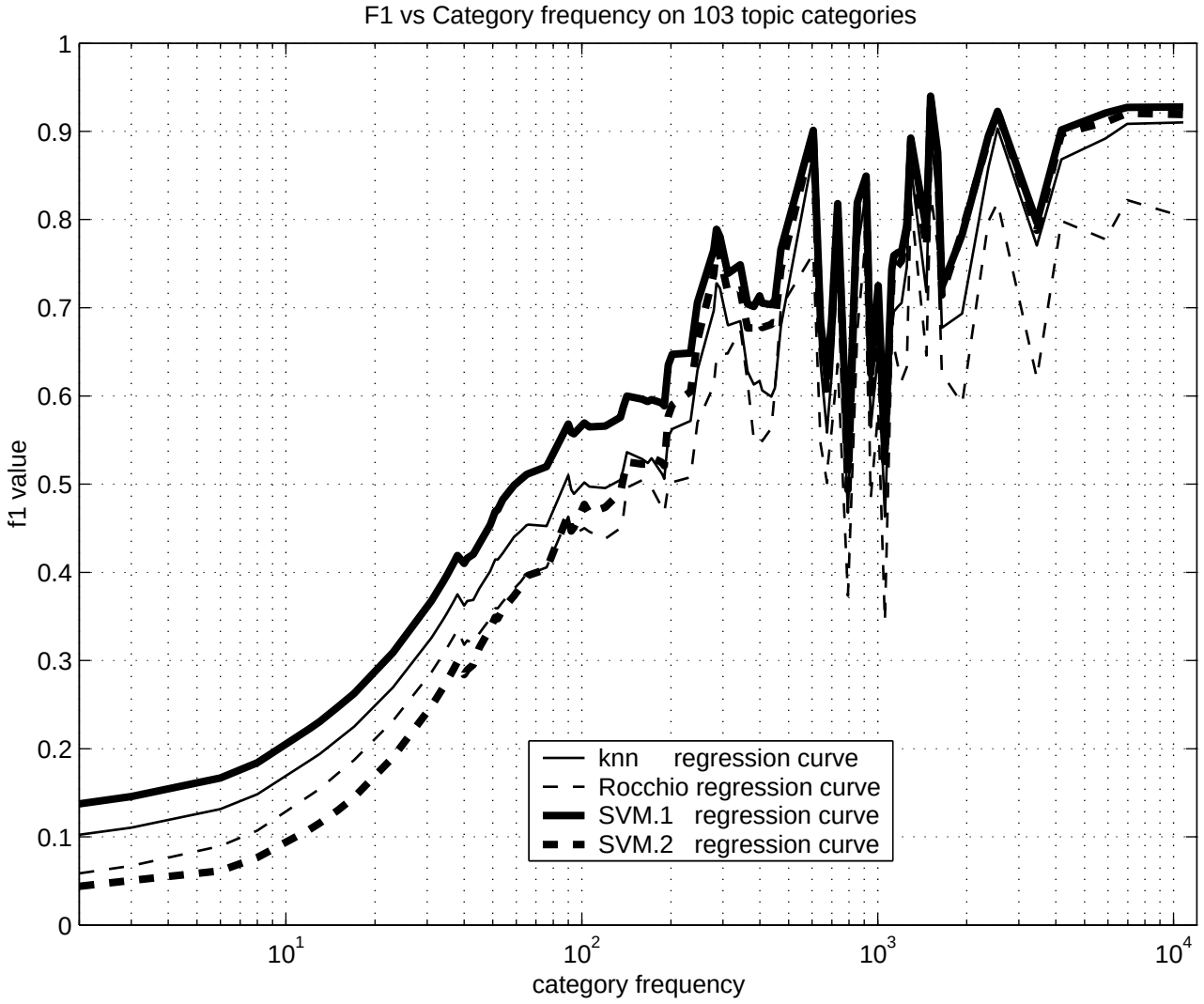


Figure 2: Test set $F_{1.0}$ for four classifier approaches on 103 RCV1-v2 Topic categories. Categories are sorted by training set frequency, which is shown on the x-axis. The $F_{1.0}$ value for a category with frequency x has been smoothed by replacing it with the output of a local linear regression over the interval $x-200$ to $x+200$.

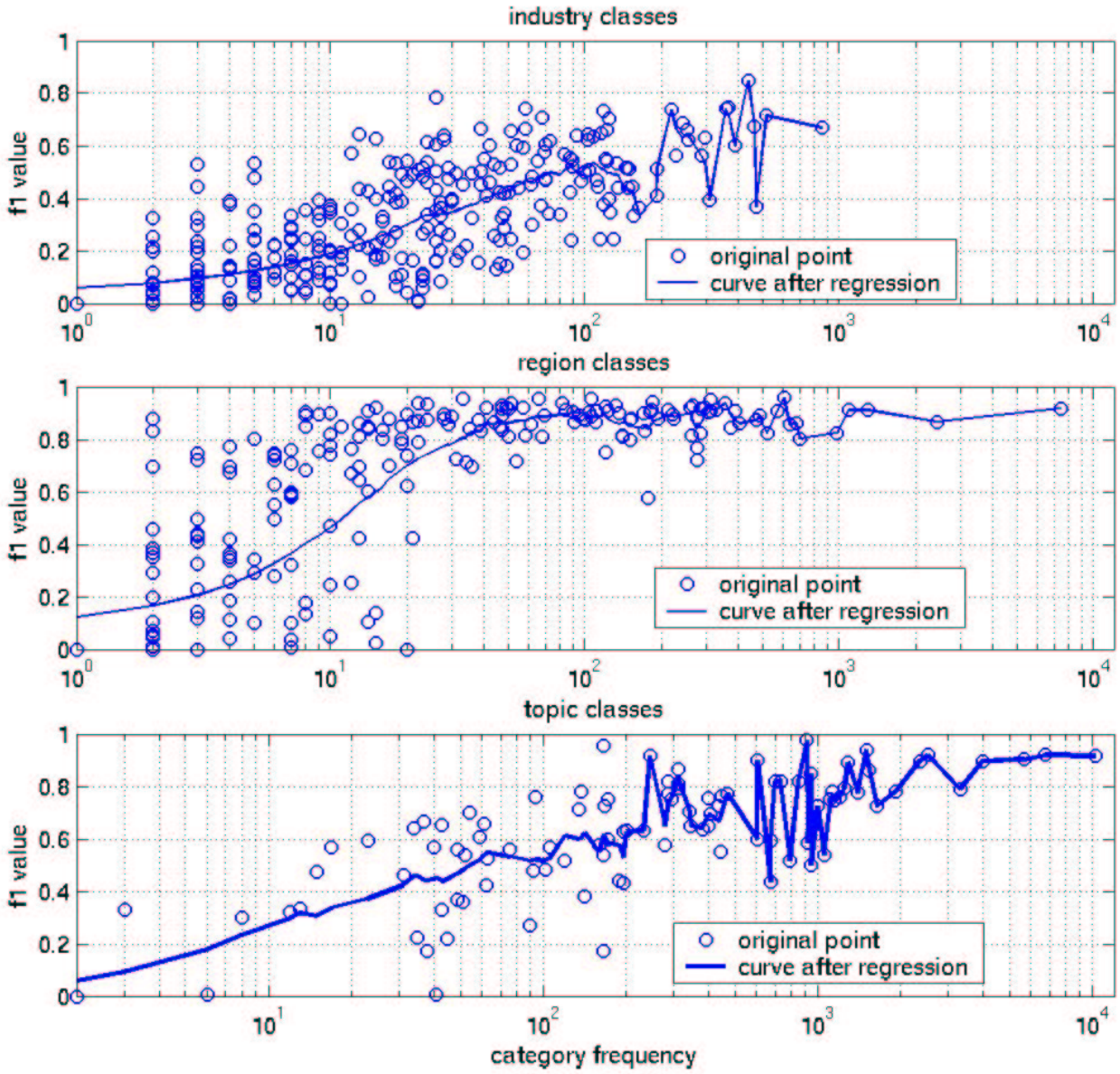


Figure 3: Raw test set $F_{1.0}$ values for the SVM.1 approach on all three category sets. The line shows corresponding smoothed (as in Figure 2) values. Training set frequency is shown on the x-axis.

In contrast, the median frequency Topic category for our test set has 7,250 test set occurrences: bigger than the entire ModApte test set. Only three categories have fewer than 100 test set occurrences, so category-level effectiveness figures are more meaningful.

Figure 2 shows smoothed $F_{1.0}$ values for our four classifier training approaches on the 103 Topic categories, sorted by training set frequency of the category. While smoothing aids comparison across the classifiers, it hides a good deal of category-to-category variation. Figure 3 instead shows raw $F_{1.0}$ values for the SVM.1 approach on all three category sets.

Since our focus is methodological we make only a few observations on this data:

- Effectiveness generally increases with increasing class frequency, but the category-to-category variation is very large (Figure 3). This variation has been noted for previous collections, but the large size of RCV1 gives more confidence in this observation. Further, some of the decrease in variation at the right of the graph results from the fact that even a poor classification on a high frequency category can yield a moderately high F-measure value (Lewis and Tong, 1992). For instance, the most frequent Topic category has a test set frequency of 0.465. A classifier that simply assigned *all* test documents to this category would have an $F_{1.0}$ of 0.635.

- Among the tested approaches, SVM classifiers and in particular SVM.1 classifiers, are dominant at all category frequencies. This fact has been obscured in some previous SVM studies, which restricted experiments to a small set of high frequency categories or presented only microaveraged effectiveness measures.

- The SCutFBR.1 approach to threshold tuning for SVMs (SVM.1) is as good or better than the more computationally expensive leave-one-out procedure (SVM.2). Interestingly, it appears that the difference in the effectiveness of SVM.1 and SVM.2 largely results from their choice of threshold rather than from the orientation of the resulting hyperplanes. We did a test (results not reported here) in which we set both SVM.1 and SVM.2 classifiers to their test set optimal thresholds, and found the resulting effectiveness to be almost identical. This similarity of effectiveness is somewhat surprising, since SVM.2 often chooses hyperplanes with substantially different orientations than those chosen by SVM.1. We found the angle between the normals of the SVM.1 and SVM.2 hyperplanes (the inverse cosine of the dot product of weight vectors normalized to have Euclidean norm of 1.0), averaged over the 103 Topic categories, to be 19.6 degrees.

- We find some support for previous suggestions (Schapire, Singer and Singhal, 1998) that Rocchio-style algorithms are at their best when relatively few positive examples are available, though in all cases they lag the other methods tested. An interesting avenue for future work, now possible with RCV1, would be teasing apart the impact of category narrowness vs. the number of positive training examples supplied (perhaps using stratified sampling).

9. Summary

Research in machine learning is heavily driven by available data sets, and supervised learning for text categorization is no exception. We believe RCV1 has the potential to support substantial research advances in hierarchical categorization, scaling of learning algorithms, effectiveness on low frequency categories, sampling strategies, and other areas. As of January 5, 2004, the collection had been distributed by Reuters to 520 groups, suggesting it is likely to be widely used.

We hope that by documenting the data production process, the nature of the coding, and the impact of these on the resulting test collection, we have contributed to the usefulness of the collection. Some of the insights here may also be of use to those producing future test collections and managing real-world text classification systems. Finally, we hope that our benchmark data will encourage replicability and transparency in future text categorization research.

Acknowledgments

We are grateful to Reuters, Ltd. for making Reuters Corpus Volume 1 available, and for supporting its design and production by Reuters employees Chris Harris and Miles Whitehead. In addition, one of us (Tony Rose) thanks Reuters for supporting his work on the corpus while a Reuters employee. We also acknowledge and thank Stephen Robertson (then of City University London) for his work with Reuters on planning the corpus.

We urge the research community to support these efforts by respecting the terms of the license agreement (*http://about.reuters.com/researchandstandards/corpus/agreement.htm*), in particular clause 3.3 on acknowledging Reuters and providing a copy of any publication. We encourage those with questions about the corpus to post them to the Reuters Corpora mailing list at *http://groups.yahoo.com/group/ReutersCorpora/*.

Many current and former Reuters employees provided information on the production of the data or on editorial processes at Reuters. These include Dave Beck, Chris Harris, Paul Hobbs, Steven Murdoch, Christopher Porter, Jo Rabin, Mark Stevenson, Miles Whitehead, Richard Willis, and Andrew Young. This paper would not have been possible without their input, and we apologize to any whose names we have missed. We also thank Tom Ault, Evgeniy Gabrilovich, Alex Genkin, Paul Kantor, Mikhail Kreines, Ray Liere, Yury Lubensky, David Madigan, Herbert Roitblat, Fabrizio Sebastiani, Bill Teahan, Benjy Weinberger, and the anonymous JMLR reviewers for comments on drafts of this paper, for sharing their data on RCV1, or for other help.

This research was supported in part by the U.S. National Science Foundation (NSF) under grant numbers KDI-9873009, IIS-9982226, EIA-0087022, and DMS-0113236. Any opinions or conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

References

D. G. Altman. *Practical Statistics for Medical Research*. Chapman & Hall/CRC, 1991.

T. Ault and Y. Yang. kNN, Rocchio and metrics for information filtering at TREC-10. In *The Tenth Text REtrieval Conference (TREC 2001)*, pages 84–93, Gaithersburg, MD 20899-0001, 2002. National Institute of Standards and Technology. *http://trec.nist.gov/pubs/trec10/papers/cmucat-correct.pdf*.

C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR 94)*, pages 292–300, 1994.

C. W. Cleverdon. The significance of the Cranfield tests of index languages. In *Proceedings of the Fourteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR 91)*, pages 3–12, 1991.

Great Britain Office for National Statistics. *Indexes to UK Standard Industrial Classification of Economic Activities 1992 UK SIC(92)*. Office for National Statistics, London, 1997.

Great Britain Office for National Statistics. *UK Standard Industrial Classification of Economic Activities UK SIC(92)*, December 20, 2002. *http://www.statistics.gov.uk/methods_quality/sic/contents.asp*.

R. Grishman and B. Sundheim. Design of the MUC-6 evaluation. In *Sixth Message Understanding Evaluation (MUC-6)*, pages 1–12. Defense Advanced Research Projects Agency, Morgan Kaufmann, 1995.

P. J. Hayes and S. P. Weinstein. CONSTRUE/TIS: A System for Content-Based Indexing of a Database of News Stories. In *Second Annual Conference on Innovative Applications of Artificial Intelligence*, pages 49–64, 1990.

W. Hersh, C. Buckley, T. J. Leone, and D. Hickman. OHSUMED: an interactive retrieval evaluation and new large text collection for research. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR 94)*, pages 192–201, 1994.

D. J. Ittner, D. D. Lewis, and D. D. Ahn. Text Categorization of Low Quality Images. In *Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, 1995.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML '98)*, pages 137–142, Berlin, 1998.

T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML'99)*, pages 200–209, San Francisco, CA, 1999.

T. Joachims. SVM Light: Support Vector Machine, May 13th, 2002. *http://svmlight.joachims.org*.

D. V. Khmelev and W. J. Teahan. A repetition based measure for verification of text collections and for text categorization. In *Proceedings of the Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 03)*, pages 104–110, 2003.

D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *International Conference on Machine Learning (ICML'97)*, pages 170–178, Nashville, 1997.

F. W. Lancaster. *Indexing and Abstracting in Theory and Practice.* Second edition. University of Illinois, Champaign, IL, 1998.

D. D. Lewis. Evaluating text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 312–318. Defense Advanced Research Projects Agency, Morgan Kaufmann, 1991.

D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 92)*, pages 37–50, 1992.

D. D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 95)*, pages 246–254, 1995.

D. D. Lewis. Reuters-21578 text Categorization test collection. Distribution 1.0. README file (version 1.2). Manuscript, September 26, 1997.
http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt

D. D. Lewis. Applying support vector machines to the TREC-2001 batch filtering and routing tasks. In *The Tenth Text REtrieval Conference (TREC 2001)*, pages 286–292, Gaithersburg, MD 20899-0001, 2002. National Institute of Standards and Technology. *http://trec.nist.gov/pubs/trec10/papers/daviddlewis-trec2001-draft4.pdf*.

D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 96)*, pages 298–306, 1996.

D. D. Lewis and R. M. Tong. Text filtering in MUC-3 and MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 51–66. Defense Advanced Research Projects Agency, Morgan Kaufmann, 1992.

M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

M. F. Porter. The Porter Stemming Algorithm, 2003.
http://www.tartarus.org/˜martin/PorterStemmer.

S. Robertson and I. Soboroff. The TREC 2001 filtering track report. In *The Tenth Text REtrieval Conference (TREC 2001)*, pages 26–37, Gaithersburg, MD 20899-0001, 2002. National Institute of Standards and Technology. *http://trec.nist.gov/pubs/trec10/papers/filtering2_track.pdf*.

J. J. Rocchio, Jr.. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, 1971.

M. Rogati and Y. Yang. High performing and scalable feature selection for text classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 659-661, 2002.

T. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1 – from Yesterday's News to Tomorrow's Language Resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002.
http://about.reuters.com/researchandstandards/corpus/LREC_camera_ready.pdf

T. Rose. Electronic mail message to *ReutersCorpora@yahoogroups.com*, June 11, 2002. *http://groups.yahoo.com/group/ReutersCorpora/message/70*.

G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of American Society for Information Sciences*, 41:288–297, 1990.

G. Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

R. E. Schapire, Y. Singer, and A. Singhal. Boosting and Rocchio applied to text filtering. In *Proceedings of the Twenty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 98)*, pages 215–223, 1998.

F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

J. M. Tague. The pragmatics of information retrieval experimentation. In K. Sparck Jones, editor, *Information Retrieval Experiment*, chapter 5. Butterworths, 1981.

C. J. van Rijsbergen. *Automatic Information Structuring and Retrieval*. PhD thesis, King's College, Cambridge, 1972.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

M. Whitehead. Electronic mail message to *ReutersCorpora@yahoogroups.com*, November 14, 2002. *http://groups.yahoo.com/group/ReutersCorpora/message/106*.

A. S. Weigend, E. D. Wiener, and J. O. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.

Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):67–88, 1999.

Y. Yang. A study on thresholding strategies for text categorization. In *The Twenty-Fourth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 01)*, pages 137–145, 2001.

Y. Yang, T. Ault, T. Pierce, and C. W. Lattimer. Improving text categorization methods for event tracking. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 00)*, pages 65–72, 2000.

Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the Twenty-Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 99)*, pages 42–49, 1999.

Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *The Fourteenth International Conference on Machine Learning (ICML'97)*, pages 412–420. Morgan Kaufmann, 1997.

T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.

Journal of Machine Learning Research 5 (2004) 399–430 Submitted 7/03; Revised 1/04; Published 4/04

Distributional Scaling: An Algorithm for Structure-Preserving Embedding of Metric and Nonmetric Spaces

Michael Quist MJQ1@CORNELL.EDU
Department of Chemistry and Biochemistry
University of California at Los Angeles
Los Angeles, CA 90095, USA

Golan Yona GOLAN@CS.CORNELL.EDU
Department of Computer Science
Cornell University
Ithaca, NY 14853, USA

Editor: Bin Yu

Abstract

We present a novel approach for embedding general metric and nonmetric spaces into low-dimensional Euclidean spaces. As opposed to traditional multidimensional scaling techniques, which minimize the distortion of pairwise distances, our embedding algorithm seeks a low-dimensional representation of the data that preserves the structure (geometry) of the original data. The algorithm uses a hybrid criterion function that combines the pairwise distortion with what we call the geometric distortion. To assess the geometric distortion, we explore functions that reflect geometric properties. Our approach is different from the Isomap and LLE algorithms in that the discrepancy in distributional information is used to guide the embedding. We use clustering algorithms in conjunction with our embedding algorithm to direct the embedding process and improve its convergence properties.

We test our method on metric and nonmetric data sets, and in the presence of noise. We demonstrate that our method preserves the structural properties of embedded data better than traditional MDS, and that its performance is robust with respect to clustering errors in the original data. Other results of the paper include accelerated algorithms for optimizing the standard MDS objective functions, and two methods for finding the most appropriate dimension in which to embed a given set of data.

Keywords: Embedding, multidimensional scaling, PCA, earth-mover's distance

1. Introduction

Embedding is concerned with mapping a given space into another space, often Euclidean, in order to study the properties of the original space. This can be especially effective when the original space is a set of abstract objects (e.g., strings, trees, graphs) related through proximity data, as a low-dimensional embedding can help in visualizing the abstract space. Embedding can also be applied when the objects are points in a vector space whose dimensionality is too large for the application of data analysis algorithms, such as clustering. In such cases, embedding can be used to lower the dimensionality of the space.

1.1 Background

In general, embedding techniques fall into two categories: linear and nonlinear. Classical **linear embedding**, as embodied by principal component analysis (PCA), reduces dimensionality by projecting high-dimensional data onto a low-dimensional subspace. The optimal p-dimensional subspace is selected by rotating the coordinate axes to coincide with the eigenvectors of the sample covariance matrix, and keeping the p axes along which the sample has the largest variance. Principal component analysis directly applies to data that already resides in a real normed space. It can also be applied to proximity data that has been appropriately preprocessed, under certain spectral conditions on the matrix of pairwise distances (Cox and Cox, 2001).

Nonlinear embedding techniques, also referred to as multidimensional scaling (MDS) techniques, apply to a broad set of data types. Generally speaking, the goal of MDS is to construct a low-dimensional map in which the distance between any two objects corresponds to their degree of dissimilarity. The method maps a given set of samples into a space of desired dimension and norm. A random mapping (or projection by PCA) can serve as the initial embedding. A stress function that compares proximity values with distances between points in the host space (usually a sum-of-squared-errors function) is used to measure the quality of the embedding, and a gradient descent procedure is applied to improve the embedding until a local minimum of the stress function is reached. Like PCA, MDS attempts to preserve all pairwise distances as well as possible; but the restriction to linear projections is removed, and arbitrary embeddings are considered. Many variants of this general approach are reported in the literature; a broad overview of the field is given by Cox and Cox (2001).

The MDS method was traditionally used to visualize high-dimensional data in two or three dimensions. It has long been employed for data analysis in the social sciences, where the generated maps tend to have only a few hundred data points, and computational efficiency is not a factor (Sammon, 1969). Practically, such procedures are not effective for more than few thousand sample points. More recently, MDS has been turned toward the visualization of large biological and chemical data sets, with thousands or even millions of points (Yona, 1999; Apostol and Szpankowski, 1999). Applying traditional MDS to very large data sets is prohibitively slow, leading several authors to propose approximations and workarounds. Linial et al. (1995) presented a randomized approach that attempts to bound the distortion. However, the bound is not tight, and in practice this approach can introduce large distortions, as no objective function is explicitly optimized. A different randomized approach, based on iteratively adjusting the lengths of randomly selected edges, was proposed by Agrafiotis and Xu (2002). This method has linear time complexity, and is therefore well-suited to extremely large data sets. Basalaj (1999) proposed an incremental method for large-scale MDS. It consists of embedding a small subset of objects carefully, then using this skeleton embedding to determine the positions of the remaining objects.

Recently, a new class of non-linear embedding techniques has emerged: the **manifold learning** algorithms, which comprise an active area of research. These algorithms are designed to discover the structure of high-dimensional data that lies on or near a low-dimensional manifold. There are several approaches. The Isomap algorithm (Tenenbaum et al., 2000) uses geodesic distances between points instead of simply taking Euclidean distances, thus "encoding" the manifold structure of the input space into the distances. The geodesic distances are computed by constructing a sparse graph in which each node is connected only to its closest neighbors. The geodesic distance between each pair of nodes is taken to be the length of the shortest path in the graph that connects

them. These approximate geodesic distances are then used as input to classical MDS. The LLE algorithm (Roweis and Saul, 2000; Saul and Roweis, 2003) uses a collection of local neighborhoods to guide the embedding. The assumption is that if the neighborhoods are small, they can be approximated as linear manifolds, and the position of each point can be reconstructed as a weighted linear combination of its k nearest neighbors. The positions of the points in the lower-dimensional space are determined by minimizing the reconstruction error in this low-dimensional space (with fixed weights that were determined in the original high-dimensional space). This is done by solving an eigenvector problem, as in PCA. Another approach is the eigenmaps method. The goal of this type of method is to minimize a quadratic form (either the squared Hessian or the squared gradient) over all functions mapping the manifold into the embedding space (Donoho and Grimes, 2003; Belkin and Niyogi, 2002). When the continuous function is approximated by a linear operator on the neighbor graph, the maximization problem becomes a sparse matrix eigenvalue problem and is readily solved.

The manifold learning methods form a powerful generalization of PCA. Unlike PCA, which is useful only when the data lies near a low-dimensional *plane*, these methods are effective for a large variety of manifolds. By using a collection of local neighborhoods, or by exploiting the spectral properties of the adjacency graph, they extract information about local manifolds from which the global geometry of the manifold can be reconstructed. In practice, preserving these local manifolds results in non-linear embeddings. The underlying principles of these methods are similar, and their power stems from the fact that they practically employ alternative representations for the data points. PCA seeks correlation between features and represents the data best in a sum-of-squared-errors sense. However, it implicitly assumes the Euclidean metric. On the other hand, the manifold learning algorithms explore the properties of the adjacency graph to form a new representation, inducing a new metric. For example, the geodesic distance in essence samples the geometry of the input manifold, and it is that definition to which one can attribute the great success of the Isomap algorithm. Similarly, the spectral approaches use the proximity data to derive the new representation that reflects collective properties. This is related to other studies that showed that encoding data through collective or transitive relations can be very effective for data representation (e.g., embedding) as well as for clustering (Smith, 1993; Wu and Leahy, 1993; Shi and Malik, 1997; Blatt et al., 1997; Gdalyahu et al., 1999; Dubnov et al., 2002).

The different types of embedding methods are inherently suited to different types of problems. PCA identifies significant coordinates and linear correlations in the original, high-dimensional data. It is therefore appropriate for finding a simple, linear, globally applicable rule for extracting information from *new* data points. It is unsuitable when the correlations are nonlinear or when no simple rule exists. General multidimensional scaling techniques are appropriate when the data is highly nonmetric and/or sparse. However, MDS is iterative, does not guarantee optimality or uniqueness of its output, does not generate a rule for interpreting new data, and is typically quite slow compared with other methods. These deficiencies are only tolerable when weighed against the greater generality and simpler formulation of multidimensional scaling. Finally, manifold-learning techniques are appropriate when a strong nonlinear relation exists in the original data. In such cases, the methods described can make use of powerful, noniterative methods, with guaranteed global optimality. They are less suitable when not enough data is available, or when the data points are inconsistent with a manifold topology (for instance, lying on a structure with branches and loops), or when the data is intrinsically nonmetric.

1.2 Method

The algorithm presented in this paper is in the class of nonlinear embedding techniques. However, unlike the manifold learning methods, our focus is on the higher-order structure of the data. The aforementioned approaches optimize an objective function that is a function of the individual pairwise distances or their derivatives. However, collective aspects of the embedding are not explicitly considered, even when local neighborhoods are used. This problem is addressed in this paper.

In a recent study by Roth et al. (2002), the authors point out that high-dimensional PCA, applied to dissimilarity data that has been shifted by an additive constant, automatically preserves some clustering properties of the original data. Specifically, they show that the optimal partition of the original data points into k clusters (using a particular cost function, which they define) is identical to the optimal partition of the embedded data points, using the standard k-means cost function. However, a subsequent reduction in the embedding dimension is often desirable, and the clustering properties are not preserved (or even considered) in this second stage.

Our interest in embedding algorithms emerges from our even stronger interest in studying high-order organization in complex spaces. In a typical application one is interested in exploratory data analysis, discovering patterns and "functional" meaningful clusters in the data. Embedding is often used to visualize complex data in a low-dimensional space, in the hope that it will be easier to discover structure or statistical regularities in the reduced data. Thus, optimal embedding should consider not only the distortion in pairwise distances that is introduced by the embedding, but also the **geometric distortion**, i.e., the disagreement on the intrinsic structure of the data. Finding the optimal embedding thus becomes a problem of optimizing a complex criterion function that seeks to jointly improve both aspects of an embedding. Our approach tackles the problem from this perspective and attempts to preserve these patterns by implicitly encoding the cluster structure into the cost function. Here we present for the first time such a criterion function and describe the means to optimize it.

Another new element of our paper is a method to deduce the right dimension for the data. Existing methods for dimensionality reduction are looking for elbows in the residual variance graph to determine the right dimensionality, however, the exact definition is subjective and qualitative. Here we introduce two quantitative methods to deduce the right dimension.

The paper is organized as follows. We first describe the two commonly-used MDS objective functions, the SAMMON and SSTRESS functions, and present improved algorithms for optimizing them. Next, we present a hierarchical method for efficiently embedding data sets that consist of many subsets or clusters of related objects. We then present the main element of this paper, a new type of MDS called **distributional scaling**, which directly addresses the problem of structure preservation during the embedding process. Distributional scaling strives to maintain the *distribution* of dissimilarities, as well as the individual dissimilarities themselves, thereby using higher-order information to create a more informative map. Next, we describe two distinct methods for ascertaining the best dimension in which to embed a given data set. Finally, we test the performance of distributional MDS on a large number of synthetic data sets. By using this new form of scaling, we demonstrate that we are able to remove undesirable artifacts from embeddings produced by traditional MDS.

2. Theory

We start with some basic definitions and a review of classical metric and nonmetric MDS. We then introduce hierarchical MDS and Distributional MDS, and discuss the measures that we use to evaluate similarity between probability distributions. We conclude this section with a method to choose the embedding dimension.

2.1 Definition and Mathematical Preliminaries

Throughout this paper we will be interested in optimizing embeddings of sets of objects in Euclidean space. An embedding of n objects in p-dimensional Euclidean space is a set of image points $\mathbf{x}_i \in \mathcal{R}^p$, where $i = 1,\ldots,n$. We take S_n^p to be the set of all such embeddings.

We primarily will be interested not in the image points themselves, but in the distances between them. Let Ω_n be the set of symmetric $n \times n$ matrices with zeros along the diagonal. For each embedding X of n objects, we can define the distance matrix $D(X) \in \Omega_n$, with matrix elements $D_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$. Since the interpoint distances are invariant under Euclidean transformations of the entire configuration of points (that is, translations, rotations, and reflections), D is many-to-one. We denote by D_n^p the image of S_n^p under the mapping D. This is the space of all possible distance matrices arising from p-dimensional embeddings of n points.

Formally, the optimization problem is defined as follow: we are given a set of n objects and their dissimilarities. Denote by Δ_{ij} the dissimilarity of objects i and j. The goal is to find a configuration of image points $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ such that the $n(n-1)/2$ distances D_{ij} between image points are as close as possible to the corresponding original dissimilarities Δ_{ij}.

2.2 Metric MDS

The simplest case is **metric MDS**, where the dissimilarity data is quantitative. We are given n objects, together with a target dissimilarity matrix $\Delta \in \Omega_n$. The goal is to find an embedding X such that the distance matrix $D(X)$ matches Δ as closely as possible. This is formulated as a weighted least-squares optimization problem: given $(\Delta, W) \in \Omega_n \times \Omega_n$, where $W = (w_{ij})$ is a symmetric matrix of weights, minimize

$$\mathcal{H}(X) = \sum_{i<j} w_{ij} \Big(f(D_{ij}(X)) - g(\Delta_{ij}) \Big)^2 \tag{1}$$

over all $X \in S_n^p$. The functions f and g determine exactly how errors are penalized. Two common choices for these functions are considered here. The stress, or SAMMON, objective function is defined by $f(x) = g(x) = x$. The squared stress, or SSTRESS, function is defined by $f(x) = g(x) = x^2$.

$$\begin{aligned} \text{SAMMON}: \quad \mathcal{H}(X) &= \sum_{i<j} w_{ij} \Big(D_{ij} - \Delta_{ij} \Big)^2, \\ \text{SSTRESS}: \quad \mathcal{H}(X) &= \sum_{i<j} w_{ij} \Big(D_{ij}^2 - \Delta_{ij}^2 \Big)^2. \end{aligned}$$

The SAMMON and SSTRESS objective functions have somewhat different advantages. While the former seems more natural, being the square of the Euclidean metric in Ω_n, and may produce more

aesthetically pleasing embeddings, the latter is more tractable from a computational standpoint, and seemingly less plagued by nonglobal minima (Malone and Trosset, 2000).

The weights contained in the weight matrix W are arbitrary. They can be used to exclude missing proximity data, or to account for data with varying confidence levels. In practice, however, the weights are often defined in terms of Δ. Three choices of this type are:

$$\begin{aligned} w_{ij}^{-1} &= \sum_{m<n} g(\Delta_{mn})^2 \,, \\ w_{ij}^{-1} &= g(\Delta_{ij}) \sum_{m<n} g(\Delta_{mn}) \,, \\ w_{ij}^{-1} &= \frac{1}{2} n(n-1) g(\Delta_{ij})^2 \,. \end{aligned}$$

All three choices normalize the metric stress function, in the sense that $\mathcal{H}(0) = 1$. We refer to the first one as global weighting, the second as intermediate (or semilocal) weighting, and the third as local weighting. Unless otherwise specified, the global weighting scheme is used in this paper.

The numerical optimization of the metric stress function is not entirely trivial. The deterministic algorithms (gradient descent) that are typically applied to solve this problem converge to local minima, which may not be globally optimal. It is possible to use stochastic techniques, like simulated annealing (Klein and Dubes, 1989), to reduce or eliminate the probability of being trapped in a nonglobal minimum, albeit at the cost of increased computation time. Recently, Klock and Buhmann (1997) have demonstrated that so-called *deterministic annealing* can be used to avoid poor minima without sacrificing too much efficiency, thus combining the merits of the stochastic and deterministic approaches. Such globalization strategies are outside the scope of the present study. Instead, we have developed an efficient method for finding *local* minima that takes advantage of special features of the SSTRESS and SAMMON objective functions. This algorithm is described in detail in Appendix A.

2.3 Nonmetric MDS

A generalization of the metric problem is **nonmetric MDS**, which is appropriate when the dissimilarity data is not quantitative, but merely ordered. In this case, we minimize an objective function like Eq. (1) over X, while also allowing g to vary over all increasing functions. As with metric MDS, the Euclidean distances will be transformed by a known function $f(x)$, which we will restrict to be x or x^2, in the SAMMON and SSTRESS cases respectively.

Note that, if we were to use Eq. (1) with fixed weights, the objective function would be trivially minimized by taking g to zero and shrinking the configuration X to a single point. Instead we use global weighting, as described above. This sets the overall weight to an appropriate functional of g, producing a scale-invariant objective function:

$$\mathcal{H}_{nm}(X,g) = \frac{\sum_{i<j} \Big(f(D_{ij}(X)) - g(\Delta_{ij}) \Big)^2}{\sum_{i<j} g(\Delta_{ij})^2} \,. \tag{2}$$

Our algorithm for optimizing metric MDS can be extended to cover the nonmetric case as well. Appendix B discusses the necessary modifications.

2.4 Hierarchical MDS

In many cases the data is naturally organized in classes that have subclasses, that are composed of subsubclasses, and so on. Such a hierarchical classification can be obtained either externally or by applying data analysis techniques, such as clustering.

When the points to be embedded are pre-grouped into clusters, it is natural to treat the measured dissimilarities *between* clusters differently from those *within* a particular cluster. The task of finding a good global embedding splits into two subtasks: (a) finding a good embedding for each individual cluster, and (b) ensuring that these embedded clusters are well-placed with respect to each other. For a cluster that can be further divided into subclusters, step (a) can be performed recursively. For clusters that cannot be further divided, step (a) is carried out with ordinary metric or nonmetric MDS, or with the distributional scaling technique we will introduce in a subsequent section. We refer to this procedure as **hierarchical MDS**.

It remains to specify the details of step (b), the placement of embedded clusters with respect to each other. This is done by searching for a transformation that will minimize the overall stress, now considering all intercluster distances, as well as the intracluster distances that are already optimized. Clearly, clusters should be allowed to undergo arbitrary Euclidean transformations, as these do not increase their internal stress. The Euclidean transformations of $\mathcal{R}^p$ are parametrized by a p-vector $\mathbf{X}$ and an orthogonal $p \times p$ matrix M, and act on an arbitrary point $\mathbf{y}$ as $\mathcal{E}_{\mathbf{X},M}(\mathbf{y}) = M \cdot \mathbf{y} + \mathbf{X}$. We choose to allow, more generally, all affine transformations. The affine transformations are parametrized in the same way, except that M need not be orthogonal. The space of affine transformations is a linear subspace of the full search space, thus simplifying the search. Moreover, the space of affine transformations is *connected*, unlike the space of Euclidean transformations.

Formally, we are given a partitioning of the target points into K clusters, and an initial embedding that was carried out for each cluster individually. Let $\{\mathbf{y}_i\}$ be the initial coordinates of the points in cluster A. We stipulate that the final coordinates $\{\mathbf{x}_i\}$ are generated by *affine* transformations of these single-cluster embeddings, where each cluster is transformed independently. That is, the final coordinates of point $i \in A$ are given by

$$\mathbf{x}_i = \mathbf{X}_A + M_A \cdot \mathbf{y}_i$$

for some affine transformation $(\mathbf{X}_A, M_A)$. Our final embedding is generated by minimizing the overall metric stress, allowing only the $(\mathbf{X}, M)$ pairs to vary, while the base coordinates $\mathbf{y}_i$ are held fixed. That is, individual clusters can be rotated and translated with respect to each other, and stretched in a small number of ways; but they cannot be split into two or otherwise fundamentally reshaped.

Restricting the allowed configurations in this way reduces the number of degrees of freedom enormously. For instance, an arbitrary two-dimensional embedding of 100 points requires 200 parameters for its description, while an arbitrary affine transformation of a *known* two-dimensional embedding requires only 6. This reduction helps us in two ways. First, optimization within a subspace usually converges much faster simply because the search space is smaller. Second, we may be able to streamline the evaluation of the objective function $\mathcal{H}$ once we have fixed the coordinates $\mathbf{y}_i$. For SSTRESS, this can be done exactly, by rewriting the stress function in terms of the $\mathbf{X}_A$ and M_A variables. Specifically, when the final coordinates $\mathbf{x}_i$ are restricted to affine images of a known base embedding $\mathbf{y}_i$, the SSTRESS function becomes

$$\mathcal{H} \quad = \quad \sum_{i,j} w_{ij} \left(||\mathbf{x}_i - \mathbf{x}_j||^2 - \Delta_{ij}^2 \right)^2$$

$$
\begin{aligned}
&= \sum_{A,B} \sum_{i\in A,\, j\in B} w_{ij} \left(||\mathbf{X}_A - \mathbf{X}_B + M_A \cdot \mathbf{y}_i - M_B \cdot \mathbf{y}_j||^2 - \Delta_{ij}^2 \right)^2 \\
&= \sum_{A,B} \sum_{\alpha,\beta} \sum_{i\in A,\, j\in B} w_{ij} \left((M_A^T M_A)_{\alpha\beta} (\mathbf{y}_i)_\alpha (\mathbf{y}_i)_\beta + \ldots + ||\mathbf{X}_A - \mathbf{X}_B||^2 - \Delta_{ij}^2 \right)^2 \\
&= \sum_{A,B} \sum_{\alpha,\beta,\gamma,\delta} P^{(AB)}_{\alpha\beta\gamma\delta} (M_A^T M_A)_{\alpha\beta} (M_A^T M_A)_{\gamma\delta} + \ldots + W^{(AB)} \,, \qquad (3)
\end{aligned}
$$

where many terms are omitted for brevity. Partial sums over ij have been performed wherever possible, leading to parameters that can be computed in advance, such as

$$
\begin{aligned}
P^{(AB)}_{\alpha\beta\gamma\delta} &= \sum_{i\in A,\, j\in B} w_{ij} (\mathbf{y}_i)_\alpha (\mathbf{y}_i)_\beta (\mathbf{y}_i)_\gamma (\mathbf{y}_i)_\delta \,, \\
W^{(AB)} &= \sum_{i\in A,\, j\in B} w_{ij} \Delta_{ij}^4 \,,
\end{aligned}
$$

and so on. The rewritten SSTRESS function is a complicated expression, but it contains a relatively small number of terms. Specifically, for an embedding problem in p dimensions, involving K clusters with N points each, the new expression is a sum over $O(K^2 p^4)$ terms, while the original metric stress function has $O(K^2 N^2)$ terms. The upshot is that for large clusters, with $N \gg p^2$ points apiece, using Eq. (3) can save computational labor.

Most importantly, hierarchical MDS proved most effective for highly frustrated data, or when embedding high dimensional data in low dimension. In such cases direct embedding of the complete data set tends to diminish any high-order structure that exists in the data, while hierarchical MDS preserves more of the structure.

2.5 Introducing Distributional MDS

Metric MDS, as defined in the previous sections, works well in many cases. When the metric stress of an embedding is sufficiently low, one knows that all embedded edges are close to their target lengths, and hence the input data is well-represented by the final map. However, cases arise in which *no* embedding has an acceptably low level of stress.[1] In such cases, the precise *quantitative* structure of the input data is impossible to maintain, and the metric stress alone does not distinguish between qualitatively good and bad maps.

An illustrative example, which will serve as our motivation for introducing a new type of multi-dimensional scaling, is shown in Figure 1. It depicts an embedding of 600 points in two dimensions, generated by applying metric SSTRESS to synthetic, random proximity data.[2] The points were originally sampled from three clusters, such that the distances between clusters tend to be greater than those within clusters, as described in the figure caption. However, as seen in the figure, the process of embedding splits the central cluster into two well-separated subclusters. This is purely an artifact of the metric scaling process, as there is no inherent difference between the points in the two subclusters. Moreover, the partitioning into subclusters is not robust, but differs from run to run

1. The amount of acceptable stress will vary from application to application and also depends on the demands of the user.
2. Note that this data is nonmetric, since the triangle inequality does not hold, and that it is represented only by its proximity matrix. This kind of data arises naturally in cases where the objects are abstract or difficult to map to a vector space (e.g., strings, graphs, biological macromolecules, DNA and protein sequences).

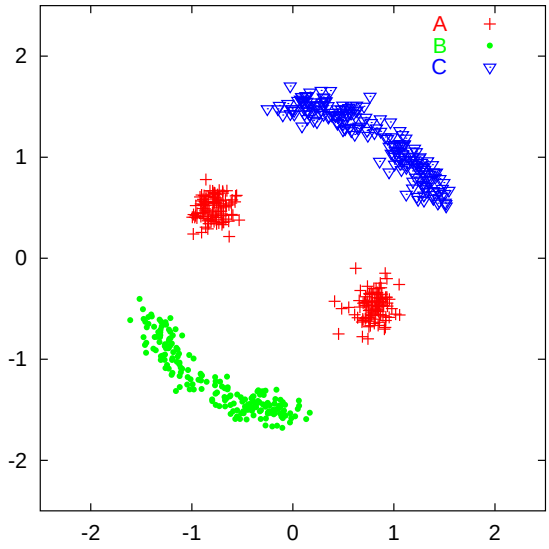


Figure 1: **Structural artifact generated by metric** SSTRESS**.** Three 200-point clusters (*A*, *B*, and *C*) were embedded in two dimensions using metric MDS and a synthetic dissimilarity matrix Δ. The central cluster (*A*) has been split into two apparent subclusters by the embedding process. To generate Δ, each target dissimilarity Δ_{ij} was drawn from one of three chi distributions. If i and j are in the same cluster, $\Delta_{ij} \sim \chi_2(1.0)$. If ij connects cluster *A* to cluster *B* or *C*, then $\Delta_{ij} \sim \chi_2(1.5)$. Finally, if ij connects clusters *B* and *C*, then $\Delta_{ij} \sim \chi_2(2.0)$.

when random starting configurations are used. Similar results can be obtained with the SAMMON criterion function, and with nonmetric MDS. This is a dramatic type of artifact, which we would like to automatically diagnose and avoid.

Our goal is to produce embeddings that preserve some notion of structure over the input space. The concept of geometry might not be clearly defined for the input space, and since the data set may be non-Euclidean or even nonmetric, it is hard to speak in general terms about the structure of the data. In our study we focus on the clustering properties of the data. The cluster structure reflects the existence of inherent order and the presence of groups and subgroups that usually can be mapped to specific subcategories of the data (for example, functional, topological, or demographic, depending on the data set). It is this notion of order that we would like to preserve. Thus, in our case, the definition of similar structures relies on the clustering profile of the data.

One way to characterize the underlying cluster structure of data is by studying the distribution of distances between and within clusters.[3] Although similar distributions do not guarantee that the embedding will have the same clustering profile, it reduces the search space to embeddings that are more likely to have the same structure. The simple example of Figure 1 demonstrates this point. Figure 2 shows histograms of the set of interpoint distances, both before and after the embedding process. From these graphs it is clear that the embedding has qualitatively altered the information present: although the target distances form a unimodal distribution, the post-embedding curve is distinctly bimodal. There is evidence that this kind of artifact is also prevalent in real applications of metric MDS (Yona, 1999).

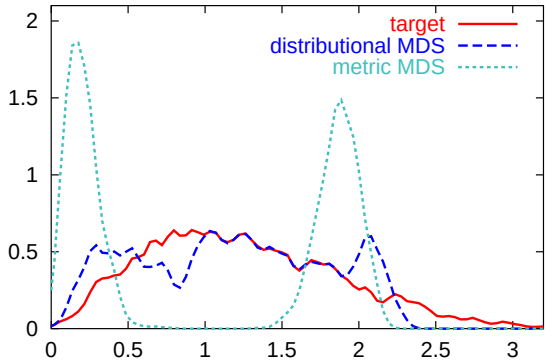


Figure 2: **Distribution of interpoint distances within a split cluster.** The three curves represent the distributions of the target distances Δ_{ij} (see the caption to Figure 1), the embedded distances D_{ij} from metric SSTRESS, and the embedded distances D_{ij} from our proposed distributional scaling. The two-dimensional embeddings from metric and distributional MDS are shown in Figure 1 and Figure 3, respectively.

To correct for artifacts of this type, and more generally to preserve the structural information we have just discussed, we propose a modified objective function that penalizes discrepancies like that shown in Figure 2. This new objective function can be used whenever cluster assignments are known, or can be estimated. For each pair of clusters, A and B, we define ρ_{AB} to be the (weighted, normalized) distribution of embedded distances between the points in cluster A and those in cluster B:

$$\rho_{AB}(x) = \frac{\sum_{i\in A}\sum_{j\in B} w_{ij}\delta(x - D_{ij})}{\sum_{i\in A}\sum_{j\in B} w_{ij}} \,. \tag{4}$$

Here $\delta(x)$ is the Dirac delta function, which describes a point mass of weight 1 localized at the origin. Similarly, we denote by $\tilde{\rho}_{AB}$ the distribution of the A–B target distances (the elements of Δ). Our proposed new objective function has the general form

$$\mathcal{H}_d(X) = (1-\alpha)\mathcal{H}(X) + \alpha \sum_{A\le B} W_{AB} D[\rho_{AB}, \tilde{\rho}_{AB}] \,, \tag{5}$$

3. Preserving just the cluster assignments, as is done by Roth et al. (2002), might miss higher order structure over clusters. Moreover, the method proposed by Roth is algorithm-dependent (tailored to the k-means algorithm).

where $D[p,q]$ is some measure of the dissimilarity between two distributions, the W_{AB} are relative weights of the target distributions, and α determines the balance between the original metric component of the stress and this new, distribution-related, component. We call the optimization of this type of objective function **distributional MDS**.

One could use any number of other measures to represent the data structure and its geometry. For example, cluster diameters, or the first and second moments of the sample points in each cluster, could be used in addition to the distributions of pairwise distances. The objective function could be modified to include these (or other, data-specific) order parameters. Rather than attempting to include all possible choices, we chose the single-parameter form of $\mathcal{H}_d$ given above. For the dissimilarity measure D, we will use the earth-mover's distance, a metric which is motivated and described in Section 2.6.

The weights W_{AB} are assigned based on the information content of the distributions. Specifically, we use the entropy

$$S_{AB} = S[\tilde{\rho}_{AB}] \equiv -\int dx\, \tilde{\rho}_{AB}(x) \log_2 \tilde{\rho}_{AB}(x)$$

as a measure for the information content of the target distribution, and we set $W_{AB} = 2^{-S_{AB}}$. Thus, the lower the entropy of a distribution, the more significant the contribution of that term to the objective function. Our motivation for this choice of weights is heuristic: high-entropy distributions are more likely to arise by chance, while low-entropy distributions are more likely to reflect a true pattern in the data. With robustness to classification errors in mind (see below), this weighting scheme attempts to minimize the sensitivity of the model to noise by emphasizing the low-entropy target distributions.

It is important to note that the availability of cluster information is by no means a hurdle or a limiting factor of this algorithm. One can use any sensible clustering algorithm (e.g., k-means), applied to the original data or to its metric embedding, to suggest a preliminary classification. If the data is sufficiently ordered, this clustering profile can provide a rough snapshot of the geometry, the quality of which depends on the clustering algorithm and the data set.[4] This clustering profile can then be used to guide the embedding process, even if it is not completely accurate. Since the distributions between all pairs of clusters are considered, the algorithm avoids embeddings that grossly distort the cluster structure, even when the higher-order structure of the data is misrepresented (e.g., when a real cluster is split into two by a clustering algorithm).

To demonstrate, we return to the previous example, supposing now that the true cluster assignments are unknown. Applying k-means clustering (with both $k = 4$ and $k = 3$) to the metric embedding (Figure 1) produces the tentative classifications shown in Figure 4. The k-means results exhibit both **overclassification**, where a single true cluster is broken into two classes, and **misclassification**, where parts of two true clusters are combined in a single class. Applying distributional scaling to the original dissimilarities, using the *tentative* cluster assignments rather than the true ones, produces the improved embeddings shown in Figure 5. Both resemble the embedding in Figure 3, which was generated using the true assignments. This is a satisfying result, since it indicates that our algorithm is robust with respect to at least some classification errors. In general, the prob-

4. Given the choice between a conservative clustering and a more permissive one (e.g., hierarchical clustering with different thresholds), one might prefer the conservative algorithm. Note that here we ignore issues of generalization and model validity of the clustering profile, as they are irrelevant at this point. Opting for structure-preserving embeddings, smaller and more compact clusters can be considered as entities of high confidence and are more amenable to undergo this process successfully.

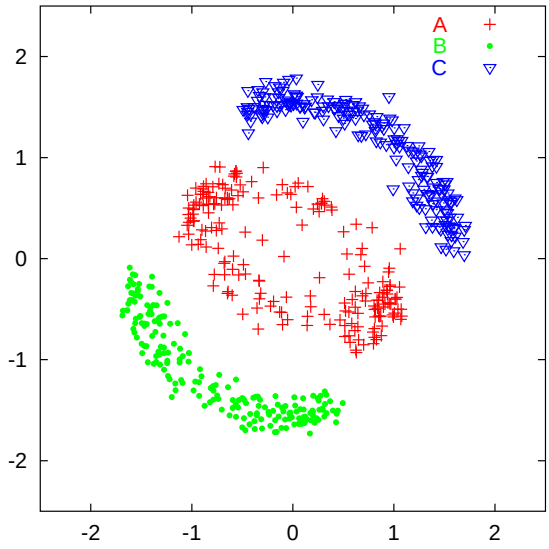


Figure 3: **Improved map from distributional scaling.** Starting from the metric embedding, the objective function defined by Eq. (5) (with $\alpha = 0.1$) was numerically optimized. The artifact seen in Figure 1 is largely corrected: cluster *A* now appears as a single cluster, as it should.

lem of overclassification is well-corrected by our algorithm. The problem of misclassification is not addressed as well; but in cases like the example, where intercluster and intracluster distances have substantially different distributions, distributional MDS gives a more reliable picture of the actual data than metric MDS alone.

2.6 The Earth-Mover's Distance Between Probability Distributions

There are several common measures to assess the statistical similarity of probability distributions, among which are the Manhattan distance (the L_1 norm) and the KL divergence (Kullback, 1959). Our first choice was the information-theoretic Jensen-Shannon divergence measure (Lin, 1991), which is a symmetric and bounded variant of the KL divergence. Formally, given two (empirical) probability distributions **p** and **q**, for every $0 \leq \lambda \leq 1$, the **λ-JS divergence** is defined as

$$D_\lambda^{JS}[\mathbf{p}||\mathbf{q}] = \lambda D^{KL}[\mathbf{p}||\mathbf{r}] + (1-\lambda) D^{KL}[\mathbf{q}||\mathbf{r}] \,,$$

where $D^{KL}[\mathbf{p}||\mathbf{q}] = \sum_i p_i \log_2(p_i/q_i)$ is the KL divergence, and $\mathbf{r} = \lambda\mathbf{p} + (1-\lambda)\mathbf{q}$ can be considered as the most likely common source distribution of both distributions **p** and **q**, with λ as a prior weight. The parameter λ reflects the *a priori* information and is set by default to 0.5.

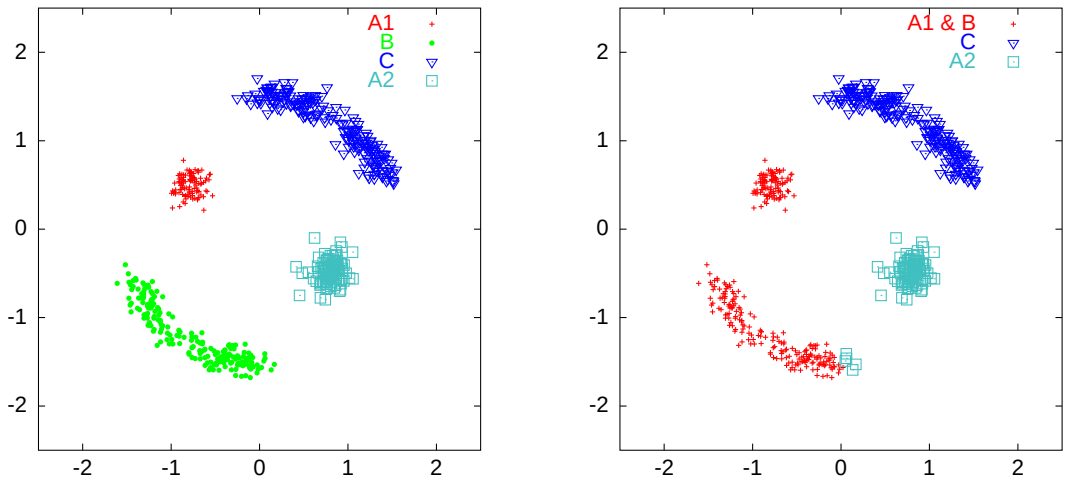


Figure 4: **Naive cluster assignments** generated by the application of k-means clustering ($k = 4$, left; $k = 3$, right) to the points in Figure 1. Note that the true cluster assignments were never used. The $k = 4$ example shows **overclassification**, where a single true cluster is broken into two classes. The $k = 3$ example shows **misclassification**, where parts of two true clusters are combined in a single class.

Despite its attractive properties as a measure of statistical similarity,[5] we learned quite early on that this measure is inappropriate when attempting to preserve the overall shape of the distribution. Specifically, this measure was found to be difficult to optimize through a local search. Since the Jensen-Shannon distance is a purely local measure of the difference between two distributions, a JS-based algorithm is easily trapped in poor local minima.

A more effective measure of dissimilarity between two distributions is the earth-mover's distance (EMD) (Rubner et al., 1998). As shown in Figure 6, the EMD is substantially easier to minimize than the Jensen-Shannon divergence. Given two probability distributions p and q over the interval $[0,K]$ (which can be thought of as distributions of "earth" and "holes" respectively), the EMD between p and q can be defined by means of the following transport or bipartite-graph flow problem. Let $f(x,y)$ be the amount of earth (flow) carried from $x \in [0,K]$ to $y \in [0,K]$, such that every hole is filled and no new holes are dug. In other words, $f(x,y)$ is a flow function that should satisfy

$$\begin{aligned} f(x,y) &\geq 0\,, \\ p(x) &= \int_0^K dy\, f(x,y)\,, \end{aligned}$$

5. Besides being bounded and symmetric, it has been shown that the JS divergence measure is proportional to minus the logarithm of the probability that the two empirical distributions represent samples drawn from the same ("common") source distribution (El-Yaniv et al., 1998).

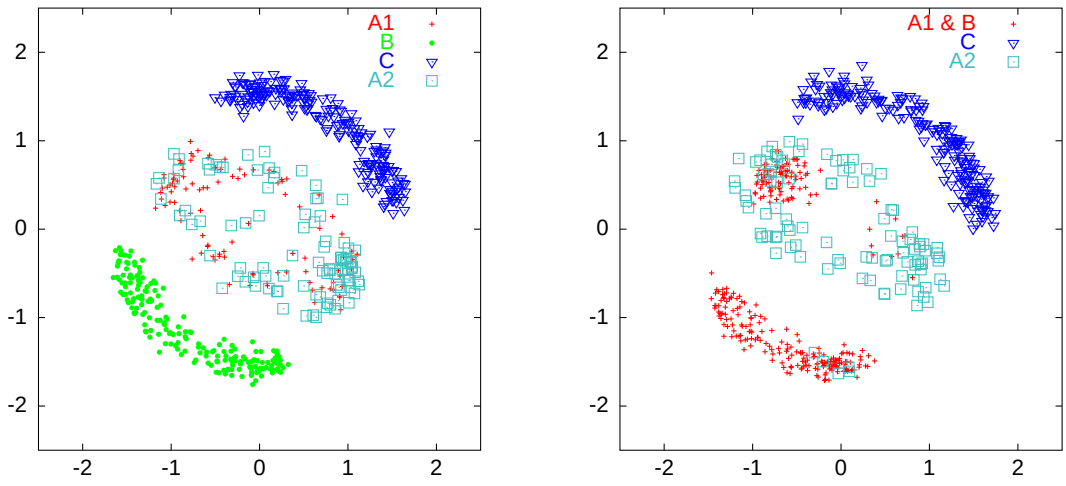


Figure 5: **Distributional scaling with naive cluster assignments.** The figure was generated in the same way as Figure 3, except that the *k*-means cluster assignments (from Figure 4) were used in place of the true ones. In both examples, the process merges the two central groups of points, while keeping them separate from the remaining two groups.

$$q(y) \quad = \quad \int_0^K dx f(x,y) \; .$$

Let $dist(x,y)$ be the "ground distance" between x and y. (In our case, $dist(x,y) = |x-y|$.) Then the EMD is the *minimum* total distance traveled by the earth,

$$\text{EMD}\,[p,q] = \min_f \int dx \int dy\, dist(x,y)\, f(x,y) \; ,$$

subject to the given constraints on f. Intuitively, the EMD can be considered as the minimal amount of work required to match p with q. It can be shown that the EMD between normalized one-dimensional distributions is the same as the L_1 distance between their *cumulative* distribution functions (Levina and Bickel, 2001). That is, the earth-mover's distance between distributions p and q is just

$$\text{EMD}\,[p,q] = \int_0^K dx \left| \int_0^x dy (p(y) - q(y)) \right| \; .$$

The result follows from the fact that there is a greedy algorithm for finding the minimal flow in one dimension (only): fill the leftmost unfilled hole with the leftmost available dirt until all holes are filled. This expression is essential for our algorithm, since it makes the EMD simple to calculate and differentiate, rendering it suitable for inclusion in the stress function.

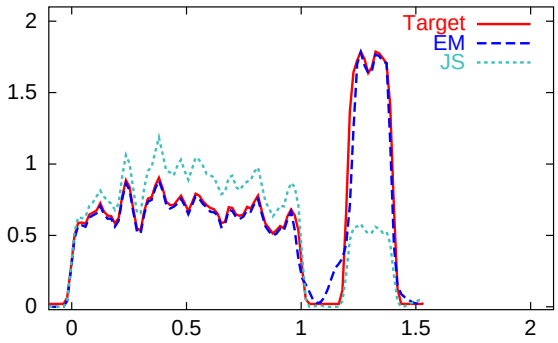


Figure 6: **Comparison of EMD-based algorithm with Jensen-Shannon algorithm.** The elements of a 200×200 dissimilarity matrix were drawn from a bimodal distribution ("Target"). Downhill search was used to find a two-dimensional embedding with interpoint distance distribution closest to the target distribution, under the EMD and JS measures. The JS-based algorithm became trapped in a local minimum: shifting weight to the right does not immediately decrease the Jensen-Shannon distance between the target and JS curves. The EMD-based algorithm, on the other hand, reproduced the target distribution accurately.

2.6.1 IMPLEMENTATION ISSUES

The naive implementation of the algorithm is impractical for large data sets, because building and storing the exact, discrete distribution defined by Eq. (4) takes a large amount of space, $O(n^2)$, and calculating the EMD between two such distributions takes $O(n^2 \log n)$ time.[6] Moreover, the earth-mover's distance between two such distributions has many nondifferentiable points along any given line, which is problematic for our (gradient-based) optimization strategy. We address both these issues by using an approximate distribution in place of the exact one.

Our approximate distributions are piecewise constant, consisting of a relatively small number of disjoint bins. We associate the k-th bin with the interval $[x_k, x_{k+1}]$. To build the necessary distribution, each delta-function in Eq. (4) is first broadened into a finite-width shape with the correct total weight. That is, $a\delta(x-b) \rightarrow ah(x-b)$, where h is a smooth function. The weight is then distributed among the relevant bins: the k-th bin is incremented by $a \int_{x_k}^{x_{k+1}} h(x-b)dx$ (see Figure 7). The result is a histogram whose bin contents are differentiable functions of the distances D_{ij}. We use this histogram in our calculation of the earth-mover's distance; using the chain rule, the EMD is then differentiable as well.[7]

6. The rate-limiting step is the sorting of the values D_{ij}, which is needed to find the cumulative distribution function.
7. More precisely, the EMD still has nondifferentiable points, but they are sparse enough that there are none on a typical ray in the search space.

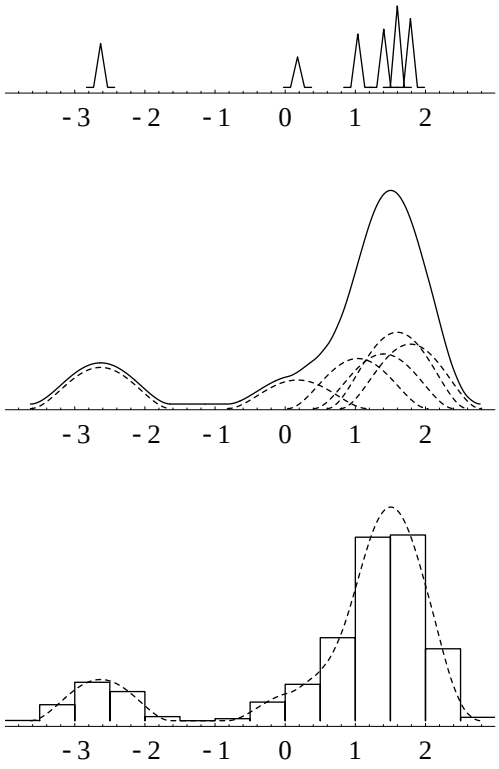


Figure 7: **Histogram construction.** To construct a histogram from a discrete distribution (top), we first broaden each point by a smooth window function (middle). The integrated weights are then used as the bin counts for the histogram (bottom).

2.7 Choosing the Initial Embedding

Since our optimization method is iterative, beginning with a low-stress embedding can save time and, potentially, improve the final result. We suggest two inexpensive ways to generate a reasonably good initial configuration.

The first method is principal component analysis (PCA). Principal component analysis is well known as the basis for classical scaling (Young and Householder, 1938; Gower, 1966); given a data set with a low-stress embedding, PCA can be used to find a good configuration very quickly. To find the principal components we first form the auxiliary matrix M, with matrix elements

$$M_{ij} = -\frac{1}{2}\Delta_{ij}^2 + \frac{1}{2n}\sum_k \left(\Delta_{ik}^2 + \Delta_{jk}^2\right) - \frac{1}{2n^2}\sum_{k,l}\Delta_{kl}^2 \,. \tag{6}$$

To generate an embedding in p dimensions we compute the p largest eigenvalues of M, together with their associated eigenvectors (λ_a and $\mathbf{u}_a$). Finally, we form an initial configuration with coordinate

components

$$(\mathbf{x}_i)_a = \sqrt{\lambda_a}(\mathbf{u}_a)_i$$

for $a = 1, \ldots p$. If Δ is, in fact, a distance matrix $D(X)$ with $X \in S_n^p$, then M will have only p nonzero eigenvalues, and this initial configuration will have zero stress. If Δ is a higher-dimensional distance matrix, this configuration will represent an optimal linear projection into p dimensions.

This analysis could be carried out by fully diagonalizing M, but this is extremely wasteful when only $p \ll n$ principal eigenvectors are wanted. Instead, we use a simple iterative method based on Hotelling's power method (Hotelling, 1933). Start with a random orthonormal set of p vectors $\mathbf{e}_a$. Multiply each by the matrix M, then orthonormalize the set using the Gram-Schmidt algorithm. As this step is repeated, the vectors $\mathbf{e}_a$ approach the p largest eigenvectors.[8] The eigenvalues are then given by $\lambda_a = \mathbf{e}_a^T \cdot M \cdot \mathbf{e}_a$. Exact diagonalization is known to take $O(n^3)$ time; this method cuts the time down to $O(pn^2)$.

The second method we use for finding a good initial embedding is the **stochastic embedding** algorithm proposed by Agrafiotis and Xu (2002). This method is also very simple and fast, and seems to work well when the data is sufficiently compatible with the embedding space. The algorithm begins with a random configuration. A random edge ij is selected, and the points $\mathbf{x}_i$ and $\mathbf{x}_j$, currently separated by a distance d_{ij}, are moved along the line connecting them so their separation becomes $\alpha\Delta_{ij} + (1-\alpha)d_{ij}$. This basic step is repeated many times, while the learning rate α decreases according to a specified schedule.

2.8 Choosing the Embedding Dimension

One of the major problems with embedding algorithms is determining the intrinsic dimensionality of the data. When the dimension of the host space is increased, the optimal metric stress will always *decrease*, as the search space is enlarged. One would like to know when the embedding dimension is sufficiently large, i.e., when any additional improvement is insignificant. Principal component analysis can sometimes suggest the appropriate embedding dimension, based on the number of "large" eigenvalues of the PCA matrix M (Eq. (6)). However, in many cases the distribution of eigenvalues is relatively flat and uninformative and the subtlety then lies in setting the correct eigenvalue threshold. To our knowledge, this has not been addressed in a statistical setting. Moreover, as a linear embedding technique, PCA explores only a small subset of all possible embeddings.

We propose two complementary approaches to this question. The first method is based on a geometric analysis of the optimization problem in the space of distance matrices, and formulates the problem in probabilistic terms. It can be used to decide whether a dimensional increase is statistically significant. This method is tailored to the case of unweighted SAMMON with small distortions. The second method, on the other hand, is information-theoretic in nature, and compares embeddings based on the principle of minimum description length (MDL). This method is more heuristic than the first, and consequently more widely applicable. In the remainder of this section, we discuss both proposed methods in detail.

2.8.1 GEOMETRIC APPROACH

In practice, one often seeks the correct embedding dimension by an iterative method: successive embeddings with decreasing metric stress are constructed, in higher and higher dimensions, until

8. Because the result will be further refined in any case, full convergence is not required.

the decrease in stress becomes negligible. We can place this iterative method on a firm statistical footing by specifying precisely what is meant by "negligible". We do this by defining a statistical null model for the decrease in stress associated with an increase in embedding dimension. For any p-dimensional embedding of a dissimilarity matrix Δ with (locally) minimum stress, our null model proposes that the remaining discrepancies between the target distances Δ_{ij} and the embedded distances are *independent* and *identically distributed* Gaussian random variables. By comparing the measured stress in a dimension $q > p$ to the stress predicted under the null model, we can assign statistical significance to the decrease in stress. When the statistical significance becomes too low, we conclude that we may well be "fitting noise," and terminate the iterative method. The details of this calculation comprise the remainder of this section.

Given a set of n points, we denote the set of all possible embeddings in p dimensions by S_n^p. The corresponding distance matrices form the manifold $D_n^p \equiv D(S_n^p) \subset \Omega_n$. This manifold is enlarged with increasing p until $p = n-1$; that is,

$$D_n^1 \subset D_n^2 \subset \cdots \subset D_n^{n-1} \equiv D_n^\infty \subset \Omega_n \ .$$

Since n points always lie in a single $(n-1)$-plane, larger values of p are never necessary. The dimension of D_n^p is the dimension of S_n^p, minus the dimension of the group of Euclidean transformations of $\mathcal{R}^p$ (i.e., the transformations $(\mathbf{X}, \hat{M})$ under which D is invariant):

$$\begin{aligned} \dim D_n^p &= \dim S_n^p - p - \frac{1}{2}p(p-1) \\ &= n \cdot p - \frac{1}{2}p(p+1) \\ &= \frac{1}{2}p\,(2n-p-1) \ . \end{aligned}$$

Equivalently, the codimension of D_n^p is

$$\begin{aligned} c_n^p &\equiv \dim \Omega_n - \dim D_n^p \\ &= \frac{1}{2}n(n-1) - \frac{1}{2}p(2n-p-1) \\ &= \frac{1}{2}(n-p)(n-p-1) \ , \end{aligned}$$

which is equal to zero when $p = n-1$, as expected.

Suppose we have found an optimal embedding $X \in S_n^p$ in p dimensions, with metric stress equal to $s(p)$. In the case of unweighted SAMMON, the stress function is simply the squared Euclidean distance between Δ and the distance matrix $D(X)$ within the encompassing space of Ω_n:

$$s(p) = ||D(X) - \Delta||^2 \ .$$

If X is a p-dimensional stress minimizer, then $D(X)$ is (locally) the closest point to Δ in D_n^p, and the **error vector** $E_p(X) \equiv \Delta - D(X)$ is perpendicular to D_n^p at that point. In other words, $E_p(X)$ lives in a space with dimension c_n^p (the codimension of D_n^p).

Assume now that we look for a q-dimensional stress minimizer ($q > p$). Starting at X, the search manifold is extended to D_n^q, adding $\dim D_n^q - \dim D_n^p = c_n^p - c_n^q$ new directions. If $E_p(X)$ is small, then a q-dimensional minimizer can be found by moving $D(X)$ so these (now unconstrained)

components of $E_p(X)$ become zero. This will lead to a new error vector $E_q(X)$ with a lower stress value $s(q)$. Note that $s(q) = \sum_i E_i^2 < \sum_j E_j^2 = s(p)$, where the second sum is over all c_n^p components of $E(X)$, while the first sum is over a particular subset of c_n^q components.

At this point we ask whether the reduction in the error is significant, i.e., greater than expected by chance alone. Our null hypothesis is that the error vector is *randomly* oriented within the space perpendicular to D_n^p at $D(X)$. That is, we hypothesize that $E_p(X)$ is given by

$$\hat{E}_p(X) = \frac{(e_1, e_2, ..., e_{c_n^p})}{\sqrt{e_1^2 + e_2^2 + ..e_{c_n^p}^2}} \sqrt{s(p)} \; ,$$

where the e_i are normally distributed with zero mean and unit variance. Setting the first c_n^q coordinate axes in this space to be those that are *also* perpendicular to D_n^q, the projection of this random vector onto the subspace where $E_q(X)$ resides is

$$\hat{E}_q(X) = \frac{(e_1, e_2, ..., e_{c_n^q}, 0, 0, ..0)}{\sqrt{e_1^2 + e_2^2 + ..e_{c_n^p}^2}} \sqrt{s(p)} \; .$$

The **random stress ratio** is therefore

$$\hat{F} \equiv \frac{\hat{s}(q)}{\hat{s}(p)} = \frac{||\hat{E}_q(X)||}{||\hat{E}_p(X)||} = \frac{\sum_{i=1}^{c_n^q} e_i^2}{\sum_{j=1}^{c_n^p} e_j^2} < 1 \; .$$

This can be rewritten as

$$\hat{F} = \frac{A}{A+B} \; ,$$

where $A = \sum_{i=1}^{c_n^q} e_i^2$ and $B = \sum_{c_n^q+1}^{c_n^p} e_i^2$. Note that A and B are two independent chi-squared random variables, with a and b degrees of freedom, where

$$\begin{aligned} a = c_n^q &= \frac{1}{2}(n-q)(n-q-1) \; , \\ b = c_n^p - c_n^q &= \frac{1}{2}(q-p)(2n-p-q-1) \; . \end{aligned}$$

Given an observed stress ratio of $F = 1/(1+\varepsilon)$, we are interested in the probability that $\hat{F} \leq F$, or (equivalently) that $\varepsilon A - B < 0$. Since the distributions of A and B are known, the significance can be calculated *exactly*. However, when $p, q \ll n$, as is often the case, it is useful to approximate the significance in terms of the normal distribution, so that tabulated Z-scores may be used. Specifically, as a and b become large, A and B approach normal variables: $A \sim N(a, \sqrt{2a})$ and $B \sim N(b, \sqrt{2b})$. Therefore, the difference $\varepsilon A - B$ is distributed as

$$\begin{aligned} x = \varepsilon A - B &\sim N(\varepsilon a, \varepsilon\sqrt{2a}) - N(b, \sqrt{2b}) \\ &\sim N(\varepsilon a - b, \sqrt{2}\sqrt{\varepsilon^2 a + b}) \\ &\equiv N(\mu_x, \sigma_x) \; . \end{aligned}$$

With the scaling $z = (x - \mu_x)/\sigma_x$, the distribution is transformed to a standard normal distribution, and

$$P(\varepsilon A - B < 0) = P\left(z > \frac{\varepsilon a - b}{\sqrt{2}\sqrt{\varepsilon^2 a + b}} \right) \; ,$$

where $z \sim N(0,1)$. The probability is $1/2$ when $\varepsilon = b/a$. For the probability to be significant (say, three standard deviations away from the mean, or smaller than $P(z > 3)$), we need to have ε greater than $(b + 3\sqrt{2b})/a$.

In summary, we have derived the significance (p-value) of a given stress ratio, based on a postulated background distribution. This p-value can be calculated exactly, or approximated in terms of the normal distribution. If the p-value associated with an increase in embedding dimension is sufficiently low, then the decrease in stress is significant, and the higher-dimensional embedding describes the data (Δ) significantly better than the lower-dimensional one. In this framework, the optimal embedding dimension has been found when an increase in dimension fails to significantly decrease the metric stress.

2.8.2 INFORMATION-THEORETIC APPROACH

An alternative method for model selection is the minimum description length (MDL) approach. The description length of a given **model (hypothesis) and data** is defined as the description length of the **model** plus the description length of the **data given the model**. In our case, we are trying to represent proximity data (Δ) in terms of the pairwise distances from a p-dimensional embedding. The model is a specific embedding $X \in S_n^p$. Given the model, the data can be reconstructed from the pairwise distortions: for concreteness, we will use the relative distortions

$$E_{ij} = \frac{D_{ij}(X) - \Delta_{ij}}{\Delta_{ij}} \ .$$

According to the MDL principle, we should select the model that minimizes the total description length. This heuristic favors low-dimensional models (short model description) that are capable of providing a fairly accurate description of the data (short description of the remaining errors, given the model).

The model is a specific embedding with the set of positions $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ in Euclidean space $\mathcal{R}^p$, for a total of $n \cdot p$ independent coordinates. Since the coordinates are not explicitly statistics of the data (we do not have an explicit mapping from Δ to X, but rather determine X implicitly, through optimization), it is difficult to specify the uncertainty in each coordinate, which could be used to define the description length. However, indirectly, they do summarize some global information about the data and in that sense they can be perceived as statistics. One can then estimate the uncertainty from the gradient curve in the vicinity of the point, or from the overall distortion in pairwise distances associated with it. For simplicity we assume a constant uncertainty for all coordinates, so the description length of the model is proportional to $n \cdot p$.

The description of the data given the model depends on the set of $n(n-1)/2$ pairwise distortions. Because this represents a large number of samples, the description length per sample will approach the information-theoretic lower bound, which is related to the entropy of the underlying distribution. For a continuous probability distribution $p(x)$, the entropy is $S[p] = -\int dx\, p(x) \log_2 p(x)$. According to Shannon's theorem, to encode a stream of samples from this distribution, with errors bounded by $\varepsilon/2$ (which must be small), one needs $-\log_2 \varepsilon + S[p]$ bits per sample. In our case, we must estimate the underlying distribution from the empirically measured distortions E_{ij}, which can be done along the lines of Section 2.6.1.

Combining the two terms, we suggest a scoring function of the form

$$\alpha n \cdot p + \frac{1}{2} n(n-1) S_E \ , \tag{7}$$

where S_E is the entropy of the error distribution, and the scaling parameter α represents the description length per coordinate of the model. We have dropped the constant term involving $-\log_2 \varepsilon$: since this term is independent of p and X, it plays no role when comparing different models.

Initially, we intended to train the parameter α to optimize the scoring function's performance; for instance, one might seek the α that most often assigns noisy data to its original dimension. However, upon reflection it is clear that the MDL method should not, in fact, be coerced into this behavior. The purpose of the method is to find the shortest encoding of the data, and often this will *not* coincide with the data's original dimensionality. For noisy and high-dimensional data in particular, the error distribution will never become very narrow, so the description length of the conditional data cannot become arbitrarily short, while each additional coordinate costs the same amount. Unless α is unreasonably small (say, less than 2 bits per coordinate), the MDL heuristic will select a lower dimension than it would for the denoised data. This behavior is acceptable and even informative. Therefore, we selected a somewhat arbitrary value of $\alpha = 10$ for use in Section 3.2, corresponding to a relative precision of 0.001, with the understanding that values anywhere from 5 to 50 would also be reasonable.

3. Test Data and Results

To test our algorithm we ran several tests. The first set of experiments tested the robustness and performance of different metric objective functions. The second set tested our method for determining the embedding dimension. Next we evaluate structural preservation when using our algorithm compared to MDS. Lastly, we test and compare the performance of our algorithm on handwriting data.

3.1 Comparison of Metric Objective Functions

We created sixteen random configurations of 1200 points in two and three dimensions (8 sets in 2d, 8 sets in 3d). Each configuration consisted of twelve gaussian clusters of 100 points each, with principal standard deviations between 0.2 and 1.0, and with intercluster separations between 1.0 and 8.0. A test distance matrix was generated from each configuration.

Our first experiment tested the robustness of metric MDS in the presence of noise, to see how frequently the algorithm failed to converge to the global minimum. Using our algorithm for metric SSTRESS with intermediate weighting, we embedded the test matrices 100 times each (from random initial configurations), both without noise and with multiplicative noise of strength 0.02, 0.1, or 0.5. The data sets were embedded in their original dimensions.

For the 2d→2d tests, the algorithm converged to the global minimum 100% of the time, for each test matrix and for each level of noise. For the 3d→3d tests, the algorithm found the global minimum 100% of the time for seven of the eight test matrices. On the eighth test matrix, the global minimum was found 70–80% of the time, depending on the noise; in the remaining trials, a single nonglobal minimizing configuration, with a low stress of 0.01–0.02, was found.

Our second experiment compared the performance of the various objective functions, and used the same test matrices, but truncated to 400 points. We embedded the distance matrices from 3d→3d, 100 times each (from random initial configurations), with no noise, using SSTRESS with intermediate and global weighting and SAMMON with intermediate and global weighting. The number of times each objective function converged to the global minimum is shown in Table 1. The results suggest that SAMMON is more liable to converge to a nonglobal minimizer than SSTRESS,

as noted by other authors. They also indicate that global weighting, which emphasizes the importance of large target distances over small ones, is more successful than intermediate weighting at recovering the original configuration.

ID	SSTRESS-i	SSTRESS-g	SAMMON-i	SAMMON-g
1	100	100	71	100
2	100	100	100	100
3	53	100	35	34
4	87	99	25	52
5	47	100	12	13
6	49	100	38	54
7	18	82	16	25
8	100	100	39	100

Table 1: Percentage of successful trials for 3d→3d embedding, for eight 400-point test sets and four different objective functions.

3.2 Dimensionality Selection

We created twenty random configurations of 250 points in 2, 3, 5, 10, and 50 dimensions. (Four for each dimensionality: a single gaussian cluster, a closely spaced pair of clusters, a widely spaced pair of clusters, and a set of eight scattered clusters.) We then generated five dissimilarity matrices from each of these configurations, using five different metrics, for a total of one hundred test matrices. The metrics we used were:

1. Euc = Euclidean metric, $\rho(\mathbf{x},\mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$,
2. EucW = Euclidean metric plus weak multiplicative noise,
3. EucS = Euclidean metric plus strong multiplicative noise,
4. Mink = Minkowski metric, $\rho(\mathbf{x},\mathbf{y}) = (\sum_i |x_i - y_i|^{3/2})^{2/3}$,
5. Manh = Manhattan metric, $\rho(\mathbf{x},\mathbf{y}) = \sum_i |x_i - y_i|$.

In this set of tests, we embedded each test matrix 10 times each (from random initial configurations) in 2, 3, 4, 5, and 10 dimensions, using SAMMON with global weighting. We took the lowest stress from each set of 10 trials, and retained the corresponding embedding.

From the stresses, we calculated the statistical significance of the dimensional transitions $2 \rightarrow 3$, $3 \rightarrow 4$, $4 \rightarrow 5$, and $5 \rightarrow 10$, as described in Section 2.8.1 (geometric approach). These significances were used to determine the best embedding dimension for each data set. An increase in dimension was considered justified if it improved the stress at the 3σ-level ($P < 0.0025$, approximately). From the final embeddings, we calculated the entropy of the distribution of errors for each embedding dimension, as described in Section 2.8.2 (information-theoretic approach). Using the measured entropies, we selected the dimensionality that minimized the MDL-based scoring function given by Eq. (7).

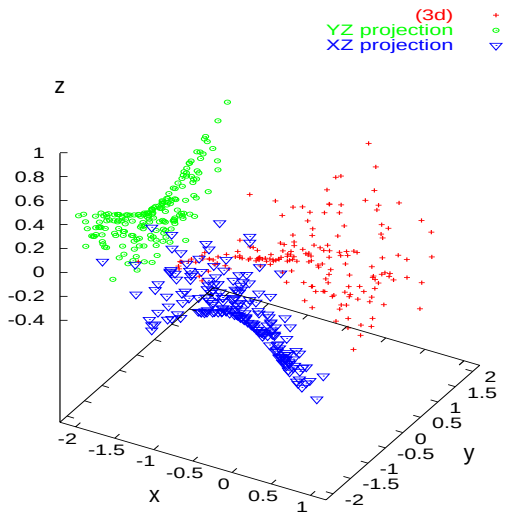


Figure 8: **Embedding a non-Euclidean space.** The dissimilarity matrix was created by applying a Minkowskian metric to a two-dimensional gaussian distribution of points. After metric embedding in 3d, the points appear to form a two-dimensional surface with negative curvature, i.e., a saddle.

Tables 2 and 3 summarize the geometric and information-theoretic results. The results were fairly consistent across the four types of test configuration (gaussian, pair, etc.). On the other hand, they depended strongly on the dimensionality of the original configuration and on the way in which dissimilarities were obtained, as seen in the Tables. Moreover, the geometric and information-theoretic approaches can lead to very different results when applied to noisy or nonmetric data.

Applying the geometric approach, our algorithm selected the original dimensionality of the data set in the Euclidean cases, both with and without noise, and indicated that higher-dimensional embeddings were significantly better at describing the Minkowski- and Manhattan-metric test sets. The results for the noisy data are not surprising; indeed, the method is designed to select the correct dimensionality for data with additive gaussian noise. For the non-Euclidean test sets, the results suggest that when a Minkowskian metric is imposed on a low-dimensional set of points, the points tend to "curl up" into a higher dimension. Visual inspection of test embeddings tends to support this idea: for instance, Figure 8 shows a 2d→3d example using the Minkowski metric in which the embedded points have formed a saddle-shaped surface.

The information-theoretic approach often proposed a *lower* embedding dimension than the geometric approach. This can best be understood by comparing the goals of the two approaches with respect to residual errors. The geometric method tries to increase the embedding dimension until the residual errors are effectively *random*, and as much information as possible has been packed into the model. On the other hand, the MDL-based method will increase the embedding dimension until a balance is struck between the residual errors and the model, such that the total description length

	Euc	EucW	EucS	Mink	Manh
$d = 2$	2	2	2	3-4	3-4
$d = 3$	3	3	3	5	5
$d = 5$	5	5	5	10	10
$d = 10$	10	10	10	10	10

Table 2: **Best embedding dimension: geometric approach.**

	Euc	EucW	EucS	Mink	Manh
$d = 2$	2	2	2	3	3
$d = 3$	3	3	3	3-4	3-4
$d = 5$	5	5	3-4	5	5
$d = 10$	10	5	3	10	10

Table 3: **Best embedding dimension: information-theoretic approach.**

is minimized. This compromise will often leave significant information in the residual errors; in any such case, the geometric approach will propose a higher dimensionality for the data.

3.3 Structural Preservation

To assess the efficacy of our distributional scaling method in preserving the structure of input data, we used the distributional method to re-embed the 100 test matrices from the previous section, in each case starting from the optimal metric embedding. For each test matrix, we calculated six different measures of structural fidelity, before and after the re-embedding. The first measure was the metric SSTRESS, which was expected to increase. The remaining measures were of the form $\sum_{A \leq B} W_{AB} D[\rho_{AB}, \tilde{\rho}_{AB}]$, where $D[p,q]$ was one of the following:

1. EMD = Earth-mover's distance,
2. JS = Jensen-Shannon distance,
3. mean = squared difference between the means of p and q,
4. max = squared difference between the maxima,
5. variance = squared difference between the variances.

Table 4 shows the percent change in each of these measures, averaged over the 100 test sets, for various embedding dimensions.

These results pertain to low-stress (metric SSTRESS ≤ 0.05) embeddings, where the agreement between distributions is rather good even without the improvements from our method. When embedded with the distributional scaling method we observe a modest increase in metric stress. However, this is compensated on average by substantial improvements in the other measures. Moreover, while we explicitly optimize only the EMD, the changes in the other measures are correlated.

For highly frustrated data, like the motivating example of Section 2.5 (shown in Figure 1, with metric SSTRESS ~ 0.5), the numbers are more dramatic. The bottom row of Table 4 shows the

Dim.	stress	EMD	JS	mean	max	variance
2	+29%	-45%	-50%	-34%	-17%	+22%
3	+27%	-59%	-61%	-44%	-24%	-24%
4	+23%	-70%	-69%	-48%	-37%	-60%
5	+20%	-76%	-74%	-47%	-45%	-83%
10	+19%	-87%	-86%	-81%	-46%	-81%
2^*	+1.7%	-49%	-60%	-79%	-17%	-22%

Table 4: **Change in six measures of structural fidelity when distributional scaling is applied.** The last data set (2^*) is the one from Figure 1.

changes in the same six measures during that example's re-embedding in $d = 2$. Here, the improvements in structural fidelity cause only a very small increase in metric stress. Our method is perhaps best suited to this type of example, where no low-stress embedding exists. In such cases, distributional MDS distinguishes among many candidate embeddings where metric MDS cannot, and selects a candidate that is faithful to the structure of the original data.

3.4 Handwriting Data

Finally, to test our algorithms on real-world data, we applied both metric and distributional SSTRESS to a subset of the MNIST database of handwritten digits.[9] Each digit is represented by a 28×28 grayscale image, where each pixel's brightness is between 0 and 255; we used the Euclidean distances between these 784-dimensional data points as input to our embedding algorithms. Figure 9 shows the two-dimensional embeddings that were generated using each method. We restrict this example to three digits, because we expect to need more than two dimensions to embed all ten digits (Saul and Roweis, 2003), making the results harder to interpret.

The general layout is similar with both methods: the digit 2 is most readily confused with the other two digits, and digits 0 and 1 are most easily distinguished from one another. However, the application of distributional scaling (right) clearly improves the embedding, in that the overlap between clusters is greatly reduced. This result suggests that the *distributions* of intercluster distances provide additional information distinguishing the handwritten digits from one another.

4. Discussion

In this paper we presented a method for structure-preserving embedding. As opposed to classical multidimensional scaling methods that are concerned only with the pairwise distances, our algorithm also monitors any higher-order structure that might exist in the data and attempts to preserve it as well.

There are many ways to characterize the structure of the data. If the data resides in a real normed space one can talk about its geometry. However, embedding is more interesting when the data is given as proximity data, where it may or may not be metric. The notion of geometry in these cases is elusive. Here we decided to focus on the clustering profile that is implied by the data. The

9. The MNIST data is available at http://yann.lecun.org/exdb/mnist.

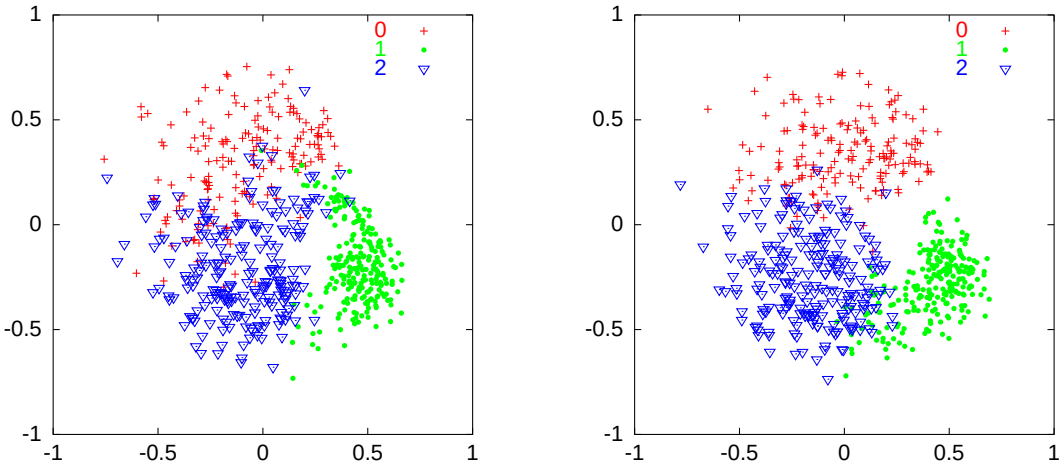


Figure 9: **Maps of handwritten digits using metric MDS (left) and distributional MDS (right).** The embeddings are of 628 examples of digits 0, 1, and 2, using the SSTRESS objective function with local weighting.

cluster structure is a strong indicator of self organization of the data and can be used to describe the structure of a variety of data types. Note that since the relative positioning of clusters with respect to each other is important in order to recover the structure, the cluster assignments alone are not sufficient.

To create embeddings that preserve the structure we defined a new objective function that considers the geometric distortion as well as the pairwise distortion. Rather than considering the error in each edge independently (as in traditional MDS techniques), we opt for embeddings that preserve the overall structure of the information contained in the matrix Δ, and specifically, the distributions of distances between and within clusters. The cluster assignments need not be known in advance, as demonstrated in Section 2.5. One can apply traditional MDS techniques to generate a preliminary embedding and use simple clustering algorithms in the host space to generate cluster assignments. Even when these assignments are imperfect, the distributional information can recover the true structure. We explored variants on this objective function, considering different functional forms, normalizations and types of dissimilarity data. Our method can be applied to proximity data as well as to high-dimensional feature vector data.

Finally, we addressed the problem of finding the "right" embedding dimension. In classical MDS techniques, the embedding dimension must be set by the user, and no bound is provided on the expected distortion of the embedding. In this paper we proposed two methods for computing the expected distortion and estimating the right dimensionality of the data: a local geometric approach, and a global heuristic based on the MDL principle.

Future directions include the study of globalization methods, other methods of assessing the structure of the data and their incorporation in the objective function, and the application of this method to real data sets.

Acknowledgments

This work is supported by the National Science Foundation under Grant No. 0133311 to Golan Yona.

Appendix A. Metric Optimization

There are numerous methods described in the literature for the numerical optimization of both the SAMMON and SSTRESS objective functions. The metric stress function is globally well-behaved: it is smooth, bounded from below, and has compact level sets. Because of this, it is easy to guarantee convergence to a local minimum. Differing strategies are distinguished not by their robustness, but by their running times, rates of convergence, and space requirements. For large data sets, evaluation of $\mathcal{H}$ takes $O(n^2)$ operations, as does the evaluation of either the gradient, $\nabla\mathcal{H}$, or the entire Hessian matrix, $\nabla^2\mathcal{H}$. As shown by Kearsley et al. (1998), linearly convergent methods, like the Guttman transform originally proposed by Sammon (1969) for SAMMON, tend to stop prematurely. On the other hand, the multidimensional Newton-Raphson method, with quadratic convergence to a local minimum, can be applied with good success. Newton's method takes $O(n^3)$ operations per iteration, most of which are spent inverting the Hessian matrix, and space of $O(n^2)$ to hold the Hessian matrix, which is not sparse. Because the latter space requirement may be prohibitive, and because we may want to check partially-converged results more frequently, we do not use Newton's method. Instead we choose a quasi-Newton minimization strategy, **conjugate gradient descent**, as an alternative.

Conjugate gradient descent shares the quadratic convergence and expected running time of Newton's method; but it has more modest storage requirements, and it produces output at shorter intervals. The theoretical basis for the method is described in many places: see, for instance, *Numerical Recipes in C* and its references (Press et al., 1993). We use the following version of the algorithm:

1. Set the iteration count k to 0, and choose an initial embedding X_0.
2. Calculate the current downhill gradient: $G_{k+1} = -\nabla\mathcal{H}(X_k)$.
3. Find new search direction, as a linear combination of the previous search direction and the gradient:
$$Y_{k+1} = G_{k+1} + \frac{(G_{k+1} - G_k)\cdot G_{k+1}}{G_k \cdot G_k} Y_k \ .$$
(For the first iteration, set $Y_1 = G_1$.)
4. Minimize $\mathcal{H}(X_k + \alpha Y_{k+1})$ with respect to the step size α. Update the embedding: $X_{k+1} = X_k + \alpha Y_{k+1}$.
5. Terminate if α, $||G_{k+1}||$, or $\mathcal{H}(X_{k+1})$ is small enough, or if k is large enough. Otherwise, increment k and return to step 2.

Because the conjugate gradient method is easy to implement and requires only first derivatives, we used it for the optimization of all the objective functions mentioned in the paper, including distributional scaling. As indicated above, it is as efficient as Newton's method and more convenient in several ways. In addition, we were able to substantially accelerate the conjugate-gradient optimization of the SSTRESS and SAMMON functions by speeding up the line minimization step, which is the bottleneck. We describe how this can be done in the following two sections.

A.1 Optimizing Metric SSTRESS

When applied to metric MDS, the conjugate gradient algorithm spends most of its time in step 4, performing line minimizations: at each iteration it calculates

$$\arg\min_{\alpha\in\mathcal{R}} \mathcal{H}(X+\alpha Y)\ ,$$

where the starting point X and the search direction Y are known. In general, pinning down each minimizing α (to sixteen digits of precision, say) will require 20–40 evaluations of $\mathcal{H}$ at different points along the ray $X+\alpha Y$. For metric SSTRESS, however, this slow process can be circumvented. Our key observation is that because $\mathcal{H}(X)$ is polynomial in the coordinates $(\mathbf{x}_i)_\mu$, its restriction to a line is also polynomial, and can be minimized in constant time once the coefficients are known. Specifically, for fixed X and Y, $\mathcal{H}(X+\alpha Y)$ is a quartic polynomial in α, with coefficients that can be found in $O(n^2)$ operations. In practice, it takes only a few times longer to find these coefficients than it does to evaluate $\mathcal{H}$ itself. As a result, by using a specialized subroutine for polynomial line minimization, we accelerate the optimization of metric SSTRESS by a factor of ten.

A.2 Optimizing Metric SAMMON

In the SAMMON case, the restriction of $\mathcal{H}$ to a line is not polynomial, so we cannot avail ourselves directly of the trick that works for SSTRESS. However, it is possible to define an auxiliary function that *is* polynomial, which can be used in place of $\mathcal{H}$ in the line minimizations. We will require the auxiliary function to be a **majorizing function** for $\mathcal{H}$; this guarantees convergence to a local minimum by ensuring that steps that decrease the auxiliary function also decrease $\mathcal{H}$.

Formally, a function $g(x,y)$ is called a majorizing function for $f(x)$ if $\forall_{x,y} g(x,y) \geq f(x)$ and $\forall_y g(y,y) = f(y)$. That is, for each fixed value of y (called the "point of support"), the values of $f(x)$ and $g(x,y)$ coincide at $x=y$, and $g(x,y)$ is never less than $f(x)$. If f and g are smooth, then clearly $\partial_1 g(y,y) = f'(y)$ and $\partial_1^2 g(y,y) \geq f''(y)$ for all y as well. Majorizing functions are of interest in minimization problems, as they give rise to the following algorithm for finding a local minimizer of f. Start at any x_0. Consider $g(x,x_0)$ as a function of x, and look for a value of x such that $g(x,x_0) < g(x_0,x_0)$. If there is none, terminate: x_0 is a (local) minimizer of f. If there is one, call it x_1. Then $f(x_1) \leq g(x_1,x_0) < g(x_0,x_0) = f(x_0)$, so we have decreased the value of f. Repeat. The potential advantage is that g can have special properties that f lacks, making it easier to minimize.

We want to find a majorizing function for $f(x;\delta) = (x-\delta)^2$ that has the additional property of being polynomial in x^2. The simplest such function is the quartic

$$g_4(x,y;\delta) = \delta^2 + \left(1 - \frac{3\delta}{y}\right)x^2 + \frac{\delta}{y^3}x^4\ .$$

At the point of support y, only the first derivatives of g and f coincide. Using g instead of f in the conjugate gradient algorithm gives a method with first-order convergence. To maintain quadratic convergence, g needs to better approximate f for small step sizes, i.e., more derivatives need to coincide. With this constraint, the next-simplest choice is the eighth-order polynomial

$$g_8(x,y;\delta) = \delta^2 + \left(1 - \frac{35\delta}{8y}\right)x^2 + \frac{35\delta}{8y^3}x^4 - \frac{21\delta}{8y^5}x^6 + \frac{5\delta}{8y^7}x^8\ .$$

This function matches f in its first three derivatives at the point of support. For fast minimization of metric SAMMON, we use the function g_8 in place of f for each line minimization.

Appendix B. Nonmetric Optimization

Nonmetric scaling is often performed by alternating between two types of steps: those that improve the configuration X, and those that improve the transformation g. Such algorithms are at best linearly convergent, since they make no use of the coupling between X and g in the objective function. Drawing on knowledge of the metric problem, we expect the nonmetric problem also to be fairly well-behaved and amenable to higher-order methods that treat X and g on the same footing. We again choose to apply conjugate gradient descent, and expect quadratic convergence to a local minimum.

In order to incorporate the function g into the set of minimization variables, we first select a parametric representation of it. For a given input matrix $\Delta = (\Delta_{ij})$, we fix $M+1$ points t_k, such that

$$t_0 < t_1 < \cdots < t_M$$

and

$$t_0 < \min_{ij} \Delta_{ij} \leq \max_{ij} \Delta_{ij} < t_M \ .$$

The t_k are chosen so that the matrix elements of Δ are distributed uniformly among the M intervals $(t_k, t_{k+1}]$. Now the function g is taken to satisfy $g(t_k) = \theta_k$ for each k, and is linearly interpolated within each interval. The requirement that g be monotonic becomes a constraint on the parameters θ:

$$\theta_0 \leq \theta_1 \leq \cdots \leq \theta_M \ . \tag{8}$$

We now minimize Eq. (2) over the range of (X, θ) admissible under the constraint (8).

Constrained minimization can be carried out in (at least) two ways consistent with our overall methodology. The first way is to employ a "simplex"-type method, analogous to that used in linear programming. Here we maintain a list of which of the M constraints $(\theta_k \leq \theta_{k+1})$ are satisfied as equalities, and take conjugate-gradient steps within that subspace. Whenever a line minimization step saturates a new inequality, we add it to the list. Whenever the downhill gradient $-\nabla\mathcal{H}$ points away from a surface $\theta_k = \theta_{k+1}$, we remove it from the list. The second way is to add a barrier function to the original objective function $\mathcal{H}(X, \theta)$. Specifically, we might minimize

$$\mathcal{H}^*(X, \theta; \mu) = \mathcal{H}(X, \theta) - \mu \sum_{k=0}^{M-1} \log\left(\frac{\theta_{k+1} - \theta_k}{\theta_M - \theta_0}\right)$$

for a sequence of barrier heights μ tending to zero. This barrier function, like $\mathcal{H}$ itself, is chosen to be scale-invariant.

Whichever method we use to enforce the constraints, we can still take advantage of efficient line minimization in the case of SSTRESS. Because of our parametrization of g, each $g(\Delta_{ij})$ is a linear function of θ; so the numerator and the denominator of Eq. (2) are polynomial in the coordinates $\mathbf{x}_{i,\mu}$ and the parameters θ_k. The restriction of $\mathcal{H}$ to a ray is

$$\mathcal{H}(X + \alpha Y, \theta + \alpha\zeta) = \frac{P_1(\alpha)}{P_2(\alpha)} \ ,$$

where P_1 and P_2 are polynomials (quartic and quadratic, in this case) with coefficients we can calculate relatively quickly. As long as the number of intervals M is small compared to n^2, evaluating the barrier function for multiple values of α will not contribute substantially to the time. However, we do not have a corresponding shortcut for nonmetric SAMMON. Therefore, our implementation of nonmetric SSTRESS is from five to ten times faster per iteration than nonmetric SAMMON.

References

D. K. Agrafiotis and H. Xu. A self-organizing principle for learning nonlinear manifolds. *Proceedings of the National Academy of Arts and Sciences*, 99:15869–15872, 2002.

I. Apostol and W. Szpankowski. Indexing and mapping of proteins using a modified nonlinear Sammon projection. *Journal of Computational Chemistry*, 20:1049–1059, 1999.

W. Basalaj. Incremental multidimensional scaling method for database visualization. In *Visual Data Exploration and Analysis VI (Proceedings of the SPIE)*, volume 3643, pages 149–158, 1999.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral eigenmaps for embedding and clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 585–591. The MIT Press, 2002.

M. Blatt, S. Wiseman, and E. Domani. Data clustering using a model granular magnet. *Neural Computation*, 9:1805–1842, 1997.

T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and Hall CRC, second edition, 2001.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100:5591–5596, 2003.

S. Dubnov, R. El-Yaniv, Y. Gdalyahu, E. Schneidman, N. Tishby, and G. Yona. A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles. *Machine Learning*, 47:35–61, 2002.

R. El-Yaniv, S. Fine, and N. Tishby. Agnostic classification of Markovian sequences. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 465–471. The MIT Press, 1998.

Y. Gdalyahu, D. Weinshall, and M. Werman. A randomized algorithm for pairwise clustering. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 424–430. The MIT Press, 1999.

J. C. Gower. Some distance properties of latent root and vector methods in multivariate analysis. *Biometrika*, 53:325–338, 1966.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441,498–520, 1933.

A. J. Kearsley, R. A. Tapia, and M. W. Trosset. The solution of the metric STRESS and SSTRESS problems in multidimensional scaling using Newton's method. *Computational Statistics*, 13: 369–396, 1998.

R. W. Klein and R. C. Dubes. Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22:213–220, 1989.

H. Klock and J. M. Buhmann. Multidimensional scaling by deterministic annealing. In *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 245–260, 1997.

S. Kullback. *Information Theory and Statistics*. John Wiley and Sons, 1959.

E. Levina and P. Bickel. The earth mover's distance is the Mallows distance: Some insights from statistics. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, pages 251–256, 2001.

J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

N. Linial, E. London, and Yu. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

S. W. Malone and M. W. Trosset. A study of the stationary configurations of the SSTRESS criterion for metric multidimensional scaling. Technical Report 00-06, Department of Computational & Applied Mathematics, Rice University, 2000.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1993.

V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of non-metric proximity data. Technical Report IAI-TR-2002-5, University of Bonn, Informatik III, 2002.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

Y. Rubner, C. Tomasi, and L. B. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth IEEE International Conference on Computer Vision*, pages 59–66, 1998.

J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.

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.

J. Shi and J. Malik. Normalized cuts and image segmentation. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 731–737, 1997.

P. S. Smith. Threshold validity for mutual neighborhood clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15:89–92, 1993.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *PAMI*, 15:1101–1113, 1993.

G. Yona. *Methods for Global Organization of the Protein Sequence Space*. PhD thesis, The Hebrew University, Jerusalem, Israel, 1999.

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

Journal of Machine Learning Research 5 (2004) 421-451 Submitted 3/03; Revised 11/03; Published 4/04

Learning Ensembles from Bites: A Scalable and Accurate Approach

Nitesh V. Chawla[*] NITESH.CHAWLA@CIBC.CA
Customer Behavior Analytics, CIBC
Commerce Court East, 11th Floor
Toronto, ON M5L 1A2, Canada

Lawrence O. Hall HALL@CSEE.USF.EDU
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620, USA

Kevin W. Bowyer KWB@CSE.ND.EDU
Department of Computer Science and Engineering
University of Notre Dame
384 Fitzpatrick Hall
Notre Dame, IN 46556, USA

W. Philip Kegelmeyer WPK@CA.SANDIA.GOV
Sandia National Labs, Biosystems Research Department
P.O. Box 969, MS 9951
Livermore, CA 94551-0969, USA

Editor: Claude Sammut

Abstract

Bagging and boosting are two popular ensemble methods that typically achieve better accuracy than a single classifier. These techniques have limitations on massive data sets, because the size of the data set can be a bottleneck. Voting many classifiers built on small subsets of data ("pasting small votes") is a promising approach for learning from massive data sets, one that can utilize the power of boosting and bagging. We propose a framework for building hundreds or thousands of such classifiers on small subsets of data in a distributed environment. Experiments show this approach is fast, accurate, and scalable.

Keywords: ensembles, bagging, boosting, diversity, distributed learning

1. Introduction

The last decade has witnessed a surge in the availability of massive data sets. These include historical data of transactions from credit card companies, telephone companies, e-commerce companies,

∗. This is the author to whom correspondence should be addressed.

and financial markets. The relatively new bioinformatics field has also opened the doors to extremely large data sets such as the Protein Data Bank (Berman et al., 2000). The availability of very large databases has constantly challenged the machine learning and data mining community to come up with fast, scalable, and accurate approaches (Provost and Kolluri, 1999; Fayyad et al., 1996). As Neal Leavitt notes (Leavitt, 2002, p. 22):

> "The two most significant challenges driving changes in data mining are scalability and performance. Organizations want data mining to become more powerful so that they can analyze and compare multiple data sets, not just individual large data sets, as is traditionally the case."

Very large data sets present a challenge for both humans and machine learning algorithms. Machine learning algorithms can be inundated by the flood of data, and become very slow in learning a model or classifier. Moreover, along with the large amount of data available, there is also a compelling need for producing results *accurately* and *fast*. Efficiency and scalability are, indeed, the key issues when designing data mining systems for very large data sets.

The machine learning community has essentially focused on two directions to deal with massive data sets: data subsampling (Musick et al., 1993; Provost et al., 1999), and the design of parallel or distributed approaches capable of handling all the data (Chan and Stolfo, 1993; Provost and Hennessy, 1996; Hall et al., 1999; Chawla et al., 2000). The subsampling approaches build on the assumption that "we don't really need all the data." The KDD-2001 conference (ACM, 2001) conducted a panel on subsampling, which overall offered positive views of subsampling. However, given 100 gigabytes of data, subsampling at 10% can itself pose a challenge. Other pertinent issues with subsampling are: What subsampling methodology to adopt? What is the right sample size? To do any intelligent subsampling, one might need to sort through the entire data set, which could take away some of the efficiency advantages. Also, some of the data mining systems are concerned with identifying interesting patterns in a large database (Hall et al., 2000; Provost and Kolluri, 1999). In such scenarios, it could be important to have enough instances of each salient case so that the learner can identify those patterns. A lot of business analysts want to identify interesting customer patterns in the data sets, so taking a subsample might not help in such a scenario. Dan Graham, IBM's director of business-intelligence solutions, notes (Leavitt, 2002, p. 22), "Data mining yields better results if more data is analyzed." While subsampling a massive data set can simplify the learning task, it can also degrade accuracy (Perlich et al., 2003). However, one can essentially build an ensemble of subsamples and observe an improvement in accuracy (Eschrich et al., 2002).

The second of the two approaches to handling massive data is to bypass the need for loading the entire data set into the memory of a single computer. Our claim is that distributed data mining can address, to a large extent, the scalability and efficiency issues presented by massive training sets. The data sets can be partitioned into a size that can be efficiently managed on a group of processors. Partitioning the data sets into random, disjoint partitions will not only overcome the issue of exceeding memory size, but will also lead to creating an ensemble of diverse and accurate classifiers, each built from a disjoint partition, but the aggregate processing all of the data (Chawla et al., 2002b,a). This can result in an improvement in performance that might not be possible by subsampling. Chawla et al. (2002b) show that it is possible to create multiple disjoint partitions of both small and very large data sets, and approach classification accuracies achievable by popular ensemble techniques such as bagging, which is normally an approach suited for small data sets.

In this paper we show that it is also possible to learn an ensemble of classifiers from each of the random disjoint partitions of data, and combine predictions from all those classifiers to achieve high classification accuracies; in some cases similar to or better than boosting or distributed boosting (Lazarevic and Obradovic, 2002). We utilize Breiman's algorithms for pasting small votes: *Ivote* and *Rvote* (Breiman, 1999). In pasting Rvotes, small random training sets are constructed from the data set and classifiers are learned. In pasting Ivotes, each subsequent small training set is constructed by importance sampling based on the quality of classifiers constructed so far. That is, the misclassified cases are given a higher probability of selection than the correctly classified cases over the learning iterations. The classifiers learned are then uniformly voted for prediction. These approaches are sequential in operation, although Rvoting can be easily implemented in a distributed environment.

We propose a distributed setting for pasting of small votes, which can also be applicable in scenarios where data is already distributed at sites, and collecting data at one location is a costly procedure. Our approach is essentially "divide and conquer": we divide the training set into n disjoint partitions, and then *Rvote* or *Ivote* on each of the disjoint subsets independently. We call our distributed approaches to pasting Ivotes and Rvotes, *DIvote* and *DRvote* respectively (Chawla et al., 2002a).

The organization of the rest of the paper is as follows. We discuss the related work in Section 2. We present the distributed paradigm of pasting Ivotes, Rvotes, and experimental details in Sections 3 to 7. We ran the distributed learning experiments on a 24-node Beowulf cluster and the ASCI Blue Supercomputer (Livermore National Laboratories), using C4.5 release 8 decision tree (Quinlan, 1992) and Cascade Correlation neural network (Fahlman and Lebiere, 1990) as the base classifiers. We show that DIvote and Ivote give very comparable accuracies, while achieving significantly better classification accuracies than a single classifier in most cases. One major advantage of DIvote is a *significant reduction in training time as compared with Ivote*. Section 7 also includes comparisons to distributed boosting. Section 8 contains the κ plots to highlight the diversity trends of DIvote and DRvote. Section 9 includes empirical evidence of applicability of Ivote with a stable method of learning such as Naive Bayes classifier (Duda et al., 2001; Good, 1965). We present the main conclusions from our work in Section 10.

2. Related Work

One popular approach towards tackling very large training sets is "divide and conquer", which can be set up in a *distributed learning* paradigm. An advantage of this approach is that the partition size of the learning task can be adjusted to fit the available computational resources. One can easily imagine a divide and conquer approach in which a data set is divided across a group of processors and a classifier is learned on each processor concurrently. The classifiers are transmitted to a central processor. The central processor can then process the predictions of the independent classifiers learned. It has been shown that combining classifiers learned on random (smaller) disjoint partitions of data can achieve the classification accuracies of the popular ensemble technique of bagging (Chawla et al., 2002b).

Domingos (1996) describes how a specific-to-general rule induction system (RISE) was sped up by applying it to disjoint training sets. This allowed the time required for learning to become linear in the number of examples. The resulting rule based classifiers were voted (with weighting) in an approach similar to bagging. The major difference was that the size of each training data set

was much smaller than the original. On a set of 7 data sets from the UCI repository using disjoint partitions of between 100 and 500 examples, they found that the resulting classification accuracy was generally as good or better than applying RISE to all the data.

Street and Kim (2001) built an ensemble of classifiers from training data treated as a stream. Each classifier is trained on a fixed amount of data from the stream. The size of the ensemble is fixed at 25 classifiers. Classifiers "compete" for entry into the ensemble based on accuracy and diversity. This approach allows an ensemble of classifiers to be built from an unlimited amount of training data. It also facilitates the building of an ensemble of classifiers on data with temporal dependencies, where the concept to be modeled may vary over time. In their experiments the ensemble of classifiers was usually slightly less accurate than a single classifier built with as much data as the current ensemble used.

Chawla et al. (2000) studied various partition strategies and found that an intelligent partitioning methodology — clustering — is generally better than simple random partitioning, and generally results in a classification accuracy comparable to learning a single C4.5 tree on the entire data set. They also found that applying bagging to the disjoint partitions, and making an ensemble of many C4.5 decision trees on each partition, can yield better results than building a decision tree by applying C4.5 on the entire data set.

Bagging, boosting, and their variants have been shown to improve classifier accuracy (Freund and Schapire, 1996; Breiman, 1996; Bauer and Kohavi, 1999; Dietterich, 2000; Latinne et al., 2001). According to Breiman, bagging exploits the instability in the classifiers, since perturbing the training set produces different classifiers using the same algorithm. However, creating 30 or more bags of 100% size can be problematic for massive data sets (Chawla et al., 2002b). We observed that for data sets too large to handle practically in the memory of a typical computer, a committee created using disjoint partitions can be expected to outperform a committee created using the same number and size of bootstrap aggregates ("bags"). Also, the performance of the committee of classifiers can be expected to exceed that of a single classifier built from all the data (Chawla et al., 2002b).

Latinne et al. (2001) proposed a combination of bagging and random feature subsets: "Bagfs". They generated bootstrap replicates (B) of a given training set, and for each such *bag* they randomly chose features without replacement F times, resulting in F feature subsets. Thus, they had $B \times F$ new learning sets. Using the McNemar test of significance they found that Bagfs never performed worse than bagging and Multiple Feature Subsets (MFS).

Boosting (Freund and Schapire, 1996) also creates an ensemble of classifiers from a single data set by utilizing different training set representations of the same data set, focusing on misclassified cases. Boosting is essentially a sequential procedure applicable to data sets small enough to fit in a computer's memory. Lazarevic and Obradovic (2002) proposed a distributed boosting algorithm to deal with massive data sets or very large distributed homogeneous data sets. In their framework, classifiers are learned on each distributed site, and broadcast to every other site. The ensemble of classifiers constructed at each site is used to compute the hypothesis, $h_{j,t}$, at the jth site at iteration t. In addition, each site also broadcasts a vector with a sum of local weights, reflecting its prediction accuracy. Each site j maintains the local distribution $\Delta_{j,t}$ and local weights $w_{j,t}$. They try to emulate the global distribution D_t by communicating the sums of weights from each site. Each site also maintains a local copy, $D_{j,t}$, of the global distribution D_t, and its local distribution Δ_j, for each boosting iteration t. The training set at site j, for each boosting iteration, is sampled according to $D_{j,t}$. At the end of all boosting iterations, hypotheses $h_{j,t}$ from different sites are combined into a final hypothesis h_{fn}. They achieved the same or slightly better prediction accuracy than

standard boosting, and they also observed a reduction in the costs of learning and the memory requirements for their data sets (Lazarevic and Obradovic, 2002). In Section 7, we compare DIvote with distributed boosting.

3. Pasting Votes

Breiman (1999) proposed pasting votes to build many classifiers from small training sets or "bites" of data. He proposed two strategies of pasting votes: Ivote and Rvote. In Ivote, the small training set (bite) of each subsequent classifier relies on the combined hypothesis of the previous classifiers, and the sampling is done with replacement. The sampling probabilities rely on the out-of-bag error, that is, a classifier is only tested on the instances not belonging to its training set. This out-of-bag estimation gives good estimates of the generalization error (Breiman, 1999), and is used to determine the number of iterations in the pasting votes procedure. Ivote is, thus, very similar to boosting, but the "bites" are much smaller in size than the original data set. Thus, Ivote sequentially generates training sets (and thus classifiers) by importance sampling.

Rvote creates many random bites, and is a fast and simple approach. Breiman found that Rvote was not competitive in accuracy to Ivote or Adaboost. The detailed algorithms behind both the approaches are presented in Section 4 as part of DIvote and DRvote.

Using CART, Breiman found that pasting Ivotes gave an accuracy comparable to running Adaboost on the whole data set (Breiman, 1999). Pasting Ivotes does not require the entire data set to be in memory for learning; instances are drawn from the pool of training data to form much smaller training sets. However, sampling from the pool of training data can entail multiple random disk accesses, which could swamp the CPU times. So Breiman proposed an alternate scheme: a sequential pass through the data set. In this scheme, an instance is read and checked to see if it will make the training set for the next classifier in the aggregate. This is repeated in a sequential fashion until all N instances (size of a bite) are accumulated. The terminating condition for the algorithm is a specified number of trees or epochs, where an epoch is one sequential scan through the entire data set. However, the sequential pass through the data set approach led to a degradation in accuracy for a majority of the data sets. Breiman also pointed out that this approach of sequentially reading instances from the disk will not work for highly skewed data sets. Thus, one important component of the power of pasting Ivotes is random sampling with replacement.

The memory requirement to keep track of which instance belongs in which small training set and the number of times the instance was given a particular class is $2JN_B$ bytes, where J is the number of classes and N_B is the number of instances in the entire data set. Let us assume we have a data set of 10^8 records(N_B), 1000 features (equivalent to 4 bytes each), the training set of size 10^5, and $J = 2$ (Breiman, 1999). The memory requirement will be close to a gigabyte, as follows:

$$\begin{aligned} A &= 2*J*N_B = 2*2*10^8 = 0.4 \text{ gigabytes;} \\ B &= 1000 \text{ features at 4 bytes each for } 10^5 \text{records} \\ &= 1000*4*10^5 = 0.4 \text{ gigabytes;} \\ C &= \text{memory required by trees stored in memory in megabytes.} \\ \text{Total} &= A+B+C \approx 1 \text{ gigabytes.} \end{aligned}$$

In our distributed approach to pasting small votes, we divide a data set into T disjoint subsets, and assign each disjoint subset to a different processor. On each of the disjoint partitions, we follow Breiman's approach of pasting small votes. We randomly sample with replacement, as we can load the entire disjoint subset of data in a processor's memory. Thus, the number of disjoint partitions can be dictated by the amount of memory available on each processor. We combine the predictions of all the classifiers by majority vote. Again, using the above framework of memory requirement, if we break up the data set into T disjoint subsets, the memory requirement will decrease by a factor of $1/T$, which is substantial. So, DIvote can be more scalable than Ivote in memory. One can essentially divide a data set into subsets easily managed by the computer's main memory.

4. Pasting DIvotes and DRvotes

The procedure for DIvote is as follows:

1. Divide the data set into T disjoint subsets.
2. Assign each disjoint subset to a unique processor.
3. On each processor build the first bite of size N by sampling with replacement from its subset, and learn a classifier.
4. Compute the out-of-bag error, $e(k)$, and the probability of selection, $c(k)$, as follows (Breiman, 1999):

$$\begin{aligned} e(k) &= p \times e(k-1) + (1-p) \times r(k). \\ p &= 0.75. \text{ We used the same p value as used by Breiman.} \\ k &= \text{number of classifiers in the aggregate or ensemble so far.} \\ r(k) &= \text{error rate of the kth aggregated classifiers on a T disjoint subset.} \\ c(k) &= e(k)/(1-e(k)), \text{ probability of selecting a correctly classified instance.} \end{aligned}$$

5. For the subsequent bites on each of the processors, an instance is drawn at random from the resident subset of data. If this instance is misclassified by a majority vote of the out-of-bag classifiers (those classifiers for which the instance was not in the training set), then it is put in the subsequent bite. Otherwise, put this instance in the bite with a probability of c(k). Repeat until N instances have been selected for the bite (Breiman, 1999).
6. Learn the $(k+1)th$ classifier on the bite created by step 5.
7. Repeat steps 4 to 6, until the out-of-bag error estimate plateaus, or for a given number of iterations, to produce a desired number of classifiers. We ran our experiments for a given number of iterations.
8. After the desired number of classifiers have been learned, combine their predictions on the test data using a voting mechanism. We used simple majority voting.

Pasting DRvotes follows a procedure similar to DIvotes. The only difference is that each bite is a bootstrap replicate of size N. Each instance through all iterations has the same probability of

being selected. DRvote is very fast, as the intermediate steps of DIvote — steps 4 and 5 in the above algorithm — are not required. However, DRvote does not provide the accuracies achieved by DIvote. This agrees with Breiman's observations on Rvote and Ivote.

Pasting DIvotes or DRvotes has the advantage of not requiring any communication between the processors, unlike the distributed boosting approach by Lazarevic and Obradovic (2002). Thus, there is no time lost in communication among processors, as trees are built independently on each processor. Furthermore, dividing the data set into smaller disjoint subsets can also mitigate the need for larger memories. Also, if the disjoint subset size is small compared to the main memory on a computer, the entire data set can be loaded in the memory. This will speed up the sampling from the data sets. Lastly, DIvote reduces the data set size on each processor, hence less examples must be tested by the aggregate classifiers during training, which also reduces the computational time.

5. Experimental Setup

We evaluated DIvote and DRvote by experiments on six small to moderate sized data sets, and two large data sets. We performed 10-fold cross-validation for almost all of our data sets, and used two-tailed paired-t-tests at the 95% confidence level to evaluate statistical significance of our results, where applicable. We also provide error bars (accuracy +/- standard error) for the Ivote, DIVote, Rvote, and DRvote in the plots. The y-axis in the plots indicates the accuracy, and the x-axis indicates the number of iterations. Please note that the number of iterations does not necessarily equate to number of classifiers. For the distributed approaches, there are *n*iterations* classifiers, where *n* is the number of disjoint partitions. For the sequential approaches, the number of iterations is equal to the number of classifiers in the ensemble.

5.1 Data Sets

The size of these data sets is summarized in Table 1. DNA, satimage, LED, pendigits, letter, waveform and covtype are available from the UCI repository. We used three of the four Statlog (D. Michie and Taylor, 1994) project data sets — DNA, satimage, and letter — used by Breiman (1999). We did not use the shuttle data set, as 10-fold cross-validation with C4.5 already gives accuracies of around 99.5%, and any improvement on it will be miniscule. Moreover, we have previously observed that simple disjoint partitioning is able to achieve that accuracy (Chawla et al., 2000). We used a subset of 60 features (features 61 — 120) for the DNA data set (D. Michie and Taylor, 1994).

We procured the training and testing set partitions of waveform, LED and covtype from Lazarevic and Obradovic (2002) to allow direct comparisons. One of our large data sets comes from the problem of predicting the secondary structure of proteins (Jones, 1999). The "train and test set one" were used in developing and validating, respectively, a neural network that won the CASP-3 (Livermore National Laboratories, 1999) secondary structure prediction contest. The Protein data set (called the "Jones data set" in the rest of the paper) contains 209,529 elements in the training set and 17,731 elements in the testing set. Each amino acid in a protein can have its structure labeled as helix (H), coil (C), or sheet (E). The features for a given amino acid are twenty values in the range -17 to 17, representing the log-likelihood of the amino acid being any one of twenty basic amino acids. Using a window of size 15 centered around the target amino acid, and an extra bit per amino acid to signify where the window spans either the N or C terminus of the protein chain, gives a feature vector of size 315 (Jones, 1999). The testing set is specially constructed to avoid

Data Set	Data Set Size	Classes	Number of Attributes
DNA	3,186	3	60
Satimage	6,435	6	36
LED	Training = 6,000; Testing = 4,000	10	7
Pendigits	10,992	10	16
Letter	20,000	26	16
Waveform	Training = 40,000; Testing = 10,000	3	21
Covtype	Training = 149,982; Testing = 431,030	7	21
Jones	Training = 209,529; Testing = 17,731	3	315

Table 1: Data set sizes, number of classes, and attributes.

any homology to the training set; this makes it a hard prediction problem for the classifier, and an appropriate test for the new protein secondary structure predictions.

We performed 10-fold cross-validation for seven of our data sets: DNA, satimage, pendigits, letter, waveform, LED, and the Jones data set. For the waveform and LED data sets, we combined the training and testing sets to perform 10-fold cross-validation. We combined the training and testing sets to increase the overall data set size for the learning procedures. For the Jones data set the 10-folds were constructed per-chain from the training set, so that non-homogenity is maintained between our 10-fold training and testing sets. This maintained the level of difficulty for the learning algorithm. For the other large data set, covtype, we only ran on the separate training and testing sets of sizes 149,982 and 431,030, respectively, as done by Lazarevic and Obradovic (2002). The size of the entire covtype data set was too large (581,102) to perform a 10-fold cross-validation in a reasonable time, given the number of approaches and classifiers.

5.2 Base Classifiers and Computing Systems

We used the C4.5 release 8 decision tree, the Cascade Correlation neural network, and Naive Bayes classifiers for our experiments. The sequential Rvote and Ivote experiments were run on a 1.4 GHz Pentium 4 Linux workstation with two gigabytes of memory, and an 8-processor Sun-Fire-880 with 32 gigabytes of main memory. We ran most of our DIvote and DRvote experiments on a 24-node Beowulf cluster. Each node on the cluster has a 900 MHz Athlon processor and 512 megabytes of memory. The cluster is connected with 100Bt ethernet. We also ran some of our DIvote experiments on the ASCI Blue Supercomputer. There are 1296 compute nodes on the ASCI Blue. The 4-CPU nodes have: 1.5 gigabyte memory, 332 MHz PowerPC 604e chip, 83 Mhz memory bus, and a compute node peak performance of 2.6 GFLOP/s (Livermore National Laboratories).

6. Experiments with the C4.5 Decision Tree Learning Algorithm

We ran 10-fold cross-validation experiments using C4.5 decision trees for all the small and moderate sized data sets. Section 6.1 contains the results on the small and moderate data sets. For the large data sets, we used separate training and testing splits. However, for one of the large data sets–Jones–we also report the 10-fold cross-validation result. Section 6.2 contains the results on the large data sets. We also present a timing analysis and comparison between DIvote and Ivote in Section 6.3.

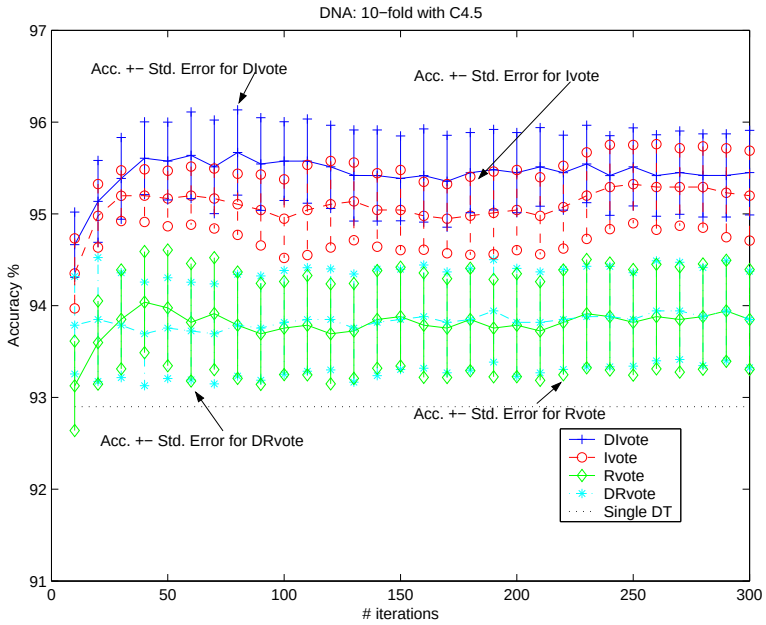


Figure 1: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for DNA.

6.1 Small and Moderate Sized Data Sets

We partitioned our small and moderate sized data sets, with the exception of DNA, into four disjoint partitions and used bites of size 800 to learn classifiers. Due to the smaller size of DNA, we partitioned it into three disjoint partitions and used bites of size 400. We used unpruned trees for all decision tree experiments as the ensemble of classifiers will override the overfitting (Breiman, 1999). Figures 1 to 5 show the 10-fold accuracy trends for six of our data sets. Each of the distributed approaches is comparable in its classification accuracy to its sequential counterpart. DIvote/Ivote achieve significantly higher classification accuracies than a single C4.5 classifier, and the simplistic approaches of DRvote/Rvote. LED is an exception, as all the approaches are comparable to each other. We believe that DIVote/IVote could be a victim of the feature noise present in the LED data set, as each feature has a 10% probability of having its value inverted (Blake and Merz, 1998).

The letter data set differed from other data sets as both DRvote and Rvote achieved significantly worse classification accuracies than C4.5. However, for letter also, DIvote and Ivote were comparable, and significantly better than Rvote, DRvote, and C4.5, in their classification accuracies. The letter data set has 26 classes with 16 dimensions; each 800 sized bite will contain an average of 30 instances (on an average) for each class. Thus, the 30 training instances may not mimic the real distribution of the data set. To test this model, we created 100% bags on each disjoint partition (Chawla et al., 2000) of the letter data set, and found that the classification accuracy increased significantly as compared to DRvote or Rvote, but was still not better than DIvote and Ivote. This shows that 100% random bags constructed on each disjoint partition of the letter data set are introducing more coverage, better individual classifiers, and diversity[1] compared to the DRvote bites (800 instances).

1. We visit the diversity issue in Section 8.

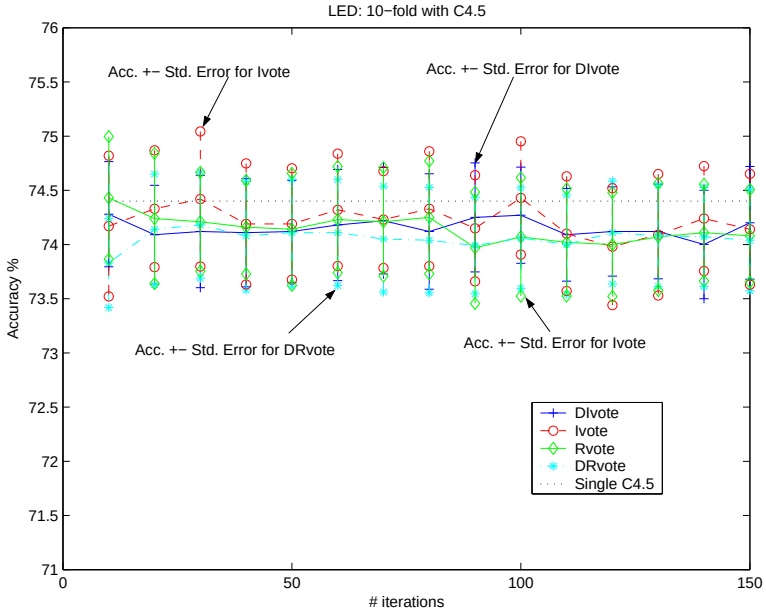


Figure 2: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for LED.

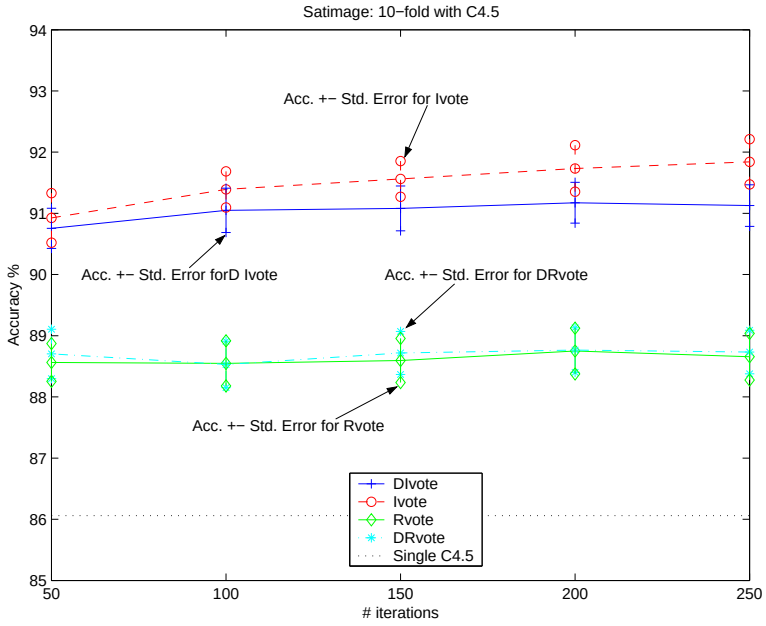


Figure 3: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for satimage.

Since DIvote and Ivote sample heavily from misclassified instances, after each iteration or series of iterations they focus on potentially different instances.

The results on the small data sets are encouraging as for almost all the data sets, DIvote is similar to Ivote in its classification accuracy, while surpassing a single C4.5 decision tree learned on

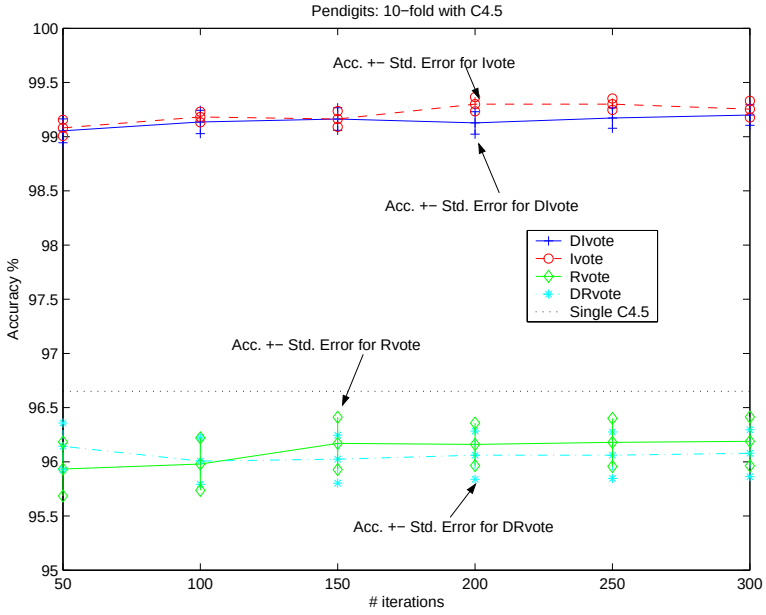


Figure 4: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for pendigits.

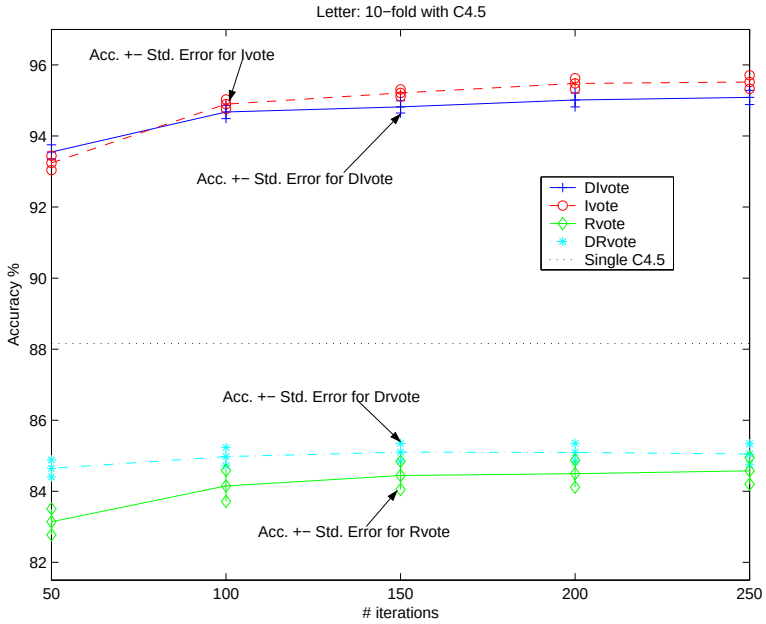


Figure 5: C4.5 Accuracy comparisons of DIvote, Ivote, Rvote, DRvote, and C4.5 for letter.

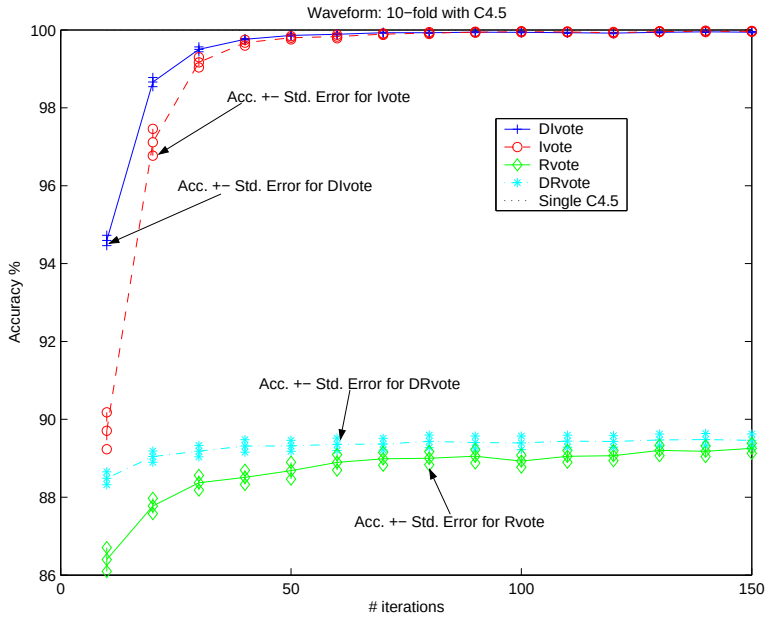


Figure 6: C4.5 Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for waveform.

the entire data set. DIvote is also significantly better than DRvote. Another important observation is that for almost all the data sets, 100 iterations of DIvote or Ivote achieve close-to-maximum accuracies, and after that the increase in accuracy is very slow compared to the addition of iterations. Given the small bite size, 100 iterations are very fast using any algorithm as the training set size is only 800 (or 400).

6.2 Large Data Sets

Due to the distinct nature of analysis on both the large data sets, we treat them separately in the subsequent subsections.

6.2.1 JONES DATA SET

We set up two different experiments on the Jones data set. One evaluates the DIvote, Ivote, DRvote, and Rvote approaches by learning and testing on the provided train and test sets. The other, 10-fold cross-validation experiment, was to understand and report statistical differences between DIvote and Ivote.

We divided the training set into 24 random disjoint partitions, as we had 24 nodes of the Beowulf cluster available for this experiment. To understand the effect of bite size on DIvote for this data set, we also set up bites at sizes 1/256 (818 instances), 1/128 (1636 instances) and 1/64 (3272 instances) of the entire training set size of 209,529. Figure 7 shows the classification accuracies of the various approaches using the given train/test split. As is evident from the figures, all the distributed approaches more accurate than the sequential approaches.

We also observe that increasing the bite sizes has no major effect on the classification accuracies of the ensemble. The experiment with increasing bite sizes is computationally cheap, as the bite size

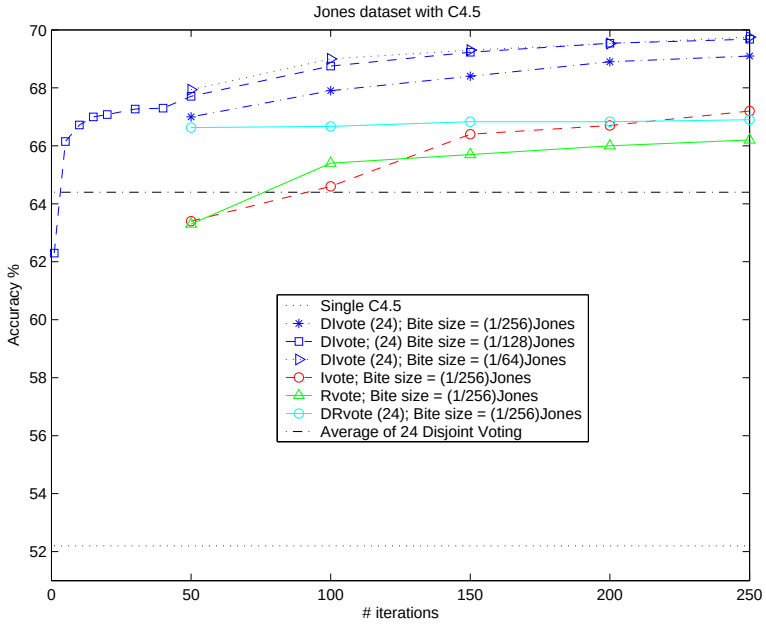


Figure 7: Accuracy comparisons of DIvote, Ivote, Rvote, DRvote, and C4.5 for the Jones data set.

remains very small (from approximately 800 to 3200). In addition, we also show the classification accuracy achieved by voting classifiers learned on each of the random disjoint partitions. Note that adding a meta-layer of DIvote on the disjoint partitions provides an absolute improvement of at least 4% in accuracy. We would also like to point out the accuracy achieved by learning a single C4.5 decision tree on the entire data set: 52.2%. It is remarkable that all the ensemble approaches, reported in the Figure 7, provide an absolute gain of at least 12% on this data set. The non-homologous nature of the testing set makes it a particularly hard problem for C4.5 with its axis parallel splits (Chawla et al., 2001). The decision tree overfits the training set, achieving a very high resubstitution accuracy of 96.7%. The ensemble of decision trees counters the effect of overfitting and improves generalization, which is very important for the protein domain (Eschrich et al., 2002).

It is also interesting to note that the average accuracy of a decision tree learned on bites of size 1/256th of the Jones data set (for DIvote) is below 50%, and the aggregate of all the not-so-good classifiers gives a performance in the range of 66% to 70% for 50 or more learning iterations. A classifier, learned on 1/256th of the entire training set, with an accuracy of $< 50\%$ is not very surprising when juxtaposed with the accuracy of the single classifier learned on the entire training set. Figure 8 shows the accuracy trends for 4800 decision trees learned in the DIvoted ensemble for the Jones data set. These plotted accuracies are sorted in an increasing order, and are not in the same order as the construction of bites.

To further evaluate the behavior of DIvote and Ivote for the large Jones data set, we report the 10-fold averaged cross-validation results in Figures 9 and 10. As with the previous experiment, we divided each training fold into 24 random disjoint partitions. Let us examine Figure 9 first. DIvote achieves significantly higher classification accuracies than Ivote for 200 iterations. However, a point of contention could be that DIvote is constructing 200*24 (4800) classifiers versus 200 classifiers for Ivote, albeit in less time (which is an advantage in itself of DIvote). So, on the ASCI Blue

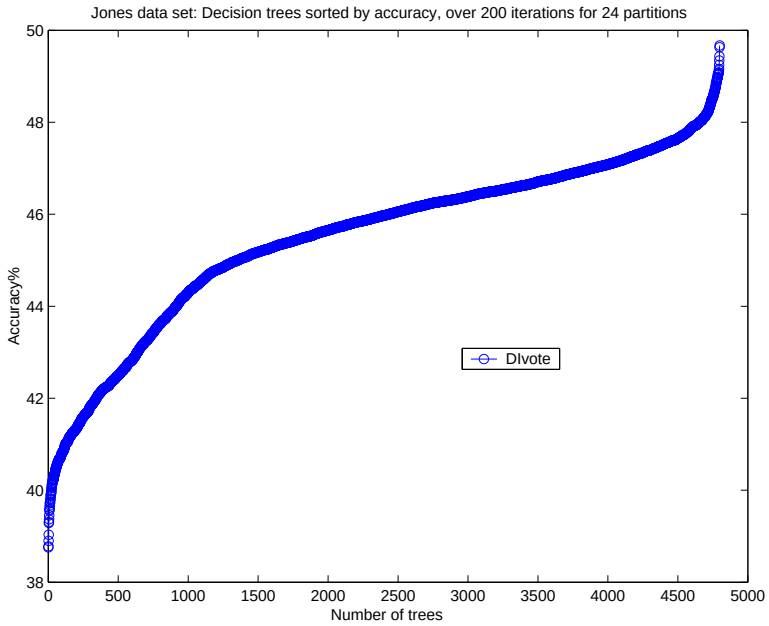


Figure 8: Individual performances on the Jones data set.

supercomputer, we let Ivote run up to 1200 iterations or six hours (per fold) of allowed compute time on a node. We then ran 50 (1200/24) iterations of DIvote, taking only six minutes (per fold), which means that we constructed only 50 decision tree classifiers on each of the 24 disjoint partitions. This gave us an ensemble size of 1200 (50*24), which is equivalent to Ivote. On performing a two-tailed paired-t-test at 95% confidence interval, we find that an ensemble of 1200 classifiers of Ivote is significantly better (in classification accuracy) than the ensemble of 1200 classifiers of DIvote.

Now, the ensemble of DIvote classifiers is really under-constructed this time, and we can still use the hours of compute time utilized by Ivote to construct the ensemble. For instance, consider Figure 10, if we compare 200 DIvote classifiers that took 30 minutes and 1200 (maximal possible, given the computation bottleneck) Ivote classifiers that took six hours, we observe that DIvote and Ivote are comparable to each other. This highlights the significant advantage of DIvote in terms of time spent on computation.

6.2.2 COVTYPE

Our second large data set, covtype, contains 149,982 instances in the training set. This data set also has a very high skew in its class distribution as reported in Table 2. We divided the training set into eight disjoint partitions, as done in the distributed boosting paper. We ran DIvote, Ivote, Rvote, and DRvote experiments using a bite size of 800. Figure 11 shows one of the results on this data set. Each of the ensemble approaches for covtype results in significantly worse classification accuracy than learning a single C4.5 decision tree on the entire training set. This is contrary to our previous observations. We believe the high skew in the class distribution is coming into play, as each of the classes is, possibly, not getting a sufficient representative coverage and interaction in the bites.

We further investigated the coverage issue by increasing the bite size for DIvote and Ivote. As one would do in subsampling (Provost et al., 1999), we tried the following different bite sizes: 800,

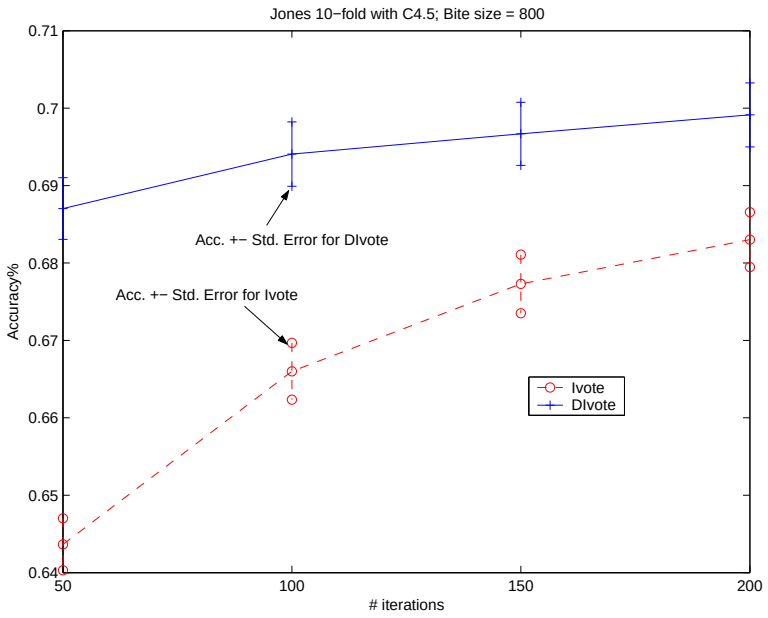


Figure 9: 10-fold averages of up to 200 iterations of Ivote and DIvote.

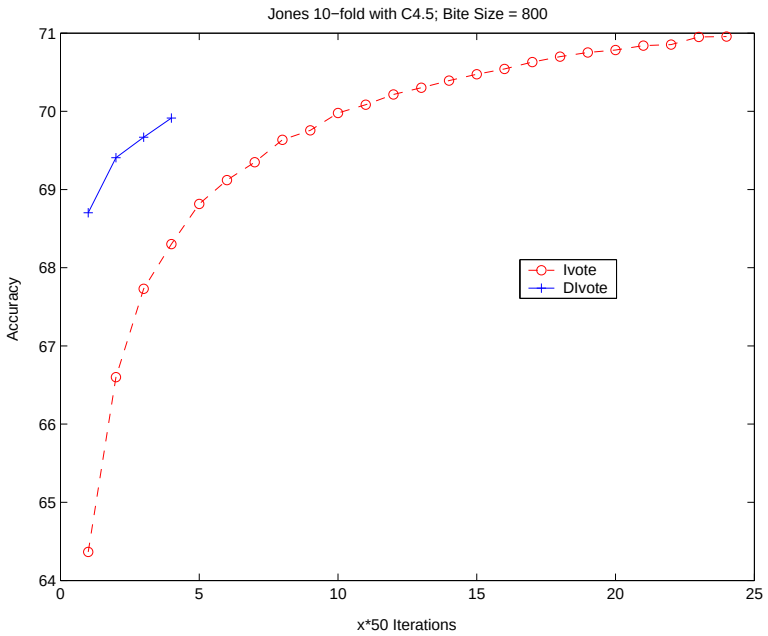


Figure 10: 10-fold averages of up to 1200 iterations of Ivote and 200 iterations of DIvote.

Class 3	670
Class 4	2,414
Class 5	4,544
Class 6	5,433
Class 2	8,971
Class 0	54,433
Class 1	73,517

Table 2: Class distribution for the covtype data set, in ascending order.

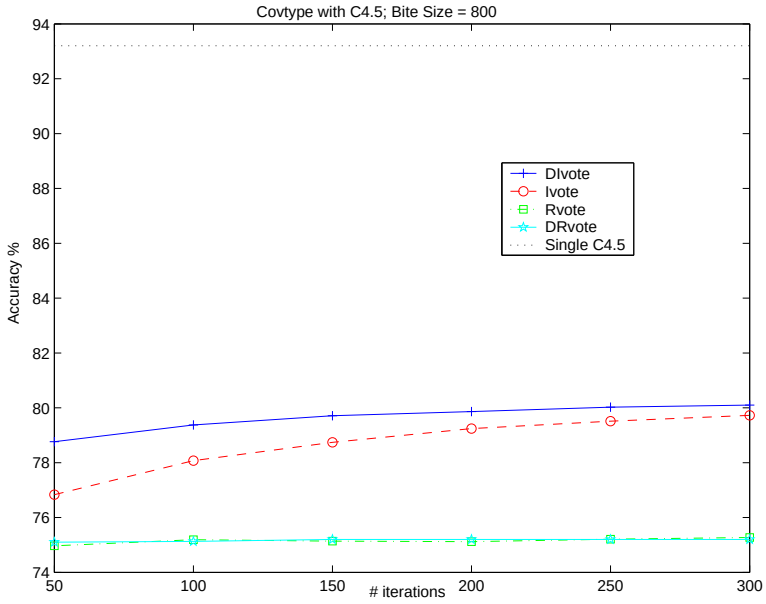


Figure 11: Covtype data set with C4.5.

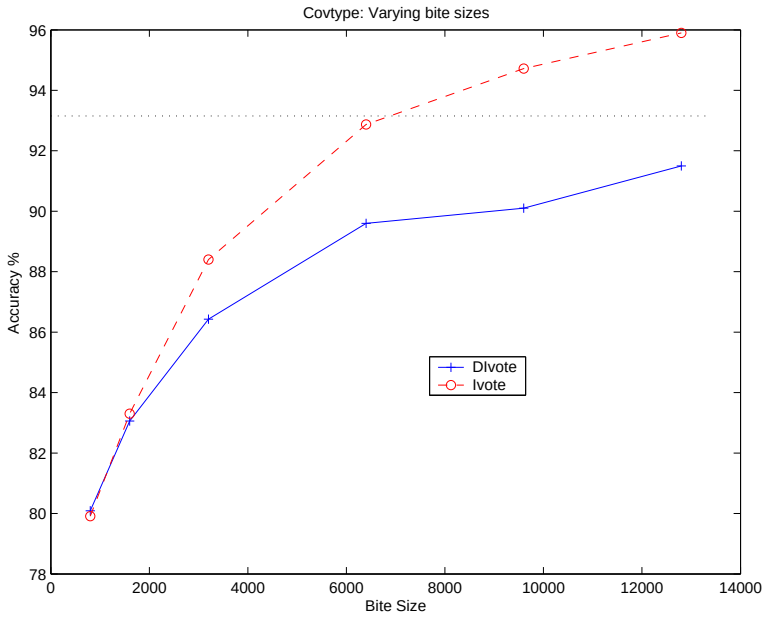


Figure 12: Effect of varying bite sizes on the covtype data set for Ivote and DIvote.

1600, 3200, 6400, 9600, 12800. We would like to point out that the bite size of 12,800 is still less than 10% of the training set of size 149,982. In the sequential domain, when searching for a single appropriate subsample of a given data set, the error (or accuracy) curve at varying subsample sizes is constructed, where each (x,y) on the curve signifies the error y at a particular subsample size x. So in the ensemble setting, each (x,y) represents the voted accuracy, y, of the ensemble constructed from subsamples (bites) of size x. This allowed us to make an accuracy (or error) curve for the improvement observed with varying bite sizes. Figure 12 summarizes the result. We constructed 300 classifiers at each bite size. Increasing the bite size improves the coverage for the classes, and leads to a significant improvement in the prediction accuracy for both DIvote and Ivote. However, Ivote achieves higher classification accuracies than DIvote for this data set, and also surpasses the accuracy achieved by the single C4.5 decision tree — 93.2%. The discrepancy in Ivote and DIvote performance is possibly arising from the limited coverage of classes in each bite from the 8 disjoint partitions; for example, each disjoint partition has, on an average, only 83 examples of class 3.

6.3 Timing

Table 3 shows the timing (user and system time during training) ratios of DIvote to Ivote, and DRvote to Rvote on the Beowulf cluster for some of our data sets using C4.5 decision trees. We report times for some of our small and large data sets, and with only C4.5, as the rest should follow similar trends. The experimental parameters were: number of iterations = 100; bite size N = 800 for the small data sets, and bite size N = (1/256) * (Jones data set size) for the Jones data set.

The time taken for DIvote and DRvote reflects the average of the time taken for 100 iterations on each of the T (T = 4 for the small data sets; T = 24 for the large data set) nodes of the cluster. For fair timing comparisons to DRvote and DIvote, we also ran 100 iterations of Ivote and Rvote on a single cluster node. It is noteworthy that we are able to build T*100 DIvote classifiers simultane-

Data Set	DIvote/Ivote	DRvote/Rvote
Satimage	0.36	0.91
Pendigits	0.29	0.95
Letter	0.23	0.89
Jones	0.048	0.90

Table 3: Ratio of time taken by DIvote to Ivote, and DRvote to Rvote, on a cluster node.

ously. One significant advantage of the proposed DIvote approach is that it requires less time than Ivote, particularly for very large data sets. Since we divide the original training set into T disjoint subsets, during training the aggregate DIvote classifiers on a processor test many fewer instances than aggregate Ivote classifiers (for the Jones data set each disjoint partition has only 8730 instances, compared to 209,529 in the entire data set). Also, a reduction in the training set size implies that the data set can be more easily handled in main memory. This is a big gain as the multiple random disk accesses can then be avoided during sampling. We pointed out earlier that Breiman found that sequential disk access was not giving as good a classification accuracy as multiple random disk access. So, to maintain the advantage of random sampling, each processor can load the relatively smaller training set partition into its main memory. Table 3 shows that as the overall training data set size increases, the ratio of DIvote time to Ivote time decreases. That is, Ivote becomes more computationally expensive than DIvote as the data set increases.

It is not surprising that the timings for DRvote and Rvote are very similar, as both the approaches essentially build many small bags from a given training set, without the intermediate book-keeping. Nevertheless, DRvote builds T times as many Rvote classifiers in less time.

We would like to note that the same software performs DIvote and Ivote, except that for the distributed runs we wrote an MPI program to load our pasting votes software on different nodes of the cluster, and collect results. So, any further improvement of the software would be applicable across the board, although it might decrease the ratios of time taken by DIvote to time taken by Ivote. Nonetheless, it is fair to assume that Ivote will hit a memory bottleneck (with an increasing data set size) faster than DIvote irrespective of the implementation.

7. Experiments with the Cascade Correlation Neural Network

To investigate whether our results generalize to other sorts of classifiers, we performed 10-fold cross-validation experiments on the small to moderate sized data sets using the Cascade Correlation (CC) neural network. We also compared the DIvote approach with the distributed boosting algorithm, which is presented in Section 7.2.

7.1 10-fold Cross-Validation Experiments

Figures 13 to 18 summarize the results of 10-fold cross-validation with CC. Using two-tailed paired-t-tests at the 95% confidence interval, we find that DIvote and Ivote are significantly better than DRvote, Rvote for satimage, letter, waveform, and pendigits, and comparable for DNA. DIvote and IVote are significantly better than a single CC for DNA, satimage, letter, waveform, and pendigits. DIvote and Ivote achieve comparable accuracies (no significant difference) for all the data sets,

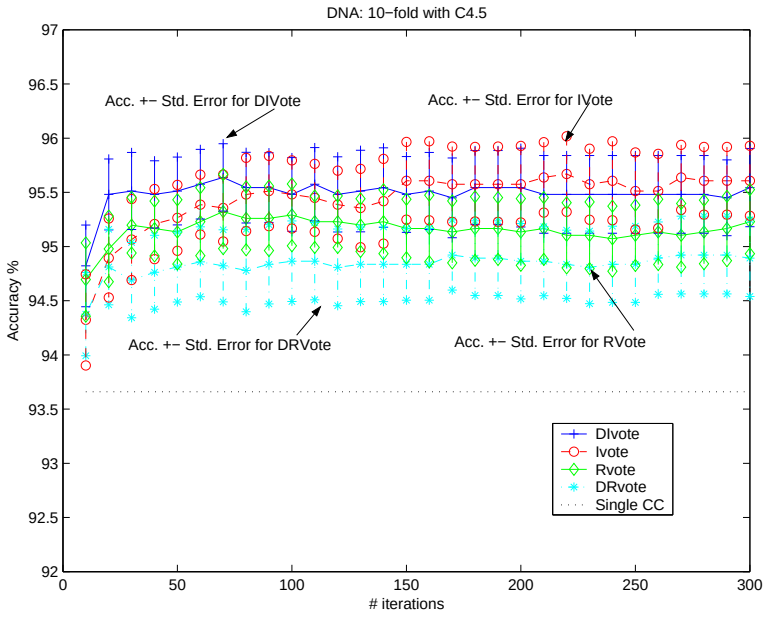


Figure 13: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for DNA.

but for pendigits. DIvote is significantly better than Ivote for pendigits. LED, again, provides an exception: all approaches are similar to each other in their classification accuracies.

We required fewer learning iterations with CC than with C4.5. Moreover, we observe that with CC, 40 to 50 iterations are sufficient, and the accuracy increases very slowly, if at all, thereafter. This is noteworthy as although neural networks are slower to learn when compared to decision trees, they require even fewer learning iterations. Essentially, an important conclusion to be drawn from all these experiments is that using importance sampling not only mitigates the need for many iterations but also limits the amount of data needed for learning.

7.2 Comparison to Distributed Boosting

We ran experiments on the four publicly available data sets used by Lazarevic and Obradovic (2002) to facilitate direct comparisons. We divided the training sets of covtype, LED and waveform into four disjoint partitions and the training set of pendigits into 6 disjoint partitions as done by Lazarevic and Obradovic (2002). We learned an ensemble of CC classifiers by running 100 iterations of DIvote with a bite size of 800 for each of the data sets. We evaluate the ensemble on the separate testing sets to compare with the distributed boosting approach.

Table 4 reports the accuracies achieved by DIvote and distributed boosting on the separate testing sets. We report the distributed boosting accuracy for the "Simple Majority" voting scheme using $p = 0$ directly from (Lazarevic and Obradovic, 2002). The table shows that using neural networks we can achieve accuracies comparable to (or better than) the distributed boosting algorithm. The inherent and compelling advantage of the DIvote approach is that it requires no inter-processor communication during learning. Although with DIvote we are learning hundreds of classifiers, we never need to learn a single classifier on a complete given partition of data. The bite size always remains

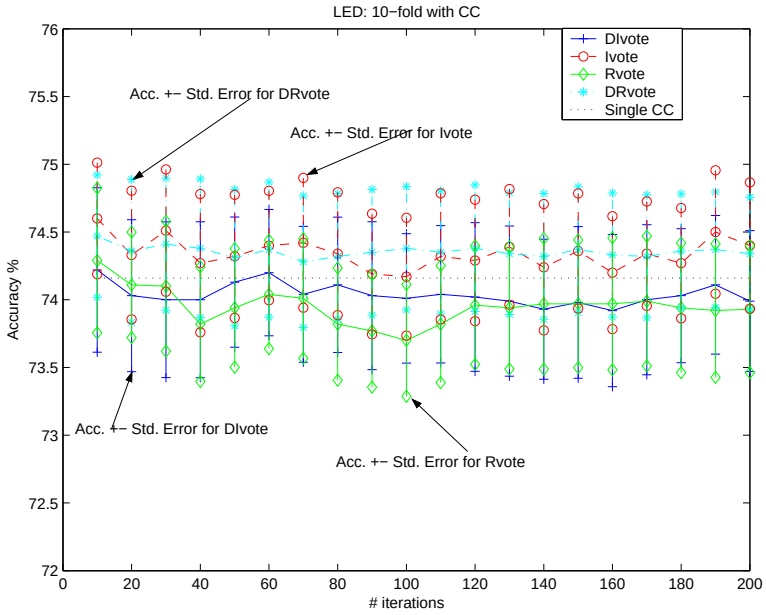


Figure 14: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for LED.

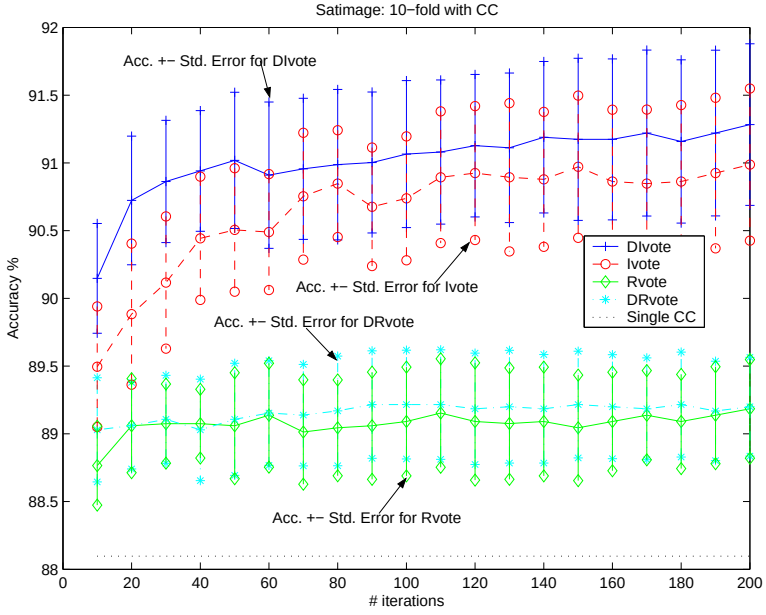


Figure 15: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for satimage.

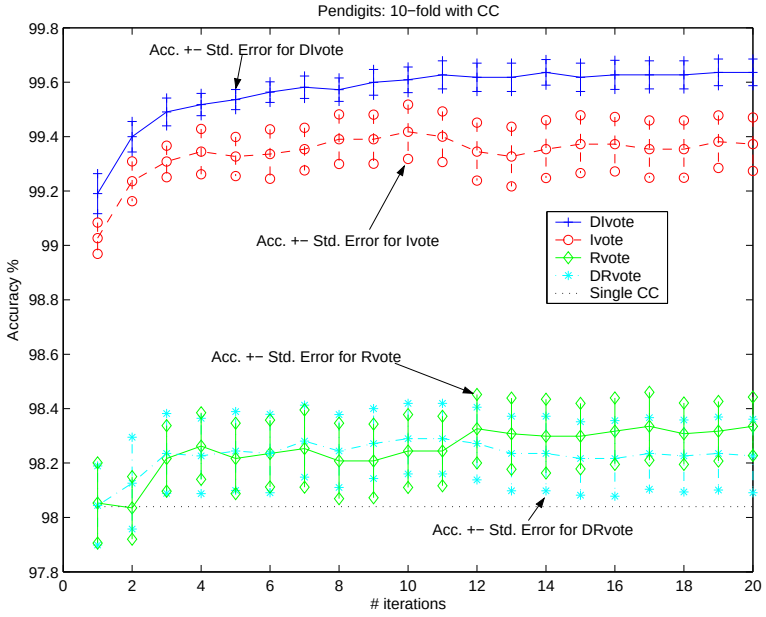


Figure 16: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for pendigits.

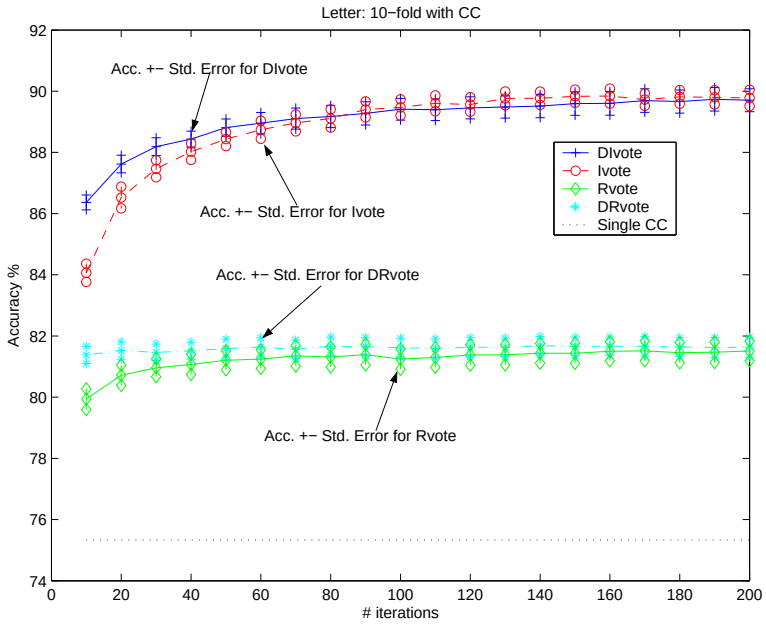


Figure 17: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for letter.

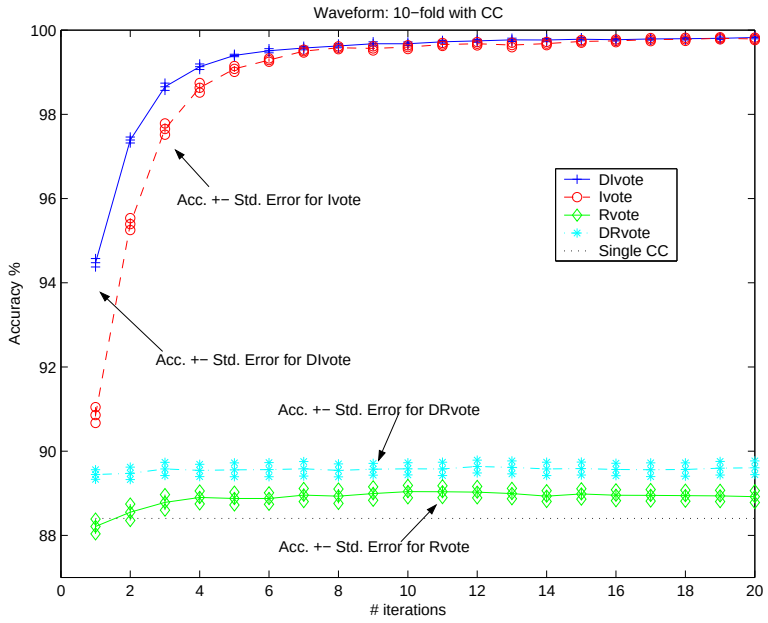


Figure 18: CC Accuracy comparisons for DIvote, Ivote, DRvote, and Rvote for waveform.

Data Set	DIvote	Distributed Boosting (Lazarevic and Obradovic, 2002)
Pendigits	96.4%	96.5%
LED	74.0%	73.4%
Waveform	87.2%	87.0%
Covtype	78.2%	72.6%

Table 4: Classification accuracy comparisons for DIvote and distributed boosting.

very small compared to the size of the entire training set or even the size of the disjoint partition. Thus, given any partition size, the learning task for a single classifier always remains small.

8. Diversity of Classifiers in the Ensemble

The diversity of classifiers in an ensemble is considered a key issue in the design of an ensemble (Kuncheva and Whitaker, 2003). Classifiers are considered diverse if they disagree on the examples for which they err. Diversity, thus, is a property of a group of classifiers with respect to a data set. The classifiers might be reporting similar accuracies, but be disagreeing on their errors. Diversity is an important aspect of the ensemble techniques — bagging, boosting, and randomization (Dietterich, 2000). There has been a significant amount of research on defining measures for diversity, and evaluating the reasons behind the success of ensemble techniques (Giacinto and Roli, 2001; Kuncheva and Whitaker, 2003; Kuncheva et al., 2000; Skalak, 1996; Ho, 1998; Banfield et al., 2003). Diversity is an important metric in addition to accuracy when evaluating and constructing an ensemble of classifiers. Breiman notes that the success of random forests lies in the interplay

of the "strength[2]" and "correlation[3]" of classifiers. Ideally, one wants lower correlation and higher strength of classifiers Breiman (2001). To understand the behavior of our ensemble of classifiers, we implemented the κ metric given by Dietterich (2000), and defined by

$$\begin{aligned}
\Theta_1 &= \frac{\sum_{i=1}^{T} C_{ii}}{m}, \\
\Theta_2 &= \sum_{i=1}^{T} (\sum_{j=1}^{T} \frac{C_{ij}}{m} . \sum_{j=1}^{T} \frac{C_{ji}}{m}), \\
\kappa &= \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}.
\end{aligned}$$

T is the number of classes, C is the $T * T$ square array, such that C_{ij} signifies the number of examples assigned to class i by the first classifier and to class j by the second classifier. Θ_1 gives the degree of agreement, and Θ_2 is the degree of agreement expected at random. κ is the statistic measuring diversity. κ equals zero when the two classifiers agree only by chance and κ equals one when the two classifiers agree for every example. Dietterich (2000) produced a scatter plot of κ and mean error of a pair of classifiers to show the spread of κ values against error. The κ plots are constructed by plotting the κ value against the mean error (or accuracy) of the n random pairs of decision tree classifiers. Each classifier in an ensemble (or a random subset of the ensemble) is combined with every other classifier to compute the mean pair-wise error and κ value. The lower the κ values, the higher the disagreement amongst the classifiers.

We built κ plots for C4.5 decision tree ensembles constructed by DIvote and DRvote. We wished to shed some light on the classification accuracy improvements observed with DIvote by virtue of more diversity in the design of the ensemble of classifiers. We only show the κ plots for two of the data sets: letter and Jones. Figure 19 shows the κ plots for C4.5 decision trees constructed with DIvote and DRvote for the letter data set. We get a broader range of κ values and mean error with DIvote, compared to DRvote, thus implying that DIvoted classifiers are essentially more diverse with each other than the DRvoted classifiers. This ties in with our observation that the ensemble constructed via DIvote is more accurate than the ensemble constructed via DRvote.

Figure 20(a) shows the diversity plots for DIvote on the Jones data set. We randomly chose 600 decision trees from the ensemble of 4800 decision trees for constructing the κ plots; the bite size was 1636 or 1/128th of the Jones data. We chose only 600 trees due to the computational infeasibility of doing pairwise comparisons for all 4800 decision trees. As is evident from the figures, the DIvoted classifiers produce a lower κ value, that is, there is a higher disagreement in the classifiers' predictions. Thus, DIvote is able to boost the classification accuracy of a multiple classifier system of weak and unstable classifiers, by inducing a high diversity amongst the classifiers. Figure 20(b) shows the diversity plot for DRvote on the Jones data set. We again selected 600 trees at random from our ensemble of 4800 decision trees constructed from bites of size 1636. As shown in the Figure 20(b), there is a high degree of disagreement amongst the classifiers, but there is still a more compact set of points in the plot as compared to the DIvoted scatter plot. There is a broader range in error as well as κ values for the DIvote classifiers. We observed that the ensemble constructed

2. Strength is the accuracy of the individual classifiers.
3. Correlation is the dependence amongst classifiers.

Data Set	Single Naive Bayes Classifier	300 Ivote	300 Rvote	50 Bags
Letter	65.58%	**68.82**%	67.02%	66.1%
Pendigits	81.1%	**91.29**%	81.36%	81.04%

Table 5: Naive Bayes classifier on the letter and pendigits data sets. The highest accuracy is given in bold.

with DRvote classifiers is able to provide an accuracy boost of at least 14% over learning a single decision tree classifier (see Figure 7). Thus, the high diversity amongst the classifiers aids the voting ensemble.

9. Stable Classifier and Ivote/DIvote

So far, we have shown that DIvoting/Ivoting unstable classifiers, such as decision trees and neural networks, achieves better classification accuracies than learning a single unstable classifier on the entire data set. Ensemble methods have been usually cited to work better with unstable classifiers due to the sensitivity of the unstable classifiers to the data presented for learning. We believe that the core issue in generating a diverse set of classifiers is how you generate the ensemble, not the base classification method. For instance, boosting has been shown to work well with a stable classifier like Naive Bayes (Bauer and Kohavi, 1999).

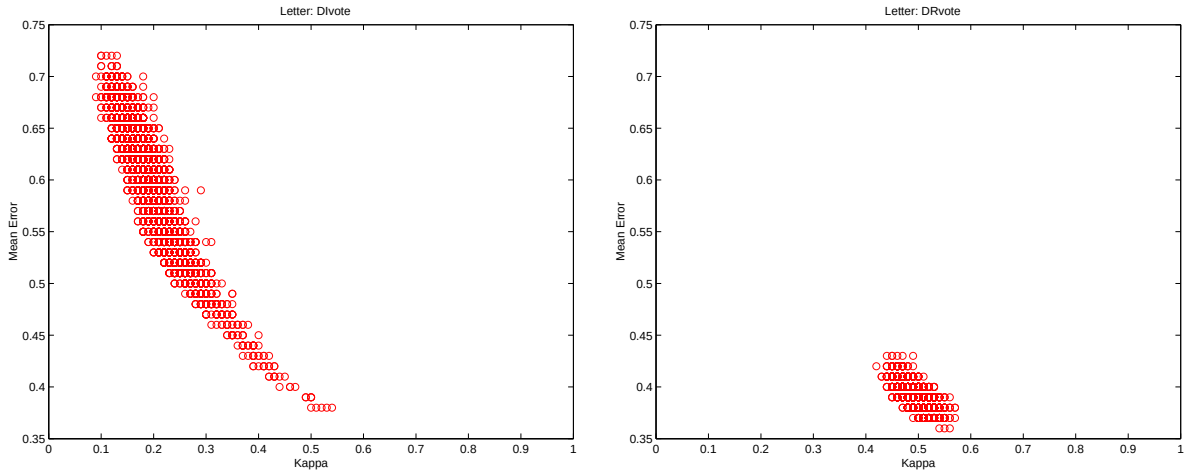
We saw in the diversity plots in the previous section, that DRvote and Rvote produce a compact set of κ values in spite of being constructed from unstable classifiers. And DIvote and Ivote produced a bigger spread of κ values. Hence, it is the importance sampling in DIvote or Ivote, which is helping the learned ensemble. To show this we used a stable method of learning a classifier, Naive Bayes[4], in conjunction with Ivote, Rvote, and bagging. For comparison purposes, we evaluated the sequential versions — Ivote, Rvote and bagging — with Naive Bayes on two of our data sets: pendigits and letter. We used the separate training and testing sets, since we were more interested in comparing the diversity trend between importance sampling and random sampling methodologies than in statistical validation of accuracies' improvement.

Table 5 shows the classification accuracies obtained from learning Naive Bayes classifiers on the letter and pendigits data sets. Figures 21 to 22 highlight the κ results on both the pendigits and letter data sets. As is evident from the figures, Ivoted classifiers are more diverse (and accurate) than both Rvoted and bagged classifiers. This result is not very surprising, as Ivote/DIvote classifiers are learned on bites derived from importance sampling, and each of these bites can look very different due to the heavier sampling from more difficult examples. Although a spread of κ values is observed for Rvote with the pendigits data set, this spread is prevalent at $0.7 \leq \kappa \leq 1.0$. Higher κ values indicate a higher degree of agreement between the different pairs of classifiers.

10. Conclusion

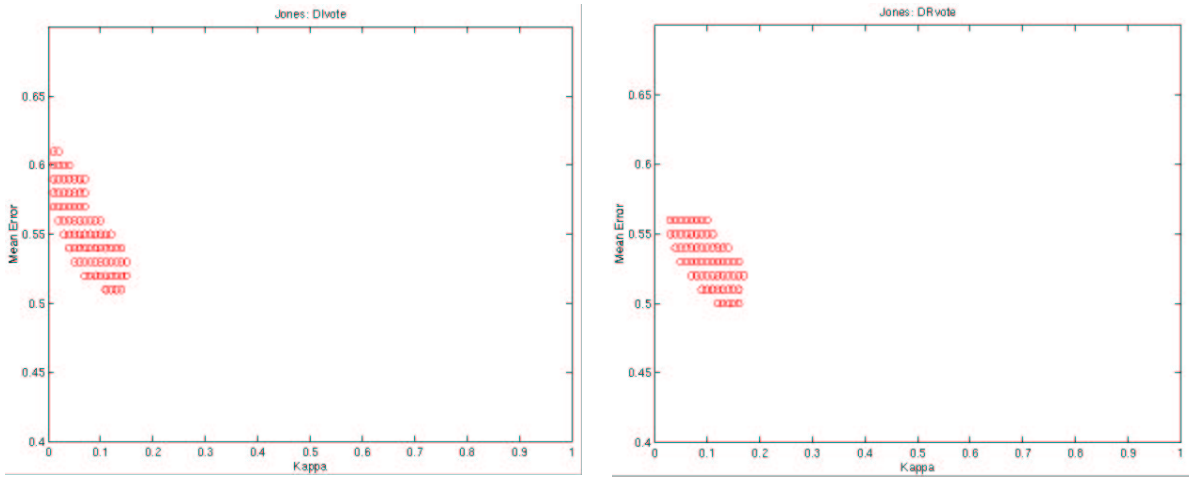
We proposed a distributed framework for pasting Ivotes by dividing a training set into *n* disjoint partitions and building hundreds of classifiers (using very small training sets) on each disjoint partition. The main conclusion of our work is that pasting DIvotes is a promising approach for very

4. The source code was downloaded from http://fuzzy.cs.uni-magdeburg.de/∼borgelt/software.html.



(a) (b)

Figure 19: a) κ Plot for DIvote with C4.5 on the letter data set. b) κ Plot for DRvote with C4.5 on the letter data set.



(a) (b)

Figure 20: a) κ plot for DIvote and Jones data set. b) κ plot for DRvote and Jones data set.

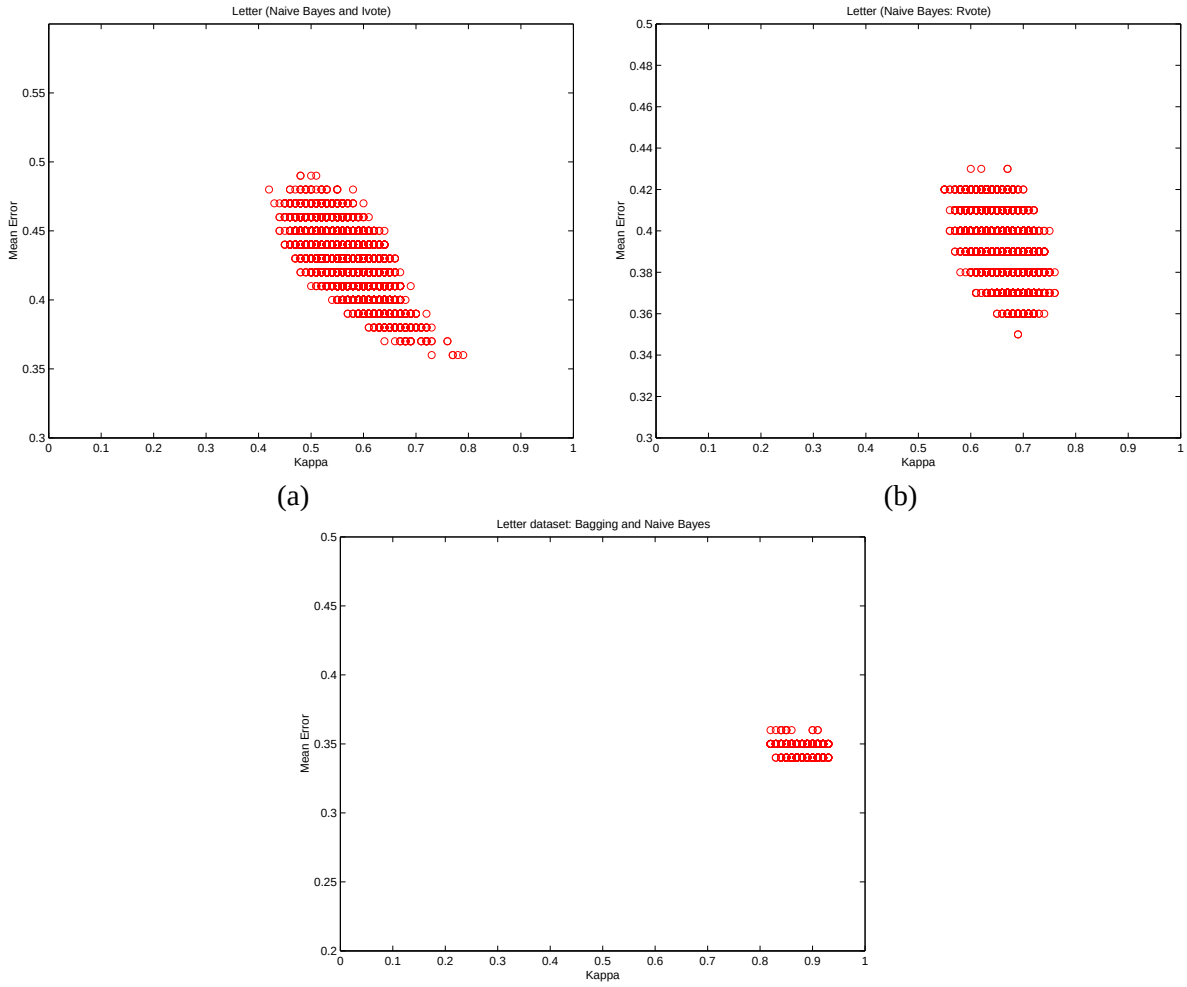


Figure 21: a) κ plot for Naive Bayes and Ivote on the letter data set. b) κ plot for Naive Bayes and Rvote on the letter data set. c) κ plot for Naive Bayes and bagging on the letter data set.

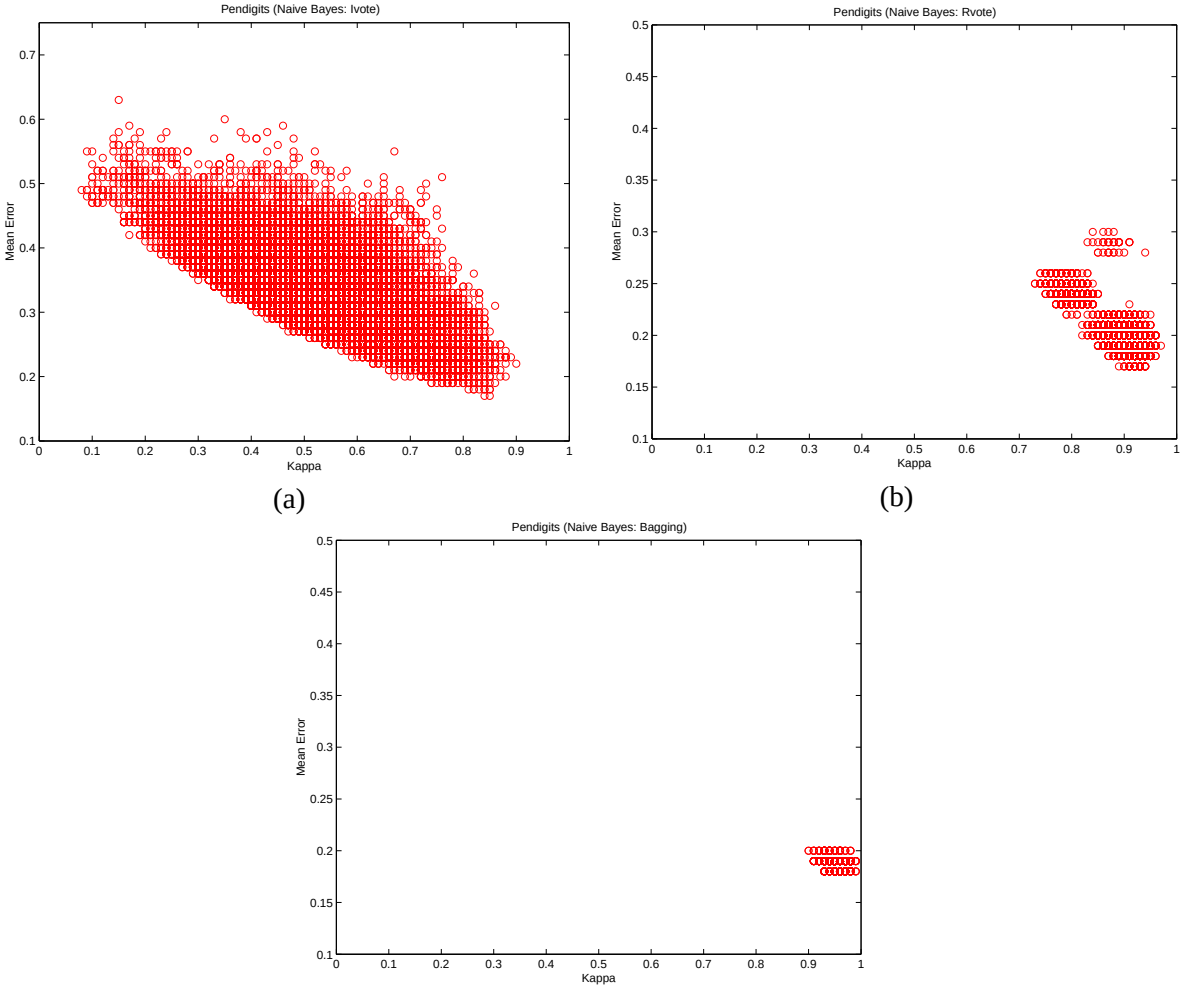


Figure 22: a) κ plot for Naive Bayes and Ivote on the pendigits data set. b) κ plot for Naive Bayes and Rvote on the pendigits data set. c) κ plot for Naive Bayes and bagging on the pendigits data set.

large data sets. Data sets too large to be handled practically in the memory of a typical computer are appropriately handled by simple partitioning into disjoint subsets, and adding another level of learning by pasting DIvotes or DRvotes on each of the disjoint subsets. We show that for almost all the data sets, using decision trees and neural networks, DIvote achieves classification accuracy comparable to Ivote (sometimes better), and almost always better than learning a single classifier. We evaluated DIvote and Ivote on data sets coming from various domains, and for almost all the cases they achieve significantly better classification accuracies than a single classifier.

DIvote is scalable as it never requires the learning set size to be greater than a very small proportion of the entire training set. Each processor works independently, without requiring communication at any stage of learning. The end result is an ensemble of hundreds of DIvote classifiers. We also conclude that pasting DIvotes is more accurate than pasting DRvotes. We believe that the combined effects of diversity, good coverage, and importance sampling are helping DIvote and Ivote. It is significant that DIvote does much better than the simplistic version of pasting small votes in a distributed scenario. Moreover, we observe that with DIvote and Ivote, the accuracy grows fast initially during learning, and then slowly plateaus. Particularly, with neural networks fewer iterations of DIvoting are needed to vastly improve over learning a single classifier from the entire training set. Neural networks, particularly, can get slowed down significantly in the training phase when learning from very large data sets. DIvote is a promising approach to reduce the learning time with neural networks even on very large data sets.

Using κ (diversity) plots, we support the theory that given an ensemble of diverse classifiers, an improvement in the accuracy can be observed. We note that DIvote provides the most significant improvement of almost 18% (relative improvement of 34%) in the difficult, non-homogeneous Protein Prediction problem over learning a single classifier using C4.5. Thus, the ensemble of diverse DIvote classifiers counters the effect of overfitting and improves the generalization.

Another important contribution of our work is comparison to the distributed boosting approach. We conclude that DIvote achieves classification accuracies similar to the distributed boosting approach, with no inter-processor communication at any stage of learning.

The DIvote framework is naturally applicable to the scenario in which data sets for a problem are already distributed. At each of the distributed sites multiple classifiers can be built, and the only communication required is the learned classifiers at the end of training. Inter-processor communication can become a bottleneck if an exchange of data is required across various nodes or processors. Moreover, one can easily conceptualize the applicability of DIvote on a cluster of workstations in a lab, where each workstation independently works on a part of the problem in its main memory.

Acknowledgments

This work was supported in part by the United States Department of Energy through the Sandia National Laboratories ASCI VIEWS Data Discovery Program, contract number DE-AC04-76DO00789. This work was also supported in part by National Science Foundation under grant EIA-0130768. We would like to thank the reviewers for their useful comments. We would like to thank Robert Banfield for his help with the experiment on ASCI Blue Supercomputer. We also thank Aleksander Lazarevic for providing us the training and testing sets of the data sets used in the distributed boosting paper.

References

Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 2001. ACM.

R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. A new ensemble diversity measure applied to thinning ensembles. In *Multiple Classifier Systems Workshop*, pages 306–316, Surrey, UK, 2003.

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1,2), 1999.

H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000. http://www.pdb.org/.

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

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Pasting bites together for prediction in large data sets. *Machine Learning*, 36(2): 85–103, 1999.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

P. Chan and S. Stolfo. Towards parallel and distributed learning by meta-learning. In *Working Notes AAAI Workshop on Knowledge Discovery and Databases*, pages 227–240, San Mateo, CA, 1993.

N. V. Chawla, S. Eschrich, and L. O. Hall. Creating ensembles of classifiers. In *First IEEE International Conference on Data Mining*, pages 581 – 583, San Jose, CA, 2000.

N. V. Chawla, L. O. Hall, K. W. Bowyer, T. E. Moore, and W. P. Kegelmeyer. Distributed pasting of small votes. In *Third International Workshop on Multiple Classifier Systems*, pages 52 – 61, Cagliari, Italy, 2002a.

N. V. Chawla, T. E. Moore, L. O. Hall, K. W. Bowyer, C. Springer, and W. P. Kegelmeyer. Distributed learning with bagging-like performance. *Pattern Recognition Letters*, 24(1-3):455 – 471, 2002b.

N. V. Chawla, T. E. Moore, Jr., L. O. Hall, K. W. Bowyer, W. P. Kegelmeyer, and C. Springer. Investigation of bagging-like effects and decision trees versus neural nets in protein secondary structure prediction. In *ACM SIGKDD Workshop on Data Mining in Bio-Informatics*, San Francisco, CA, 2001.

D. J. Spiegelhalter D. Michie and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

T. Dietterich. An empirical comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. *Machine Learning*, 40(2):139 – 157, 2000.

P. Domingos. Using partitioning to speed up specific-to-general rule induction. In *AAAI Workshop on Integrating Multiple Learned Models*, pages 29–34, Portland, OR, 1996.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

S. Eschrich, N. V. Chawla, and L. O. Hall. Learning to predict in complex biological domains. *Journal of System Simulation*, 2:1464 – 1471, 2002.

S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, Vancouver, Canada, 1990. Morgan Kaufmann.

U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery and Data Mining*, chapter From data mining to knowledge discovery: An overview. AAAI Press, Menlo Park, CA, 1996.

Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, Bari, Italy, 1996.

G. Giacinto and F. Roli. An approach to automatic design of multiple classifier systems. *Pattern Recognition Letters*, 22:25 – 33, 2001.

I. J. Good. *The Estimation of Probabilities: An essay on modern Bayesian methods*. MIT Press, 1965.

L. O. Hall, K. W. Bowyer, N. V. Chawla, T. E. Moore, and W. P. Kegelmeyer. Avatar: Adaptive Visualization Aid for Touring and Recovery. Technical Report SAND2000-8203, Sandia National Labs, 2000.

L. O. Hall, N. V. Chawla, K. W. Bowyer, and W.P Kegelmeyer. Learning rules from distributed data. In *Workshop of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 1999.

T. Ho. Random subspace method for constructing decision forests. *IEEE Transactions on PAMI*, 20(8):832 – 844, 1998.

D. T. Jones. Protein secondary structure prediction based on decision-specific scoring matrices. *Journal of Molecular Biology*, 292:195–202, 1999.

L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51:181 – 207, 2003.

L. Kuncheva, C. Whitaker, C. Shipp, and R. Duin. Is independence good for combining classifiers? In *Proceedings of 15th International Conference on Pattern Recognition*, pages 168 – 171, Barcelona, Spain, September 2000.

P. Latinne, O. Debeir, and C. Decaestecker. Limiting the number of trees in random forests. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems, Second International Workshop*, pages 178 – 187, Cambridge, UK, 2001. Springer.

A. Lazarevic and Z. Obradovic. Boosting algorithms for parallel and distributed learning. *Distributed and Parallel Databases Journal, Special Issue on Parallel and Distributed Data Mining*, 11:203 – 229, 2002.

N. Leavitt. Data mining for the corporate masses. In *IEEE Computer*. IEEE Computer Society, May 2002.

Lawrence Livermore National Laboratories. ASCI Blue Pacific. *http://www.llnl.gov/asci/platforms/bluepac*.

Lawrence Livermore National Laboratories. Protein Structure Prediction Center. *http://predictioncenter.llnl.gov/*, 1999.

R. Musick, J. Catlett, and S. Russell. Decision theoretic subsampling for induction on large databases. In *Proceedings of Tenth International Conference on Machine Learning*, pages 212 – 219, Amherst, MA, 1993.

C. Perlich, F. Provost, and J. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research*, 4:211–255, 2003.

F. Provost and D. N. Hennessy. Scaling up: Distributed machine learning with cooperation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI '96*, pages 74–79, Portland, Oregon, 1996.

F. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131–169, 1999.

F. J. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 23–32, San Diego, CA, 1999.

J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1992.

D. B. Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *AAAI Integrating Multiple Learned Models Workshop*, Portland, Oregon, 1996.

W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of seventh International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001. San Francisco, CA.

Journal of Machine Learning Research 5 (2004) 453–471 Submitted 4/03; Revised 10/03; Published 5/04

Robust Principal Component Analysis with Adaptive Selection for Tuning Parameters

Isao Higuchi HIGUCHI@AMATH.HIROSHIMA-U.AC.JP
Department of Applied Mathematics
Hiroshima University
1-4-1, Kagamiyama
Higashi-Hiroshima 739-8527, Japan

Shinto Eguchi EGUCHI@ISM.AC.JP
Institute of Statistical Mathematics and
Graduate University of Advanced Studies
4-6-7, Minami-Azabu, Minato-ku
Tokyo 106-8569, Japan

Editor: David Madigan

Abstract

The present paper discusses robustness against outliers in a principal component analysis (PCA). We propose a class of procedures for PCA based on the minimum psi principle, which unifies various approaches, including the classical procedure and recently proposed procedures. The reweighted matrix algorithm for off-line data and the gradient algorithm for on-line data are both investigated with respect to robustness. The reweighted matrix algorithm is shown to satisfy a desirable property with local convergence, and the on-line gradient algorithm is shown to satisfy an asymptotical stability of convergence. Some procedures in the class involve tuning parameters, which control sensitivity to outliers. We propose a shape-adaptive selection rule for tuning parameters using K-fold cross validation.

Keywords: K-fold cross validation, on-line algorithm, reweighted matrix algorithm, influence function, data contamination

1. Introduction

In both neural network and statistical studies, PCA is one of the most fundamental tools of dimensionality reduction for extracting effective features from high-dimensional vectors of input data. See Croux and Haesbroeck (2000) and De la Torre and Black (2001) for recent discussions. PCA is implemented by projecting input data onto the most informative subspace of lower dimension so that the hidden structure behind the input data may be clarified. The procedure of detecting principal components from input data in an on-line manner is related to the mechanism of a single neuron by the Hebbian adaptation rule, for which the learning theory has been discussed (Amari, 1977; Oja, 1982).

One of the frequently occurring difficulties in PCA is that a few outliers give disturbance in finding the effective features in a bulk of input vectors. The usual PCA satisfies the statistical optimality only under the assumption of a Gaussian distribution for all of the input data. A small departure from the assumption produces a gross error in the performance of the principal component

vector or subspace in the PCA. This motivates our study of robust PCA procedures. In what follows, we discuss an ε-contamination model for a data distribution F_ε defined by

$$F_\varepsilon = (1-\varepsilon)N(\mu,V) + \varepsilon H, \tag{1}$$

where $N(\mu,V)$ is a Gaussian distribution with mean vector μ and covariance matrix V, and H is a distribution of possible outliers. In this modeling, it is assumed that ε, or the probability of outliers, is small and that the distribution H is unspecified but qualitatively different from the supposed distribution $N(\mu,V)$. Thus, the model (1) lies in a kind of ε-neighborhood surrounding $N(\mu,V)$ taking into account all of the possible distributions H. If H has the mean vector μ_H and the covariance matrix V_H, then the data distribution F_ε has the covariance matrix

$$V_\varepsilon = (1-\varepsilon)V + \varepsilon V_H + \varepsilon(1-\varepsilon)(\mu-\mu_H)(\mu-\mu_H)^T. \tag{2}$$

Thus, the classical procedure works properly as long as $V_H \simeq V$ and $\mu_H \simeq \mu$, because the procedure basically searches for the dominant eigenvectors of V_ε by learning from input data generated by the assumed distribution F_ε. However, even if the probability ε is quite small, the classical procedure often breaks down when input data have a distribution F_ε, such that μ_H or V_H is far from the assumed μ or V, as observed from (2).

A variety of outlier distributions H have infinite dimensionality, and the simplest candidate is a point-mass distribution $\delta_\xi(x)$ degenerated at $x=\xi$. This choice corresponds to a situation in which the outliers occur deterministically in a singleton ξ with probability ε. The influence function of a procedure in the PCA is defined as the derivative at $\varepsilon = 0$ of the procedure under F_ε in (1) with $H = \delta_\xi$. This concept will be more explicitly explored in a subsequent discussion. See Higuchi and Eguchi (1998) for the case of the PCA, and see Hampel (1974) and Hampel *et al.* (1986) for the general case.

We discuss a class of principal component analyzers defined using generic functions which contain tuning parameters. For example if we adopt a log-sigmoidal function as a generic function, the tuning parameters are the inverse temperature and saturation value parameters, as will be discussed in detail. In general the tuning parameter set makes a delicate trade between loss of information and degree of insensitivity to outliers. The main objective in the present paper is to provide a reasonable selection of tuning parameters of principal component analyzers. The basic idea is to craft a loss function that reflects as appropriate trade off between loss of information and robustness to outliers. We introduce K-fold cross validation for estimating the expected loss based on a given data set. As a result we build a method of data-adaptive selection of tuning parameters. In a simulation study, we examine the performance of the adaptive selection under three types of outlier distributions H displaying deterministic, structural and distributional contaminations based on (2). The three types of outliers are simulated in a numerical experiment, and we test the performance in a few cases of principal component analyzers. We provide an S implementation of the basic robust PCA at `http://home.hiroshima-u.ac.jp/oxbow/RobustPCA/`.

The present paper is organized as follows. Section 2 introduces a class of procedures in PCA derived by the minimum psi method. In Section 3 we discuss the robustness of the procedure in the class, and in Section 4 we present an adaptive method of selecting tuning parameters. Finally, in Section 5 we provide the results of a simple simulation study to validate our theoretical discussion in previous sections and tests numerical behavior in three types of departures from the Gaussian distributional assumption.

2. A Class of Principal Component Vectors

In this section we propose a class of principal component vectors. In general, PCA aims to extract the most informative k-dimensional output vector y from an input vector x of p-dimension. This is achieved by learning the matrix Γ which connects x to $y = \Gamma^{\mathrm{T}}(x-\mu)$ based on input data $\{x_t; t = 1,2,\cdots\}$, where μ is a vector of center of the input data and Γ is a $p \times k$ orthonormal matrix, or $\Gamma^{\mathrm{T}}\Gamma = I$ (the k-identity matrix). In neural networks, Γ is interpreted as the matrix of coefficients connecting p neurons to k neurons, where a learning process works by renewing Γ according to a batch of inputs in an off-line manner or sequential input vectors in an on-line manner (Oja, 1982, 1989 and §8 in Haykin, 1999). By combining these approaches, we propose a certain class of procedures for PCA.

We present a concise review of the classical PCA for detecting the principal k-subspace. Let

$$z(x,\mu,\Gamma) = \frac{1}{2}\{\|x-\mu\|^2 - \|\Gamma^{\mathrm{T}}(x-\mu)\|^2\} \tag{3}$$

be half the square of the residual distance of $x-\mu$ from the subspace spanned by the columns of Γ. We note that $z(x,\mu,\Gamma) = \frac{1}{2}\min_{\beta\in R^k}\|x-\mu-\Gamma\beta\|^2$. See Hotelling (1933) for the original derivation. The classical PCA is simply given by minimizing

$$\frac{1}{n}\sum_{t=1}^{n} z(x_t,\mu,\Gamma)$$

with respect to μ and Γ, which reduces to solving k dominant eigenvectors of the sample covariance matrix

$$S = \frac{1}{n}\sum_{t=1}^{n}(x_t - \bar{\mu})(x_t - \bar{\mu})^{\mathrm{T}}, \tag{4}$$

where the centralized vector $\bar{\mu}$ is given by $\sum_{t=1}^{n} x_t/n$. Thus, we obtain a solution Γ by stacking the k dominant eigenvectors of S, which we write in the form

$$\Gamma = \mathrm{eigen}(S).$$

We propose a variant of this classical procedure for PCA obtained by minimizing an objective function

$$\mathcal{E}(\mu,\Gamma) = \frac{1}{n}\sum_{t=1}^{n}\Psi(z(x_t,\mu,\Gamma)), \tag{5}$$

where $\Psi(z)$ is assumed to be a monotonic increasing function of $z > 0$. Various Ψ yield various procedures for PCA. As typical examples, the identity function $\Psi_0(z) = z$ reduces to the classical PCA and

$$\Psi_1(z) = \log\frac{1}{1+\exp\{-\beta(z-\eta)\}} \tag{6}$$

defines Xu and Yuille's self-organizing rule, where β and η are tuning parameters, referred to as the inverse temperature and saturation value, respectively (Xu and Yuille, 1995). Another possible function is

$$\Psi_2(z) = \frac{1-\exp(-\beta z)}{\beta}. \tag{7}$$

In general, Ψ is interpreted as the generic function which gives the total function $\mathcal{E}$, and we refer to the minimization of $\mathcal{E}$ in (5) as the "minimum psi principle generated by Ψ".

Based on an argument similar to that of the classical PCA, we observe that the minimizer $(\tilde{\mu}, \tilde{\Gamma})$ of $\mathcal{E}(\mu, \Gamma)$ satisfies the stationary equations

$$\tilde{\mu} = \sum_{t=1}^{n} p_t(\tilde{\mu}, \tilde{\Gamma}) x_t, \tag{8}$$

$$\tilde{\Gamma} = \text{eigen}(S(\tilde{\mu}, \tilde{\Gamma})), \tag{9}$$

where

$$p_t(\mu, \Gamma) = \frac{\psi(z(x_t, \mu, \Gamma))}{\sum_{s=1}^{n} \psi(z(x_s, \mu, \Gamma))},$$

$$S(\mu, \Gamma) = \sum_{t=1}^{n} p_t(\mu, \Gamma)(x_t - \mu)(x_t - \mu)^{\mathrm{T}}, \tag{10}$$

with $\psi(z) = (\partial/\partial z)\Psi(z)$. Thus, the equilibrium point $(\tilde{\mu}, \tilde{\Gamma})$ is expressed by the weighted mean and the covariance matrix, where the weight function p_t depends upon $\tilde{\mu}$ and $\tilde{\Gamma}$, except for the case of $\psi(z) = 1$, which yields the classical procedure. In effect, (8) is determined only up to the addition of a vector in the subspace associated with $\tilde{\Gamma}$. Thus $\mu^* = \tilde{\mu} + \gamma$ is also a solution of (8) if $\gamma \in \text{Im}(\tilde{\Gamma}) \equiv \{\tilde{\Gamma} c | c \in R^k\}$, because $\mu^* + \text{Im}(\tilde{\Gamma}) = \tilde{\mu} + \text{Im}(\tilde{\Gamma})$. However, (9) is independent of the choice of possible solutions, so we adopt the expression (8) for convenience.

In PCA research, centralization of input data by a vector other than a sample mean has not been considered. In the present paper, we investigate the problem of centralization and explore the usefulness of a method using a vector such as (8). In conventional robust statistics, the estimation of location has been studied extensively. (See, for example, Huber, 1981.) Our estimator $\tilde{\mu}$ can be viewed as one of several variants for robust estimation. However, our main objective concerning $\tilde{\mu}$ is not the location estimation itself, but rather the data centralization for the extraction of principal components. Thus, $\tilde{\mu}$ is naturally linked to $\tilde{\Gamma}$ in the optimization of $\mathcal{E}(\mu, \Gamma)$.

For a batch of data $\{x_t : 1 \le t \le n\}$, we propose a fixed-point algorithm. See Hyvarinen and Oja (1997) for the related discussion on a fixed-point algorithm for ICA. This algorithm alternates two steps associated with the stationary equations (8) and (9) in the following:

Step 1: Given (μ_1, Γ_1), calculate

$$p_t^{(1)} = \frac{\psi(z(x_t, \mu_1, \Gamma_1))}{\sum_{s=1}^{n} \psi(z(x_s, \mu_1, \Gamma_1))}.$$

Step 2: Using the estimated $\{p_t^{(1)}\}$ in step 1, perform the same task as in the classical PCA:

$$\mu_2 = \sum_{t=1}^{n} p_t^{(1)} x_t \quad \text{and}$$

$$\Gamma_2 = \text{eigen}(S^{(1)}), \tag{11}$$

where $S^{(1)}$ is a weighted matrix defined by $\sum p_t^{(1)}(x_t - \mu_2)(x_t - \mu_2)^{\mathrm{T}}$. In this way, the algorithm alternates between two steps, and we refer to this as the reweighted matrix (RM) algorithm.

Assuming hereafter that the generic function $\Psi(z)$ is strictly concave in z, we have

$$\mathcal{E}(\mu_2,\Gamma_2)-\mathcal{E}(\mu_1,\Gamma_1)<\frac{\sum_{t=1}^n\psi(z(x_t,\mu_1,\Gamma_1))}{n}\{\sum_{t=1}^n p_t^{(1)}z(x_t,\mu_2,\Gamma_2)-\sum_{t=1}^n p_t^{(1)}z(x_t,\mu_1,\Gamma_1)\}$$

because, by assumption, $\Psi(z_2)-\Psi(z_1)<\psi(z_1)(z_2-z_1)$ for $z_1<z_2$. In step 2, the procedure is equivalent to minimization of

$$\sum p_t^{(1)}z(x_t,\mu,\Gamma)$$

with respect to (μ,Γ). Therefore, we conclude that the RM algorithm generated by $\{(\mu_j,\Gamma_j):j\geq 1\}$ is responsible for the strict decrease of the objective function

$$\mathcal{E}(\mu_1,\Gamma_1)>\cdots>\mathcal{E}(\mu_j,\Gamma_j)>\cdots.$$

This desirable property is mathematically the same as that of the EM algorithm. The possible region of (μ,Γ) that the algorithm works in is $\mathcal{X}\times O_{p,k}$, where $\mathcal{X}$ is the convex hull of data and $O_{p,k}$ is the space of $p\times k$ orthonormal matrices. We can easily check a condition for the convergence to the solution of the equations such that a set

$$\{(\mu,\Gamma)\in\mathcal{X}\times O_{p,k}:\mathcal{E}(\mu,\Gamma)\leq c\}$$

is compact for any fixed $c\leq\mathcal{E}(\mu_1,\Gamma_1)$. This is referred to as the regularity condition of Wu (1983), found in §3.4.2 of McLachlan and Krishnan (1997), and this condition implies that the sequence $\{(\mu_j,\Gamma_j):j\geq 1\}$ is convergent to a set of solutions of equations (8) and (9). However, even when this regularity condition holds, the computational complexity with high dimensional data may be prohibitive since each iteration requires the solution of (11).

2.1 Stability of On-line Gradient Algorithm

We next discuss the on-line gradient algorithm. The gradient vector of the objective function is the sum of

$$G(\mu,\Gamma)(x_t)=\psi(z(x_t,\mu,\Gamma))G_1(\mu,\Gamma)(x_t)$$

over $t=1,2,\cdots$, where

$$G_1(\mu,\Gamma)(x)=\left[\begin{array}{c} x-\mu \\ (x-\mu)(x-\mu)^{\mathrm{T}}\Gamma-\Gamma\,\mathcal{LT}[yy^{\mathrm{T}}]\end{array}\right]$$

with $y=\Gamma^{\mathrm{T}}(x-\mu)$, where the operator $\mathcal{LT}[\cdot]$ sets all of the elements above the diagonal of its matrix argument to zero. Hence, the on-line gradient algorithm is given by

$$\left[\begin{array}{c}\mu_{t+1}\\ \Gamma_{t+1}\end{array}\right]=\left[\begin{array}{c}\mu_t\\ \Gamma_t\end{array}\right]+r_tG(\mu_t,\Gamma_t)(x_t) \tag{12}$$

for $t=1,2,\cdots$ with a learning rate r_t. If we apply the classical procedure this algorithm reduces to the Oja algorithm (1982). See §8 in Haykin (1999) for the related algorithmic developments in PCA. The gradient algorithm (12) is different from the Oja algorithm only with respect to the factor $\psi(z)$, which depends on the t-step (μ_t,Γ_t) and the t-th example x_t through $z=z(x_t,\mu_t,\Gamma_t)$ defined in (3). In the classical procedure, the μ-part of the algorithm (12) reduces to the usual

centralization and has no connection to Γ. However, in the case of a non-constant weight function $\psi(z)$, the μ-part is essentially connected not only to μ itself, but also to Γ through the z-variable. For the case of non-constant $\psi(z)$, we confine ourselves to the Xu-Yuille rule, which is generated by $\psi_1(z) = \beta/(1+e^{\beta(z-\eta)})$. Xu and Yuille implemented the on-line algorithm in the same fashion as (12) for the Γ-part, but the centralizing mean $\bar{\mu}$ was used for the μ-part. We will make a simple comparison between the two methods for the μ-part in a subsequent discussion.

The on-line gradient algorithm does not satisfy the property of uniform decrease of the objective function possessed by the reweighted algorithm as shown above. We first discuss the asymptotic convergence of (12) for the case of $k = 1$, $\Gamma = \gamma$. See §8.4 in Haykin (1999) for the proof for the classical procedure. The on-line gradient algorithm (12) is a special case of the generic stochastic approximation algorithm

$$\begin{bmatrix} \mu_{t+1} \\ \gamma_{t+1} \end{bmatrix} = \begin{bmatrix} \mu_t \\ \gamma_t \end{bmatrix} + r_t h(\gamma_t, \mu_t, x_t), \tag{13}$$

where

$$h(\gamma, \mu, x) = \psi(z(x,\mu,\gamma)) \begin{bmatrix} x-\mu \\ (x-\mu)(x-\mu)^{\mathrm{T}}\gamma - \{\gamma^{\mathrm{T}}(x-\mu)(x-\mu)^{\mathrm{T}}\gamma\}\gamma \end{bmatrix}.$$

If $\psi(z) \equiv 1$, then (13) leads to the classical PCA. It is assumed that ψ is a finite function, so the convergence is proved using an argument similar to that used in the case of classical PCA.

Take the expectation of $h(\gamma_t, \mu_t, x_t)$ over x, and then in the limit we have

$$\begin{aligned} \bar{h}(\gamma_\infty, \mu_\infty) &= \lim_{t\to\infty} E[h(\gamma_t, \mu_t, x_t)] \\ &= \begin{bmatrix} m(\gamma_\infty, \mu_\infty) - \kappa(\gamma_\infty, \mu_\infty)\mu_\infty \\ R(\gamma_\infty, \mu_\infty)\gamma_\infty - \{\gamma_\infty^{\mathrm{T}} R(\gamma_\infty, \mu_\infty)\gamma_\infty\}\gamma_\infty \end{bmatrix}, \end{aligned}$$

where

$$\kappa(\gamma, \mu) = E\{\psi(z(x,\mu,\gamma))\}, \quad m(\gamma,\mu) = E\{\psi(z(x,\mu,\gamma))x\}$$

and

$$R(\gamma, \mu) = E[\psi(z(x,\mu,\gamma))(x-\mu)(x-\mu)^{\mathrm{T}}].$$

Thus, our differential equation is

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} \mu_t \\ \gamma_t \end{bmatrix} &= \bar{h}(\gamma_t) \\ &= \begin{bmatrix} m(\gamma_t, \mu_t) - \kappa(\gamma_t, \mu_t)\mu_t \\ R(\gamma_t, \mu_\infty)\gamma_t - \{\gamma_t^{\mathrm{T}} R(\gamma_t, \mu_\infty)\gamma_t\}\gamma_t \end{bmatrix}. \end{aligned} \tag{14}$$

In the μ-part, we observe that μ_t behaves asymptotically as $e^{-\kappa_\infty t}a + m_\infty/\kappa_\infty$, which implies that m_∞/κ_∞ is the stable-point, where $\kappa_\infty = \kappa(\gamma_\infty, \mu_\infty)$ and $m_\infty = m(\gamma_\infty, \mu_\infty)$. Therefore, we consider only

$$\frac{d}{dt}\gamma_t = R(\gamma_t, \mu_\infty)\gamma_t - \{\gamma_t R(\gamma_t, \mu_\infty)\gamma_t\}\gamma_t.$$

We expand γ_t in terms of the set of eigenvectors $\{q_k(\infty) : k = 1, \cdots, p\}$ of $R(\gamma_\infty, \mu_\infty)$ with the dominant eigenvector $q_1(\infty)$ as follows:

$$\gamma_t = \sum \theta_k(t) q_k(\infty),$$

Let us decompose $\bar{h}(\gamma)$ into $\bar{h}_1(\gamma,\gamma_\infty)+\bar{h}_2(\gamma,\gamma_\infty)$, where

$$\begin{aligned}\bar{h}_1(\gamma,\gamma_\infty) &= R(\gamma_\infty,\mu_\infty)\gamma-\{\gamma^{\mathrm{T}}R(\gamma_\infty,\mu_\infty)\gamma\}\gamma \\ \bar{h}_2(\gamma,\gamma_\infty) &= \{R(\gamma,\mu_\infty)-R(\gamma_\infty,\mu_\infty)\}\gamma \\ &\quad -[\gamma^{\mathrm{T}}\{R(\gamma,\mu_\infty)-R(\gamma_\infty,\mu_\infty)\}\gamma]\gamma.\end{aligned} \tag{15}$$

Then the equilibrium condition $\bar{h}_1(\gamma_\infty,\gamma_\infty)=0$ implies that γ_∞ reduces to one of k eigenvectors of $R(\gamma_\infty,\mu_\infty)$; $h_2(\gamma_\infty,\gamma_\infty)=0$ holds identically. The differential equation

$$\frac{d}{dt}\gamma_t=\bar{h}_1(\gamma_t,\gamma_\infty)$$

has an asymptotically stable point $q_1(\infty)$ through the same discussion established in §8.4 in Haykin (1999). Hence the differential equation (14) leads to stable convergence to $q_1(\infty)$.

Secondly, we observe that the case of k principal component vectors also satisfies the stable convergence, noting that our differential equation is

$$\frac{d}{dt}\Gamma_t=R(\Gamma_t,\mu_\infty)\Gamma_t-\mathcal{LT}[\Gamma_t^{\mathrm{T}}R(\Gamma_t,\mu_\infty)\Gamma_t]\Gamma_t,$$

where

$$R(\Gamma,\mu)=E\{\psi(z(x,\mu,\Gamma))(x-\mu)(x-\mu)^{\mathrm{T}}\}.$$

In effect, the RM algorithm is applicable to on-line data by solving the eigen problem for batch data with a new observation incorporated in each step. The computational burden is quite heavy relative to the on-line gradient algorithm, but we will pursue more rapid convergence property in a simulation study.

In the statistical literature another type of PCA methods has been proposed by minimizing

$$\frac{1}{n}\sum_{t=1}^{n}\Psi(d(x_t,\mu,V))$$

with respect to (μ,V), where d is Mahalanobis squared distance, that is,

$$d(x_t,\mu,V)=\frac{1}{2}(x_t-\mu)^{\mathrm{T}}V^{-1}(x_t-\mu).$$

See Campbell(1980), Devlin *et al.* (1981), Caussinus and Ruiz (1990), and Croux and Haesbroeck (2000). The use of the nonlinear generic function Ψ is the same, but the essential difference is that our method aims at estimating the principal component vectors rather than estimating the scatter matrix V. One advantage of our method is that it does not need all the information of V. In fact only the first k dominant eigenvalues and the corresponding eigenvectors are needed in the algorithm, which is easily implemented by the singular-value decomposition algorithm even if the data set is of high dimension.

3. Robustness of the Proposed Principal Component Vectors

Data analysts have frequently found that the classical PCA breaks down in the presence of outliers. It can happen that a single outlier changes the principal component subspace into the orthogonal

complement. As a result, the PCA fails to capture an important feature of the bulk of the data, which will be observed from a simple simulation study in Section 5. In the statistical literature, the robustification of the classical likelihood-based procedures has been discussed and well established: see Huber (1981) for some notions on robustness. Such contamination is typically expressed by the ε -contamination model,

$$(1-\varepsilon)N(\mu,V)(x)+\varepsilon\delta_{\xi}(x),$$

with the point-mass distribution δ_{ξ} as discussed in the Introduction. In the expression, ε is undetectably small; nevertheless, the likelihood procedure based on the density function of $N(\mu,V)$ may sometimes break down for an extreme vector ξ. We next explore which first principal component vector or principal subspace is robust against outliers in our class. First, we consider the case of the first principal component vector, or $k=1$. In Higuchi and Eguchi (1998), the influence function of the Xu and Yuille rule defined by Ψ_1 in (6) is given by

$$\mathrm{IF}_{\Psi_1}(\xi)=-\psi_1(z(\xi,\mu,\gamma_1))\tilde{\gamma}_1^{\mathrm{T}}(\xi-\mu)\sum_{j=2}^{p}\frac{\hat{\lambda}_j\tilde{\gamma}_j^{\mathrm{T}}(\xi-\mu)}{\tilde{\lambda}_j(\hat{\lambda}_j-\hat{\lambda}_1)}\tilde{\gamma}_j,$$

where $(\hat{\lambda}_j,\hat{\gamma}_j)$ is the pair of the j-th dominant eigenvalue and its associated eigenvector of S defined at (4) and $(\tilde{\lambda}_j,\tilde{\gamma}_j)$ is that of $S(\mu,\Gamma)$ defined at (10), and

$$\psi_1(z)=\frac{\partial\Psi_1(z)}{\partial z}=\frac{\beta}{1+\exp\{\beta(z-\eta)\}}. \tag{16}$$

The influence function assesses the effect on the principal component subspace of the contamination of the data $\{x_t|1\le t\le n\}$ by the outlier ξ.

Secondly, we discuss a general case of k principal component vectors $\Gamma=(\gamma_1,\ldots,\gamma_k)$. We consider the matrix

$$P=\Gamma\Gamma^{\mathrm{T}}.$$

The matrix P is the projection operator onto the subspace spanned by the eigenvectors γ_I, see Tanaka (1988). Our estimator $\tilde{P}=\tilde{\Gamma}\tilde{\Gamma}^{\mathrm{T}}$ has the influence function

$$\mathrm{IF}_{\Psi_1}(\xi;\tilde{P})=-\psi_1(z(\xi,\tilde{\mu},\tilde{\Gamma}))\sum\frac{\hat{\lambda}_j\tilde{\gamma}_I{}^{\mathrm{T}}(\xi-\tilde{\mu})(\xi-\tilde{\mu})^{\mathrm{T}}\tilde{\gamma}_j}{\tilde{\lambda}_j(\hat{\lambda}_j-\hat{\lambda}_I)}(\tilde{\gamma}_I\tilde{\gamma}_j^{\mathrm{T}}+\tilde{\gamma}_j\tilde{\gamma}_I^{\mathrm{T}}), \tag{17}$$

where the summation is taken over $\{(I,j):I=1,\cdots,k,j=1,\cdots,p,I\neq j\}$, and we will also use this summation convention in a subsequent discussion.

The formula is valid for the minimum psi principle for a general Ψ, see Kamiya and Eguchi (2001) for a detailed discussion, as well as a discussion of relative efficiency under a Gaussian distribution. The influence function, as a function of ξ, assesses the smoothness of the principal component subspace around the supposed distribution $N(\mu,V)$. The boundedness of the influence function in ξ qualitatively guarantees robustness for the target principal component subspace.

In PCA, μ needs to be estimated in order to centralize the data before extracting the principal component subspace. This is usually estimated as $\sum x_t/n$, which can be expressed in the form of a functional $\int xdF(x)$ with $F=F_n$ (empirical distribution). This usual estimation is Fisher consistent since $\int xdF(x)=\mu$ when $F=N(\mu,V)$, but is quite sensitive to the outlier ξ, since the functional evaluated at F is $(1-\varepsilon)\mu+\varepsilon\xi$, and so the influence function is $\xi-\mu$. In contrast, we will show that

our PCA procedure automatically leads to a robust centralization of data. We typically observe an unbounded case for the usual principal component subspace, and a bounded case for Xu and Yuille's principal component subspace.

The boundedness of IF_Ψ conditional on

$$\|\tilde{\Gamma}^{\mathrm{T}}(\xi-\tilde{\mu})\|^2 \leq d^2 \tag{18}$$

with a positive constant d guarantees robustness against any outliers ξ satisfying (18) in the contamination model, cf. Higuchi and Eguchi (1998) for the justification for this robustness. Using the formula (17) with general Ψ, we find a sufficient condition for the robustness

$$\sup_{z>0} \sqrt{z}\psi(z) < \infty \tag{19}$$

with $\psi(z) = \partial\Psi(z)/\partial z$. The proof is given as follows. First, we obtain

$$\begin{aligned}
\|\mathrm{IF}_\Psi(\xi;\tilde{P})\| &\leq \psi(z(\xi,\tilde{\mu},\tilde{\Gamma}))\sum\left|\frac{\hat{\lambda}_j\tilde{\gamma}_I^{\mathrm{T}}(\xi-\tilde{\mu})(\xi-\tilde{\mu})^{\mathrm{T}}\tilde{\gamma}_j}{\tilde{\lambda}_j(\hat{\lambda}_j-\hat{\lambda}_I)}\right|\|\tilde{\gamma}_I\tilde{\gamma}_j^{\mathrm{T}}+\tilde{\gamma}_j\tilde{\gamma}_I^{\mathrm{T}}\| \\
&\leq d\sum\left|\frac{\hat{\lambda}_j\|\tilde{\gamma}_I\tilde{\gamma}_j^{\mathrm{T}}+\tilde{\gamma}_j\tilde{\gamma}_I^{\mathrm{T}}\|}{\tilde{\lambda}_j(\hat{\lambda}_j-\hat{\lambda}_I)}\right||(\xi-\tilde{\mu})^{\mathrm{T}}\tilde{\gamma}_j|\psi(z(\xi,\tilde{\mu},\tilde{\Gamma}))
\end{aligned}$$

from the assumption of (18). Since

$$z(\xi,\tilde{\mu},\tilde{\Gamma}) = \frac{1}{2}\sum_{j=k+1}^{p}\{\tilde{\gamma}_j^{\mathrm{T}}(\xi-\tilde{\mu})\}^2 \geq \frac{1}{2}\{\tilde{\gamma}_j^{\mathrm{T}}(\xi-\tilde{\mu})\}^2$$

for any j, $k+1 \leq j \leq p$ by the definition of z at (3), we obtain

$$\|\mathrm{IF}_\Psi(\xi;\tilde{P})\| \leq \sum\left|\frac{\hat{\lambda}_j\|\tilde{\gamma}_I\tilde{\gamma}_j^{\mathrm{T}}+\tilde{\gamma}_j\tilde{\gamma}_I^{\mathrm{T}}\|}{\tilde{\lambda}_j(\hat{\lambda}_j-\hat{\lambda}_I)}\right|\left(d\sqrt{2}\sqrt{z(\xi,\tilde{\mu},\tilde{\Gamma})}\psi(z(\xi,\tilde{\mu},\tilde{\Gamma}))+d^2\psi(z(\xi,\tilde{\mu},\tilde{\Gamma}))\right).$$

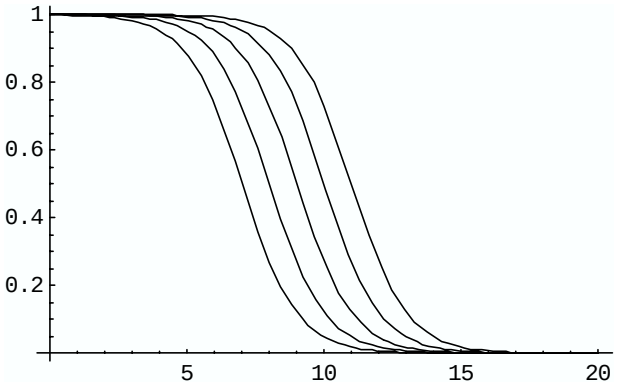
Finally, we conclude that

$$\|\mathrm{IF}_\Psi(\xi;\tilde{P})\| \leq C\left(d\sqrt{2}\sup_{z>0}\sqrt{z}\psi(z)+d^2\sup_{z>0}\psi(z)\right)$$

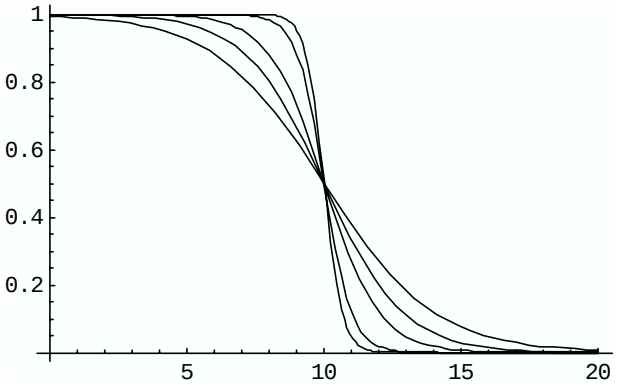
for any outlier ξ in R^p satisfying (18), where

$$C = \sum\left|\frac{\hat{\lambda}_j\|\tilde{\gamma}_I\tilde{\gamma}_j^{\mathrm{T}}+\tilde{\gamma}_j\tilde{\gamma}_I^{\mathrm{T}}\|}{\tilde{\lambda}_j(\hat{\lambda}_j-\hat{\lambda}_I)}\right|.$$

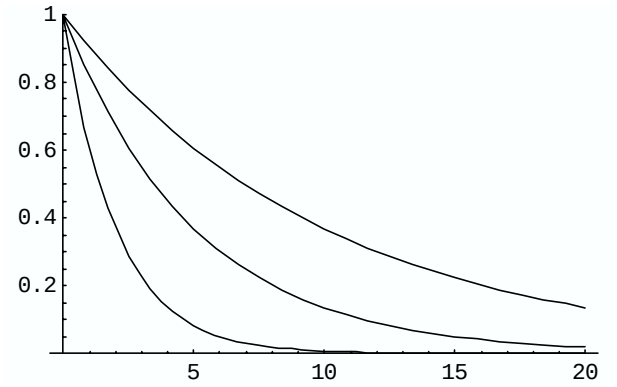
Therefore, the condition (19) for Ψ leads to the boundedness of the influence function IF_Ψ conditional on the condition (18).



(1) φ_1 with $\beta = 1,\ \eta = 7, 8, 9, 10, 11$.



(2) φ_1 with $\beta = 0.5, 0.7, 1, 2, 3,\ \eta = 10$.



(3) φ_2 with $\beta = 0.1, 0.2, 0.5$.

Figure 1: Graphs of ϕ for several tuning parameters.

4. Adaptive Selection for Tuning Parameters

Let α be a parameter in the generic function Ψ which defines our objective function (5). In this section, we focus on the role of the tuning parameter α. The performance of PCA by the proposed method generated by Ψ_α depends on the value of the tuning parameter α. As shown in (6) and (7), the generic function Ψ_1 involves tuning parameters β and η, and Ψ_2 involves β.

Thus, generic functions control the sensitivity to outliers by these tuning parameters. See Figure 1 for the graphs of the derivatives ψ_1 and ψ_2 of Ψ_1 and Ψ_2 for several tuning parameters.

The generic function Ψ_1 where $\eta \to \infty$ or $\beta \to 0$ yields the classical PCA. If we exactly assume Gaussian distribution, or $\varepsilon = 0$ in (1), then the classical PCA or $\psi(z) \equiv 1$, is recommended as the standard method. See Kamiya and Eguchi (2001) for a detailed discussion. This suggests that under the situation in which $\varepsilon \neq 0$, there exists an optimal selection for tuning parameters giving a generic function other than $\psi(z) \equiv 1$.

We propose herein a method of determining tuning parameters as in (6) and (7), based on K-fold cross-validation. See Subsection 7.10 in Hastie *et al.* (2001) for the detailed discussion. Throughout the section, we focus on batch data.

Let $F(x)$ be a data distribution which is assumed to generate input data x. Then, we adopt a generalization error function for assessing the performance of a given $\hat{\mu}$ and $\hat{\Gamma}$ using

$$L(\hat{\theta}, F) = \int \cdots \int \Psi_0(z(x, \hat{\mu}, \hat{\Gamma}))dF(x)$$

where

$$\Psi_0(z) = \log \frac{1}{1 + \exp\{-\beta_0(z \cdot \eta_0)\}}$$

with $\alpha_0 = (\beta_0, \eta_0) = (50, 10)$. This choice of the error function is intended to achieve mild robustness to outliers. This is because we cannot obtain a sensible result if the error function itself is sensitive to outliers. The empirical error function is

$$L_{\text{emp}}(\hat{\theta}) = \frac{1}{n} \sum_{t=1}^{n} \Psi_0(z(x_t, \hat{\mu}, \hat{\Gamma}))$$

for given data $\{x_1, \ldots, x_n\}$ and would be unchanged by data contamination if $(\hat{\mu}, \hat{\Gamma})$ is a robust estimator. The choice of β_0 is universal, but that of η_0 should be adaptive. For example, the median of $\|x_I - \bar{\mu}\|$ with the usual centralizing vector $\bar{\mu}$ as a default value. Given a class of estimators $\hat{\theta}_\alpha$ by the generic function Ψ_α with the tuning parameter α, for example $\alpha = (\beta, \eta)$ in (6) or (7), we attempt to estimate the expected loss function, or the risk function associated with an estimator $\hat{\theta}_\alpha$, which is essentially

$$R(\hat{\theta}_\alpha, F) = E_F\{L(\hat{\theta}_\alpha, F)\},$$

where E_F denotes the mathematical expectation when input data $x_1, \cdots, x_n$ follow from the underlying distribution F.

Here we provide a method of selecting α^* which generates $\hat{\theta}_{\alpha^*}$ with good performance. We use K-fold validation to get a estimator of the generalization error $L(\hat{\theta}_\alpha, F)$. Here, we divide the data set $D = \{x_1, \ldots, x_n\}$ into K subsets $\{D_k = \{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\} : k = 1, \cdots, K\}$ and so $D = \bigcup_{k=1}^{K} D_k$. Define

$$CV(\alpha) = \frac{1}{K} \sum_{k=1}^{K} L_{\text{emp}}^{(k)}(\hat{\theta}_\alpha^{(-k)}),$$

where $\hat{\theta}_\alpha^{(-k)}$ is the estimator based on the data set $\bigcup_{k' \neq k} D_{k'}$ and

$$L_{\mathrm{emp}}^{(k)}(\theta) = \frac{1}{n_k} \sum_{I=1}^{n_k} L(x_I^{(k)}, \theta).$$

In this way, the estimator $\hat{\theta}_\alpha$ and D_k are statistically independent, which implies elimination of the bias in the empirical error $L_{\mathrm{emp}}(\hat{\theta}_\alpha)$. In this formulation, we now define the optimal α^* by

$$\alpha^* = \operatorname*{argmin}_\alpha CV(\alpha).$$

We will explore the performance of this selection method for synthetic data situations.

5. Simulation Study

We explore the robustness of our class of principal component subspaces in numerical experiments, focusing on the classical rule ($\psi(z) = 1$), the Xu-Yuille rule and the Gaussian kernel rule defined in (7). For our simulation study, we consider the following three types of outlier distributions H in the ε-contamination model defined by (1):

(i) Deterministic contamination: a sum of point-mass distributions at $x = \xi_j$ for $j = 1, \cdots, M$.

(ii) Structural contamination: the same Gaussian distribution $N(\mu_1, V_1)$, but with the structure in μ_1 and V_1 being quite different from with that in μ and V, that is to say, $\|\mu_1 - \mu\|$ or $\mathrm{tr}(V_1 - V)^2$ is substantially large.

(iii) Distributional contamination: the same structure as in (μ, V) but the distribution is totally different from the Gaussian distribution $N(\mu, V)$.

First, we investigate the case in which ε is undetectably small, for example we take $\varepsilon = 0.03$. We have performed a numerical study for the behavior of our procedure for seven-dimensional data in the following setting: $\mu = (0, \cdots, 0)^{\mathrm{T}}, V = \mathrm{diag}(5, 2, 3, 1, 0.5, 0.5, 0.5)$. As for (i), $\xi_1 = (0, 0, 0, 0, 0, 0, b)$, $\xi_2 = (0, 0, 0, 0, 0, b, 0)$ with a probability 0.5 for each.
In (ii) the distribution H is a Gaussian distribution with

$$\mu_1 = (1, \cdots, 1)^{\mathrm{T}}, V_1 = \mathrm{diag}(0.5, 2, 3, 1, 0.5, 0.5, 0.5).$$

In (iii) the outlier, ξ has a distribution H of Cauchy-type of which the location-scatter structure is the same as (μ, V), that is, all the components of $V^{-\frac{1}{2}}(\xi - \mu)$ are independently and identically distributed according to a standard Cauchy distribution with density function $1/(\pi(1 + x^2))$.

We observe in a series of simulations in the above setting that the classical procedure ($\psi(z) = 1$) breaks down for a batch of data with most observations from $N(\mu, V)$ and a few outliers from H. The first principal component vector extracted only from the 98 simulated vectors is completely changed into the space of minor components after mixing with two outliers. In our experience, PCA has never been weakly perturbed by outlier contamination under the situation of setting (i) with a small b. Whether the classical PCA resists or completely breaks down against outliers is determined by b. If b ranges from 14 to 16, the breakdown occurs with about 50 percent proportion. If input data are simulated by setting (iii), then the classical PCA tends to break down with a higher frequency of

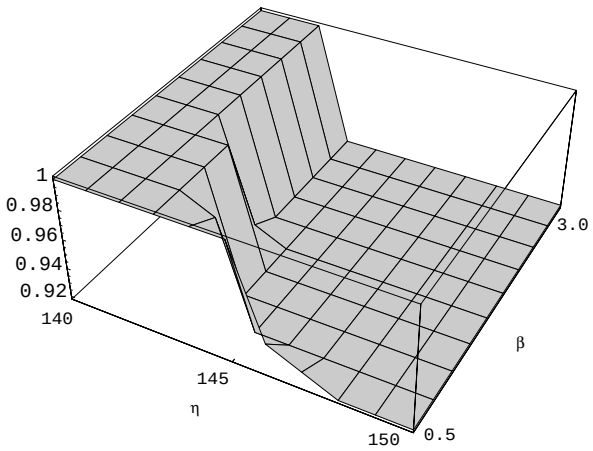


Figure 2: The plot of inner product of the PC vectors with/without outliers against β and η.

occurrence than for setting (ii). In almost all of the cases of setting (iii), the PCA breaks down. Thus, we observe that the distributional contamination is more severe than the structural contamination for the classical PCA.

We confine ourselves to a typical case of input vectors from the structural contamination. We obtained 270 input vectors of 200-dimension from $N(\mu, V)$, where

$$\mu = (0, \cdots, 0)^{\mathrm{T}}, \quad V = \mathrm{diag}(10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0.5, \cdots, 0.5),$$

and 30 outliers from $N(\mu_1, V_1)$, where

$$\mu_1 = (1, \cdots, 1)^{\mathrm{T}}, \quad V_1 = \mathrm{diag}(1, 9, 8, 7, 6, 5, 4, 3, 2, 1, 1, 1, \cdots, 1),$$

and observed that the inner product of the first principal component vectors based on the data of 270 vectors and on the data added to 30 outliers is 0.678 when using the classical PCA. On the other hand the inner product is 0.999 using procedure defined by (16) with $\beta = 0.5$ and $\eta = 130$. The RM algorithm is started with the initial vector $\gamma = (1/\sqrt{200}, \cdots, 1/\sqrt{200})^{\mathrm{T}}$ and $\mu = (0, \cdots, 0)$, which assigns less weight to the 30 outliers after 10 iterations.

In this procedure, we heuristically choose the tuning parameters β and η. We observe in Figure 2 that the performance is not so sensitive to the choice of β if $\eta < 145$. In the subsequent simulation, we will investigate the data-adaptive selection for tuning parameters using the 10-fold CV method in Section 4.

Secondly, we investigate the case in which ε is fairly large. Hence, we take $\varepsilon = 0.5$, which can be viewed as the worst case in our context. We focus on the case of structural contamination with the same setting as that with $\varepsilon = 0.03$. The situation is really an extreme case, and is beyond the usual context of outlier detection. Thus, we have several simulations involving this situation as the first step, as seen in Figure 3. A typical result gives 0.101 as the inner product between the first

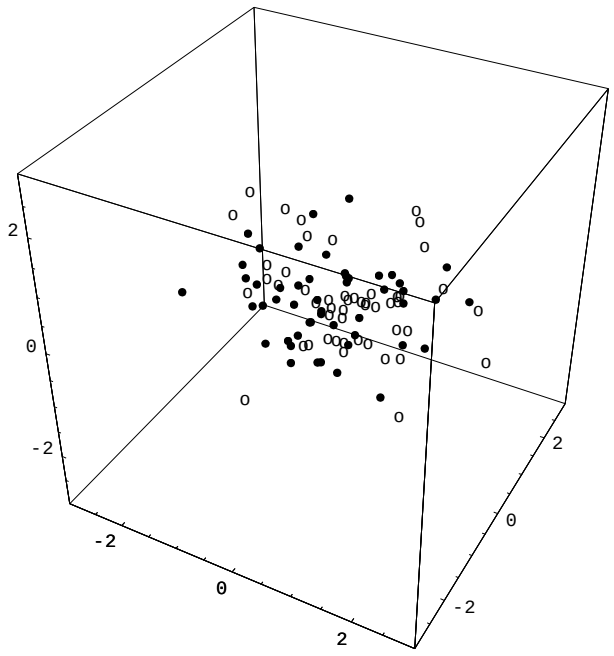


Figure 3: The plot of 50 observations and 50 outliers in the minor subspace.

principal component vectors with and without 50 outliers by the classical procedure. Alternatively our procedure gives 0.833, so we see that the procedure can detect a more sensible direction vector than the classical procedure. The proposed procedure detects the heterogeneity of structure, as indicated in Figure 4. If we have more information on the contamination or outlying structure, then we can build a shaper model for the outlier distribution. For example, we might suggest a two component Gaussian mixture model and estimate the structure in a more complete situation via the EM algorithm. However, to expect exact information on the outliers is often unrealistic in the present situation.

We apply the RM algorithm to on-line input vectors for comparison with the gradient learning algorithm. For the simulation study, we assume a specific form of model (ii) with $\varepsilon = 0.1$ and Gaussian density with $\mu = 0$, $\mu_1 = (30,0,0,0,30)^{\mathrm{T}}$, $V = \mathrm{diag}(9,7,5,3,1)$, and $V_1 = \mathrm{diag}(1,1,3,3,5)$. Thus, the true principal component vector of V is $(1,0,0,0,0)$ in the simulation design. We observe that the RM algorithm is stable and attains rapid convergence from these on-line input vectors. However, the computational burden is much heavier than the on-line gradient algorithm, as shown in Figure 5.

We next investigate the numerical performance of the 10-fold CV method discussed in Section 4 under the model of the structural contamination as follows: 45 input vectors are simulated from a Gaussian density $N(0, \mathrm{diag}(9,7,5,3,1))$ and five outliers from $N((0,0,0,0,10), \mathrm{diag}(9,7,5,3,1))$. We observe that the 10-fold CV method has detected the optimal tuning parameter $\eta = 46$, as shown in Figures 6 (1) and (2), while the PCA is much less sensitive to β than to η, so we fix the optimal tuning parameter as $\beta = 1$.

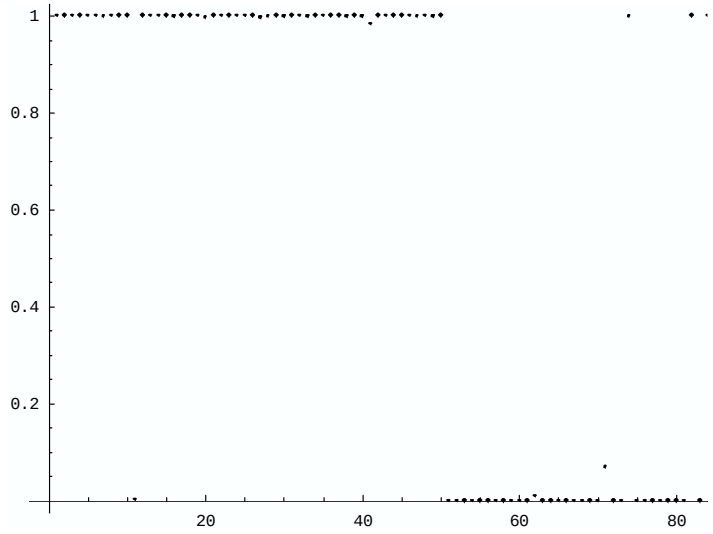


Figure 4: The plot of the weight function of ψ over 50 data with 50 outliers.

We propose a robust procedure for centralization of the data in (12). In the neural networks literature, such a variant for centralizing data has been ignored until now. Using the usual centralization is correct if all of the data are generated from Gaussian distribution. This is because the mean vector and covariance matrix are orthonormal as parameters, so that any influence on the principal vectors is independent of that on the mean vector. However, if the data in the mean vector are structured, this sometimes has a significant impact on the PCA. Here we consider a simple simulation study for investigating the difference between two procedures defined by adopting the weighted sample mean vector $\hat{\mu}$ in the RM algorithm and the sample mean vector $\bar{\mu}$ in classical PCA as the centralizer. We generate 180 observations from a Gaussian density $N(0,V)$ with $V = \mathrm{diag}(9,7,5,3,1)$ and 20 outliers from $N((0,0,20,0,0)^{\mathrm{T}},V)$. Figure 7 shows the two-dimensional score plot produced by the classical PCA based on only the 180 observations without any outliers, where the horizontal axis is taken as taken exactly as the first principal component vector. We observe that the first principal component vector by $\hat{\mu}$-centralization yields a proper direction.

6. Discussion

We have discussed a class of procedures for PCA based on the generic functions Ψ. The derivative ψ gives a weight expressing the degree of confidence for each input vector being an outlier. The robust procedure for the PCA gives less weights to input vectors having long residual vectors when projected onto the principal component subspace. We emphasis that the μ-portion is defined to be a weighted mean with the same weights as in the Γ-portion while the usual PCA employs the naive centralization with constant weights.

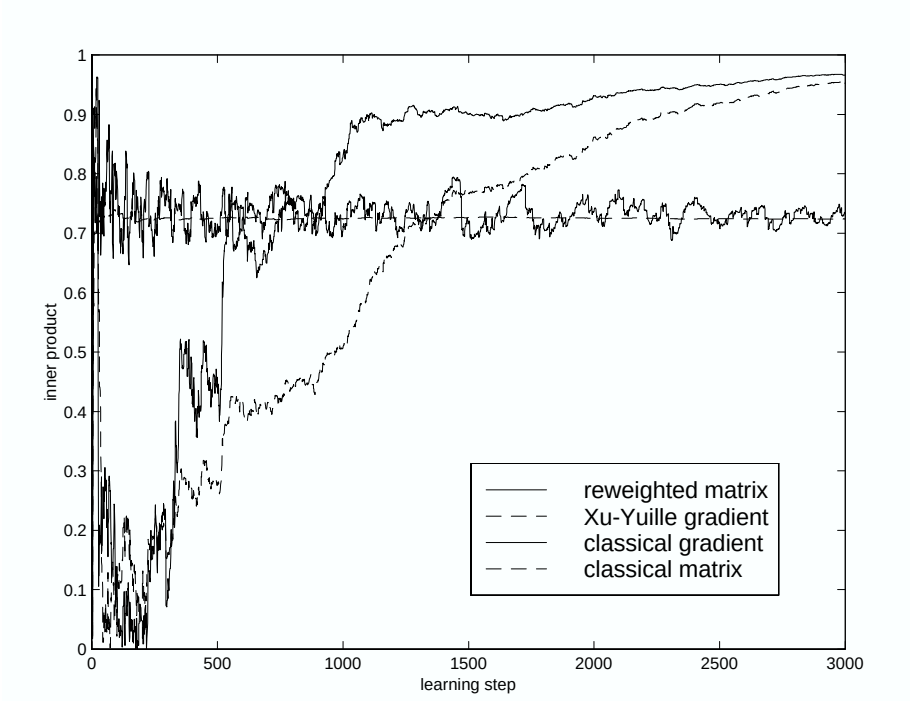


Figure 5: The inner products of the true vector $(1,0,0,0)$ and the PC vectors by RM, Xu-Yuille gradient, the classical gradient and classical matrix algorithms.

Our major point is the adaptive selection of a set of tuning parameters which control the degree of robustification. In empirical studies, we observe that the robustness performance is sensitive to the selection of tuning parameters. K-fold cross validation properly gives the adaptive selection for tuning parameters in accordance with data. However the selection method is done only for batch data but it cannot be applied to on-line data, which we must post as a future research. The RM algorithm needs the evaluation of eigenvalues and eigenvectors of the full matrix. In this respect it requires heavy computational burdens, whereas the convergence is stable and rapid relative to the gradient algorithm. The RM algorithm must be improved when the dimension of the input vector is considerably high. There is room for improvement in solving the *k*-dominant eigenvectors from a computational point of view.

Another interesting issue would be the breakdown point of the method proposed in the present paper as a global measure of robustness. In the previous literature the breakdown point has been considered as estimation of covariance (scatter) matrix other than estimation of principal component vector. However our method does not directly fit the theory since the method is not only a function of covariance matrix. We will need more discussion for this problem to be challenged as a future problem.

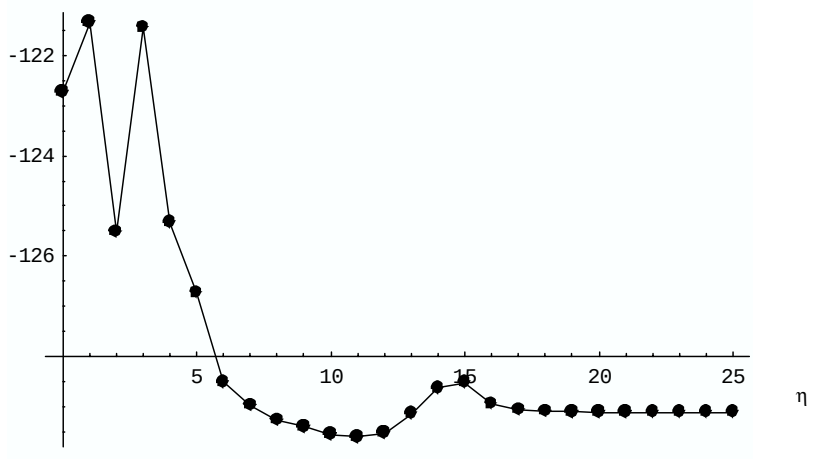


Figure 6: The plot of 10-fold CV for the Xu-Yuille rule for $\beta = 1$ fixed.

References

Amari, S. -I. Neural theory of association and concept formation. *Biological Cybernetics,* **26,** 175–185, 1977.

Campbell, N. A. Robust procedures in multivariate analysis 1: Robust covariance estimation. *Applied Statistics* **29**, 231-237, 1980.

Caussinus, H. and Ruiz, A. Interesting projections of multidimensional data by means of generalized principal component analysis. *COMPSTAT 90,* 121–126, 1990.

Croux, C. and Haesbroeck, G. Principal component analysis based on robust estimators of the covariance or correlation matrix: Influence functions and efficiencies. *Biometrika,* **87**, 603-618, 2000.

De la Torre, F. and Black, M. Robust principal component analysis for computer vision. *International Conference on Computer Vision,* 2001.

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

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J. and Stahel, W. A. *Robust Statistics: the Approach Based on Influence Functions.* Wiley, 1986.

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

Haykin, S. *Neural Networks.* Prentice Hall, 1999.

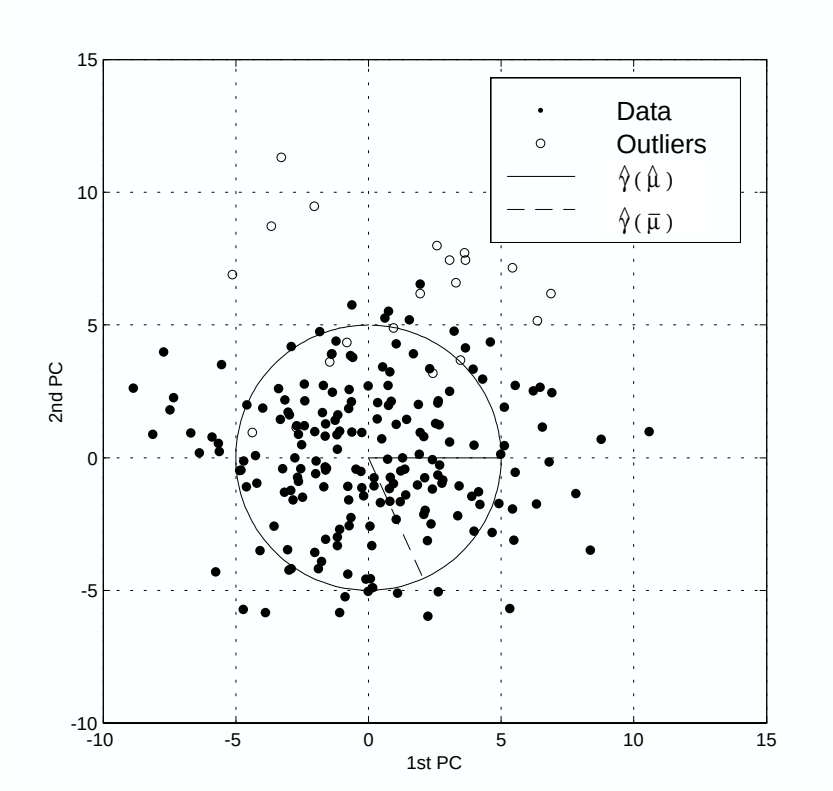


Figure 7: The effect of centralization ways.

Higuchi, I. and Eguchi, S. The influence function of principal component analysis by self-organizing rule. *Neural Computation*, **10,** 1435–1444, 1998.

Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology,* **24,** 417–441, 1933.

Huber, P. J. *Robust Statistics.* Wiley, 1981.

Hyvarinen, A. and Oja, E. A fast fixed-point algorithm for independent component analysis. *Neural Computation,* **9,** 1483–1492, 1997

Kamiya, H. and Eguchi, S. A class of robust principal component vectors. *Journal of Multivariate Analysis,* **76**, 239–269, 2001.

McLachlan, G. J. and Krishnan, T. *The EM Algorithm and Extensions.* Wiley, 1997.

Oja, E. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology,* **15,** 267–273, 1982.

Oja, E. Neural networks, principal components and subspaces. *International Journal of Neural Systems,* **1,** 61–68, 1989.

Tanaka, Y. Sensitivity analysis in principal component analysis: influence on the subspace spanned by principal components. *Communications in Statistics -Theory and Methods,* **17,** 3157–3175, 1988.

Wu, C. F. J. On the convergence properties of the EM algorithm. *Annals of Statistics,* **11,** 95–103, 1983.

Xu, L. and Yuille, A. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Transactions on Neural Networks,* **6,** 131–143, 1995.

Journal of Machine Learning Research 5 (2004) 473-497 Submitted 10/03; Revised 3/04; Published 5/04

PAC-learnability of Probabilistic Deterministic Finite State Automata

Alexander Clark ASC@ACLARK.DEMON.CO.UK
ISSCO/TIM, Université de Genève 40
Bvd du Pont d'Arve CH-1211
Genève 4, Switzerland

Franck Thollard THOLLARD@UNIV-ST-ETIENNE.FR
EURISE, Université Jean Monnet, 23
Rue du Docteur Paul Michelon
42023 Saint-Etienne Cédex 2, France

Editor: Dana Ron

Abstract

We study the learnability of Probabilistic Deterministic Finite State Automata under a modified PAC-learning criterion. We argue that it is necessary to add additional parameters to the sample complexity polynomial, namely a bound on the expected length of strings generated from any state, and a bound on the distinguishability between states. With this, we demonstrate that the class of PDFAs is PAC-learnable using a variant of a standard state-merging algorithm and the Kullback-Leibler divergence as error function.

Keywords: Grammatical inference, PAC-learnability, finite state automata, regular languages.

1. Introduction

Probabilistic Deterministic Finite State Automata (PDFAs) are widely used in a number of different fields, including Natural Language Processing (NLP), Speech Recognition (Mohri, 1997) and Bio-informatics (Durbin et al., 1999). The deterministic property here means that at any point in producing a string they are in a single state, which allows for very efficient implementations and accounts for much of their appeal. Efficient algorithms for inducing the models from data are thus important. We are interested primarily in algorithms which learn from positive data under stochastic presentation, where the data are drawn from the distribution defined by the target automaton. In addition we are concerned with automata that generate finite strings of unbounded length. This is necessary in many problem classes, particularly in NLP, where the sequences can be words or sentences with particular global structures.

Here we provide a proof that a particular algorithm Probably Approximately Correctly (PAC) learns the class of PDFAs, using the Kullback-Leibler divergence as the error function, when we allow the algorithm to have amounts of data polynomial in three quantities associated with the complexity of the problem: first, the number of states of the target, secondly the distinguishability of the automata — a lower bound on the L_∞ norm between the suffix distributions of any pair of states of the target — and thirdly a bound on the expected length of strings generated from *any*

state of the target, a quantity that can be used to bound the variance of the length of the strings generated by the target. The algorithm uses polynomial amounts of computation. We will motivate these additional parameters to the sample complexity polynomial with reference to specific counter-examples.

The algorithm we present here is a state-merging algorithm of a fairly standard type (Carrasco and Oncina, 1994; Ron et al., 1995; Thollard et al., 2000; Kermorvant and Dupont, 2002). The convergence properties of this class of algorithm have been studied before, but proofs have been restricted either to the subclass of acyclic automata (Ron et al., 1995, 1998), or using only the identification in the limit with probability one paradigm (de la Higuera et al., 1996; Carrasco and Oncina, 1999; de la Higuera and Thollard, 2000), which is generally considered less interesting as a guide to their practical usefulness. We wish to note here that we shall follow closely the notation and techniques of Ron et al. (1995). Unfortunately, our technique is too different to merely present modifications to their algorithm, but in points we shall adopt wholesale their methods of proof. We shall also use notation as close as possible to the notation used in their paper.

1.1 Learning Paradigm

We are interested in learning in the Probably Approximately Correct (PAC) framework (Valiant, 1984), more specifically in cases where the amount of data and the amount of computation required can be bounded by polynomials in the various parameters, including some characterisation of the complexity of the target.

We will use, as is standard, the Kullback-Leibler Divergence (KLD) (Cover and Thomas, 1991) between the target and the hypothesis to measure the error. Intuitively, this is the hardest measure to use since it is unbounded, and bounds other plausible distance measures (quadratic, variational etc.) (Cover and Thomas, 1991; Abe et al., 2001).

The problem class $\mathcal{C}$ is a subset of the set of all distributions over Σ^*, where Σ is a finite alphabet. The algorithm is presented with a sequence S_m of m strings from Σ^* that are drawn identically and independently from the target distribution c. Such a sequence is called a sample. Given this sample, the algorithm returns a hypothesis $H(S_m)$.

Definition 1 (KL-PAC) *Given a class of stochastic languages or distributions $\mathcal{C}$ over Σ^*, an algorithm A* KL-Probably Approximately Correctly (KL-PAC)-learns *$\mathcal{C}$ if there is a polynomial q such that for all c in $\mathcal{C}$, all $\varepsilon > 0$ and $\delta > 0$, A is given a sample S_m and produces a hypothesis H, such that $Pr[D(c||H) > \varepsilon] < \delta$ whenever $m > q(1/\varepsilon, 1/\delta, |c|)$, where $|c|$ is some measure of the complexity of the target, with running time bounded by a polynomial in m plus the total length of the strings in S_m.*

Note here two points: first, the examples are drawn from the target distribution so this approach is very different from the normal distribution-free approach. Indeed here we are interested in learning the distribution itself, so the approach is closely related to the traditional problems of density estimation, and also to the field of language modelling in speech recognition and NLP. It can also be viewed as a way of learning languages from positive samples under a restricted class of distributions, which is in line with other research which has found that the requirement to learn under all distributions is too stringent. Secondly, we are concerned with distributions over Σ^* not over Σ^n, for some fixed n, which form a finite set, which distinguishes us from Ron et al. (1995). Note that

though Ron et al. (1995) study distributions over Σ^* since they study acyclic automata of bounded size and depth, the distributions are, as a result, limited to distributions over Σ^n.

1.2 Negative Results

For PAC-learning stochastic languages there are some negative results that place quite strict limits on how far we can hope to go. First, let us recall that the stochastic languages generated by PDFAs are a proper subclass of the class of all stochastic regular languages. The results in Abe and Warmuth (1992) establish that robust learning of general (non-deterministic) finite state automata is hard. This strongly suggests that we cannot hope to learn all of this class, though the class of Probabilistic Residual Finite State Automata (Esposito et al., 2002) is a promising intermediate class between stochastic deterministic regular languages and stochastic regular languages. Secondly, Kearns et al. (1994) show that under certain cryptographic assumptions it is impossible to efficiently learn PDFAs defining distributions over two letters. They define a correspondence between noisy parity functions and a certain subclass of automata. Since the complexity results they rely upon are generally considered to be true, this establishes that the class of all PDFAs is not PAC-learnable using polynomial computation. These results apply to distributions over Σ^n for some fixed n, and thus also to our more general case of distributions over Σ^*. However, Ron et al. (1995) show that if one restricts one's attention to a class of automata that have a certain distinguishability criterion between the states, that we shall define later, it is possible to PAC-learn acyclic PDFAs. Formally, there are two ways to define this: either one defines a subclass of the problem class where the distinguishability is bounded by some inverse polynomial of the number of states or we allow the sample complexity polynomial to have another parameter. We follow Ron et al. (1995) in using the latter approach.

However the extension from acyclic PDFAs to all PDFAs produces an additional problem. As we shall see, the KLD between the target and the hypothesis can be decomposed into terms based on the contributions of particular states. The contribution of each state is related to the expected number of times the target automaton visits the state and not the probability that the target automaton will visit this state. Therefore there is a potential problem with states that are both rare (have a low probability of being reached) and yet have a high expected number of visits, a combination of properties that can happen, for example, when a state is very rare but has a transition to itself with probability very close to one. If the state is very rare we cannot guarantee that our sample will contain any strings which visit the state, but yet this state can make a non-negligible contribution to the error.

In particular, as we show in Appendix A, it is possible to construct families of automata, with bounded expected length, that will with high probability produce the same samples, but where some targets must have large KLD from any hypothesis. We refer the reader to the appendix for further details. This is only a concern with the use of the KLD as error function; with other distance measures the error on a set of strings with low aggregate probability is also low. In particular with the variation distance, while it is straightforward to prove a similar result with a bound on the overall expected length, we conjecture that it is also possible with no length bound at all – i.e. with a sample complexity that depends only on the number of states, the distinguishability and the alphabet size.

Accordingly, we argue that, in the case of learning with the KLD, it is necessary to accept an upper bound on the expected length of the strings from any state of the target automaton, or a bound on the expected length of strings from the start state and a bound on the variance. It also seems

reasonable that in most real-world applications of the algorithm, the expectation and variance of the length will be close to the mean observed length and sample variance.

1.3 Positive Results

Carrasco and Oncina (1999) proposed a proof of the identification in the limit with probability one of the structure of the automaton. In de la Higuera and Thollard (2000), the proof of the identification of the probabilities was added achieving the complete identification of the class of the PDFA with rational probabilities. With regard to the more interesting PAC-learnability criterion with respect to the KLD (KL-PAC), Ron et al. proposed an algorithm that can KL-PAC-learn the class of distinguishable acyclic automata. The distinguishability is a guarantee that the distributions generated from any state differ by at least μ in the L_∞ norm. This is sufficent to immunise the algorithm against the counterexample discussed by Kearns et al. (1994).

Our aim is to extend the work of Ron et al. (1995) to the full class of PDFAs. This requires us to deal with cycles, and with the ensuing problems caused by allowing strings of unbounded length, since with acyclic automata a bound on the number of states is also a bound on the expected length. These are of three types: first, the support of the distribution can be infinite, which rules out the direct application of Hoeffding bounds at a particular point in the proof; secondly, the automaton can be in a particular state more than once in the generation of a particular string, which requires a different decomposition of the KLD, and thirdly deriving the bound on the KLD requires slightly different constraints. Additionally, we simplify the algorithm somewhat by drawing new samples at each step of the algorithm, which avoids the use of "reference classes" in Ron et al. (1995), which are necessary to ensure the independence of different samples.

The article is organized as follow: we first start with the definitions and notations we will use (Section 2). The algorithm is then presented in Section 3 and its correctness is proved in Section 4. We then conclude with a discussion of the relevance of this result. The reader will find on Page 481 a glossary for the notation.

2. Preliminaries

We start by defining some basic notation regarding languages, distributions over languages and finite state automata of the type we study in this paper.

2.1 Languages

We have a finite alphabet Σ, and Σ^* is the free monoid generated by Σ, i.e. the set of all words with letters from Σ, with λ the empty string (identity). For $s \in \Sigma^*$ we define $|s|$ to be the length of s. The subset of Σ^* of strings of length d is denoted by Σ^d. A distribution or stochastic language D over Σ^* is a function $D : \Sigma^* \to [0,1]$ such that $\sum_{s\in\Sigma^*} D(s) = 1$. The *Kullback-Leibler Divergence* (KLD) is denoted by $D_{KL}(D_1||D_2)$ and defined as

$$D_{KL}(D_1||D_2) = \sum_s D_1(s) \log\left(\frac{D_1(s)}{D_2(s)}\right). \tag{1}$$

The L_∞ norm between two distributions is defined as

$$L_\infty(D_1, D_2) = \max_{s\in\Sigma^*} |D_1(s) - D_2(s)|.$$

We will use σ for letters and s for strings.

If S is a multiset of strings from Σ^* for any $s \in \Sigma^*$ we write $S(s)$ for the multiplicity of s in S and define $|S| = \sum_{s\in\Sigma^*} S(s)$, and for every $\sigma \in \Sigma$ define $S(\sigma) = \sum_{s\in\Sigma^*} S(\sigma s)$. We also write $S(\zeta) = S(\lambda)$. We will write $\hat{S}$ for the empirical distribution of a non-empty multiset S which gives to the string s the probability $S(s)/|S|$. This notation is slightly ambiguous for strings of length one; we will rely on the use of lower-case Greek letters to signify elements of Σ to resolve this ambiguity.

2.2 PDFA

A probabilistic deterministic finite state automaton is a mathematical object that stochastically generates strings of symbols. It has a finite number of states one of which is a distinguished start state. Parsing or generating starts in the start state, and at any given moment makes a transition with a certain probability to another state and emits a symbol. We have a particular symbol and state which correspond to finishing.

Definition 2 *A PDFA A is a tuple $(Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$, where*

- *Q is a finite set of states,*
- *Σ is the alphabet, a finite set of symbols,*
- *$q_0 \in Q$ is the single initial state,*
- *$q_f \notin Q$ is the final state,*
- *$\zeta \notin \Sigma$ is the final symbol,*
- *$\tau : Q \times \Sigma \cup \{\zeta\} \to Q \cup \{q_f\}$ is the transition function and*
- *$\gamma : Q \times \Sigma \cup \{\zeta\} \to [0,1]$ is the next symbol probability function. $\gamma(q,\sigma) = 0$ when $\tau(q,\sigma)$ is not defined.*

We will sometimes refer to automata by the set of states. All transitions that emit ζ go to the final state. In the following τ and γ will be extended to strings recursively as follows:

$$\begin{aligned}\tau(q,\sigma_1\sigma_2\ldots\sigma_k) &= \tau(\tau(q,\sigma_1),\sigma_2\ldots\sigma_k),\\ \gamma(q,\sigma_1\sigma_2\ldots\sigma_k) &= \gamma(q,\sigma_1)\times\gamma(\tau(q,\sigma_1),\sigma_2\ldots\sigma_k).\end{aligned}$$

Also we define $\tau(q,\lambda) = q$ and $\gamma(q,\lambda) = 1$. If $\tau(q_0,s) = q$ we say that *s reaches q*.

The sum of the output probabilities from each states must be one, so for all $q \in Q$,

$$\sum_{\sigma\in\Sigma\cup\{\zeta\}} \gamma(q,\sigma) = 1.$$

Assuming further that there is a non-zero probability of reaching the final state from each state, i.e.

$$\forall q \in Q\ \exists s \in \Sigma^* : \tau(q,s\zeta) = q_f \wedge \gamma(q,s\zeta) > 0,$$

the PDFA then defines a probability distribution over Σ^*, where the probability of generating a string $s \in \Sigma^*$ is

$$P^A(s) = \gamma(q_0, s\zeta). \tag{2}$$

We will also use $P_q^A(s) = \gamma(q, s\zeta)$ which we call the suffix distribution of the state q. We will omit the automaton symbol when there is no risk of ambiguity. Note that $\gamma(q_0, s)$ where $s \in \Sigma^*$ is the prefix probability of the string s, i.e. the probability that the automaton will generate a string that *starts* with s.

Definition 3 (Distinguishability) *For $\mu > 0$ two states $q_1, q_2 \in Q$ are μ-distinguishable if there exists a string s such that $|\gamma(q_1, s\zeta) - \gamma(q_2, s\zeta)| \geq \mu$. A PDFA A is μ-distinguishable if every pair of states in it is μ-distinguishable.*

Note that any PDFA has an equivalent form in which indistinguishable states have been merged, and thus $\mu > 0$.

3. Algorithm

We will now describe the algorithm we use here to learn PDFAs. It is formally described in pseudocode starting at Page 481.

We are given the following parameters

- an alphabet Σ, or strictly speaking an upper bound on the size of the alphabet,
- an upper bound on the expected length of strings generated from any state of the target L,
- an upper bound on the number of states of the target n,
- a lower bound μ for the distinguishability,
- a confidence δ, and
- a precision ε.

We start by computing the following quantities. First, we compute m_0, which is a threshold on the size of a multiset. When we have a multiset whose size is larger than m_0, which will be a sample drawn from a particular distribution, then with high probability the empirical estimates derived from that sample will be sufficiently close to the true values. Secondly, we compute N, which is the size of the sample we draw at each step of the algorithm. Finally, we compute γ_{min}, which is a small smoothing constant. These are defined as follows, using some intermediate variables to simplify the expressions slightly:

$$\gamma_{min} = \frac{\varepsilon}{4(L+1)(|\Sigma|+1)}, \tag{3}$$

$$\delta' = \frac{\delta}{2(n|\Sigma|+2)}, \tag{4}$$

$$\varepsilon_1 = \frac{\varepsilon^2}{16(|\Sigma|+1)(L+1)^2}, \tag{5}$$

$$m_0 = \max\left(\frac{8}{\mu^2}\log\left(\frac{96n|\Sigma|}{\delta'\mu}\right), \frac{1}{2\varepsilon_1^2}\log\left(\frac{12n|\Sigma||\Sigma+1|}{\delta'}\right)\right), \tag{6}$$

$$\varepsilon_3 = \frac{\varepsilon}{2(n+1)\log(4(L+1)(|\Sigma|+1)/\varepsilon)}, \tag{7}$$

$$N = \frac{4n|\Sigma|L^2(L+1)^3}{\varepsilon_3^2}\max\left(2n|\Sigma|m_0, 4\log\frac{1}{\delta'}\right). \tag{8}$$

The basic data structure of the algorithm represents a digraph $G = (V,E)$ with labelled edges, V being a set of vertices (or nodes) and $E \subseteq V \times \Sigma \times V$ a set of edges. The graph holds our current hypothesis about the structure of the target automaton. We have a particular vertex in the graph, $v_0 \in V$ that corresponds to the initial state of the hypothesis. Each arc in the graph is labelled with a letter from the alphabet, and there is at most one edge labelled with a particular letter leading from any node. Indeed, the graph can be thought of as a (non-probabilistic) deterministic finite automaton, but we will not use this terminology to avoid confusion. We will use $\tau_G(v,\sigma)$ as a transition function in this graph to refer to the node reached by the arc labelled with σ that leads from v, if such a node exists, and we will extend it to strings as above. At each node $v \in V$ we associate a multiset of strings S_v that represents the suffix distribution of the state that the node represents. At any given moment in the algorithm we wish the graph to be isomorphic to a subgraph of the target automaton. Initially we have a single node that represents the initial state of the target automaton, together with a multiset that is just a sample of strings from the automaton.

At each step we are given a multiset of N data points (i.e. strings from Σ^*) generated independently by a target PDFA T. For each node u in the graph, and each letter σ in the alphabet that does not yet label an arc out of u, we hypothesize a candidate node. We can refer to this node by the pair (u,σ). The first step is to compute the multiset of suffixes of that node. For each string in the sample, we trace its path through the graph, deleting symbols from the front of the string as we proceed, until we arrive at the node u, or the string is empty, in which case we discard it. If the string then starts with σ we delete this letter and then add the resulting string to the multiset associated with (u,σ). Intuitively, this should be a sample from the suffix distribution of the relevant state. More formally, for any input string s, if $\tau_G(v_0,s)$ is defined then we discard the string. Otherwise we take the longest prefix r such that there is a node u such that $\tau_G(v_0,r) = u$ and $s = r\sigma t$ and add the string t to the multiset of the candidate node (u,σ). If this multiset is sufficiently large (at least m_0), then we compare it with each of the nodes in the graph. The comparison operator computes the L_∞-norm between the empirical distributions defined by the multisets. When we first add a node to the graph we keep with it the multiset of strings it has at that step, and it remains with this multiset for the rest of the algorithm.

Definition 4 (Candidate node) *A candidate node is a pair (u,σ) where u is a node in the graph and $\sigma \in \Sigma$ where $\tau_G(u,\sigma)$ is undefined. It will have an associated multiset $S_{u,\sigma}$. A candidate node (u,σ) and a node v in a hypothesis graph G are similar if and only if, for all strings $s \in \Sigma^*$, $|S_{u,\sigma}(s)/|S_{u,\sigma}| - S_v(s)/|S_v|| \leq \mu/2$, i.e. if and only if $L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) \leq \mu/2$.*

We will later see that the value of $\mu/2$, given that any two states in the target are at least μ apart in the L_∞-norm, allows us to ensure that the nodes are similar if and only if they are representatives of the same state. We will define this notion of representation more precisely below. If the candidate

node (u,σ) is similar to one of the nodes in the graph, say v then we add an arc labelled with σ from u to v. If it is not similar to any node, then we create a new node in the graph, and add an arc labelled with σ from u to the new node. We attach the multiset to it at this point. We then delete all of the candidate nodes, sample some more data and continue the process, until we draw a sample where no candidate node has sufficiently large a multiset.

Completing the Graph If the graph is incomplete, i.e. if there are strings not accepted by the graph, we add a new node called the ground node which represents all the low frequency states. We then complete the graph by adding all possible arcs from all states leading to the ground node, including from the ground node to itself. Since the hypothesis automaton must accept every string, every state must have an arc leading out of it for each letter in the alphabet. We then define for each node in the graph a state in the automaton. We add a final state $\hat{q}_f$, together with a transition labelled with ζ from each state to $\hat{q}_f$. The transition function τ is defined by the structure of this graph.

Estimating Probabilities The transition probabilities are then estimated using a simple additive smoothing scheme (identical to that used by Ron et al., 1995). For a state u, $\sigma \in \Sigma \cup \{\zeta\}$,

$$\hat{\gamma}(\hat{q},\sigma) = (S_u(\sigma)/|S_u|) \times (1-(|\Sigma|+1)\gamma_{min}) + \gamma_{min}. \quad (9)$$

This is also used for the ground node where of course the multiset is empty.

4. Analysis of the Algorithm

We now proceed to a proof that this algorithm will learn the class of PDFAs. We first state our main result.

Theorem 5 *For every PDFA A with n states, with distinguishability $\mu > 0$, such that the expected length of the string generated from every state is less than L, for every $\delta > 0$ and $\varepsilon > 0$, Algorithm LearnPDFA outputs a hypothesis PDFA $\hat{A}$, such that with probability greater than $1-\delta$, $D_{KL}(A,\hat{A}) < \varepsilon$.*

We start by giving a high-level overview of our proof. In order to bound the KLD we will use a decomposition of the KLD between two automata presented in Carrasco (1997). This requires us to define various expectations relating the number of times that the two automata are in various states.

- We define the notion of a good sample – this means that certain quantities are close to their expected values. We show that a sample is likely to be good if it is large enough.
- We show that if all of the samples we draw are good, then at each step the hypothesis graph will be isomorphic to a subgraph of the target.
- We show that when we stop drawing samples, there will be in the hypothesis graph a state representing each frequent state in the target. In addition we show that all frequent transitions will also have a representative edge in the graph.
- We show that the estimates of all the transition probabilities will be close to the target values.
- Using these bounds we derive a bound for the KLD between the target and the hypothesis.

We shall use a number of quantities calculated from the parameters input to the algorithm to bound the various quantities we are estimating. Table 1 summarises these.

Symbol	Description	Definition
ε	Precision	input
δ	Confidence	input
L	Bound on the expected length of strings	input
μ	Distinguishability of states	input
n	Bound on the number of states of the target automaton	input
Φ	Function from states of the target to nodes in graph	Definition 8
$W_d(q)$	Probability to reach state q after d letters	Equation 10
$W(q)$	Expected number of times state q is reached	Equation 12
$W_d(q,\hat{q})$	Expected number of times a path is in q and $\hat{q}$	Equation 14
$P(q)$	Probability that a state q is reached	Equation 4.1
$P(s), P^A(s)$	Probability that a string is generated	Equation 2
$P_q(s)$	Probability that a string is generated from q	$\gamma(q, s\zeta)$
S_u	Multiset of node u	Page 477
$\hat{S_u}$	Empirical distribution of multiset of node u	Page 477
$\lvert S_u \rvert$	Size of multiset of node u	Page 477
$S_u(s)$	Count of string s in multiset	Page 477
$S_u(\sigma)$	Count of letter σ in multiset	Page 477
N	One step sample size	Equations 8
m_0	Threshold for size of multiset of candidate node	Equation 6
γ_{min}	Smoothing constant for transition probabilities	Equation 9
ε_1	Bound on transition counts	$\varepsilon_4^2/4(\lvert\Sigma\rvert+1)$
ε_2	Threshold for weight of frequent states	$\varepsilon_3/2nL(L+1)$
ε_3	Bound on difference between weights in target and hypothesis	Equation 7
ε_4	Bound on smoothed estimates of transition probabilities	$\varepsilon/2(L+1)$
ε_5	Threshold for frequent transitions	$\varepsilon_3/2\lvert\Sigma\rvert L(L+1)$
ε_6	Bound on exit probability	$\varepsilon_2\varepsilon_5/(L+1)$
δ'	Fraction of confidence	Equation 4

Table 1: Glossary for notation

4.1 Weights of States

We will use a particular decomposition of the KLD, that allows us to represent the divergence as a weighted sum of the divergences between the distributions of the transition probabilities between states in the target and states in the hypothesis. This requires us to define a number of quantities which relate to the expected number of times the automaton will be in various states.

We will define below the *weight* of a state. Informally, this is the expected number of times the automaton will reach, or pass through, that state. Here we encounter one of the differences that cyclicity introduces. With acyclic automata, the automaton can reach each state at most once. Thus the expectation is equal to the probability – and thus we can operate with sets of strings, and simply add up the probabilities. With cyclic automata, the automata can repeatedly be in a particular state while generating a single string. We start by defining

$$W_d(q) = \sum_{s \in \Sigma^d : \tau(q_0,s)=q} \gamma(q_0,s). \tag{10}$$

Informally, $W_d(q)$ is the probability that it will generate at least d characters and be in the state q after having generated the first d. This is an important difference in notation to that of Ron et al. (1995), who use W to refer to a set of strings.

Since s does not end in ζ, we are not summing over the probability of strings but rather the probability of prefixes. Note that $W_0(q_0)$ is by definition 1 and that we can also calculate this quantity recursively as

$$W_d(q) = \sum_{p \in Q} W_{d-1}(p) \sum_{\sigma : \tau(p,\sigma)=q} \gamma(p,\sigma). \tag{11}$$

We now define the probability that the automaton will generate a string of length at least d as W_d:

$$W_d = \sum_{q \in Q} W_d(q).$$

Note that the probability that it will be of length exactly d is $W_d - W_{d+1}$, and that $W_0 = 1$. We also define here the *expected* number of times the automaton will be in the state q as

$$W(q) = \sum_{d=0}^{\infty} W_d(q) = \sum_{s \in \Sigma^* :\ \tau(q_0,s)=q} \gamma(q_0,s). \tag{12}$$

We shall refer to this quantity as the *weight* of the state q. Therefore the expectation of the length of strings is

$$E[|s|] = \sum_{d=0}^{\infty} d(W_d - W_{d+1}) = \sum_{d=1}^{\infty} W_d = \left(\sum_{d=0}^{\infty} W_d\right) - W_0 = \left(\sum_{d=0}^{\infty} W_d\right) - 1 = \sum_{s \in \Sigma^*} \gamma(q_0,s) - 1.$$

The expected length of strings generated from any state can be defined likewise. We will be given a bound on the expected length of strings generated from any state, denoted by L. Formally, for all q we shall require that

$$\sum_{s \in \Sigma^*} \gamma(q,s) \leq L + 1.$$

Using this bound we can establish that for any length k,

$$\sum_{d>k}^{\infty} W_d = \sum_q W_k(q) \sum_{d>0} \sum_{s \in \Sigma^d} \gamma(q,s) \leq L W_k.$$

We will later need to bound the quantity $\sum_d dW_d(q)$, which we can do in the following way:

$$\sum_{d=0}^{\infty} dW_d(q) \leq \sum_{d=0}^{\infty} dW_d = \sum_{d=0}^{\infty} \sum_{k>d} W_k \leq \sum_{d=0}^{\infty} LW_d \leq L(L+1). \tag{13}$$

In addition to these expected values, we will also need to lower bound the probability of the automaton being in a state at least once in terms of the expected number of times it will be in that state. The probability of it being in a state at some point can be defined in terms of the set of strings that have a prefix that reaches the state,

$$S(q) = \{s \in \Sigma^* : \exists r,t \in \Sigma^* \text{ s.t. } s = rt \wedge \tau(q_0, r) = q\},$$

or in terms of the set of strings that reach q for the first time (i.e. have no proper prefix that also reaches q):

$$R(q) = \{s \in \Sigma^* : \tau(q_0, s) = q \wedge (\nexists u, v \in \Sigma^* \text{ s.t. } v \neq \lambda \wedge uv = s \wedge \tau(q_0, u) = q)\}.$$

The probability of it being in the state at least once can therefore be written in two ways

$$P(q) = \sum_{s \in S(q)} \gamma(q_0, s\zeta) = \sum_{s \in R(q)} \gamma(q_0, s).$$

Note that in the absence of a bound on the expected length of strings, a state can have arbitrarily small probability of being visited but have large weight.

Lemma 6 *For any automaton such that the expected length from any state is less than L, for all states q, $P(q) \geq W(q)/(L+1)$.*

Proof Intuitively, after we reach the state q, the expected number of times we reach the state q again will be at most the expected number of times we reach any state after this, which is bounded by L. Formally,

$$\begin{aligned}
W(q) &= \sum_{s \in S(q):\ \tau(q_0,s)=q} \gamma(q_0, s) \\
&= \sum_{r \in R(q)} \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q_0, rs) \\
&= \sum_{r \in R(q)} \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q_0, r)\gamma(q, s) \\
&= \sum_{r \in R(q)} \gamma(q_0, r) \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q, s) \\
&= P(q) \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q, s) \\
&\leq P(q) \sum_{s \in \Sigma^*} \gamma(q, s) \\
&\leq P(q)(L+1).
\end{aligned}$$

■

We also define here a related quantity given two automata. Given two automata $A, \hat{A}$ with sets of states $Q, \hat{Q}$, the joint weight $W(q, \hat{q})$ is defined to be the expected number of times the automata

are simultaneously in the states $q \in Q$ and $\hat{q} \in \hat{Q}$, when strings are being generated by A, and parsed by $\hat{A}$.

Define

$$W_d(q,\hat{q}) = \sum_{\substack{s:\tau(q_0,s)=q \\ \hat{\tau}(\hat{q}_0,s)=\hat{q} \\ |s|=d}} \gamma(q_0,s). \tag{14}$$

We can also define this recursively with $W_0(q_0,\hat{q}_0) = 1$ and

$$W_d(q,\hat{q}) = \sum_{p\in Q}\sum_{\hat{p}\in\hat{Q}} W_{d-1}(p,\hat{p}) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\hat{p},\sigma)=\hat{q}}} \gamma(p,\sigma).$$

We now define the expected number of times the first automaton will be in state q and the second in state $\hat{q}$ thus

$$W(q,\hat{q}) = \sum_d W_d(q,\hat{q}).$$

Note also that

$$W(q) = \sum_{\hat{q}\in\hat{Q}} W(q,\hat{q}). \tag{15}$$

Given these quantities we can now use the following decomposition of the KLD (Carrasco, 1997) to bound the error:

$$D_{KL}(T||H) = \sum_{q\in Q_T}\sum_{\hat{q}\in Q_H}\sum_{\sigma\in\Sigma\cup\{\zeta\}} W(q,\hat{q})\gamma(q,\sigma)\log\frac{\gamma(q,\sigma)}{\hat{\gamma}(\hat{q},\sigma)}. \tag{16}$$

4.2 Probably Correct

We define a notion of *goodness* of a multiset which is satisfied if certain properties of the multiset are close to their expected values.

Definition 7 (Good multiset) *We say that a multiset S is μ-ε_1-*good *for a state q iff $L_\infty(\hat{S},P_q) < \mu/4$ and for every $\sigma \in \Sigma\cup\{\zeta\}$, $|S(\sigma)/|S| - \gamma(q,\sigma)| < \varepsilon_1$.*

The algorithm produces a sequence of graphs $G_0, G_1, \ldots, G_k$, with multisets attached to each node. We will first show that all of these graphs have the right structure, and then we will show that the final graph will have all of the important structure and finally we will bound the KLD.

Definition 8 (Good hypothesis graph) *We say that a hypothesis graph G for an automaton A is* good *if there is a bijective function Φ from a subset of states of A to all the nodes of G such that $\Phi(q_0) = v_0$, and if $\tau_G(u,\sigma) = v$ then $\tau(\Phi^{-1}(u),\sigma) = \Phi^{-1}(v)$, and for every node u in the graph, the multiset attached to u is μ-ε_1-good for the state $\Phi^{-1}(u)$.*

Note that if there is such an isomorphism then it is unique. When $\Phi(q) = u$ we shall call u a representative of the state q. When we have a candidate node (u,σ) we shall call this also a representative of the state $\tau(q,\sigma)$. We will extend the use of Φ to a mapping from the states of the target to the states of the final hypothesis automaton. In this case it will no longer be bijective since in general more than one state can be mapped to the ground state.

Lemma 9 *If a graph G_i is good and we draw a sample of size N such that for every candidate node with multiset S such that $|S| > m_0$ is μ-ε_1-good for the state $\tau(\Phi^{-1}(u),\sigma)$, and there is at least one such candidate node, then the graph G_{i+1} is also good.*

Consider a candidate node (u,σ) and a node v Suppose these are both representatives of the same state q in which case $\Phi^{-1}(v) = \tau(\Phi^{-1}(u),\sigma)$. Then as $L_\infty(\hat{S}_{u,\sigma}, P_q) < \mu/4$ and $L_\infty(\hat{S}_v, P_q) < \mu/4$ (by the goodness of the multisets), the triangle inequality shows that $L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) < \mu/2$, and therefore the comparison will return true. On the other hand, let us suppose that they are representatives of different states q and q_v. We know that $L_\infty(\hat{S}_{u,\sigma}, P_q) < \mu/4$ and $L_\infty(\hat{S}_v, P_{q_v}) < \mu/4$ (by the goodness of the multisets), and $L_\infty(P_q, P_{q_v}) \geq \mu$ (by the μ-distinguishability of the target). By the triangle inequality $L_\infty(P_q, P_{q_v}) \leq L_\infty(\hat{S}_{u,\sigma}, P_q) + L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) + L_\infty(\hat{S}_v, P_{q_v})$, therefore $L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) > \mu/2$ and the comparison will return false. If there is a node in the graph that is similar to the candidate node, then this is because there is the relevant transition in the target automaton, which means the new graph will also be good. If there is no such node, then that is because there is no node v such that $\Phi^{-1}(v) = \tau(\Phi^{-1}(u),\sigma)$ in which case we can define a new node v and define $\Phi^{-1}(v) = \tau(\Phi^{-1}(u),\sigma)$, and thereby show that the new graph is good. Additionally since the candidate node multisets are μ-ε_1-good, the multiset of this node will also be good. ■

Since G_0 is good if the initial sample is good, using this we can prove by induction on i that the final hypothesis graph will be μ-ε_1-good if all the samples satisfy the criteria defined above.

Definition 10 (exiting the graph) *A string exits a graph G if there is no node v such that $\tau_G(v_0, s) = v$. The exit probability of a graph G (with respect to an automaton A) which we write $P_{exit}(G)$ is defined as*

$$P_{exit}(G) = \sum_{s:s \text{ exits } G} \gamma(q_0, s\zeta).$$

Definition 11 (Good sample) *We say that a sample of size N is good given a good graph G if for every candidate node (u,σ), such that $|S_{u,\sigma}| > m_0$, $S_{u,\sigma}$ is μ-ε_1-good for the state $\tau(\Phi^{-1}(u),\sigma)$ and that if $P_{exit}(G) > \varepsilon_6$ then the number of strings that exit the graph is more than $\frac{1}{2}NP_{exit}(G)$.*

Note that if there are no candidate nodes with multisets larger than m_0, then the total number of strings that exited the graph must be less than $n|\Sigma|m_0$ (since there are at most n nodes in a good graph, and therefore at most $n|\Sigma|$ candidate nodes). Therefore in this circumstance, if the samples are good, we can conclude that either $P_{exit}(G) \leq \varepsilon_6$ or $P_{exit}(G) < 2n|\Sigma|m_0/N$.

We then prove that the hypothesis graph will probably contain a node representing each state with non-negligible weight, and an edge representing each transition from a state with non-negligible weight that has a non-negligible transition probability. We define two intermediate quantities: ε_2, a bound on the weight of states, and ε_5, a bound on transition weights. We have

$$\varepsilon_2 = \frac{\varepsilon_3}{2nL(L+1)} \text{ and } \varepsilon_5 = \frac{\varepsilon_3}{2|\Sigma|L(L+1)}. \tag{17}$$

Lemma 12 *For any state q in the target such that $W(q) > \varepsilon_2$, if all the samples are good, then there will be a node u in the final hypothesis graph such that $\Phi(q) = u$. Furthermore, for such a state for any $\sigma \in \Sigma$, such that $\gamma(q,\sigma) > \varepsilon_5$, then $\tau_G(u,\sigma)$ is defined and is equal to $\Phi(\tau(q,\sigma))$.*

Proof Since $W(q) > \varepsilon_2$, we know by Lemma 6 that $P(q) > \varepsilon_2/(L+1)$.

From the definitions of N and ε_6

$$\frac{2n|\Sigma|m_0}{N} < \varepsilon_6 = \frac{\varepsilon_2\varepsilon_5}{(L+1)} < P(q)\varepsilon_5 < P(q).$$

Clearly if there were no representative node u for state q, then all strings that reached the state would exit the graph, and thus $P_{exit}(G) \geq P(q)$. Similarly, if there were no edge labelled σ from the node, then $P_{exit}(G) \geq P(q)\varepsilon_5$. By the goodness of the sample we know that either $P_{exit}(G) \leq \varepsilon_6$ or $P_{exit}(G) < 2n|\Sigma|m_0/N$, in both cases $P_{exit}(G) < P(q)\varepsilon_5$. Therefore there is a suitable state u and edge in the graph, and since the final hypothesis graph is μ-ε_1-good, τ_G will have the correct value. ■

4.3 Close Transition Probabilities

Assuming that all the samples are good, we now want to show that the final smoothed transition probabilities will be close to the correct values. More precisely, we want to show that for all transitions, the ratio of the true value to the estimate is not much greater than unity.

For every state q with $W(q) > \varepsilon_2$, with $u = \Phi(q)$ and for every symbol $\sigma \in \Sigma \cup \{\zeta\}$, we will have, by the goodness of the multisets

$$\left|\gamma(q,\sigma) - \frac{S_u(\sigma)}{|S_u|}\right| \leq \varepsilon_1. \tag{18}$$

Define

$$\varepsilon_4 = \frac{\varepsilon}{2(L+1)},$$

$$\gamma_{min} = \frac{\varepsilon_4}{2(|\Sigma|+1)} \text{ and } \varepsilon_1 = \frac{\varepsilon_4^2}{4(|\Sigma|+1)}.$$

We use exactly the argument in Lemma 5.3 of Ron et al. (1995): if $\gamma(q,\sigma)$ is small (at most $\gamma_{min}+\varepsilon_1$) then the estimate will anyway be at least γ_{min} and thus the ratio will be not much larger than the true value, and if it is large then it will be close to the true value. We can verify that

$$\frac{\gamma(q,\sigma)}{\hat{\gamma}(q,\sigma)} \leq 1+\varepsilon_4. \tag{19}$$

4.4 Close Weights

We now come to the most complex part of the proof. We want to show for every frequent state q in the target, which will have a corresponding state in the hypothesis $\hat{q} = \Phi(q)$, that (again under the assumption that all the samples are good)

$$W(q) - W(q,\hat{q}) \leq \varepsilon_3. \tag{20}$$

As a consequence of this and Equation 15, we will have

$$\sum_{\hat{q}:\Phi(q)\neq\hat{q}} W(q,\hat{q}) \leq \varepsilon_3. \tag{21}$$

So we will do this by showing that for any d

$$W_d(q) - W_d(q,\hat{q}) < \frac{\varepsilon_3 d W_d(q)}{L(L+1)}.$$

Then, since we know that $\sum_d dW_d(q) < L(L+1)$ by Equation 13, we can sum with respect to d and derive the result.

Lemma 13 *For all states $q \in Q$ such that $W(q) > \varepsilon_2$, there is a state $\hat{q}$ in the hypothesis such that $\Phi(q) = \hat{q}$ and for all $d \geq 0$*

$$W_d(q) - W_d(q,\hat{q}) \leq \frac{\varepsilon_3 d W_d(q)}{L(L+1)}.$$

Proof We will prove the lemma by induction on d. For $d = 0$ this is clearly true. For the inductive step, we assume that it is true for $d-1$.

We can rewrite the joint weight as

$$W_d(q,\hat{q}) = \sum_{p\in Q} \sum_{\hat{p}\in\hat{Q}} W_{d-1}(p,\hat{p}) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\hat{p},\sigma)=\hat{q}}} \gamma(p,\sigma). \tag{22}$$

If we consider only the cases where $W(p) > \varepsilon_2$ and $\Phi(p) = \hat{p}$ we can see that

$$W_d(q,\hat{q}) \geq \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p,\Phi(p)) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)=\hat{q}}} \gamma(p,\sigma). \tag{23}$$

And then by the inductive hypothesis, for these frequent states

$$W_d(q,\hat{q}) \geq \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \left(1 - \frac{\varepsilon_3(d-1)}{L(L+1)}\right) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)=\hat{q}}} \gamma(p,\sigma). \tag{24}$$

Using the fact that $W_d \leq 1$ for all d, and using the recursive definition of W_d we can write this as

$$W_d(q,\hat{q}) \geq \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)=\hat{q}}} \gamma(p,\sigma) - \frac{\varepsilon_3(d-1)W_{d-1}}{L(L+1)} \tag{25}$$

$$\begin{aligned} &\geq \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \sum_{\sigma:\tau(p,\sigma)=q} \gamma(p,\sigma) \\ &\quad - \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)\neq\hat{q}}} \gamma(p,\sigma) \\ &\quad - \frac{\varepsilon_3(d-1)}{L(L+1)}. \end{aligned} \tag{26}$$

So we need to show that most of the weight from W_{d-1} must be from states p with $W(p) > \varepsilon_2$. Then we can change from $W_{d-1}(p)$ to $W_d(q)$.

Using Equation 11 we can see that

$$\begin{aligned} W_d(q) &= \sum_{p\in Q:W(p)\leq\varepsilon_2} W_{d-1}(p) \sum_{\sigma:\tau(p,\sigma)=q} \gamma(p,\sigma) + \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \sum_{\sigma:\tau(p,\sigma)=q} \gamma(p,\sigma) \\ &\leq n\varepsilon_2 + \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \sum_{\sigma:\tau(p,\sigma)=q} \gamma(p,\sigma). \end{aligned}$$

Using this to replace the first term on the right hand side of Equation 26 we get

$$W_d(q,\hat{q}) \geq W_d(q) - n\varepsilon_2 - \frac{\varepsilon_3(d-1)}{L(L+1)} - \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)\neq\hat{q}}} \gamma(p,\sigma).$$

Since by Lemma 12 all of the transitions from frequent states with probability greater than ε_5 must go to the correct states, we know that the values of $\gamma(p,\sigma)$ in the final term must be less than ε_5.

$$\begin{aligned} W_d(q,\hat{q}) &\geq W_d(q) - n\varepsilon_2 - \frac{\varepsilon_3(d-1)}{L(L+1)} - \sum_{p\in Q:W(p)>\varepsilon_2} W_{d-1}(p) \sum_{\sigma} \varepsilon_5 \\ &\geq W_d(q) - n\varepsilon_2 - \frac{\varepsilon_3(d-1)}{L(L+1)} - |\Sigma|\varepsilon_5. \end{aligned}$$

By our definitions of ε_2 and ε_5

$$n\varepsilon_2 + |\Sigma|\varepsilon_5 \leq \frac{\varepsilon_3}{L(L+1)},$$

and therefore the lemma will hold. ■

5. Proof that the Divergence is Small

We are now in a position to show that the KLD between the target and the hypothesis is small. We use Carrasco's decomposition of the KLD (Carrasco, 1997) to bound the error:

$$D_{KL}(T||H) = \sum_{q\in Q_T} \sum_{\hat{q}\in Q_H} \sum_{\sigma\in\Sigma\cup\{\zeta\}} W(q,\hat{q})\gamma(q,\sigma) \log \frac{\gamma(q,\sigma)}{\hat{\gamma}(\hat{q},\sigma)}.$$

We are going to divide the summands into three parts. We define

$$D(q,\hat{q}) = W(q,\hat{q}) \sum_{\sigma\in\Sigma\cup\{\zeta\}} \gamma(q,\sigma) \log \frac{\gamma(q,\sigma)}{\hat{\gamma}(\hat{q},\sigma)},$$

$$\begin{aligned} D_1 &= \sum_{q\in Q_T:W(q)>\varepsilon_2} \sum_{\hat{q}\in Q_H:\Phi(q)=\hat{q}} D(q,\hat{q}), \\ D_2 &= \sum_{q\in Q_T:W(q)>\varepsilon_2} \sum_{\hat{q}\in Q_H:\Phi(q)\neq\hat{q}} D(q,\hat{q}), \\ D_3 &= \sum_{q\in Q_T:W(q)\leq\varepsilon_2} \sum_{\hat{q}\in Q_H} D(q,\hat{q}). \end{aligned}$$

Note that for the frequent states with $W(q) > \varepsilon_2$, Φ will be well-defined.

$$D_{KL}(T||H) = D_1 + D_2 + D_3$$

We will bound these separately. We bound D_1 by showing that for these matching pairs of states q, $\Phi(q)$, all of the transition probabilities are close to each other. We can bound D_2 by showing that $W(q,\hat{q})$ is small when $\hat{q} \neq \Phi(q)$, and D_3 since $W(q)$ is small.

Using Equation 19 and recalling that $W(q) \geq W(q,\hat{q})$ we bound D_1.

$$D_1 \leq \sum_{q \in Q_T : W(q) > \varepsilon_2} W(q) \log(1+\varepsilon_4) \leq (L+1)\log(1+\varepsilon_4) \leq (L+1)\varepsilon_4.$$

Bounding D_2 is done using Equation 21, the fact that $\gamma(q,\sigma) \leq 1$ and that $\hat{\gamma}(\hat{q},\sigma) \geq \gamma_{min}$ by the smoothing technique.

$$\begin{aligned} D_2 &\leq \sum_{q \in Q_T, W(q) > \varepsilon_2} \sum_{\hat{q} \in Q_H : \Phi(q) \neq \hat{q}} W(q,\hat{q}) \sum_{\sigma \in \Sigma \cup \{\zeta\}} \gamma(q,\sigma) \log \frac{1}{\gamma_{min}} \\ &\leq \sum_{q \in Q_T, W(q) > \varepsilon_2} \varepsilon_3 \log \frac{1}{\gamma_{min}} \leq n\varepsilon_3 \log \frac{1}{\gamma_{min}}. \end{aligned}$$

With regard to D_3, using Equation 15 and the bound on $W(q)$ we can see that

$$D_3 \leq \sum_{q \in Q_T, W(q) \leq \varepsilon_2} \sum_{\hat{q} \in Q_H} W(q,\hat{q}) \sum_{\sigma \in \Sigma \cup \{\zeta\}} \gamma(q,\sigma) \log \frac{1}{\gamma_{min}} \leq n\varepsilon_2 \log \frac{1}{\gamma_{min}}.$$

Substituting in Equation 17, and assuming that $L > 1$,

$$\begin{aligned} D_{KL}(T||H) &< (L+1)\varepsilon_4 + \left(n\varepsilon_3 + \frac{\varepsilon_3}{2nL(L+1)} \right) \log \frac{1}{\gamma_{min}} \\ &< (L+1)\varepsilon_4 + (n+1)\varepsilon_3 \log \frac{1}{\gamma_{min}}. \end{aligned}$$

Substituting in the values of γ_{min}, ε_3 and ε_4 gives us $D_{KL}(T||H) < \varepsilon$ as desired. ■

We have thus shown that if all of the samples are good at each step, the resulting hypothesis will be approximately correct. We must now show that all of the samples will be good with high probability, by showing that N and m_0 are large enough.

6. Bounding the Probability of an Error

We need to show that with high probability a sample of size N will be good for a given good graph G. We assume that the graph is good at each step. Each step of the algorithm will increase the number of transitions in the graph by at least 1. There are at most $n|\Sigma|$ transitions in the target; so there are at most $n|\Sigma|+2$ steps in the algorithm since we need an initial step to get the multiset for u_0 and another at the end when we terminate. So we want to show that a sample will be good with probability at least $1 - \delta/(n|\Sigma|+2) = 1 - 2\delta'$.

There are two sorts of errors that can make the sample bad. First, one of the multisets could be bad, and secondly too few strings might exit the graph. There are at most $n|\Sigma|$ candidate nodes, so we ensure that the probability of getting a bad multiset at a particular candidate node is less than $\delta'/n|\Sigma|$, and we will ensure that the probability of the second sort of error is less than δ'.

6.1 Good Multisets

First we bound the probability of getting a bad multiset. Recall from Definition 7 that we have two requirements. First, for every string we must have the empirical probability within $\mu/4$ of its true value. Secondly, for each $\sigma \in \Sigma \cup \{\zeta\}$ the empirical probability must be within ε_1 of the true value.

A difficulty here is that in the first case we will be comparing over an infinite number of strings. Clearly only finitely many strings will have non-zero counts, but nonetheless, we need to show that for every string in Σ^* the empirical probability is close to the true probability. This will be true if all of the strings with probability less than $\mu/8$ have empirical probability less than $\mu/4$ and all of the strings with probability greater than this have empirical probability within $\mu/4$ of their true values.

Consider the strings of probability greater than $\mu/8$. There are at most $8/\mu$ of these. Therefore for a given string s, by Hoeffding bounds (Hoeffding, 1963):

$$Pr\left[\left|\frac{S_u(s)}{|S_u|} - \gamma(q, s\zeta)\right| > \mu/4\right] < 2e^{-m_u\mu^2/8} < 2e^{-m_0\mu^2/8}. \tag{27}$$

The probability of making an error on one of these frequent strings is less than $\frac{16}{\mu}e^{-m_0\mu^2/8}$. We also need to bound the probability of all of the rare strings–those with $\gamma(q, s\zeta) \leq \mu/8$.

Consider all of the strings whose probability is in $[\mu 2^{-(k+1)}, \mu 2^{-k})$.

$$S_k = \{s \in \Sigma^* : \gamma(q, s\zeta) \in [\mu 2^{-(k+1)}, \mu 2^{-k})\}.$$

Define $S_{rare} = \bigcup_{k=3}^{\infty} S_k$. We bound each of the S_k separately, using the binomial Chernoff bound where $n = |S_u|\mu/4 > |S_u|p$ (which is true since $p < \mu/4$):

$$Pr\left[\frac{S_u(s)}{|S_u|} \geq \frac{\mu}{4}\right] \leq \left(\frac{|S_u|p}{n}\right)^n e^{n-|S_u|p}.$$

This bound decreases with p, so we can replace this for all strings in S_k with the upper bound for the probability, and we can replace $|S_u|$ with m_0. We write $\mu' = \mu/4$ to reduce clutter.

$$Pr\left[\frac{S_u(s)}{|S_u|} \geq \mu'\right] \leq \left(\frac{m_0\mu' 2^{-k}}{m_0\mu'}\right)^{m_0\mu'} e^{m_0\mu' - m_0\mu' 2^{-k}} \tag{28}$$

$$\leq \left(2^{-k}e^{1-2^{-k}}\right)^{m_0\mu'} < 2^{-km_0\mu'}. \tag{29}$$

Assuming that $m_0\mu' > 3$ we can see that

$$Pr\left[\frac{S_u(s)}{|S_u|} \geq \mu'\right] \quad < \quad 2^{-2k}2^{k(2-m_0\mu')} \leq 2^{-2k}2^{2-m_0\mu'} \tag{30}$$

$$Pr\left[\exists s \in S_k : \frac{S_u(s)}{|S_u|} \geq \mu'\right] \quad \leq \quad |S_k|2^{-2k}2^{2-m_0\mu'} \leq \frac{1}{\mu}2^{-k+1}2^{2-m_0\mu'}. \tag{31}$$

The 2^{-k} factor allows us to sum over all of the k to calculate the probability that there will be an error with any rare string:

$$Pr\left[\exists s \in S_{rare} : \frac{S_u(s)}{|S_u|} \geq \mu'\right] < \frac{1}{\mu}\sum_{k=3}^{\infty} 2^{-k+1}2^{2-m_0\mu'} < \frac{2}{\mu}2^{-m_0\mu'}. \tag{32}$$

The next step is to show that for every $\sigma \in \Sigma \cup \{\zeta\}$,

$$\left|\gamma(q,\sigma) - \frac{S_u(\sigma)}{|S_u|}\right| \leq \varepsilon_1.$$

We can use Chernoff bounds to show that the probability of an error will be less than $e^{-2m_0\varepsilon_1^2}$. Putting these together we can show that the probability of a single multiset being bad can be bounded as in the equation below, and that this is satisfied by the value of m_0 defined above in Equation 6:

$$\frac{16}{\mu}e^{-m_0\mu^2/8} + \frac{2}{\mu}2^{-m_0\mu/4} + (|\Sigma|+1)e^{-2m_0\varepsilon_1^2} < \frac{\delta'}{n|\Sigma|}. \tag{33}$$

6.2 Exit Probability Errors

We next need to show that roughly the expected number of strings exit the graph, if the exit probability is greater than $\varepsilon_6 = \varepsilon_2\varepsilon_5/(L+1)$. Again we can use Chernoff bounds to show that the probability of this sort of error will be less than $e^{-NP_{exit}(G)/4} < e^{-N\varepsilon_6/4}$. It is easy to verify that for our chosen value of N,

$$e^{-N\varepsilon_6/4} < \delta'. \tag{34}$$

6.3 Complexity

Since when the algorithm runs correctly there are at most $n|\Sigma|+2$ steps, the sample complexity will be $N_{total} = N(n|\Sigma|+2)$ which is polynomial by inspection.

As the strings can be of unbounded length the computational complexity has to be bounded in terms of the total lengths of the strings as well. If S_{total} is the multiset representing the overall sample of size at most $N(n|\Sigma|+2)$, then define

$$N_l = \sum_{s\in\Sigma^*} |s|S(s).$$

We denote the maximum observed length L_{max} (clearly $L_{max} \leq N_l$).

Since there are at most n nodes in the graph at any time, and at most $n|\Sigma|$ candidate nodes, the number of comparisons will be at most $n^2|\Sigma|$ at each step and thus $n^2|\Sigma|(n|\Sigma|+2)$ in all. For each multiset we only need to store at most m_0 so there will be at most m_0 distinct strings in each multiset, so the comparison will need to compare at most $2m_0$ strings, and each comparison will take at most L_{max}. The construction of the candidate nodes can be done in less than N_l time. These observations suffice to show that the algorithm is polynomial in N_l and N_{total}. We have assumed here that $|\Sigma|$ is sufficiently small that two characters can be compared in constant time.

7. Conclusion

We have presented an algorithm and a proof that it KL-PAC-learns the class of PDFAs given certain reasonable additional parameters for the sample complexity. We argue that these additional parameters or some substitutes are necessary given the counter-examples previously studied and presented here. Furthermore the classes we present here cover the whole space of distributions defined by PDFAs. Convergence properties of these sorts of state merging algorithms have been studied before in the identification in the limit paradigm (Carrasco and Oncina, 1999), but not in the KL-PAC framework.

The work presented here can be compared to traditional types of PAC learning. Distribution free learning of regular languages has been shown to be hard under some standard cryptographic assumptions (Kearns and Valiant, 1994), but learnability under limited classes of distributions is still an open question. We can view the algorithm presented here, if the graph completion and smoothing is removed, as learning regular languages from positive samples only, where the distributions are restricted to be generated by PDFAs which define the target language. We can compare the result presented in Parekh and Honavar (2001), where it is shown that DFAs can be PAC-learned from positive and negative samples, using a *simple* distribution. This means a distribution where examples of low Kolmogorov complexity, given a representation of the target, have a high probability. Indeed, Denis (2001) shows that using this approach they can also be probably exactly learned from positive examples only. Since the Solomonoff-Levin distribution dominates every other computable distribution, up to a constant factor, this is in some sense the easiest distribution to learn under. Moreover, the use of simple distributions has a number of flaws: first, there are some very large non-computable constants that appear in the sample complexity polynomial. Secondly, representations of the target will also have low complexity with respect to the target, and thus will almost certainly occur in polynomial samples (what is sometimes called "collusion"), which does allow trivial learning algorithms in the case of the positive and negative samples. This may also be the case with positive samples alone. This is an open question, and beyond the scope of this paper, but certainly if the support of the language is large enough, collusive learning becomes possible. Thirdly, the examples provided will be those that are comparatively predictable – i.e. carry little information; in many realistic situations, the strings are used to carry information and thus generally will have high complexity. Thus we feel that though mathematically quite correct, as a characterisation of the convergence properties of algorithms, these approaches are too abstract.

One further extension we intend to study is to the class of the probabilistic residual automata (Esposito et al., 2002) which should be tractable with matrix perturbation theory.

Acknowledgments

We are grateful to anonymous reviewers of this paper and an earlier draft, for helpful comments. We would also like to thank Codrin Nichitiu, Colin de la Higuera and Marc Sebban.

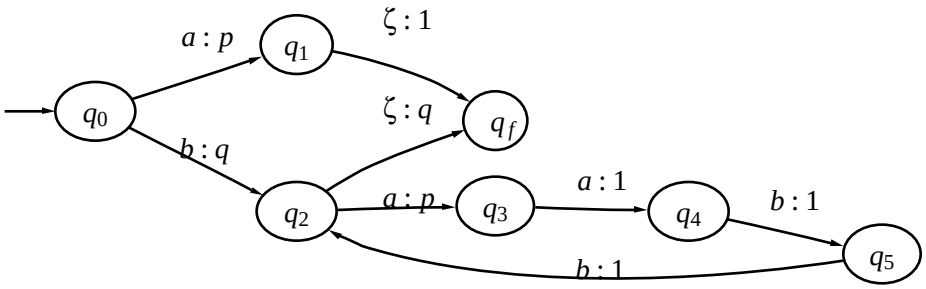


Figure 1: Example automaton with $k = 4$ and $s = aabb$. The arcs are labelled with the symbol and then the probability. p will be very close to 1 and $q = 1 - p$.

Appendix A.

In this appendix we establish our counter-example that justifies our addition of a bound on the expected length of the strings generated from every state. Our basic argument is as follows: we construct a countable family of automata such that given any sample size N, even exponentially large, all but finitely many automata in the family will give the same sample set with probability greater than 0.5. Given this sample set, the algorithm will produce a particular hypothesis. We show that whatever hypothesis, H, a learning algorithm produces must have $D(H||T) > 0.05$ for some of the targets in the subset that will generate this same set. Thus arguing by contradiction, we see that it is not possible to have a PAC-learning algorithm unless we allow it to have a sample complexity polynomial that includes some additional parameter relating in some way to the expected length. We neglect the possibility that the algorithm might be randomised, but it is easy to deal with that eventuality.

It is quite straightforward to construct such a family of automata if we merely want to demonstrate the necessity of a bound on the overall expected length of the strings generated by the automaton. Consider a state of the target that is reached with prob n^{-1}, but that generates strings of length n^2 through a transition to itself of probability $1 - n^{-2}$. For such a state q, the weight $W(q)$ will be n. Thus any small differences in the transitions can cause unbounded increases in the KLD. Here we want to establish a slightly sharper result, which applies even when we have a bound on the overall expected length.

We define a family of automata $\mathcal{F}_k$ for some $k > 0$, over a two letter alphabet $\Sigma = \{a, b\}$, $\mathcal{F}_k = \{A_p^s | s \in \Sigma^k, p \in (0,1)\}$. Each of these automata $A = A_p^s$ defines a distribution as follows:

$$\begin{aligned} P_A(t) = &\quad p \text{ when } t = a; \\ P_A(t) = &\quad (1-p)^2 p^i \text{ when } t = bs^i \text{ for } i \geq 0\ ; \\ P_A(t) = &\quad 0 \text{ otherwise }. \end{aligned}$$

The automata will have $k + 2$ states, and the expected length of the strings generated by these automata will be $k + 1$. Figure 7 shows a simple example for $k = 4$.

Suppose we have an algorithm H that can learn a superset of $\mathcal{F}$. Set $\varepsilon = \frac{1}{8}\log(3/2)$ and $\delta = \frac{1}{2}$. Since we have a polynomial algorithm there must be a polynomial bound on the number of states

of the target $q(\varepsilon,\delta,|\Sigma|,k+2,L)$, when there is an upper bound on the length of the strings in the sample, L. This additional caveat is necessary because the strings in the sample in general can be of unbounded length. Here we will be considering a sample where all the strings are of length 1. Fixing these values of $\varepsilon,\delta,|\Sigma|$ and L we will have a polynomial in k, so we can find a value of k such that $2^k > q(\varepsilon,\delta,|\Sigma|,k+2)$. Denote the smallest such k by k_0. Let N be the sample complexity of the algorithm H for these values. We can set p to be so close to 1 that $p^N > 0.5$, which means that with probability greater than 0.5 the sample generated by the target will consist of N repetitions of the string a. Let $A_H = (\hat{Q},\hat{\gamma},\hat{\tau})$ be the hypothesis automaton produced by the algorithm H on such a data set. By construction $2^k > |\hat{Q}|$.

It is not enough just to show that for some string s the hypothesis will give a low probability to strings of the form bs^i. We must also show that this probability decreases exponentially as i increases. We must therefore show, using a pumping argument, that there is a low probability transition in the cycles in the hypothesis automaton traced by the strings for large values of i. We can assume, without loss of generality, that the hypothesis assigns a non-zero probability to all the strings bs^i, or the KLD will be infinite. For each string $s \in \Sigma^{k_0}$, consider the states in $\hat{Q}$ defined by $\hat{\tau}(\hat{q_0},bs^i)$. There must be two distinct values $i < j \leq |\hat{Q}|$ such that $\hat{\tau}(\hat{q_0},bs^i) = \hat{\tau}(\hat{q_0},bs^j)$, by the pigeonhole principle. Select the smallest i such that this is true, denote these values of i and j by s_i and s_j, and let q_s denote the state $\hat{\tau}(\hat{q_0},bs^{s_i})$. By construction $0 < s_j - s_i \leq |\hat{Q}|$. The state q_s will therefore be in a suitable cycle since

$$q_s = \hat{\tau}(\hat{q_0},bs^{s_i}) = \hat{\tau}(\hat{q_0},bs^{s_j}) = \hat{\tau}(\hat{q_0},bs^{s_i+k(s_j-s_i)})$$

for all $k \geq 0$.

We now want to show that for some string s there is transition in the cycle with a probability at most $\frac{1}{2}$. Since the number of strings is larger than $|\hat{Q}|$ there must be two distinct strings, s,s', such that $q_s = q_{s'}$.

We can write $s = u\sigma v$ and $s' = u\sigma' v'$, where $u,v,v' \in \Sigma^*, \sigma,\sigma' \in \Sigma$. and $\sigma \neq \sigma'$, u here being the longest common prefix of s and s'. Consider the transitions from the state $\hat{\tau}(q_s,u)$. At least one of the two values $\hat{\gamma}(\hat{\tau}(q_s,u),\sigma)$, $\hat{\gamma}(\hat{\tau}(q_s,u),\sigma')$ must be less than or equal to 0.5. Without loss of generality we can say it is σ, which means that $\hat{\gamma}(q_s,s) \leq 1/2$.

This means that we can say the probability of a string of the form $bs^{k|\hat{Q}|+l}$ for any $k,l \geq 0$ must be less than or equal to 2^{-k}. For $k = 0$ this is trivially true. For $k > 0$ define $n = k(|\hat{Q}| + s_i - s_j) + l - s_i \geq 0$ we can write the probability as

$$\begin{aligned}
\hat{\gamma}(\hat{q_0},bs^{k|\hat{Q}|+l}\zeta) &= \hat{\gamma}(q_0,bs^{s_i})\hat{\gamma}(q_s,s^{s_j-s_i})^k\hat{\gamma}(q_s,s^n)\hat{\gamma}(\hat{\tau}(bs^{k|\hat{Q}|+l}),\zeta) \\
&\leq \hat{\gamma}(q_s,s^{s_j-s_i})^k \\
&\leq \hat{\gamma}(q_s,s)^{(s_j-s_i)k} \\
&\leq 2^{-k}.
\end{aligned}$$

We can now use this upper bound on the probability that the hypothesis gives the strings to lower bound the divergence with respect to the target.

Expanding out the definition of the KLD and the target automaton, we have

$$D_{KL}(A_p^s,A_H) = p\log\frac{p}{\hat{\gamma}(\hat{q_0},a\zeta)} + \sum_{i=0}^{\infty}\sum_{j=0}^{|\hat{Q}|-1}(1-p)^2p^{i|\hat{Q}|+j}\log\frac{(1-p)^2p^{i|\hat{Q}|+j}}{\hat{\gamma}(\hat{q_0},bs^{i|\hat{Q}|+j}\zeta)}.$$

Substituting in the bound above, and the fact that $\hat{\gamma}(\hat{q_0}, a\zeta) \leq 1$ yields

$$\begin{aligned} D_{KL}(A_p^s, A_H) &\geq p\log p + \sum_{i=0}^{\infty}\sum_{j=0}^{|\hat{Q}|-1}(1-p)^2 p^{i|\hat{Q}|+j}\log(1-p)^2 p^{i|\hat{Q}|+j}2^i \\ &\geq p\log p + |\hat{Q}|\sum_{i=0}^{\infty}(1-p)^2 p^{i|\hat{Q}|+|\hat{Q}|}\log(1-p)^2 p^{i|\hat{Q}|+|\hat{Q}|}2^i \\ &\geq p\log p + |\hat{Q}|(1-p)^2 p^{|\hat{Q}|}\left(\log(1-p)^2 p^{|\hat{Q}|}\sum_{i=0}^{\infty}p^{i|\hat{Q}|} + \log 2p^{|\hat{Q}|}\sum_{i=0}^{\infty}ip^{i|\hat{Q}|}\right). \end{aligned}$$

Recall that $\sum_{i=0}^{\infty} ip^i = p(1-p)^{-2}$ and $\sum_{i=0}^{\infty} p^i = (1-p)^{-1}$, so that

$$D_{KL}(A_p^s, A_H) \geq p\log p + |\hat{Q}|(1-p)^2 p^{|\hat{Q}|}\left(\log(1-p)^2 p^{|\hat{Q}|}(1-p^{|\hat{Q}|})^{-1} + \log 2p^{|\hat{Q}|}p^{|\hat{Q}|}(1-p^{|\hat{Q}|})^{-2}\right).$$

We can take p to be large enough that $p^{|\hat{Q}|} > 3/4$, giving

$$D_{KL}(A_p^s, A_H) \geq p\log p + |\hat{Q}|\frac{(1-p)^2}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\left(\log(1-p)^2 + (1-p^{|\hat{Q}|})^{-1}\log 3/2\right).$$

Now if we write $p = (1-1/n)$,

$$D_{KL}(A_p^s, A_H) \geq \log p + |\hat{Q}|\frac{(1-p)^2}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\left(\frac{n}{|\hat{Q}|}\log\frac{3}{2} - 2\log n\right).$$

Now using a simple linear bound on the logarithm it can be shown that for any $\beta > 1$ if $n > 2\beta\log 2\beta$ then $\log n < n/\beta$. If we set $\beta = 4|\hat{Q}|/\log(3/2)$ and $n > 8|\hat{Q}|\log|\hat{Q}|$, and $p^{|\hat{Q}|} > (1-|\hat{Q}|/n)$ and assuming that $p > (2/3)^{1/8}$ we have

$$\begin{aligned} D_{KL}(A_p^s, A_H) &\geq \log p + |\hat{Q}|\frac{(1-p)^2}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\frac{n}{2|\hat{Q}|}\log\frac{3}{2} \\ &\geq \log p + \frac{(1-p)}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\frac{1}{2}\log\frac{3}{2} \\ &\geq \log p + \left(\frac{3}{4}\right)^2\frac{1}{2}\log\frac{3}{2} \geq \frac{1}{8}\log\frac{3}{2}. \end{aligned}$$

Thus for sufficiently large values of n, and thus for values of p sufficiently close to 1, there must be an automaton A_p^s such that the algorithm will with probability at least 0.5 produce a hypothesis with an error of at least $\frac{1}{8}\log\frac{3}{2}$. ■

References

N. Abe, J. Takeuchi, and M. Warmuth. Polynomial learnability of stochastic rules with respect to the KL-divergence and quadratic distance. *IEICE Transactions on Information and Systems*, E84-D (3), 2001.

N. Abe and M. K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260, 1992.

P. Adriaans, H. Fernau, and M. van Zaannen, editors. *Grammatical Inference: Algorithms and Applications, ICGI '02*, volume 2484 of *LNAI*, Berlin, Heidelberg, 2002. Springer-Verlag.

R. C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *RAIRO (Theoretical Informatics and Applications)*, 31(5):437–444, 1997.

R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In R. C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications, ICGI-94*, number 862 in LNAI, pages 139–152, Berlin, Heidelberg, 1994. Springer Verlag.

R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *Theoretical Informatics and Applications*, 33(1):1–20, 1999.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, 1991.

C. de la Higuera, J. Oncina, and E. Vidal. Identification of DFA: data-dependent vs data-independent algorithms. In *Proceedings of 3rd Intl Coll. on Grammatical Inference*, LNAI, pages 313–325. Springer, sept 1996. ISBN 3-540-61778-7.

C. de la Higuera and F. Thollard. Identification in the limit with probability one of stochastic deterministic finite automata. In A. de Oliveira, editor, *Grammatical Inference: Algorithms and Applications, ICGI '00*, volume 1891 of *LNAI*, Berlin, Heidelberg, 2000. Springer-Verlag.

F. Denis. Learning regular languages from simple positive examples. *Machine Learning*, 44(1/2): 37–66, 2001.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of proteins and nucleic acids*. Cambridge University Press, 1999.

Y. Esposito, A. Lemay, F. Denis, and P. Dupont. Learning probabilistic residual finite state automata. In Adriaans et al. (2002), pages 77–91.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

M. Kearns and G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *JACM*, 41(1):67–95, January 1994.

M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing*, pages 273–282, 1994.

C. Kermorvant and P. Dupont. Stochastic grammatical inference with multinomial tests. In Adriaans et al. (2002), pages 140–160.

M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(4), 1997.

R. Parekh and V. Honavar. Learning DFA from simple examples. *Machine Learning*, 44(1/2):9–35, 2001.

D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *COLT 1995*, pages 31–40, Santa Cruz CA USA, 1995. ACM.

D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences (JCSS)*, 56(2):133–152, 1998.

F. Thollard, P. Dupont, and C. de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In Pat Langley, editor, *Seventh Intl. Conf. on Machine Learning*, San Francisco, June 2000. Morgan Kaufmann.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134 – 1142, 1984.

Journal of Machine Learning Research 5 (2004) 499-527 Submitted 3/02; Revised 3/03; Published 5/04

Sources of Success for Boosted Wrapper Induction

David Kauchak DKAUCHAK@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093-0144, USA

Joseph Smarr JSMARR@STANFORD.EDU
Symbolic Systems Program
Stanford University
Stanford, CA 94305-2181, USA

Charles Elkan ELKAN@CS.UCSD.EDU
Department of Computer Science and Engineering
University of California at San Diego
La Jolla, CA 92093-0144, USA

Editor: Jaz Kandola

Abstract

In this paper, we examine an important recent rule-based information extraction (IE) technique named Boosted Wrapper Induction (BWI) by conducting experiments on a wider variety of tasks than previously studied, including tasks using several collections of natural text documents. We investigate systematically how each algorithmic component of BWI, in particular boosting, contributes to its success. We show that the benefit of boosting arises from the ability to reweight examples to learn specific rules (resulting in high precision) combined with the ability to continue learning rules after all positive examples have been covered (resulting in high recall). As a quantitative indicator of the regularity of an extraction task, we propose a new measure that we call the SWI ratio. We show that this measure is a good predictor of IE success and a useful tool for analyzing IE tasks. Based on these results, we analyze the strengths and limitations of BWI. Specifically, we explain limitations in the information made available, and in the representations used. We also investigate the consequences of the fact that confidence values returned during extraction are not true probabilities. Next, we investigate the benefits of including grammatical and semantic information for natural text documents, as well as parse tree and attribute-value information for XML and HTML documents. We show experimentally that incorporating even limited grammatical information can increase the regularity of natural text extraction tasks, resulting in improved performance. We conclude with proposals for enriching the representational power of BWI and other IE methods to exploit these and other types of regularities.

Keywords: Boosted wrapper induction, information extraction, boosting

1. Introduction

Computerized text is abundant. Thousands of new web pages appear everyday. News, magazine, and journal articles are constantly being created online. Email has become one of the most popular ways of communicating. All these trends result in an enormous amount of text available in digital form, but these repositories of text are mostly untapped resources of information, and identifying specific desired information in them is a difficult task. Information extraction (IE) is the task of extracting relevant fragments of text from larger documents, to allow the fragments to be processed further in some automated way, for example to answer a user query. Examples of IE tasks include identifying the speaker featured in a talk announcement, finding the proteins mentioned in a biomedical journal article, and extracting the names of credit cards accepted by a restaurant from an online review.

A variety of systems and techniques have been developed to address the information extraction problem. Successful techniques include statistical methods such as n-gram models, hidden Markov models, probabilistic context-free grammars (Califf, 1998), and rule-based methods that employ some form of machine learning. Rule-based methods have been especially popular in recent years. They use a variety of different approaches, but all recognize a number of common key facts. First, creating rules by hand is difficult and time-consuming (Riloff, 1996). For this reason, most systems generate their rules automatically given labeled or partially labeled data. Second, generating a single, general rule for extracting all instances of a given field is often impossible (Muslea et al., 1999). Therefore, most systems attempt to learn a number of rules that together cover the training examples for a field, and then combine these rules in some way.

Some recent techniques for generating rules in the realm of text extraction are called "wrapper induction" methods. These techniques have proved to be fairly successful for IE tasks in their intended domains, which are collections of highly structured documents such as web pages generated from a template script (Muslea et al., 1999; Kushmerick, 2000). However, wrapper induction methods do not extend well to natural language documents because of the specificity of the induced rules.

Recent research on improving the accuracy of weak classifiers using boosting (Schapire, 1999) has led to methods for learning classifiers that focus on examples that previous classifiers have found difficult. The AdaBoost algorithm works by iteratively learning a new classifier using a base learner and then reweighting the training examples to emphasize those that the current set of classifiers does not perform well on. The classifiers learned are then combined by voting with weights, where the weight of each classifier depends on its accuracy on its reweighted training set. Boosting has been shown theoretically to converge to an optimal combined classifier on the training set, and typically generalizes well in practice.

Boosted Wrapper Induction (BWI) is an IE technique that uses the AdaBoost algorithm to generate a general extraction procedure that combines a set of specific wrappers (Freitag and Kushmerick, 2000). Its authors showed that BWI performs well on a variety of tasks with moderately structured and highly structured documents. They compared BWI to several other IE methods, but exactly how boosting contributes to the success of BWI has not been investigated previously. Furthermore, BWI has been proposed as an IE method for unstructured natural language documents, but little has been done to examine its performance in this challenging direction.

In this paper, we investigate the benefit of boosting in BWI and also the performance of BWI on natural text documents. We compare the use of boosting in BWI with the most common approach

to combining individual extraction rules: sequential covering. With sequential covering, extraction rules are ordered in some way according to performance on the training set. The best rule is chosen, and all examples that this rule correctly classifies are removed from the training set. A new best rule is learned, and the process is repeated until the entire training set has been covered. For a more detailed description of sequential covering see Cardie and Mooney (1998). Sequential covering has been used in some modern IE systems (Muslea et al., 1999) and is a common algorithmic idea in a wide range of other learning methods (Clark and Niblett, 1989; Michalski, 1980; Quinlan, 1990). Sequential covering is popular because it is simple to implement, tends to generate understandable and intuitive rules, and has achieved good performance.

This paper is divided into a number of sections. In Section 2, we briefly describe BWI and related techniques, providing a formalization of the problem and a review of relevant terminology. In Section 3, we present experimental results comparing these different rule-based IE methods on a wide variety of document collections. In Section 4, we analyze the results of the experiments in detail, with specific emphasis on how boosting affects the performance of BWI, and how performance relates to the regularity of the extraction task. In Section 5, we introduce the SWI ratio as a measure of task regularity, and further examine the connection between regularity and performance. In Section 6, we broaden the discussion and investigate important limitations of BWI and other current IE methods, focusing on what information is used and how it is represented, how results are scored and selected, and the efficiency of training and testing. Finally, in Section 7 we suggest ways to address these limitations, and provide experimental results that show that including grammatical information in the extraction process can increase exploitable regularity and therefore performance.

2. Overview of Algorithms and Terminology

In this section we present a brief review of the BWI approach to information extraction, including a precise problem description, a summary of the algorithms used, and important terminology. We also present a simplified variant of the BWI algorithm, called SWI, which will be used to analyze BWI and related algorithms.

2.1 Information Extraction as a Classification Task

Most of the material in this section is based on Freitag and Kushmerick (2000). We present an abridged version here for convenience, starting with a review of relevant vocabulary and assumptions. A document is a sequence of tokens. A token is one of three things: an unbroken string of alphanumeric characters, a punctuation character, or a carriage return. The problem of information extraction is to extract subsequences of tokens matching a certain pattern from test documents. To learn the pattern, we reformulate the IE problem as a classification problem. Instead of thinking of the objective as a substring of tokens, we look at the problem as a function over boundaries. A boundary is the space between two tokens. Notice that a boundary is not something that is actually in the text (such as white space), but a notion that arises from the parsing of the text into tokens. We want to learn two classifiers, that is two functions from boundaries to the binary set $\{0,1\}$: one function that is 1 iff the boundary is the beginning of a field to be extracted, and one function that is 1 iff the boundary is the end of a field. This transformation from a segmentation problem to a boundary classification problem is also used, for example, by Shinnou (2001) to find word boundaries in Japanese text, since written Japanese does not use spaces.

```
SWI: training sets S and E -> two lists of detectors and length histogram
  F = GenerateDetectors(S)
  A = GenerateDetectors(E)
  H = field length histogram from S and E
  return <F,A,H>

GenerateDetectors: training set Y(S or E) -> list of detectors
   prefix pattern p = []
   suffix pattern s = []
   list of detectors d

   while(positive examples are still uncovered)
     <p,s> -> FindBestDetector()
     add <p,s> to d
     remove positive examples covered by <p,s> from Y
     <p,s> = []

   return d
```

Figure 1: The SWI and GenerateDetectors algorithms.

Each of the two classifiers is represented as a set of boundary detectors, also called just detectors. A detector is a pair of token sequences $\langle p, s \rangle$. A detector matches a boundary iff the prefix string of tokens, *p*, matches the tokens before the boundary and the suffix string of tokens, *s*, matches the tokens after the boundary. For example, the detector $\langle$Who:, Dr.$\rangle$ matches "Who: Dr. John Smith" at the boundary between the colon ':' and the 'Dr'. Given beginning and ending classifiers, extraction is performed by identifying the beginning and end of a field and taking the tokens between the two points.

BWI separately learns two sets of boundary detectors for recognizing the alternative beginnings and endings of a target field to extract. During each iteration, BWI learns a new detector, and then uses ADABOOST to reweight all the examples in order to focus on portions of the training set that the current rules do not match well. Once this process has been repeated for a fixed number of iterations, BWI returns the two sets of learned detectors, called fore detectors, F, and aft detectors, A, as well as a histogram, H, over the lengths (number of tokens) of the target field. The next subsection explains in more detail the BWI algorithm, with pseudocode, and identifies the differences between BWI and SWI.

2.2 Sequential Covering Wrapper Induction (SWI)

SWI uses the same framework as BWI with some modifications to the choice of individual detectors. The basic algorithm for SWI can be seen in Figure 1. SWI generates F and A independently by calling the GENERATEDETECTORS procedure, which is a sequential covering rule learner. During each iteration through the while loop, a new boundary detector is learned by the FINDBESTDETECTOR method. This method works by separately learning a prefix and suffix portion. As with BWI, this is done by starting with an empty sequence. Then, a single token is added by exhaustively

enumerating all possible extensions of length L, picking the best one based on the scoring function (scoring functions will be discussed below) and adding the first token of that extension. This process is repeated, adding one token at a time, until the detector being generated no longer improves, as measured by the scoring function.

Once the best boundary detector has been found, it is added to the set of detectors to be returned, either F or A. Before continuing, all positive examples covered by the new rule are removed, leaving only uncovered examples. As soon as the set of detectors covers all positive training examples, this set is returned.

Given the sets of detectors F and A and the field length histogram, a field is extracted by applying the detectors in F and A to each boundary in the test document. For each boundary, an F score and an A score is computed as the sum of the confidence values of the detectors that match the boundary. A field is extracted if the product of the F score at the beginning of the field, the A score at the end of the field and the length histogram probability is greater than a predefined threshold (usually zero).

To clarify, there is a distinction between the actual text, which is a sequence of tokens, and the two functions to be learned, which indicate whether a boundary is the start or end of a field to be extracted. When the SWI algorithm "removes" the boundaries that are matched by a detector, these boundaries are relabeled in the training documents, but the actual sequence of tokens is not changed.

There are two key differences between BWI and SWI. First, as mentioned above, with sequential covering covered positive training examples are removed. In BWI, examples are reweighted but never relabeled. Second, the scoring functions for the extensions and detectors are slightly different. We investigate two different versions of SWI. The basic SWI algorithm uses a simple greedy method. The score of an extension or detector is the number of positive examples that it covers if it covers no negative examples, zero otherwise. So, if a detector misclassifies even one example (i.e. covers a negative example), it is given a score of zero. We call this version Greedy-SWI. A second version of SWI, called Root-SWI, uses the same scoring function as BWI. Given the sum of the weights of the positive examples covered by the extension or detector, W^+, and the sum of the weights for the negative examples covered, W^-, the score is calculated to minimize training error (Cohen and Singer, 1999):

$$score = \sqrt{W^+} - \sqrt{W^-}.$$

Notice that for Root-SWI the two sums are the numbers of examples covered because examples are all given the same weight. The final difference between BWI and SWI is that BWI terminates after a predetermined number of boosting iterations. In contrast, SWI terminates as soon as all of the positive examples have been covered. This turns out to be a particularly important difference between the two different methods.

2.3 Wildcards

One additional aspect of both BWI and SWI that needs further explanation is the use of wildcards. Both BWI and SWI extend boundary detectors using tokens as found in the training examples. However, using literal tokens does not allow these algorithms to recognize common regularities in the documents. For example, given the task of identifying phone numbers in a document, the algorithms as described above would be unable to match a future phone numbers unless that specific number had already been seen in a training example. To overcome this limitation, wildcards that represent categories of tokens are used. Freitag and Kushmerick describe eight basic wildcards, which are also used in our work, representing alphabetic tokens, alphanumeric tokens, tokens be-

ginning with a capitalized letter, tokens beginning with a lower-case letter, single character tokens, numeric tokens, punctuation tokens, and any token.

3. Experiments and Results

In this section, we describe the data sets used for our set of experiments, our setup and methods, and the numerical results of our tests. We provide an analysis of the results in the next section.

3.1 Document Collections and IE Tasks

In order to compare BWI, Greedy-SWI and Root-SWI we examine the three algorithms on 15 different information extraction tasks using 8 different document collections. Many of these tasks are standard and have been used in testing a variety of IE techniques. The document collections can be categorized into three groups: natural (unstructured) text, partially structured text, and highly structured text. The breadth of these collections allows for a comparison of the algorithms on a wider spectrum of tasks than has previously been studied. The natural text documents are biomedical journal abstracts and contain little obvious formatting information. The partially structured documents resemble natural text, but also contain document-level structure and have standardized formatting. These documents consist primarily of natural language, but contain regularities in formatting and annotations. For example, it is common for key fields to be preceded by an identifying label (e.g. "Speaker: Dr. X"), though this is not always the case. The highly structured documents are web pages that have been automatically generated. They contain reliable regularities, including location of content, punctuation and style, and non-natural language formatting such as HTML tags. The partially structured and highly structured document collections were obtained primarily from RISE (1998) while the natural text documents were obtained from the National Library of Medicine MEDLINE abstract database (National Library of Medicine, 2001).

We use two collections of natural language documents, both consisting of individual sentences from MEDLINE abstracts. The first collection consists of 555 sentences tagged with the names of proteins and their subcellular locations. These sentences were automatically selected and tagged from larger documents by searching for known proteins and locations contained in the Yeast-Protein-Database (*YPD*). Because of the automated labeling procedure, the labels are incomplete and sometimes inaccurate. Ray and Craven (2001) estimate an error rate in labeling between 10% and 15%. The second collection consists of 891 sentences, containing genes and their associated diseases, as identified using the Online-Mendelian-Inheritance-In-Man (*OMIM*) database, and is subject to the same noisy labeling. Both these collections have been used in other research, including Ray and Craven (2001) and Eliassi-Rad and Shavlik (2001).

It should be noted that the original *OMIM* and *YPD* document collections contain many sentences labeled with no training examples of fields to extract, because the collections were created for extracting field pairs only. Therefore, only in cases where both fields in a pair were found was any training label added. Because BWI is designed to extract only single fields, we remove these sentences. Many of them in fact contain examples of desired fields, so when unlabeled, they present incorrect training information. Moreover, the unlabeled documents originally outnumber the labeled documents by almost an order of magnitude. Our restricted sentence collections still pose challenging information extraction tasks, but performance on these collections cannot be compared to performance reported for extracting field pairs in the original collections.

For the partially structured extraction tasks, we use three different document collections. The first two are taken from RISE, and consist of 486 speaker announcements (*SA*) and 298 Usenet job announcements (*Jobs*). In the speaker announcement collection, four different fields are extracted: the speaker's name (*SA-speaker*), the location of the seminar (*SA-location*), the starting time of the seminar (*SA-stime*) and the ending time of the seminar (*SA-etime*). In the job announcement collection, three fields are extracted: the message identifier code (*Jobs-id*), the name of the company (*Jobs-company*) and the title of the available position (*Jobs-title*).

We created the third collection of partially structured documents from 630 MEDLINE article citations. These documents contain full MEDLINE bibliographic data for articles on hip arthroplasty surgery, i.e. author, journal, and publication information, MeSH headings (a standard controlled vocabulary of medical subject keywords), and other related information, including the text of the paper's abstract about 75% of the time. The task is to identify the beginning and end of the abstract, if the citation includes one (*AbstractText*). The abstracts are generally large bodies of text, but without any consistent beginning or ending marker, and the information that precedes or follows the abstract text varies from citation to citation.

Finally, the highly structured tasks are taken from three different document collections, also in RISE: 20 web pages containing restaurant descriptions from the Los Angeles Times (*LATimes*), where the task is to extract the list of accepted credit cards (*LATimes-cc*); 91 web pages containing restaurant reviews from the Zagat Guide to Los Angeles Restaurants (*Zagat*), where the task is to extract the restaurant's address (*Zagat-addr*); and 10 web pages containing responses from an automated stock quote service (*QS*), where the task is to extract the date of the response (*QS-date*). Although these collections contain a small number of documents, the individual documents typically comprise several separate entries, i.e. there are often multiple stock quote responses or restaurant reviews per web page.

3.2 Experimental Design

The performance of the different rule-based IE methods on the tasks described above is measured using three different standard metrics: precision, recall, and F1, the harmonic mean of precision and recall. F1 is used as the principal metric to compare different algorithms for several reasons: it balances precision and recall, it is common in previous research on information extraction and information retrieval, and it allows experimental results to be quantified with a single number.

Given that we analyze BWI, SWI and related algorithms in a range of applications, and in many domains both precision and recall are important, we consider the F1 measure to be better than alternative single measures, but it is still somewhat arbitrary. For many applications, there are asymmetric costs associated with false positives and false negatives. For example, high precision is necessary when IE is used to populate a database automatically, while high recall is important when IE is used to find descriptions of rare events that are later reviewed by a human analyst. In these applications it is beneficial to make explicit the relative costs of false positives and false negatives, and to use a cost-sensitive learning method (Elkan, 2001).

Each result presented is the average of ten random 75% train/25% test splits. For all algorithms, a lookahead value, L, of 3 tokens is used. For the natural and partially structured texts (*YPD*, *OMIM*, *SA*, *Jobs*, and *AbstractText*), the default BWI wildcard set is used (the eight wildcards described in Section 2.3), and for the web pages (*LATimes*, *Zagat*, and *QS*), three lexical wildcards are used in addition; these represent last names, first names, and words not found in the standard Unix list

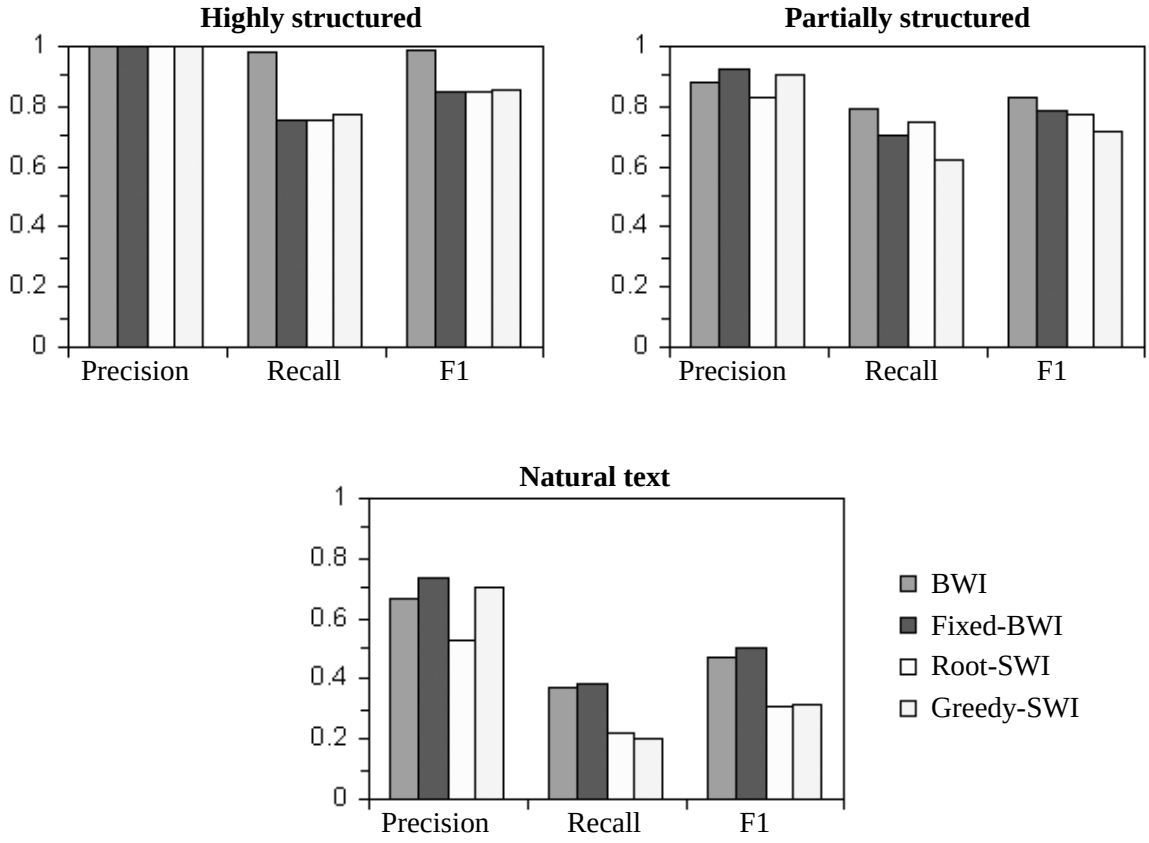


Figure 2: Performance of the four IE methods examined in this paper, separated by overall regularity of document collection (indicated above each chart), averaged over all tasks within the given collections.

of words /USR/DICT/WORDS. A graphical summary of our results can be seen in Figure 2. For complete numerical results, see Table 1 at the end of this paper.

To understand the differences between BWI and SWI, we performed four sets of tests. First, we ran BWI with the same number of rounds of boosting used by Freitag and Kushmerick (2000): for the *YPD*, *OMIM*, *SA*, *Jobs*, and *AbstractText* document collections, 500 rounds of boosting were used; for the *LATimes*, *Zagat*, and *QS* document collections, 50 rounds were used. Next, we ran Greedy-SWI and Root-SWI on the same tasks until all of the examples were covered. The actual number of iterations for which these algorithms ran varied across the different IE tasks. These numbers are presented in Table 2 at the end of the paper. Finally, we ran BWI for the same number of iterations that it took for Root-SWI to complete each task: 323 rounds for *YPD-protein*, one round for *QS-date*, and so on; this method is called Fixed-BWI.

3.3 Experimental Results

To investigate whether BWI outperforms the set-covering approach of SWI, we compare Greedy-SWI and BWI. BWI appears to perform better, since the F1 values for BWI are higher for all the highly structured and natural text tasks studied. Greedy-SWI has slightly higher F1 performance on two of the partially structured tasks, but considerably lower performance on the other three. Generally, Greedy-SWI has slightly higher precision, while BWI tends to have considerably higher recall.

Given these results, the next logical step is determining what part of the success of BWI comes from the difference in scoring functions used by the two algorithms, and what part comes from how many rules are learned. To explore the first of these differences, we compare Greedy-SWI and Root-SWI, which differ only by their scoring function. Greedy-SWI tends to have higher precision while Root-SWI tends to have higher recall. However, they have similar F1 performance, a result that is illustrated further by comparing BWI to Root-SWI: BWI still has higher F1 performance than Root-SWI in all but three partially structured tasks, despite the fact that they use an identical scoring function.

There are only two differences between BWI and Root-SWI. First, after each iteration Root-SWI removes all positive examples covered, whereas BWI merely reweights them. Second, Root-SWI terminates as soon as all positive examples have been covered, whereas BWI continues to boost for a fixed number of iterations. Table 2 shows that Root-SWI always terminates after many fewer iterations than were set for BWI. Freitag and Kushmerick (2000) examined the performance of BWI as the number of rounds of boosting increases, and found improvement in many cases even after more than 500 iterations.

To investigate this issue further, we ran BWI for the same number of iterations as it takes Root-SWI to complete each task; we call this method Fixed-BWI. Recall that the number of rounds Fixed-BWI runs for depends on the extraction task. Here the results vary qualitatively with the level of structure in the domain. In the highly structured tasks, Fixed-BWI and Root-SWI perform nearly identically. In the partially structured tasks, Fixed-BWI tends to exhibit higher precision but lower recall than Root-SWI, resulting in similar F1 values. In the unstructured tasks, Fixed-BWI has considerably higher precision and recall than Root-SWI.

3.4 Statistical Significance of Experiments

In this section and in following sections (4 and 7) we compare the performance of algorithmic variations on the different data sets. When comparing the performance of algorithms, particularly with limited data, it is common to verify that the differences in performance are not due to chance. This is usually done using a statistical test of significance. We checked the validity of all of our comparisons using the popular paired *t*-test. All comparisons we examined are statistically significant based on this test. The worst significant level found was 0.0005 and most comparisons were significant with much greater confidence. We chose the *t*-test because it has been commonly used in the past. Dietterich (1998) mentions a number of concerns regarding the *t*-test, however, the ten 75%/25% splits lessens these problems and all the tests examined were much more confident than the common 0.01 significance level.

4. Analysis of experimental results

In this section, we discuss our experimental results from two different angles. We examine the effect of the algorithmic differences between the IE methods studied, and we look at how overall performance is affected by the inherent difficulty of the IE task.

4.1 Why BWI Outperforms SWI

The experiments performed yield a detailed understanding of how the algorithmic differences between BWI and SWI allow BWI to achieve consistently higher F1 values than SWI. The most important difference is that BWI uses boosting to learn rules as opposed to set covering. Boosting has two complementary effects. First, boosting continually reweights all positive and negative examples to focus on the increasingly specific problems that the existing set of rules is unable to solve. This tends to yield high precision rules, as is clear from the fact that Fixed-BWI consistently has higher precision than Root-SWI, even though they use the same scoring function and learn the same number of rules. While Greedy-SWI also has high precision, this is achieved by using a scoring function that does not permit any negative examples to be covered by any rules. This results in lower recall than with the other three methods, because general rules with wide coverage are hard to learn without covering some negative examples.

Second, boosting allows BWI to continue learning even when the existing set of rules already covers all the positive training examples. Unlike BWI, SWI is limited in the total number of boundary detectors it can learn from a given training set. This is because every time a new detector is learned, all matching examples are removed, and training stops when there are no more positive examples left to cover. Since each new detector must match at least one positive example, the number of boundary detectors SWI can learn is at most equal to the number of positive examples in the training set. (Usually many fewer detectors are learned because multiple examples are covered by single rules.) In contrast, after each iteration BWI reweights examples, but does not remove them entirely, so there is no limit in principle to how many rules can be learned. The ability to continue learning rules means that BWI can not only learn to cover all the positive examples in the training set, but it can widen the margin between positive and negative examples, learning redundant and overlapping rules, which together better separate the positive and negative examples.

4.2 The Consequences of Task Difficulty

As just explained, boosting gives BWI the consistent advantage in F1 performance observed in the experiments of Section 3. However, the difficulty of the extraction task also has a pronounced effect on the performance of different IE methods. We now turn to a specific investigation of each document collection.

4.2.1 HIGHLY STRUCTURED IE TASKS

These tasks are the easiest for all methods. It is no coincidence that they are often called "wrapper tasks" as learning the regularities in automatically generated web pages was precisely the problem for which wrapper induction was originally proposed as a solution. All methods have near-perfect precision, and SWI tends to cover all examples with only a few iterations.

Surprisingly, SWI does not achieve near-perfect recall. While these tasks are highly regular, the regularities learned on the first couple of iterations are not the only important ones. Neither changing

the scoring function from "greedy" to "root" nor changing the iteration method from set covering to boosting solves this problem, as Root-SWI and Fixed-BWI have virtually identical performance to that of Greedy-SWI. However, because BWI continues to learn new and useful rules even after all examples are covered, it is able to learn secondary regularities that increase its recall and capture the cases that are missed by a smaller set of rules. Thus BWI achieves higher recall than the other methods on these extraction tasks.

4.2.2 PARTIALLY STRUCTURED IE TASKS

For these tasks, we see the largest variation in performance between the IE algorithms investigated. Compared to Greedy-SWI, Root-SWI has increased recall, decreased precision, and similar F1. By allowing rules to cover some negative training examples, more general rules can be learned that have higher recall, but also cover some negative test examples, causing lower precision. Root-SWI consistently terminates after fewer iterations than Greedy-SWI, confirming that it is indeed covering more examples with fewer rules.

Changing from set covering to boosting as a means of learning multiple rules results in a precision/recall tradeoff with little change in F1. As mentioned earlier, boosting reweights the examples to focus on the hard cases. This results in rules that are very specific, with higher precision but also lower recall. Boosting also results in a slower learning curve, measured in performance versus number of rules learned, because positive examples are often covered multiple times before all examples have been covered once. SWI removes all covered examples, focusing at each iteration on a new set of examples. The consequence is that Fixed-BWI cannot learn enough rules to overcome its bias towards precision in the number of iterations Root-SWI takes to cover all the positive examples. However, if enough iterations of boosting are used, BWI compensates for its slow start by learning enough rules to ensure high recall without sacrificing high precision.

4.2.3 UNSTRUCTURED TEXT IE TASKS

Extensive testing of BWI on natural text has not been previously done, though it was the clear vision of Freitag and Kushmerick to pursue research in this direction. As can be seen in Figure 2, on the natural text tasks all the IE methods investigated have relatively low precision, despite the bias towards high precision of the boundary detectors. All the methods also have relatively low recall. This is mainly due to the fact that the algorithms often learn little more than the specific examples seen during training, which usually do not appear again in the test set. Looking at the specific boundary detectors learned, most simply memorize individual labeled instances of the target field, ignoring context altogether: the fore detectors have no prefix, and the target field as the suffix, and vice versa for the aft detectors. Changing the SWI scoring function from "greedy" to "root" results in a marked decrease in precision with only a small increase in recall. Even when negative examples may be covered, the rules learned are not sufficiently general to increase recall significantly.

Unlike on the partially and highly structured IE tasks, Fixed-BWI has both higher precision and higher recall than Root-SWI on the natural text tasks. Higher precision with boosting is explainable as above. Boosting also increases recall because while already covered examples are down-weighted during boosting, they are not removed entirely. Thus, BWI remains sensitive to the performance on all training examples of all the rules it learns. In contrast, SWI quickly captures the regularities that can be easily found, and then begins covering positive training examples one at a time, ignoring the performance of new rules on already covered positive examples. The problem that SWI learns

many rules with low recall is compounded by the fact that it never removes negative examples. This behavior of SWI is particularly troublesome because the labelings of the collections of natural text documents are partially inaccurate, as explained in Section 3.1. Unlike on the more structured tasks, allowing BWI to run for 500 iterations causes it to perform worse than Fixed-BWI, suggesting that the extra iterations result in overfitting. Because of the irregularity of the data, Fixed-BWI runs for more iterations on natural text tasks than on easier tasks. The last rounds of boosting tend to concentrate on only a few highly weighted examples, meaning that unreliable rules are learned, which are more likely to be artifacts of the training data than true regularities.

5. Quantifying Task Regularity

In addition to the observed variations resulting from changes to the scoring function and iteration method of BWI and SWI, it is clear that the inherent difficulty of the extraction task is also a strong predictor of performance. All the algorithms we consider easily handle highly structured documents such as those generated from a database and an HTML template, while natural text documents are uniformly more difficult. It is thus interesting to investigate methods for measuring the regularity of an information extraction task numerically and objectively, as opposed to subjectively and qualitatively. In this section we provide such a measure, which we find to be useful in our work and which we expect to be useful in future research.

5.1 SWI Ratio as a Measure of Task Regularity

We propose that a good measure of the regularity of an extraction task is the number of iterations SWI takes to cover all the positive examples in a training set, divided by the total number of positive examples in the training set. We call this measure the *SWI ratio*. In the most regular limit, a single rule would cover an infinite number of documents, and thus the SWI ratio would be $1/\infty = 0$. At the other extreme, in a completely irregular set of documents, SWI would have to learn a separate rule for each positive example (i.e. there would be no generalization possible), so for N positive examples, SWI would need N rules, for an SWI ratio of $N/N = 1$. Since, each SWI rule must cover at least one positive example, the number of SWI rules learned for a given document set will always be less than or equal to the total number of positive examples. So, the SWI ratio will always be between 0 and 1, with a lower value indicating a more regular extraction task.

The SWI ratio is a sensible and well-motivated measure of task regularity for several reasons. First, it is a relative measure with respect to the number of training examples, so one can use it to compare the regularity of tasks with small and large training collections of documents. Second, it is simple and objective, because SWI is a straightforward algorithm, with no parameters to set other than the lookahead for extending boundary-detector rules, which is fixed at $L = 3$ in all our tests, and the set of wildcards, for which using the default set of eight makes sense in a wide range of domains. Third, the SWI ratio is efficient to compute, and it is practical to run SWI before or while running any other technique. Finally, and most importantly, the SWI ratio correlates with task regularity as perceived intuitively by humans. Table 1 shows the SWI ratios for all of the extraction tasks. The highly structured tasks have an average SWI ratio of 0.027, the partially structured tasks have an average SWI ratio of 0.178 and the unstructured tasks have an average SWI ratio of 0.539. The tasks that are perceived by humans as being the most regular, such as dates, times, credit card numbers and addresses, have low SWI ratio values. Rules can easily be generated by hand to extract these

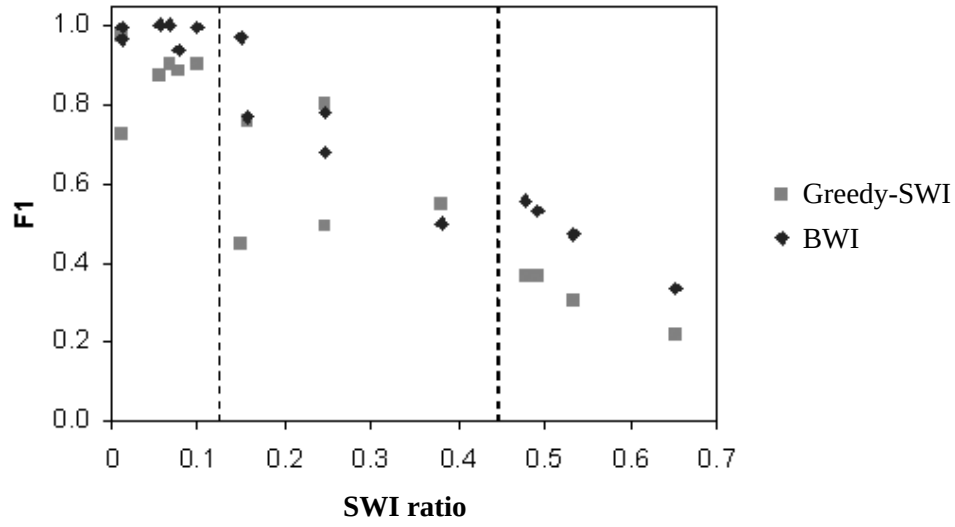


Figure 3: F1 performance for BWI and Greedy-SWI on the 15 different extraction tasks, plotted versus the SWI ratio of the task, with separations between highly structured, partially structured, and natural text.

fields. On the other hand, the tasks involving the MEDLINE articles, which are human-generated documents, have the highest SWI ratio values.

The SWI ratio has a similar motivation to other measures of regularity. Rendell and Cho (1990) examine the concentration of the data set, which is the distribution of the positive instances throughout the instance space. They introduce the idea of a peak (a neighborhood of values that has a class membership that is greater than average) and show that as the number of peaks increases (i.e. the concentration of the data decreases) the performance decreases. The SWI ratio can be seen as a similar, though not identical, measure of concentration. The SWI ratio is also similar to a VC bound on generalization error. Intuitively, it is the capacity (i.e. complexity) of the space of classifiers needed to achieve zero training error divided by the number of training examples; this ratio is expected theoretically to be related to the generalization error rate (Vapnik et al., 1994).

5.2 Performance and Task Regularity

In addition to the clear differences in performance between the three classes of extraction tasks presented in the previous sections, there is also wide variation within each class of tasks. Each value in Figure 2 is an average over several extraction tasks. Using the SWI ratio we can compare algorithms on a per-task basis. This reveals in greater detail both how performance is related to task difficulty, and whether BWI reliably outperforms SWI.

In Figure 3, F1 values for Greedy-SWI and BWI are plotted for each task versus its SWI ratio. As noted in Section 4, BWI performs better than Greedy-SWI on all but two tasks, and the F1 difference on those tasks is small. By plotting the performance of each algorithm versus the SWI ratio, we can examine how the two algorithms succeed as the regularity of the tasks decreases.

There is a consistent decline in performance as the tasks decrease in human-perceived regularity, for both algorithms. Also, there is no clear divergence in performance between the two algorithms as the regularity of the task decreases. So, although BWI appears to reliably outperform SWI, both algorithms are still limited by the regularity of the task to be solved.

5.3 The SWI Ratio as a Tool for Future Research

A numerical and objective measure of the regularity of information extraction tasks is useful not only for predicting how well a known IE method will do on a new task, but also for guiding research to improve IE methods. In Sections 5.1 and 5.2 above, the human perception of task regularity is confirmed by measurement with the SWI ratio, while in Section 7.1 below, the SWI ratio is used to show how the regularity of an IE task can be increased under certain conditions. We hope that future research will show how to use the SWI ratio to help in choosing the right technique for a given IE task. For example, a given IE method may tend to be particularly suitable for tasks whose SWI ratio falls in a particular range.

6. Opportunities for Improving BWI

Having investigated BWI as an important recent information extraction technique, and having understood its advantages over variant methods, we now broaden the discussion to investigate a number of limitations of BWI. In particular, we examine types of information that BWI fails to use, or does not represent completely, as well as speed and efficiency problems. We examine the limitations of BWI in the hope of creating a useful list of important issues to consider when designing and evaluating many future IE systems.

As shown in Section 3 above and in numerous other papers, many IE systems perform quite well on fairly structured tasks. If the target field to be extracted is canonically preceded or followed by a given set of tokens, or by tokens of a distinct type, represented by the wildcards available, boundary detectors easily represent this context. This is a common occurrence in highly structured and partially structured documents, where fields are often preceded by identifying labels (e.g. “Speaker: Dr. X”), or followed by identifiable pieces of information (e.g. in the Zagat survey, a restaurant address is almost always followed by its telephone number, which is easily distinguished from the rest of the text).

Surprisingly, there is a good deal of this type of regularity to be exploited even in more natural texts. For example, when extracting the locations where proteins are found, “located in” and “localizes to” are common prefixes, and are learned early by BWI. In general, human authors often provide a given type of information only in a certain context, and specific lead-in words or following words are used repeatedly.

While rule-based IE methods are primarily designed to identify contexts outside target fields, BWI and other methods do also learn about some of the regularities that occur inside the fields being extracted. In the case of BWI, boundary detectors extend into the edge of the target field as well as into the local context. So the fore detectors can learn what the first few tokens of a target field look like, if the field tends to have a regular beginning, and the aft detectors can learn what the last few tokens look like.

With short fields, individual boundary detectors often memorize instances of the target field. This happens in the MEDLINE tasks, where specific gene names, protein names, etc. are memorized when their context is not otherwise helpful. However, with longer fields, there is still important

information learned. For example, in the MEDLINE citations, the abstract text often starts with phrases like "OBJECTIVE: ", or "We investigated..."

In addition to learning the typical starting and ending tokens of a target field, BWI learns a probability distribution for the number of tokens in the field. Specifically, the field length is recorded for each training example, and this histogram is normalized into a probability distribution once training is complete. In BWI, length is the only piece of information that ties fore detectors and aft detectors together. Length information can be extremely useful. When BWI runs on a new test document, it first notes every fore and aft detector that fire, and then pairs them up to find specific token sequences. BWI only keeps a match that is consistent with the length distribution. Specifically, it drops matches whose fore and aft boundaries are farther apart or closer together than seen during training. Also, given two overlapping matches of equal detector confidence, it prefers the match whose length has been seen more times.

In all but the most regular IE tasks, a single extraction rule is insufficient to cover all the positive examples, so it is necessary to learn a set of rules that capture different regularities. An important piece of information to capture, then, is the relative prominence of the different pattern types, to distinguish the general rules from the exceptional cases. In the case of BWI, this information is learned explicitly by assigning each boundary detector a confidence value, which is proportional to the total weight of positive examples covered by the rule during training, and so reflects the generality of the rule. In SWI, rules that cover the most examples are typically learned first, so there is a ranking of rules.

6.1 Representational Expressiveness

BWI learns sets of fore and aft boundary detectors and a histogram of the lengths of the fields. The boundary detectors are short sequences of specific words or wildcards that directly precede or follow the target field or that are found within the target field. As a consequence of this representation, there are valuable types of regularity that cannot be captured, including a detailed model of the field to be extracted, the global location of the field within documents, and structural information such as that exposed by a grammatical parse or an HTML/XML document tree.

6.1.1 LIMITED MODEL OF THE CONTENT BEING EXTRACTED

Clearly an important clue in finding and extracting a particular fact is recognizing what relevant fragments of text look like. As mentioned in the previous section, BWI can learn the canonical beginnings and endings of a target field, as well as a distribution over the length of the field. However, both these pieces of information are learned in a superficial manner, and much of the regularity of the field is ignored entirely.

Currently, BWI normalizes its frequency counts of field lengths in the training data without any generalization. This means that any candidate field to be extracted whose length is not precisely equal to that of at least one target field in the training data will be dismissed, no matter how compelling the boundary detection. This does not cause problems for very short fields, when there are only a few possible lengths, but it is a major problem for fields with a wide variance of lengths. For example, in the MEDLINE citations task, the abstract texts to be extracted have a mean length of 230 tokens, with a standard deviation of 264 tokens. With only 462 positive examples of abstracts, in 630 citations, many reasonable abstract lengths are never seen in training data.

In addition to the length distribution, the only other information BWI learns about the content of the target field comes from boundary detectors that overlap into the field. Often, more information about the canonical form of the target field could be exploited. There has been a great deal of work on modeling fragments of text to identify them in larger documents, most under the rubric Named Entity Extraction. For example, Baluja et al. (1999) reports impressive performance finding proper names in free text, using only models of the names themselves, based on capitalization, common prefixes and suffixes, and other word-level features. NYMBLE is another notable effort in this field (Bikel et al., 1997). Combining these field-finding methods with the BWI context-finding methods should yield superior performance.

6.1.2 LIMITED EXPRESSIVENESS OF BOUNDARY DETECTORS

Boundary detectors are designed to capture the flat, nearby context of a field to be extracted, by learning short sequences of surrounding tokens. They are good at capturing regular, sequential information around the field to be extracted, resulting in high precision. However, current boundary detectors are not effective in partially structured and natural texts, where regularities in context are less consistent and reliable. In these domains, many detectors only cover one or a few examples, and collectively the detectors have low recall.

A second limitation of existing boundary detectors is that they cannot represent the grammatical structure of sentences, or most structural information present in HTML or XML documents. Boundary detectors can only capture information about the tokens that appear directly before or after a target field in the linear sequence of tokens in a document. BWI cannot represent or learn any information about the parent nodes, siblings, or child position in the grammar, or XML or HTML tree, to which target fields implicitly belong. In addition, the boundary detector representation does not allow BWI to take advantage of part-of-speech information, and other similar features that have been shown to contain important sources of regularity.

While context information is often the most relevant for recognizing a field, global information about the relative position of the field within the entire document can also be important. For example, despite the impressive performance of BWI on the abstract detection task, the most obvious clue to finding an abstract in a MEDLINE citation is, "look for a big block of text in the middle of the citation." There is no way to capture this global location knowledge using the existing BWI representations. Additional knowledge that BWI is unable to learn or use includes "the field is in the second sentence of the second paragraph if it is anywhere," "the field is never more than one sentence long," "the field has no line breaks inside it," and so on.

6.2 Scoring of Candidate Matches

Each match that BWI returns when applied to a test document has an associated confidence score, which is the product of the confidences of the fore and aft boundary detectors, and the learned probability of the field length of the match. These scores are useful for identifying the most important detectors, and also for choosing a threshold to obtain a precision/recall tradeoff.

While the score of a candidate match is correlated with the chance that the proposed match is correct, these scores are not as useful as actual probabilities would be. First, the confidence scores are unbounded positive real numbers, so it is difficult to compare the scores of matches in different document collections, where the typical range of scores is different. For the same reason, it is hard to set absolute confidence thresholds for accepting proposed matches.

Second, it is hard to compare the confidence of full matches and partial matches, because the components of the confidence score have dramatically different ranges. This obscures the "decision boundary" of the learned system, which would otherwise be an important entity for analysis to improve performance. Finally, without true probabilities, it is difficult to use match scores in a larger framework, such as Bayesian evidence combination or decision-theoretic choice of action. The basic problem is that while confidence scores are useful for stating the relative merit of two matches in the same document collection, they are not useful for comparing matches across collections, nor are they useful for providing an absolute measure of merit.

Another problem with confidence scores is that no distinction is made between a fragment of text that has appeared numerous times in the training set as a negative example, a fragment unlike any seen before, and a fragment where one boundary detector has confidently fired but the other has not. These all have confidence score zero, so there is no way to tell a "confident rejection" from a "near miss."

BWI also implicitly assumes that fields of just one type are extracted from a given document collection. Most documents contain multiple pieces of related information to be extracted, and the position of one field is often indicative of the position of another. However, BWI can only extract fields one at a time, and ignores the relative position of different fields. For example, in the speaker announcements domain, even though four fields are extracted from each document, the field extractors are all trained and tested independently, and the presence of one field is not used to predict the location of another. It is difficult to extend methods like BWI to extract relationships in text, a task that is often as important as extracting individual fields (Muslea et al., 1999).

6.3 Efficiency

In addition to precision and recall, the speed and efficiency of training and testing is also a critical issue in making IE systems practical for solving real-world problems.

6.3.1 BWI IS SLOW TO TRAIN AND TEST, AND NOT INCREMENTAL

Even on a modern workstation, training BWI on a single field with a few hundred documents can take several hours, and testing can take several minutes. The slowness of training and testing makes using larger document collections, or larger lookahead for detectors, prohibitive, even though both may be necessary to achieve high performance on complex tasks.

There are a number of factors that make BWI slower than may be necessary. The innermost loop of training BWI is finding extensions to boundary detectors. This is done in a brute-force manner, with a specified lookahead parameter, L, and repeated until no better rule can be found. Finding a boundary detector extension is exponential in L, because every combination of tokens and wildcards is enumerated and scored. So, even modest lookahead values are prohibitively expensive. A value of $L = 3$ is normally used to achieve a balance between efficiency and performance. Freitag and Kushmerick note that $L = 3$ is usually sufficient, but for some tasks a value up to $L = 8$ is required to achieve reasonable results.

Although testing is considerably faster than training, it too is inefficient. One technique for improving testing speed is to compress the set of boundary detectors learned during training, before applying them to a set of test documents. For example, one could eliminate redundancy by combining duplicate detectors, or eliminating detectors that are logically subsumed by a set of other detectors. This is a practice used by Cohen and Singer (1999) in SLIPPER, and by Ciravegna (2001)

in $(LP)^2$ for partially structured tasks. It is doubtful how useful redundancy elimination would be for less regular document collections, which are the ones for which many detectors must be learned. There are also more sophisticated methods to compress a set of rules, many of which are somewhat lossy (Margineantu and Dietterich, 1997; Yarowsky, 1994). For these methods, the tradeoff between speed and accuracy would need to be measured.

After BWI has been trained on a given set of labeled documents, if more documents are labeled and added, there is currently no way to avoid training on the entire set from scratch again. In other words, there is no way to learn extra information from a few additional documents, even though the majority of what will be learned in the second run will be identical to what was learned before. This is more a problem with boosting than with BWI specifically, because the reweighting that occurs at each round depends on the current performance on all training examples, so adding even a few extra documents can mean that training takes a very different direction.

6.3.2 BWI CANNOT DETERMINE WHEN TO STOP BOOSTING

While the ability of BWI to boost for a fixed number of rounds instead of terminating as soon as all positive examples are covered is clearly one of its advantages, it is hard to decide how many rounds to use. Depending on the difficulty of the task, a given number of rounds may be insufficient to learn all the cases, or it may lead to overfitting, or to redundancy. Ideally, BWI would continue boosting for as long as useful, and then stop when continuing to boost would do more harm than good.

Cohen and Singer (1999) address the problem of when to stop boosting directly in the design of SLIPPER by using internal five-fold validation on a held-out part of the training set. For each fold, they first boost up to a specified maximum number of rounds, testing on the held-out data after each round, and then they select the number of rounds that leads to the lowest average error on the held-out set, finally training for that number of rounds on the full training set. This process is reasonable, and sensitive to the difficulty of the task, but it has several drawbacks: the process of training is made six times slower and the free parameter of the number of boosting rounds is replaced by another free parameter, the maximum number of rounds to try during cross-validation.

It may be possible to use the BWI detector confidences to tell when continuing to boost is futile or harmful. Since detector scores tend to decrease as the number of boosting rounds increases, it may be possible to set an absolute or relative threshold below which to stop boosting. A relative threshold would measure when the curve has flattened out, and new detectors are only picking up exceptional cases one at a time. For example, Ciravegna (2001) prunes rules that cover less than a specified number of examples, because they are unreliable, and too likely to propose spurious matches.

7. Extending Rule-Based IE Methods

In this section we present several suggestions for overcoming the deficiencies of BWI analyzed in the previous section. These suggestions are applicable to BWI and to alternative learning algorithms for IE. We concentrate on identifying new sources of information to consider, constructing new representations for handling them, and producing more meaningful output. In some cases, we present preliminary results that corroborate our hypotheses. In other cases, we cite existing work by other researchers that we believe represent steps in the right direction.

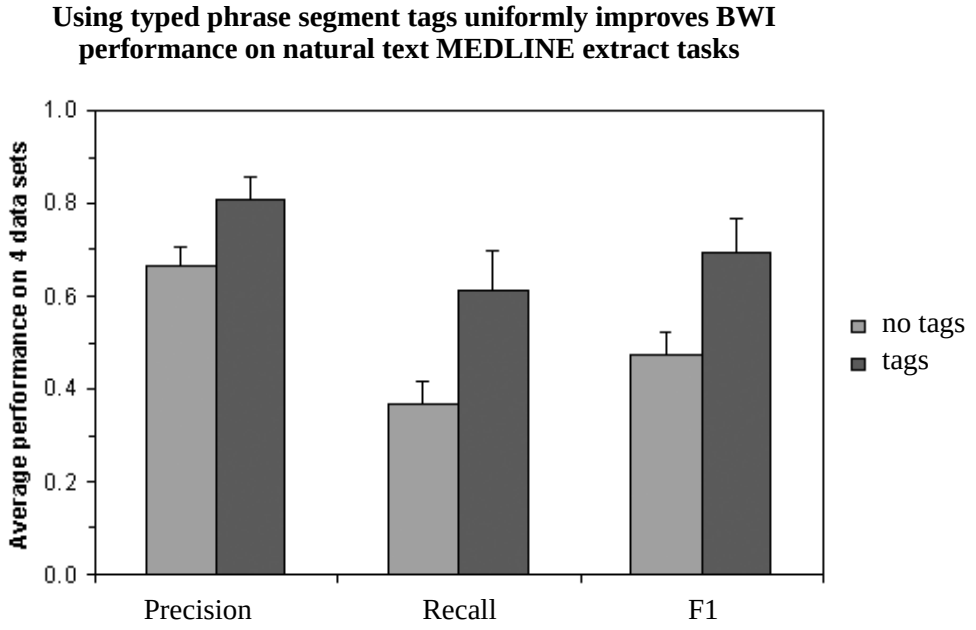


Figure 4: Performance of BWI averaged across the four natural text extraction tasks, with and without the use of typed phrase segments. Means are shown with standard error bars.

7.1 Exploiting the Grammatical Structure of Sentences

Section 6.1.2 discusses the importance of using grammatical structure in natural language or hierarchical structure in HTML and XML documents. Ray and Craven (2001) have taken an important first step in this direction by preprocessing natural text with a shallow parser, and then flattening the parser output by dividing sentences into typed phrase segments. The text is then marked up with this grammatical information, which is used as part of the information extraction process. Ray and Craven use hidden Markov models, but their technique is generally applicable. For example, using XML tags to represent these phrase segments, they construct sentences from MEDLINE articles such as:

<NP_SEG>**Uba2p**</NP_SEG> <VP_SEG>**is located largely**</VP_SEG>
<PP_SEG>**in**</PP_SEG> <NP_SEG>**the nucleus**</NP_SEG>**.**

While the parses produced are not perfect, and the flattening often causes further distortion, this procedure is fairly consistent in labeling noun, verb, and prepositional phrases. An information extraction system can then learn rules that include these tags, allowing it, for example, to represent the constraint that proteins tend to be found in noun phrases, and not in verb phrases. Ray and Craven report that their results "suggest that there is value in representing grammatical structure in the HMM architectures, but the Phrase Model [with typed phrase segments] is not definitively more accurate."

In addition to the results presented in Section 3, which use the document collections of Ray and Craven without grammatical information, we also obtained results using phrase segment information. We used BWI to extract the four individual fields in the two relations that Ray and Craven

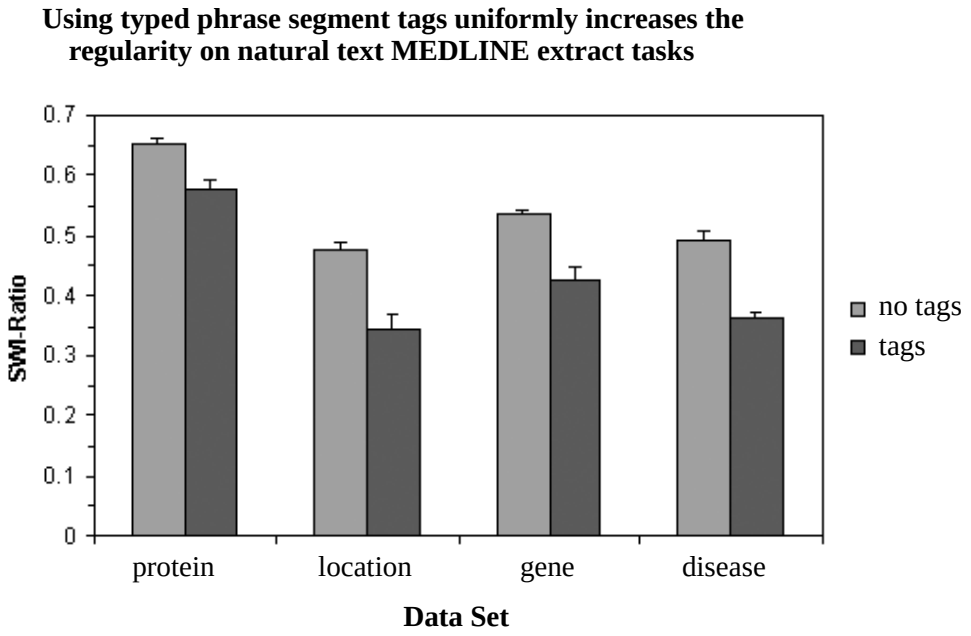


Figure 5: Average SWI ratio for the four natural extraction tasks, with and without the use of typed phrase segments. Means are shown with standard error bars.

study (proteins and their localizations, genes and their associated diseases). We ran identical tests with the grammatical tags inserted as single tokens on either side of a segment (a different tag is specified for the end and start), and without these tags. Including the tags uniformly and considerably improves both precision and recall, for all four extraction tasks (Figure 4). In fact, all four tasks see double-digit percentage increases in precision, recall, and F1, with average increases of 21%, 65%, and 46% respectively.

The fact that precision improves when using the phrase segment tags means that BWI is able to use this information to reject possible fields that it would otherwise return. The fact that recall also improves suggests that having segment tags helps BWI to find fields that it would otherwise miss. The combination of these results is somewhat surprising. For example, while not being a noun phrase may be highly correlated with not being a protein, the inverse is not necessarily the case, since there are many non-protein noun phrases in MEDLINE articles.

We hypothesize that including typed phrase segment information actually regularizes the extraction task, enabling rules to gain greater positive coverage without increasing negative coverage. Using the SWI ratio described in Section 5, we can test this hypothesis quantitatively. Figure 5 shows the SWI ratios for all four extraction tasks with and without phrase segment tags. As expected, there are double-digit percentage decreases in the SWI ratio for all four tasks, with an average reduction of 21%. Recall that a lower SWI ratio indicates a more regular domain, because it means that the same number of positive examples can be perfectly covered with fewer rules. We conclude that including grammatical information, even with existing rule-based IE methods, can be a considerable advantage for both recall and precision, and is worth investigating in more detail.

Our experiments show that grammatical information might be exploited even more. Given how the BWI tokenizer is implemented, all grammatical tags are encoded as capitalized neologisms such as LLLNPSEGMENT. Surprisingly, the rules learned by BWI use the "all upper-case" wildcard more often than they mention a specific grammatical tag. In other words, BWI learns rules of the form "start matching at the boundary of a phrasal chunk" more often than rules that specify a particular grammatical construct such as a noun phrase. An explanation for this behavior is that fragments of text to be extracted are usually well-formed syntactically, so "chunking" the text helps BWI focus on fragments of text that do not cross syntactic boundaries. This focus helps eliminate incorrect fragments and helps identify correct fragments, thus improving both precision and recall. This is similar to the benefit that information retrieval techniques achieve from segmenting documents into coherent regions.

In order to use grammatical information more, the existing formulation of boundary detectors may need to be changed. While inserting XML tags to indicate typed phrase segments is useful, this approach is unprincipled, as it loses the distinction between meta-level information and the text itself. The knowledge representation used by boundary detectors should be extended to express regularities in implicit higher-level information, as well as in explicit token information in a document. When using typed phrase segments, we can increase performance without changing the simple, flat representation currently used. However, this approach cannot make recursive structure explicit, so it can only be taken so far.

The success of including even limited grammatical information immediately raises the question of what additional grammatical information can be used, and how beneficial it might be. It is plausible that regularities in field context such as argument position in a verb phrase, subject/object distinction, and so on can be valuable. For example, Charniak (2001) has shown that probabilistically parsing sentences is greatly aided by conditioning on information about the linguistic head of the current phrase, even if the head is several tokens away in the flat representation. Charniak's finding is evidence that linguistic head information is an important source of regularity, which is exactly what rule-based IE methods are designed to exploit. There have also been several other attempts to exploit limited grammatical analysis for information extraction, including AutoSlog (Riloff, 1996), FASTUS (Appelt et al., 1995), and CRYSTAL (Soderland et al., 1995), all of which demonstrate the potential value of this type of information. As pointed out by Grishman (1997), improved performance is often achieved only when such information is used in a conservative way, for example looking only for specific grammatical patterns, or using grammatical information only in contexts that are unambiguous.

7.2 Binning Histograms of Field Length

Section 3 presents results using the field lengths seen in the training set when computing confidence values for possible extractions. When the variance of field lengths is small, this method works well. However, as mentioned in Section 6.1.1, some tasks involve a wide range of field lengths, for example the task of identifying abstracts in documents.

On the AbstractText task BWI gives an F1 of only 71%. The main reason for this poor performance is a recall of 62%. Because BWI gives a score of zero to any potential field with length never seen in the training set, many correct fields are excluded in test documents even though boundary detectors indicate that these fields should be extracted: relying on raw length counts from the train-

ing data causes 45% of the predictions made by boundary detectors on the test set to be dropped from consideration.

A simple response is to ignore the field lengths and extract fields based only upon the information provided by the boundary detectors. For the AbstractText task, this method increases F1 to 97.8%. However, this method requires a human to analyze the task in advance to decide whether lengths should be used. A more general solution would instead generalize automatically from the lengths seen in the training set.

The difficulty is that the length histogram does not generalize from the lengths seen in the training set. To generalize, the histogram bins must be expanded to allow for lengths never seen in the training set. We take a simple approach by using a fixed number of equal width bins.

We would like to make the bins start at the true minimum length and end at the true maximum. Unfortunately, we do not know the true minimum and maximum because we only have a sample of the data. The maximum likelihood estimates (MLEs) for these parameters are the minimum and maximum of the sample (i.e. the smallest and largest lengths seen in the training data). Unfortunately, the MLEs are biased: they tend to underestimate the maximum and overestimate the minimum, particularly for small sample sizes. Fortunately, we can compute unbiased estimates for the true max and min:

$$\text{max\_estimate}(x) = max(x)\frac{n+1}{n} \qquad \text{min\_estimate}(x) = min(x)\frac{n-1}{n},$$

where x is the sample of size n.

The unbiased estimates above are valid for a uniform distribution. For bell-shaped distributions, these estimates need to be expanded further beyond the sample minimum and maximum. For this reason, in our experiments we use ten bins of uniform width with 1/18th of the range beyond the sample max and min. In other words, we extend half a bin width beyond the sample minimum and maximum.

These bin sizes give F1 of 97.0% on the AbstractText task. This is slightly worse than the F1 achieved by ignoring length, but the binning method performs well on both highly irregular tasks such as the AbstractText task and more regular tasks like those examined in Section 3 and does not require a human to examine the method a priori.

7.3 Handling XML or HTML Structure and Information

Just as the grammatical structure of a sentence presents opportunities for exploiting structural regularities, the hierarchical structure and attribute information available in an XML or HTML document also contain important clues for locating target fields. However, this information is essentially lost when an XML document is parsed as a linear sequence of tokens instead of using a document object model (DOM). For example, tags like `<tag>` or `</tag>` that should be treated as single tokens are instead broken into pieces. More problematic, however, is the fact that many tags contain namespace references, and attribute-value pairs inside the starting tag, such as `<name:tag att1=''val1'' att2=''val2''>`. When using flat tokenization, lookahead becomes a serious problem, and there is no way to intelligently generalize over such tags, e.g. to match a tag with the same name, to require specific attributes and/or values, etc.

Muslea et al. (1999) made some important contributions to solving this problem with STALKER, which constructs an "embedded catalog" for web pages (a conceptual hierarchy of content in a document) that allows it to take advantage of global landmarks for finding and extracting text fields.

They accomplish this by use of the "Skip-To" operator, which matches a boundary by ignoring all text until a given sequence of tokens and/or wildcards. While the token sequences learned with Skip-To operators are similar to the boundary detectors used in BWI, they can be combined sequentially to form an extraction rule that relies on several distinct text fragments, which can be separated by arbitrary amounts of intermediate text. This is an excellent first approach to parsing hierarchically arranged documents, but using Skip-To only captures some of the information available in a complete DOM representation.

A second approach to exploiting the structural information embedded in web pages and XML documents is due to Yih (1997), who uses hierarchical document templates for IE in web pages, and finds fields by learning their position within the template's node tree. The results presented are impressive, but currently the document templates must be constructed manually from web pages of interest, because the hierarchies in the templates are more subjective than just the HTML parse. Constructing templates may be less of a problem with XML documents, but only if their tag-based structure corresponds in some relevant way to the content structure that is necessary to exploit for information extraction. Similar results are presented by Liu et al. (2000) with their "XWRAP", a rule-based learner for web page information extraction that uses heuristics like font size, block positioning of HTML elements, and so on to construct a document hierarchy and extract specific nodes.

7.4 Extending the Expressiveness of Boundary Detectors

One simple solution to the problem of exploiting more grammatical and structural information is the development of a more sophisticated set of wildcards. Currently, wildcards only represent word-level syntactic classes, such as "all upper-case", "begins with a lower-case letter", "contains only digits", and so on. While these are useful generalizations over matching individual tokens, they are also extremely broad. A potential middle ground consists of wildcards that match words of a given linguistic part of speech (e.g. "noun"), a given semantic class (e.g. "location/place"), or a given lexical feature (e.g. specific prefix/suffix, word frequency threshold, etc.).

Encouraging results in this direction are already available. For example, Ciravegna's $(LP)^2$ system uses word morphology and part-of-speech information to generalize the rules it initially learns (Ciravegna, 2001), a process similar to using lexical and part-of-speech wildcards. Ciravegna's results are comparable to those with other state of the art methods, including BWI. While $(LP)^2$ also does rule correction and rule pruning, Ciravegna says that "the use of NLP for generalization" is the most responsible for the performance of the system. Another clever use of generalizations can be found in RAPIER (Califf and Mooney, 1999), which uses WordNet (Miller, 1995) to find hypernyms (a semantic class of words to which the target word belongs), then uses these in its learned extraction rules. Yarowsky (1994) reports that including semantic word classes like "weekday" and "month" to cover sets of tokens improves performance for lexical disambiguation, suggesting that there are indeed useful semantic regularities to be exploited for IE.

7.5 Converting BWI Match Scores into Probabilities

While improving the accuracy of IE methods is an important goal, progress will be limited in its usefulness by the output representation used. As mentioned in Section 6.2, BWI (like most other IE methods) does not attach probabilities to the matches it returns. Probability is the *lingua franca* for combining information processing systems, because probabilities have both absolute and relative

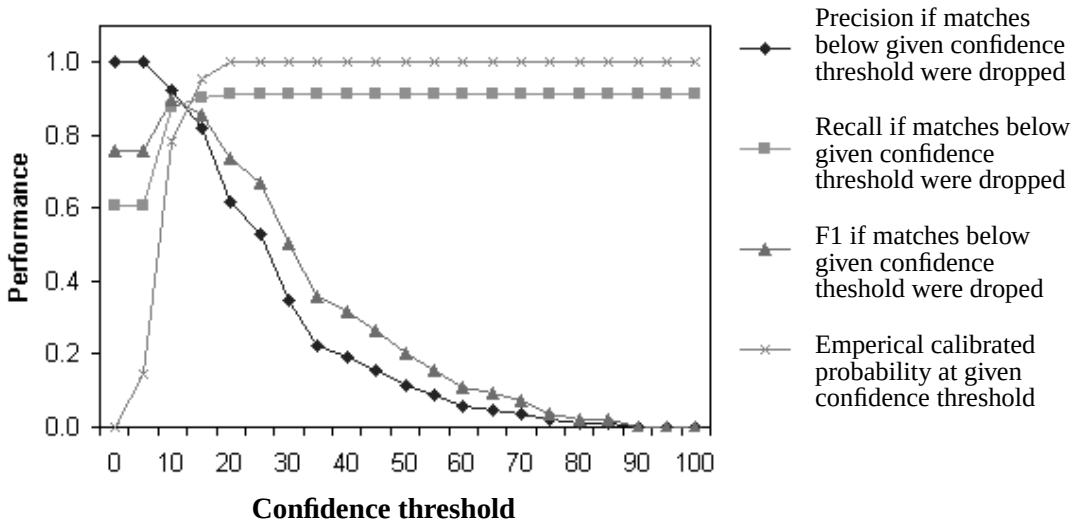


Figure 6: Precision, recall, and F1 for BWI on the *AbstractText* task, versus confidence threshold below which to ignore matches. The empirical probability of match correctness within each confidence interval is also shown.

meaning, and because there are powerful mathematical frameworks for dealing with them, such as Bayesian evidence combination and decision theory. For example, question-answering systems commonly use probabilities (Ponte and Croft, 1998).

Thus, it is worth asking the question, can BWI confidence scores be transformed into probabilities? This question really comes in two pieces. First, are BWI confidence scores meaningful and consistent? That is, does BWI produce incorrect matches with high confidence, or does its accuracy increase with its confidence? Second, can BWI learn the correlation between confidence scores and the chance of correctness of a match, and thus automatically calibrate its scores into probabilities?

The first question, whether BWI scores are consistent, can be answered empirically, by training and testing BWI on different document collections. Ideally, most incorrect predictions have low confidence compared to correct predictions. Such a distribution is desirable for two reasons. First, it resembles a true probability distribution, where a higher score is correlated with a higher chance of being correct. Second, it allows users to set a confidence threshold above which to accept predictions, and below which to examine predictions further.

Our results suggest that BWI scores are fairly consistent and amenable to threshold setting for tasks with high regularity, but on hard tasks it is difficult to separate correct and incorrect predictions based only on match confidences. For the *AbstractText* task, confidence scores range from 0 to 100, but all incorrect predictions have confidence scores below 20 (Figure 6). This means that if BWI ignores predictions below a confidence threshold of 20, it obtains perfect precision. However, this threshold results in only 59% recall, compared with 95% recall with threshold 0. A confidence threshold of 10 maximizes F1, because most incorrect answers can be pruned without eliminating correct guesses, as can be seen in Figure 6. The confidence scores are behaving roughly as probabilities should, because as confidence increases, so does the fraction of correct guesses.

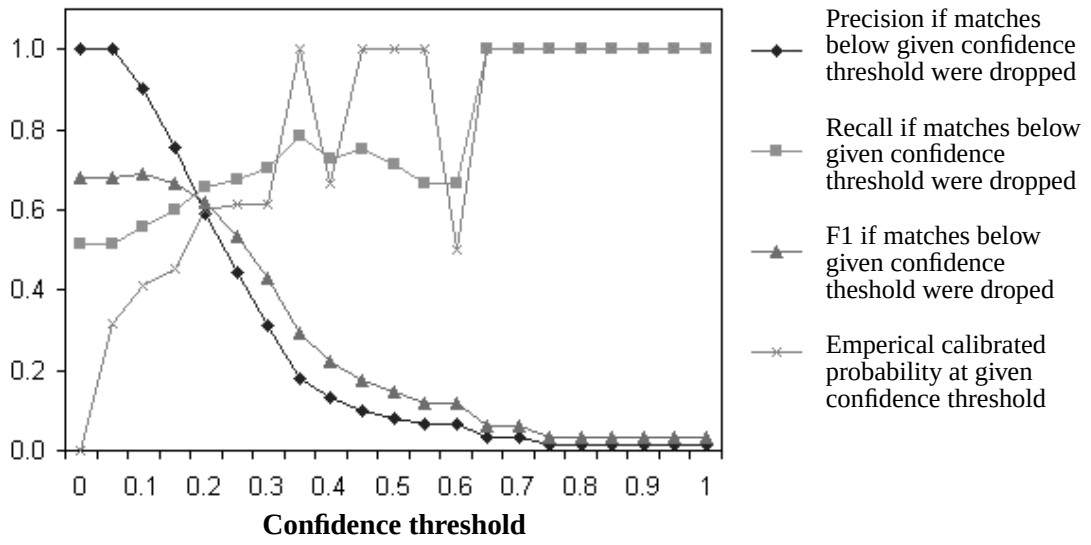


Figure 7: Precision, recall, and F1 for BWI on the *YPD-protein* task, versus confidence threshold below which to ignore matches. The empirical probability of match correctness within each confidence interval is also shown.

Note that the precision for this task is lower than reported in Table 1 because here we consider all BWI matches, whereas normally BWI eliminates overlapping matches, keeping only the match with higher confidence.

Unfortunately, the correlation between confidence and probability of correctness is weaker for more difficult tasks. On the protein task, the most difficult as measured both by performance and by SWI ratio, Figure 7 shows that there is no clean way to separate the correct and incorrect guesses. The highest confidence negative match has score 0.65, but above this threshold, BWI only achieves 0.8% recall, because almost all correct matches have confidence score below 0.35. The highest F1 in this case comes from a confidence threshold of 0, with precision 52% and recall 24%. (These numbers are taken from an individual train/test fold and thus differ slightly from the averages presented in Table 1.) There is no way to improve performance with a non-zero confidence threshold, because there is no smooth transition to a higher density of correct predictions as confidence scores increase.

There is a direct way of calibrating scores into probabilities–divide BWI predictions on a validation set of documents into bins by confidence score, and estimate the probability for each bin as the fraction of correct predictions in the bin. Essentially if for a given range of confidence scores, say 75% of predictions are correct, then a prediction on a test document with similar confidence is estimated to have a 75% chance of being correct. Even for difficult tasks for which confidence scores are not consistent, i.e. as scores increase there is no smooth increase in probability, the calibrated probabilities are still meaningful, because they make explicit the uncertainty in predictions.

		BWI			**Fixed-BWI**			**Root-SWI**			**Greedy-SWI**		
Data set	SWI-ratio	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
LATimes-cc	0.013	0.996	1.000	0.998	1.000	0.985	0.993	1.000	0.975	0.986	1.000	0.948	0.973
Zagat-addr	0.011	1.000	0.937	0.967	1.000	0.549	0.703	1.000	0.549	0.703	1.000	0.575	0.724
QS-date	0.056	1.000	1.000	1.000	1.000	0.744	0.847	1.000	0.744	0.847	1.000	0.783	0.875
AbstractText Raw histogram	NA	0.860	0.616	0.716	NA	NA	NA	NA	NA	NA	NA	NA	NA
AbstractText No binning	0.149	0.990	0.967	0.978	0.966	0.495	0.654	0.846	0.585	0.685	0.847	0.317	0.447
AbstractText 10-bins	NA	0.976	0.965	0.971	NA	NA	NA	NA	NA	NA	NA	NA	NA
SA-speaker	0.246	0.791	0.592	0.677	0.887	0.446	0.586	0.777	0.457	0.565	0.904	0.342	0.494
SA-location	0.157	0.854	0.696	0.767	0.927	0.733	0.818	0.800	0.766	0.780	0.924	0.647	0.759
SA-stime	0.098	0.996	0.996	0.996	0.991	0.949	0.969	0.975	0.952	0.964	0.979	0.842	0.902
SA-etime	0.077	0.944	0.949	0.939	0.993	0.818	0.892	0.912	0.793	0.843	0.987	0.813	0.885
Jobs-id	0.068	1.000	1.000	1.000	1.000	0.956	0.978	1.000	0.956	0.978	0.996	0.829	0.902
Jobs-company	0.246	0.884	0.701	0.782	0.955	0.733	0.824	0.794	0.838	0.784	0.904	0.751	0.802
Jobs-title	0.381	0.596	0.432	0.501	0.661	0.479	0.547	0.480	0.658	0.546	0.660	0.477	0.549
YPD-protein	0.651	0.567	0.239	0.335	0.590	0.219	0.319	0.516	0.154	0.228	0.594	0.134	0.218
YPD-location	0.478	0.738	0.446	0.555	0.775	0.418	0.542	0.633	0.246	0.347	0.774	0.240	0.365
OMIM-gene	0.534	0.655	0.368	0.470	0.826	0.480	0.606	0.469	0.249	0.324	0.646	0.199	0.304
OMIM-disease	0.493	0.707	0.428	0.532	0.741	0.411	0.528	0.487	0.251	0.327	0.785	0.241	0.369

Table 1: SWI ratio and performance of the four IE methods examined on the 15 tasks.

8. Concluding Remarks

Current information extraction methods are able to discover and exploit regularities in text that come from local word ordering, punctuation usage, and distinctive word-level features like capitalization. This information is often sufficient for partially and highly structured documents because their regularity is at the surface level-in the use and ordering of words themselves. However, current IE methods perform considerably worse when dealing with natural text documents, because the regularities in these are less apparent. Nevertheless, there *are* still important regularities in natural text documents, at the grammatical and semantic levels. We have shown that making explicit even limited grammatical information results in considerably higher precision and recall for information extraction from natural text documents.

All IE methods perform differently on document collections of different regularity, but there has been no quantitative analysis in previous papers of the relationship between document regularity and IE accuracy. We propose the SWI ratio as a quantitative measure of document regularity, and we use it to investigate the relationship between regularity and accuracy in greater detail than previously possible. Since the SWI ratio objectively measures the relative regularity of a document collection, it is suitable for comparison of IE tasks on document collections of varying size and content.

While all the IE algorithms we study perform worse on less regular document collections, they still exhibit consistent differences. Many current rule-based IE methods, including the SWI method proposed in this paper, employ some form of set covering to combine multiple extraction rules. These methods are relatively simple to implement, but they are all fundamentally limited in that they remove positive examples as soon as they are covered once. Boosting overcomes this limitation by reducing the weight of covered examples instead of removing them. We have shown that BWI exploits this property to learn additional useful rules even after all examples have been cov-

	cc	addr	date	abst.	speak.	loc.	stime	etime	id	comp.	title	prot.	loc.	gene	disease
BWI	50	50	50	500	500	500	500	500	500	500	500	500	500	500	500
SWI	5.1	1.3	1.0	51.8	139.4	75.6	72.1	25.0	15.2	46.9	132.1	333.1	235.0	357.0	280.1
Root-SWI	3.6	1.3	1.0	48.4	110.7	66.8	62.0	17.9	1.0	46.3	119.6	322.5	229.7	344.6	278.8

Table 2: Number of boosting iterations used BWI, SWI, and Root-SWI on the 15 data sets.

ered, and consistently outperforms set covering methods. Reweighting also helps by focusing BWI on learning specific rules for the exceptional cases missed by general rules, resulting in higher precision. The combination of learning accurate rules, and continuing training to broaden coverage, is the source of the success of BWI.

While this paper focuses on variants of BWI, many of our observations hold for information extraction as a whole. For example, while hidden Markov models employ a different representation, they tend, like BWI, to learn regularities that are local and flat, concerning word usage, punctuation usage, and other surface patterns. Thus the sections above about sources of information that are currently ignored are relevant to HMMs and rule-based IE methods. There are many opportunities for improving the usefulness of current information extraction methods. We hope this paper is a contribution in this direction.

Acknowledgments

The authors would like to thank Dayne Freitag for his assistance and for making the BWI code available, Mark Craven for giving us the natural text MEDLINE documents with annotated phrase segments, and MedExpert International, Inc. for its financial support of this research. The anonymous reviewers provided very detailed and valuable comments that have helped us improve the paper considerably. This work was conducted with support from the California Institute for Telecommunications and Information Technology, Cal-(IT)^2.

References

D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, and M. Tyson. SRI international FASTUS system: MUC-6 test results and analysis. In *Proceedings of the 6th Message Understanding Conference*, pages 237–248, 1995.

S. Baluja, V. Mittal, and R. Sukthankar. Applying machine learning for high performance named-entity extraction. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics*, pages 365–378, 1999.

D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. NYMBLE: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

M. Califf. *Relational Learning Techniques for Natural Language Information Extraction*. PhD thesis, Department of Computer Science, Univeristy of Texas at Austin, 1998.

M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, 1999.

C. Cardie and R. Mooney. Tutorial on symbolic machine learning for natural language processing. In *ACL-98*, 1998. URL `http://www.cs.cornell.edu/Info/People/cardie/tutorial/tutorial.html`.

E. Charniak. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

F. Ciravegna. $(LP)^2$, an adaptive algorithm for information extraction from web-related texts. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning 3*, pages 261–283, 1989.

W.W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.

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

T. Eliassi-Rad and J. Shavlik. A theory-refinement approach to information extraction. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 577–583, 2000.

R. Grishman. Information extraction: Techniques and challenges. *Springer-Verlag, Lecture Notes in Artificial Intelligence, Rome*, 1997.

N. Kushmerick. Wrapper induction: Efficiency and expressiveness. In *Artificial Intelligence*, volume 118, pages 15–68, 2000.

L. Liu, C. Pu, and W. Ilan. XWRAP: An XML-enabled wrapper construction system for web information sources. In *Proceedings of the International Conference on Data Engineering*, 2000.

D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 211–218, 1997.

S. Michalski. Pattern recognition as rule-guided inductive inference. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 2, pages 349–361, 1980.

G. Miller. Wordnet: A lexical database for english. In *Communications of the ACM*, volume 38, pages 39–41, 1995.

I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 190–197, 1999.

National Library of Medicine, 2001. Medline database. URL http://www.ncbi.nlm.gov/Pubmed/.

J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Research and Development in Information Retrieval*, pages 275–281, 1998.

J. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

S. Ray and M. Craven. Representing sentence structure in hidden markov models for information extraction. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

L. Rendell and H. Cho. Empirical learning as a function of concept character. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1990.

E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1996.

RISE. A repository of online information sources used in information extraction tasks, 1998. URL http://www.isi.edu/info-agents/RISE/.

R. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1999.

H. Shinnou. Detection of errors in training data by using a decision list and adaboost. In *IJCAI-2001 Workshop on Text Learning: Beyond Supervision*, pages 61–65, 2001.

S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. CRYSTAL: Inducing a conceptual dictionary. In *IJCAI'95 - Proc. of the 14th Intl. Joint Conf. on Artificial Intelligence*, volume 2, pages 1314–1319, 1995.

V. Vapnik, E. Levin, and Y. LeCun. Measuring the VC dimension of a learning machine. *Neural Computation*, 6:851–876, 1994.

D. Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Proceedings of the ACL '94*, pages 77–95, 1994.

W-T. Yih. Template-based information extraction from tree-structured HTML documents. Master's thesis, National Taiwan University, 1997.

Journal of Machine Learning Research 5 (2004) 529-547 Submitted 1/03; Revised 8/03; Published 5/04

Computable Shell Decomposition Bounds

John Langford JL@TTI-C.ORG
David McAllester MCALLESTER@TTI-C.ORG
Toyota Technology Institute at Chicago
1427 East 60th Street
Chicago, IL 60637, USA

Editor: Manfred Warmuth

Abstract

Haussler, Kearns, Seung and Tishby introduced the notion of a shell decomposition of the union bound as a means of understanding certain empirical phenomena in learning curves such as phase transitions. Here we use a variant of their ideas to derive an upper bound on the generalization error of a hypothesis computable from its training error and the histogram of training errors for the hypotheses in the class. In most cases this new bound is significantly tighter than traditional bounds computed from the training error and the cardinality of the class. Our results can also be viewed as providing a rigorous foundation for a model selection algorithm proposed by Scheffer and Joachims.

Keywords: Sample complexity, classification, true error bounds, shell bounds

1. Introduction

For an arbitrary finite hypothesis class we consider the hypothesis of minimal training error. We give a new upper bound on the generalization error of this hypothesis computable from the training error of the hypothesis and the histogram of the training errors of the other hypotheses in the class. This new bound is typically much tighter than more traditional upper bounds computed from the training error and cardinality of the class.

As a simple example, suppose that we observe that all but one empirical error in a hypothesis space is $1/2$ and one empirical error is 0. Furthermore, suppose that the sample size is large enough (relative to the size of the hypothesis class) that with high confidence we have that, for all hypotheses in the class, the true (generalization) error of a hypothesis is within $1/5$ of its training error. This implies, that with high confidence, hypotheses with training error near $1/2$ have true error in $[3/10, 7/10]$. Intuitively, we would expect the true error of the hypothesis with minimum empirical error to be very near to 0 rather than simply less than $1/5$ because none of the hypotheses which produced an empirical error of $1/2$ could have a true error close enough to 0 that there exists a significant probability of producing 0 empirical error. The bound presented here validates this intuition. We show that you can ignore hypotheses with training error near $1/2$ in calculating an "effective size" of the class for hypotheses with training error near 0. This new effective class size allows us to calculate a tighter bound on the difference between training error and true error for hypotheses with training error near 0. The new bound is proved using a distribution-dependent application of the union bound similar in spirit to the shell decomposition introduced by Haussler et al. (1996).

We actually give two upper bounds on generalization error — an uncomputable bound and a computable bound. The uncomputable bound is a function of the unknown distribution of true error rates of the hypotheses in the class. The computable bound is, essentially, the uncomputable bound with the unknown distribution of true errors replaced by the known histogram of training errors. Our main contribution is that this replacement is sound, i.e., the computable version remains, with high confidence, an upper bound on generalization error.

When considering asymptotic properties of learning theory bounds it is important to take limits in which the cardinality (or VC dimension) of the hypothesis class is allowed to grow with the size of the sample. In practice, more data typically justifies a larger hypothesis class. For example, the size of a decision tree is generally proportional the amount of training data available. Here we analyze the asymptotic properties of our bounds by considering an infinite sequence of hypothesis classes $\mathcal{H}_m$, one for each sample size m, such that $\frac{\ln|\mathcal{H}_m|}{m}$ approaches a limit larger than zero. This kind of asymptotic analysis provides a clear account of the improvement achieved by bounds that are functions of error rate distributions rather than simply the size (or VC dimension) of the class.

We give a lower bound on generalization error showing that the uncomputable upper bound is asymptotically as tight as possible — any upper bound on generalization error given as a function of the unknown distribution of true error rates must asymptotically be greater than or equal to our uncomputable upper bound. Our lower bound on generalization error also shows that there is essentially no loss in working with an upper bound computed from the true error distribution rather than expectations computed from this distribution as used by Scheffer and Joachims (1999).

Asymptotically, the computable bound is simply the uncomputable bound with the unknown distribution of true errors replaced with the observed histogram of training errors. Unfortunately, we can show that in limits where $\frac{\ln|\mathcal{H}_m|}{m}$ converges to a value greater than zero, the histogram of training errors need not converge to the distribution of true errors — the histogram of training errors is a "smeared out" version of the distribution of true errors. This smearing loosens the bound even in the large-sample asymptotic limit. We give a precise asymptotic characterization of this smearing effect for the case where distinct hypotheses have independent training errors. In spite of the divergence between these bounds, the computable bound is still significantly tighter than classical bounds not involving error distributions.

The computable bound can be used for model selection. In the case of model selection we can assume an infinite sequence of finite model classes $\mathcal{H}_0, \mathcal{H}_1, \ldots$ where each $\mathcal{H}_j$ is a finite class with $\ln|\mathcal{H}_j|$ growing linearly in j. To perform model selection we find the hypothesis of minimal training error in each class and use the computable bound to bound its generalization error. We can then select, among these, the model with the smallest upper bound on generalization error. Scheffer and Joachims propose (without formal justification) replacing the distribution of true errors with the histogram of training errors. Under this replacement, the model selection algorithm based on our computable upper bound is asymptotically identical to the algorithm proposed by Scheffer and Joachims.

The shell decomposition is a distribution-dependent use of the union bound. Distribution-dependent uses of the union bound have been previously exploited in so-called self-bounding algorithms. Freund (1998) defines, for a given learning algorithm *and data distribution*, a set S of hypotheses such that with high probability over the sample, the algorithm always returns a hypothesis in that set. Although S is defined in terms of the unknown data distribution, Freund gives a way of computing a set S' from the given algorithm and the sample such that, with high confidence, S' contains S and hence the "effective size" of the hypothesis class is bounded by $|S'|$. Langford and

Blum (1999) give a more practical version of this algorithm. Given an algorithm and data distribution they conceptually define a weighting over the possible executions of the algorithm. Although the data distribution is unknown, they give a way of computing a lower bound on the weight of the particular execution of the algorithm generated by the sample at hand. In this paper we consider distribution dependent union bounds defined independent of any particular learning algorithm.

The bounds given in this paper apply to finite concept classes. Of course more sophisticated measures of the complexity of a concept class, such as VC dimension or Rademacher complexity, are possible and can sometimes result in tighter bounds.

However, insight into finite classes remains useful in at least two ways. Finite class analysis is useful as a pedagogical tool, teaching about directions in which to look for the removal of slack from these more sophisticated bounds. Indeed, various localized Rademacher complexity results (Bartlett et al., 2002) and the "peeling" technique (van de Geer, 1999) appear to (roughly) correspond to the orthogonal combination of shell bounds and earlier Rademacher complexity results. One advantage of the shell bounds is the KL-divergence form of the bounds which smoothly interpolates between the linear bounds of the realizable case and the quadratic bounds of the unrealizable case. This realizable-unrealizable interpolation is orthogonal to the shell principle that concepts with large empirical error are unlikely to be confused with concepts with low error rate. The shell bound also supports intuitions that are difficult to achieve in more complex settings. For example, the simple shell bounds clearly exhibit phase transitions in the learning bound, something which does not appear to be well-elucidated for localalized Rademacher bounds. In summary, the simplicity of finite classes (and a shell bound analysis on a finite class) provides a clarity that is difficult to achieve with more complex structure-exploiting bounds.

Finite class analysis is also useful in a more practical sense. In practice a finite VC dimension class usually has a finite parameterization. Given that these real parameters are typically represented as 32 bit floating point numbers, the class becomes finite and the log of the class size is linear in the number of parameters. Since many of the more sophisticated infinite-class techniques are loose by large multiplicative constants, a finite class analysis applied to a VC class discretized to a small number of bits can actually yield *tighter* bounds as shown in Figure 1.

2. Mathematical Preliminaries

For an arbitrary measure on an arbitrary sample space we use the notation[1] $\forall^{\delta} S\ \Phi[S,\delta]$ to mean that with probability at least $1-\delta$ over the choice of the sample S we have that $\Phi[S,\delta]$ holds. In practice S is the training sample of a learning algorithm. Note that $\forall x\ \forall^{\delta} S\ \Phi[x,\ S,\ \delta]$ does not imply $\forall^{\delta} S\ \forall x\ \Phi[x,\ S,\ \delta]$. If X is a finite set, and for all $x \in X$ we have the assertion $\forall \delta > 0\ \forall^{\delta} S\ \Phi[S,x,\delta]$ then by a standard application of the union bound we have the assertion $\forall \delta > 0\ \forall^{\delta} S\ \forall x \in X\ \Phi[S,x,\frac{\delta}{|X|}]$. We call this the quantification rule. If $\forall \delta > 0\ \forall^{\delta} S\ \Phi[S,\delta]$ and $\forall \delta > 0\ \forall^{\delta} S\ \Psi[S,\ \delta]$ then by a standard application of the union bound we have $\forall \delta > 0\ \forall^{\delta} S\ \Phi[S,\frac{\delta}{2}] \wedge \Psi[S,\frac{\delta}{2}]$. We call this the conjunction rule.

The KL-divergence of p from q, denoted $D(q||p)$, is $q\ln(\frac{q}{p}) + (1-q)\ln(\frac{1-q}{1-p})$ with $0\ln(\frac{0}{p}) = 0$ and $q\ln(\frac{q}{0}) = \infty$. Let $\hat{p}$ be the fraction of heads in a sequence S of m tosses of a biased coin where

1. This can be read as "for all but δ sets S, the predicate $\Phi[S,\delta]$ holds" or "with probability $1-\delta$ over the draw of S, the predicate $\Phi[S,\delta]$ holds".

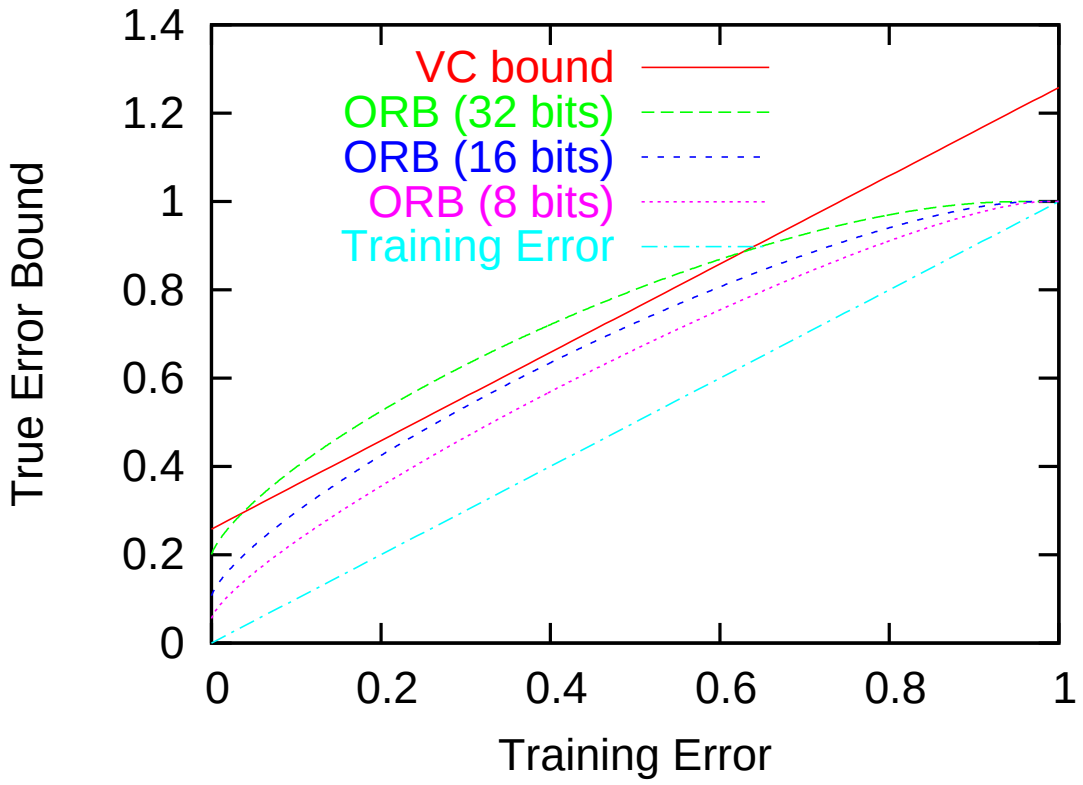


Figure 1: A graph comparing the (infinite hypothesis) VC bound to the finite hypothesis Occam's razor bound. For all curves we use VC dimension $d = 10$, bound failure probability, $\delta = 0.1$, and $m = 1000$ examples. For the VC bound calculation (see Moore, 2004, for details) the formula is true error $\leq$ train error $+ \sqrt{(d \ln \frac{2m}{d} + \ln \frac{4}{\delta})/m}$. For the Occam's Razor Bound (see Langford, 2003, for details) calculation, we use a uniform distribution over the 2^{kd} discrete classifiers which might be representable when we discretize d parameters to $k = 8, 16, 32$ bits per dimension. The basic formula is: $\mathrm{KL}(\text{train error}||\text{true error}) \leq (kd \ln 2 + \ln \frac{1}{\delta})/m$. This graph is approximatly the same for any similar ratio of d/m with smaller values favoring the Occam's Razor Bound.

the probability of heads is p. We have the following inequality given by Chernoff (1952):

$$\forall q \in [p,1]: \ \ Pr(\hat{p} \geq q) \quad \leq \quad e^{-mD(q||p)}. \tag{1}$$

This bound can be rewritten as follows:

$$\forall \delta > 0 \, \forall^{\delta} S \ \ D(\max(\hat{p},p)||p) \leq \frac{\ln(\frac{1}{\delta})}{m}. \tag{2}$$

To derive (2) from (1) note that $Pr(D(\max(\hat{p},p)||p) \geq \frac{\ln(\frac{1}{\delta})}{m})$ equals $Pr(\hat{p} \geq q)$ where $q \geq p$ and $D(q||p) = \frac{\ln(\frac{1}{\delta})}{m}$. By (1) we then have that this probability is no larger than $e^{-mD(q||p)} = \delta$. It is just as easy to derive (1) from (2) so the two statements are equivalent. By duality, i.e., by considering the problem defined by replacing p by $1-p$, we get

$$\forall \delta > 0 \, \forall^{\delta} S \ \ D(\min(\hat{p},p)||p) \leq \frac{\ln(\frac{1}{\delta})}{m}. \tag{3}$$

Conjoining (2) and (3) yields the following corollary of (1):

$$\forall \delta > 0 \, \forall^{\delta} S \ \ D(\hat{p}||p) \leq \frac{\ln(\frac{2}{\delta})}{m}. \tag{4}$$

Using the inequality $D(q||p) \geq 2(q-p)^2$ one can show that (4) implies the better known form of the Chernoff bound

$$\forall \delta > 0 \, \forall^{\delta} S \ \ |p - \hat{p}| \leq \sqrt{\frac{\ln(\frac{2}{\delta})}{2m}}. \tag{5}$$

Using the inequality $D(q||p) \geq \frac{(p-q)^2}{2q}$, which holds for $q \leq p$, we can show that (3) implies the following:[2]

$$\forall \delta > 0 \, \forall^{\delta} S \ \ p \leq \hat{p} + \sqrt{\frac{2\hat{p}\ln(\frac{1}{\delta})}{m}} + \frac{2\ln(\frac{1}{\delta})}{m}. \tag{6}$$

Note that for small values of $\hat{p}$ formula (6) gives a tighter upper bound on p than does (5). The upper bound on p implicit in (4) is somewhat tighter than the minimum of the bounds given by (5) and (6).

We now consider a formal setting for hypothesis learning. We assume a finite set $\mathcal{H}$ of hypotheses and a space $\mathcal{X}$ of instances. We assume that each hypothesis represents a function from $\mathcal{X}$ to $\{0,1\}$ where we write $h(x)$ for the value of the function represented by hypothesis h when applied to instance x. We also assume a distribution D on pairs $\langle x, y \rangle$ with $x \in \mathcal{X}$ and $y \in \{0,1\}$. For any hypothesis h we define the error rate of h, denoted $e(h)$, to be $P_{\langle x, y \rangle \sim D}(h(x) \neq y)$. For a given sample S of m pairs drawn from D we write $\hat{e}(h)$ to denote the fraction of the pairs $\langle x, y \rangle$ in S such that $h(x) \neq y$. Quantifying over $h \in \mathcal{H}$ in (4) yields the following second corollary of (1):

$$\forall^{\delta} S \ \ \forall h \in \mathcal{H} \ \ D(\hat{e}(h)||e(h)) \leq \frac{\ln|\mathcal{H}| + \ln(\frac{2}{\delta})}{m}. \tag{7}$$

2. A derivation of this formula can be found in Mansour and McAllester (2000) or McAllester and Schapire (2000). To see the need for the last term consider the case where $\hat{p} = 0$.

By considering bounds on $D(q||p)$ we can derive the more well known corollary of (7),

$$\forall^{\delta} S \quad \forall h \in \mathcal{H} \quad |e(h) - \hat{e}(h)| \leq \sqrt{\frac{\ln|\mathcal{H}| + \ln(\frac{2}{\delta})}{2m}}. \tag{8}$$

These two formulas both limit the distance between $\hat{e}(h)$ and $e(h)$. In this paper we work with (7) rather than (8) because it yields an (implicit) upper bound on generalization error that is optimal up to asymptotic equality.

3. The Upper Bound

Our goal now is to improve on (7). Our first step is to divide the hypotheses in $\mathcal{H}$ into m disjoint sets based on their true error rates. More specifically, for $p \in [0,1]$ define $\lceil\lceil p \rceil\rceil$ to be $\frac{\max(1,\lceil mp \rceil)}{m}$. Note that $\lceil\lceil p \rceil\rceil$ is of the form $\frac{k}{m}$ where either $p = 0$ and $k = 1$ or $p > 0$ and $p \in (\frac{k-1}{m}, \frac{k}{m}]$. In either case we have $\lceil\lceil p \rceil\rceil \in \{\frac{1}{m}, \ldots, \frac{m}{m}\}$ and if $\lceil\lceil p \rceil\rceil = \frac{k}{m}$ then $p \in [\frac{k-1}{m}, \frac{k}{m}]$. Now we define $\mathcal{H}(\frac{k}{m})$ to be the set of $h \in \mathcal{H}$ such that $\lceil\lceil e(h) \rceil\rceil = \frac{k}{m}$. We define $s(\frac{k}{m})$ to be $\ln(\max(1, |\mathcal{H}(\frac{k}{m})|))$. We now have the following lemma.

Lemma 3.1 *With high probability over the draw of S, for all hypotheses, the deviation between the empirical error $\hat{e}(h)$, and true error $e(h)$, of every hypothesis is bounded by $s(\lceil\lceil e(h) \rceil\rceil)$. More precisely,*

$\forall \delta > 0 \; \forall^{\delta} S \quad \forall h \in \mathcal{H}$,

$$D(\hat{e}(h)||e(h)) \leq \frac{s(\lceil\lceil e(h) \rceil\rceil) + \ln(\frac{2m}{\delta})}{m}.$$

Proof Quantifying over $p \in \{\frac{1}{m}, \ldots, \frac{m}{m}\}$ and $h \in \mathcal{H}(p)$ in (4) gives $\forall \delta > 0$, $\forall^{\delta} S$, $\forall p \in \{\frac{1}{m}, \ldots, \frac{m}{m}\}$, $\forall h \in \mathcal{H}(p)$,

$$D(\hat{e}(h)||e(h)) \leq \frac{\ln s(p) + \ln(\frac{2m}{\delta})}{m}.$$

But this implies the lemma. ■

Lemma 3.1 imposes a constraint, and hence a bound, on $e(h)$. More specifically, we have the following where $\mathtt{lub}\,\{x : \Phi[x]\}$ denotes the least upper bound (the maximum) of the set $\{x : \Phi[x]\}$:

$$e(h) \leq \mathtt{lub}\,\{q : D(\hat{e}(h)||q) \leq \frac{s(\lceil\lceil q \rceil\rceil) + \ln(\frac{2m}{\delta})}{m}\}. \tag{9}$$

This is our uncomputable bound. It is uncomputable because the m numbers $s(\frac{1}{m})$, ..., $s(\frac{m}{m})$ are unknown. Ignoring this problem, however, we can see that this bound is typically significantly tighter than (7). More specifically, we can rewrite (7) as

$$e(h) \leq \mathtt{lub}\,\{q : D(\hat{e}(h)||q) \leq \frac{\ln|\mathcal{H}| + \ln(\frac{2}{\delta})}{m}\}. \tag{10}$$

Since $s(\frac{k}{m}) \leq \ln|\mathcal{H}|$, and since $\frac{\ln m}{m}$ is small for large m, we have that (9) is never significantly looser than (10). Now consider a hypothesis h such that the bound on $e(h)$ given by (7), or equivalently,

(10), is significantly less than 1/2. Assuming m is large, the bound given by (9) must also be significantly less than 1/2. But for q significantly less than 1/2 we typically have that $s(\lceil\lceil q\rceil\rceil)$ is significantly smaller than $\ln|\mathcal{H}|$. For example, suppose $\mathcal{H}$ is the set of all decision trees of size $m/10$. For large m, a random decision tree of this size has error rate near 1/2. The set of decision trees with error rate significantly smaller than 1/2 is an exponentially small fraction of the set of all possible trees. So for q small compared to 1/2 we get that $s(\lceil\lceil q\rceil\rceil)$ is significantly smaller than $\ln|H|$. This makes the bound given by (9) significantly tighter than the bound given by (10).

We now show that the distribution of true errors can be replaced, essentially, by the histogram of training errors. We first introduce the following definitions:

$$\hat{\mathcal{H}}\left(\frac{k}{m},\delta\right) \equiv \left\{h\in\mathcal{H}:\ \left|\hat{e}(h)-\frac{k}{m}\right| \le \frac{1}{m}+\sqrt{\frac{\ln(\frac{16m^2}{\delta})}{2m-1}}\right\},$$

$$\hat{s}\left(\frac{k}{m},\,\delta\right) \equiv \ln\left(\max\left(1,\,2\left|\hat{\mathcal{H}}\left(\frac{k}{m},\,\delta\right)\right|\right)\right).$$

The definition of $\hat{s}\left(\frac{k}{m},\,\delta\right)$ is motivated by the following lemma.

Lemma 3.2 *With high probability over the draw of S, for all q, $s(q)\le\hat{s}(q,2\delta)$. More precisely,*
$\forall\delta>0$, $\forall^{\delta}S$, $\forall q\in\{\frac{1}{m},\,\ldots,\,\frac{m}{m}\}$,

$$s(q)\le\hat{s}(q,\,2\delta).$$

Before proving Lemma 3.2 we note that by conjoining (9) and Lemma 3.2 we get the following. This is our main result.

Theorem 3.3 *With high probability over the draw of S, for all hypotheses, the deviation between the empirical error $\hat{e}(h)$, and true error $e(h)$, of every hypothesis is bounded by $\hat{s}(\lceil\lceil q\rceil\rceil,\delta)$. More precisely,*
$\forall\delta>0$, $\forall^{\delta}S$, $\forall h\in\mathcal{H}$,

$$e(h)\le \texttt{lub}\left\{q: D(\hat{e}(h)||q)\le\frac{\hat{s}(\lceil\lceil q\rceil\rceil,\,\delta)+\ln(\frac{4m}{\delta})}{m}\right\}.$$

As for Lemma 3.1, the bound implicit in Theorem 3.3 is typically significantly tighter than the bound in (7) or its equivalent form (10). The argument for the improved tightness of Theorem 3.3 over (10) is similar to the argument for (9). More specifically, consider a hypothesis h for which the bound in (10) is significantly less than 1/2. Since $\hat{s}(\lceil\lceil q\rceil\rceil,\,\delta)\le\ln|\mathcal{H}|$, the set of values of q satisfying the condition in Theorem 3.3 must all be significantly less than 1/2. But for large m we have that $\sqrt{\frac{\ln(16m^2/\delta)}{2m-1}}$ is small. So if q is significantly less than 1/2 then all hypotheses in $\hat{\mathcal{H}}(\lceil\lceil q\rceil\rceil,\delta)$ have empirical error rates significantly less than 1/2. But for most hypothesis classes, e.g., decision trees, the set of hypotheses with empirical error rates far from 1/2 should be an exponentially small fraction of the class. Hence we get that $\hat{s}(\lceil\lceil q\rceil\rceil,\,\delta)$ is significantly less than $\ln|\mathcal{H}|$ and Theorem 3.3 is tighter than (10).

The remainder of this section is a proof of Lemma 3.2. Our departure point for the proof is the following lemma from McAllester (1999).

Lemma 3.4 (McAllester 99) *For any measure on any hypothesis class we have the following where* $\mathsf{E}_h f(h)$ *denotes the expectation of* $f(h)$ *under the given measure on* h*:*

$$\forall \delta > 0 \; \forall^{\delta} S \quad \mathsf{E}_h e^{(2m-1)(\hat{e}(h)-e(h))^2} \leq \frac{4m}{\delta}.$$

Intuitively, this lemma states that with high confidence over the choice of the sample most hypotheses have empirical error near their true error. This allows us to prove that $\hat{s}(\lceil\lceil q\rceil\rceil, \delta)$ bounds $s(\lceil\lceil q\rceil\rceil)$. More specifically, by considering the uniform distribution on $\mathcal{H}(\frac{k}{m})$, Lemma 3.4 implies

$$\begin{aligned}
\mathsf{E}_{h\sim\mathcal{H}(\frac{k}{m})}\left(e^{(2m-1)(\hat{e}(h)-e(h))^2}\right) &\leq \frac{4m}{\delta} \\
Pr_{h\sim\mathcal{H}(\frac{k}{m})}\left(e^{(2m-1)(\hat{e}(h)-e(h))^2} \geq \frac{8m}{\delta}\right) &\leq \frac{1}{2} \\
Pr_{h\sim\mathcal{H}(\frac{k}{m})}\left(e^{(2m-1)(\hat{e}(h)-e(h))^2} \leq \frac{8m}{\delta}\right) &\geq \frac{1}{2} \\
\left|\left\{h \in \mathcal{H}(\frac{k}{m}) : \; |\hat{e}(h)-e(h)| \leq \sqrt{\frac{\ln(\frac{8m}{\delta})}{2m-1}}\right\}\right| &\geq \frac{1}{2}|\mathcal{H}(\frac{k}{m})| \\
\left|\left\{h \in \mathcal{H}(\frac{k}{m}) : \; |\hat{e}(h)-\frac{k}{m}| \leq \frac{1}{m}+\sqrt{\frac{\ln(\frac{8m}{\delta})}{2m-1}}\right\}\right| &\geq \frac{1}{2}|\mathcal{H}(\frac{k}{m})| \\
\left|\mathcal{H}(\frac{k}{m})\right| \leq 2\left|\hat{\mathcal{H}}\left(\frac{k}{m}, \; 2m\delta\right)\right|.
\end{aligned}$$

Lemma 3.2 now follows by quantification over $q \in \{\frac{1}{m}, \ldots, \frac{m}{m}\}$. □

4. Asymptotic Analysis and Phase Transitions

This section and the two that follow give an asymptotic analysis of the bounds presented earlier. The asymptotic analysis is stated in Theorem 4.1 and Statement 6.1. To develop the asymptotic analysis, however, a preliminary discussion is needed regarding the phenomenon of phase transitions. The bounds given in (9) and Theorem 3.3 exhibit phase transitions. More specifically, the bounding expression can be discontinuous in δ and m, e.g., arbitrarily small changes in δ can cause large changes in the bound. To see how this happens consider the constraint on the quantity q:

$$D(\hat{e}(h)||q) \leq \frac{s(\lceil\lceil q\rceil\rceil)+\ln(\frac{2m}{\delta})}{m}. \tag{11}$$

The bound given by (9) is the least upper bound of the values of q satisfying (11). Assume that m is sufficiently large that we can think of $\frac{s(\lceil\lceil q\rceil\rceil)}{m}$ as a continuous function of q which we write as $\bar{s}(q)$. We can then rewrite (11) where λ is a quantity not depending on q and $\bar{s}(q)$ does not depend on δ:

$$D(\hat{e}(h)||q) \leq \bar{s}(q)+\lambda. \tag{12}$$

For $q \geq \hat{e}(h)$ we know that $D(\hat{e}(h)||q)$ is a monotonically increasing function of q. It is reasonable to assume that for $q \leq 1/2$ we also have that $\bar{s}(q)$ is a monotonically increasing function of q. But even under these conditions it is possible that the feasible values of q, i.e., those satisfying (12), can be divided into separated regions. Furthermore, increasing λ can cause a new feasible region to come into existence. When this happens the bound, which is the least upper bound of the feasible values, can increase discontinuously. At a more intuitive level, consider a large number of high error concepts and smaller number of lower error concepts. At a certain confidence level the high error concepts can be ruled out. But as the confidence requirement becomes more stringent suddenly (and discontinuously) the high error concepts must be considered. A similar discontinuity can occur in sample size. Phase transitions in shell decomposition bounds are discussed in more detail by Haussler et al. (1996).

Phase transitions complicate asymptotic analysis. But asymptotic analysis illuminates the nature of phase transitions. As mentioned in the introduction, in the asymptotic analysis of learning theory bounds it is important that one does not hold $\mathcal{H}$ fixed as the sample size increases. If we hold $\mathcal{H}$ fixed then $\lim_{m\to\infty} \frac{\ln|\mathcal{H}|}{m} = 0$. But this is not what one expects for large samples in practice. As the sample size increases one typically uses larger hypothesis classes. Intuitively, we expect that even for very large m we have that $\frac{\ln|\mathcal{H}|}{m}$ is far from zero.

For the asymptotic analysis of the bound in (9) we assume an infinite sequence of hypothesis classes $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$ ... and an infinite sequence of data distributions D_1, D_2, D_3, ... Let $s_m(\frac{k}{m})$ be $s(\frac{k}{m})$ defined relative to $\mathcal{H}_m$ and D_m. In the asymptotic analysis we assume that the sequence of functions $\frac{s_m(\lceil\lceil q\rceil\rceil)}{m}$, viewed as functions of $q \in [0,1]$, converge uniformly to a continuous function $\bar{s}(q)$. This means that for any $\varepsilon > 0$ there exists a k such that for all $m > k$ we have

$$\forall q \in [0,1] \;\; |\frac{s_m(\lceil\lceil q\rceil\rceil)}{m} - \bar{s}(q)| \leq \varepsilon.$$

Given the functions $\frac{s_m(\lceil\lceil p\rceil\rceil)}{m}$ and their limit function $\bar{s}(p)$, we define the following functions of an empirical error rate $\hat{e}$:

$$B_m(\hat{e}) \equiv \mathtt{lub}\left\{q : D(\hat{e}||q) \leq \frac{s_m(\lceil\lceil q\rceil\rceil) + \ln(\frac{2m}{\delta})}{m}\right\},$$

$$B(\hat{e}) \equiv \mathtt{lub}\,\{q : D(\hat{e}||q) \leq \bar{s}(q)\}.$$

The function $B_m(\hat{e})$ corresponds directly to the upper bound in (9). The function $B(\hat{e})$ is intended to be the large m asymptotic limit of $B_m(\hat{e})$. However, phase transitions complicate asymptotic analysis. The bound $B(\hat{e})$ need not be a continuous function of $\hat{e}$. A value of $\hat{e}$ where the bound $B(\hat{e})$ is discontinuous corresponds to a phase transition in the bound. At a phase transition the sequence $B_m(\hat{e})$ need not converge. Away from phase transitions, however, we have the following theorem.

Theorem 4.1 *If the bound $B(\hat{e})$ is continuous at the point $\hat{e}$ (so we are not at a phase transition), and the functions (parameterized by m) $\frac{s_m(\lceil\lceil q\rceil\rceil)}{m}$, viewed as functions of $q \in [0,1]$, converge uniformly to a continuous function $\bar{s}(q)$, then we have*

$$\lim_{m\to\infty} B_m(\hat{e}) = B(\hat{e}).$$

Proof Define the set $F_m(\hat{e})$ as

$$F_m(\hat{e}) \equiv \left\{ q : D(\hat{e}||q) \leq \frac{s_m(\lceil\lceil q \rceil\rceil) + \ln(\frac{2m}{\delta})}{m} \right\}.$$

This gives

$$B_m(\hat{e}) = \mathtt{lub}\, F_m(\hat{e}).$$

Similarly, define $F(\hat{e}, \varepsilon)$ and $B(\hat{e}, \varepsilon)$ as

$$\begin{aligned} F(\hat{e}, \varepsilon) &\equiv \{q \in [0,1] : D(\hat{e}||q) \leq \bar{s}(q) + \varepsilon\}. \\ B(\hat{e}, \varepsilon) &\equiv \mathtt{lub}\, F(\hat{e}, \varepsilon). \end{aligned}$$

We first show that the continuity of $B(\hat{e})$ at the point $\hat{e}$ implies the continuity of $B(\hat{e}, \varepsilon)$ at the point $\langle \hat{e}, 0 \rangle$. We note that there exists a continuous function $f(\hat{e}, \varepsilon)$ with $f(\hat{e}, 0) = \hat{e}$ and such that for any ε sufficiently near 0 we have

$$D(f(\hat{e}, \varepsilon)||q) = D(\hat{e}||q) - \varepsilon.$$

We then have

$$B(\hat{e}, \varepsilon) = B(f(x, \varepsilon)).$$

Since f is continuous, and $B(\hat{e})$ is continuous at the point $\hat{e}$, we get that $B(\hat{e}, \varepsilon)$ is continuous at the point $\langle \hat{e}, 0 \rangle$.

We now prove the lemma. The functions of the form $\frac{s_m(\lceil\lceil q \rceil\rceil) + \ln \frac{2m}{\delta}}{m}$ converge uniformly to the function $\bar{s}(q)$. This implies that for any $\varepsilon > 0$ there exists a k such that for all $m > k$ we have

$$F(\hat{e}, -\varepsilon) \subseteq F_m(\hat{e}) \subseteq F(\hat{e}, \varepsilon).$$

But this in turn implies that

$$B(\hat{e}, -\varepsilon) \leq B_m(\hat{e}) \leq B(\hat{e}, \varepsilon). \tag{13}$$

The lemma now follows from the continuity of the function $B(\hat{e}, \varepsilon)$ at the point $\langle \hat{e}, 0 \rangle$. ■

Theorem 4.1 can be interpreted as saying that for large sample sizes, and for values of $\hat{e}$ other than the special phase transition values, the bound has a well defined value independent of the confidence parameter δ and determined only by a smooth function $\bar{s}(q)$. A similar statement can be made for the bound in Theorem 3.3 — for large m, and at points other than phase transitions, the bound is independent of δ and is determined by a smooth limit curve.

For the asymptotic analysis of Theorem 3.3 we assume an infinite sequence $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \ldots$ of hypothesis classes and an infinite sequence $S_1, S_2, S_3, \ldots$ of samples such that sample S_m has size m. Let $\mathcal{H}_m(\frac{k}{m}, \delta)$ and $\hat{s}_m(\frac{k}{m}, \delta)$ be $\mathcal{H}(\frac{k}{m}, \delta)$ and $\hat{s}(\frac{k}{m}, \delta)$ respectively defined relative to hypothesis class $\mathcal{H}_m$ and sample S_m. Let $U_m(\frac{k}{m})$ be the set of hypotheses in $\mathcal{H}_m$ having an empirical error of exactly $\frac{k}{m}$ in the sample S_m. Let $u_m(\frac{k}{m})$ be $\ln(\max(1, |U_m(\frac{k}{m})|)$. In the analysis of Theorem 3.3 we allow that the functions $\frac{u_m(\lceil\lceil q \rceil\rceil)}{m}$ are only locally uniformly convergent to a continuous function $\bar{u}(q)$, i.e., for any $q \in [0,1]$ and any $\varepsilon > 0$ there exists an integer k and real number $\gamma > 0$ satisfying

$$\forall m > k,\ \forall p \in (q-\gamma, q+\gamma)\ |\frac{u_m(\lceil\lceil p \rceil\rceil)}{m} - \bar{u}(p)| \leq \varepsilon.$$

Locally uniform convergence plays a role in the analysis in Section 6.

Theorem 4.2 *If the functions* $\frac{u_m(\lceil\lceil q\rceil\rceil)}{m}$ *converge locally uniformly to a continuous function* $\overline{u}(q)$ *then, for any fixed value of* δ*, the functions* $\frac{\hat{s}_m(\lceil\lceil q\rceil\rceil,\,\delta)}{m}$ *also converge locally uniformly to* $\overline{u}(q)$*. If the convergence of* $\frac{u_m(\lceil\lceil q\rceil\rceil)}{m}$ *is uniform, then so is the convergence of* $\frac{\hat{s}_m(\lceil\lceil q\rceil\rceil,\,\delta)}{m}$*.*

Proof Consider an arbitrary value $q \in [0,1]$ and $\varepsilon > 0$. We construct the desired k and γ. More specifically, select k sufficiently large and γ sufficiently small that we have the properties

$$\forall m > k,\ \forall p \in (q-2\gamma,\ q+2\gamma)\ \left|\frac{u_m(\lceil\lceil p\rceil\rceil)}{m} - \overline{u}(p)\right| < \frac{\varepsilon}{3},$$

$$\forall p \in (q-2\gamma,\ q+2\gamma)\ |\overline{u}(p) - \overline{u}(q)| \leq \frac{\varepsilon}{3},$$

$$\frac{1}{k} + \sqrt{\frac{\ln(\frac{16k^2}{\delta})}{2k-1}} < \gamma,$$

$$\frac{\ln k}{k} \leq \frac{\varepsilon}{3}.$$

Consider an $m > k$ and $p \in (q-\gamma,\ q+\gamma)$. It now suffices to show that

$$\left|\frac{\hat{s}_m(\lceil\lceil p\rceil\rceil,\ \delta)}{m} - \overline{u}(p)\right| \leq \varepsilon.$$

Because $U_m(\lceil\lceil p\rceil\rceil)$ is a subset of $\mathcal{H}_m(\lceil\lceil p\rceil\rceil,\ \delta)$ we have

$$\frac{\hat{s}_m(\lceil\lceil p\rceil\rceil,\ \delta)}{m} \geq \frac{u_m(\lceil\lceil p\rceil\rceil)}{m} \geq \overline{u}(p) - \frac{\varepsilon}{3}.$$

We can also upper bound $\frac{\hat{s}_m(\lceil\lceil p\rceil\rceil,\,\delta)}{m}$ as follows:

$$\begin{aligned}
|\mathcal{H}_m(\lceil\lceil p\rceil\rceil,\delta)| &\leq \sum_{|\frac{k}{m}-p|\leq\gamma} \left|U_m\left(\frac{k}{m}\right)\right| \\
&\leq \sum_{|\frac{k}{m}-p|\leq\gamma} e^{u_m(\frac{k}{m})} \\
&\leq \sum_{|\frac{k}{m}-p|\leq\gamma} e^{m(\overline{u}(\frac{k}{m})+\frac{\varepsilon}{3})} \\
&\leq \sum_{|\frac{k}{m}-p|\leq\gamma} e^{m(\overline{u}(p)+\frac{2\varepsilon}{3})} \\
&\leq m e^{m(\overline{u}(p)+\frac{2\varepsilon}{3})} \\
\\
\frac{\hat{s}(\lceil\lceil p\rceil\rceil,\ \delta)}{m} &\leq \overline{u}(p) + \frac{2\varepsilon}{3} + \frac{\ln m}{m} \\
&\leq \overline{u}(p) + \varepsilon.
\end{aligned}$$

A similar argument shows that if $\frac{u_m(\lceil\lceil q\rceil\rceil)}{m}$ converges uniformly to $\overline{u}(q)$ then so does $\frac{u_m(\lceil\lceil q\rceil\rceil)}{m}$. ■

Given quantities $\frac{\hat{s}_i(\lceil\lceil q\rceil\rceil,\,\delta)}{m}$ that converge uniformly to $\overline{u}(q)$ the remainder of the analysis is identical to that for the asymptotic analysis of (9). We define the upper bounds

$$\hat{B}_m(\hat{e}) \equiv \mathtt{lub}\left\{q : D(\hat{e}||q) \leq \frac{\hat{s}_m(\lceil\lceil q\rceil\rceil,\,\delta)+\ln\left(\frac{4m}{\delta}\right)}{m}\right\}.$$

$$\hat{B}(\hat{e}) \equiv \mathtt{lub}\,\{q : D(\hat{e}||q) \leq \overline{u}(q)\}.$$

Again we say that $\hat{e}$ is at a phase transition if the function $\hat{B}(\hat{e})$ is discontinuous at the value $\hat{e}$. We then get the following whose proof is identical to that of Theorem 4.1.

Theorem 4.3 *If the bound $\hat{B}(\hat{e})$ is continuous at the point $\hat{e}$ (so we are not at a phase transition), and the functions $\frac{u_m(\lceil\lceil q\rceil\rceil)}{m}$ converge uniformly to $\overline{u}(q)$, then we have that*

$$\lim_{m\to\infty} \hat{B}_m(\hat{e}) = \hat{B}(\hat{e}).$$

5. Asymptotic Optimality of (9)

Formula (9) can be viewed as providing an upper bound on $e(h)$ as a function of $\hat{e}(h)$ and the function s. In this section we show that for any curve s and value $\hat{e}$ there exists a hypothesis class and data distribution such that the upper bound in (9) is realized up to asymptotic equality. Up to asymptotic equality, (9) is the tightest possible bound computable from $\hat{e}(h)$ and the m numbers $s(\frac{1}{m}), \ldots, s(\frac{m}{m})$.

The classical VC dimensions bounds are nearly optimal over bounds computable from the chosen hypothesis error rate $\hat{e}(h^*)$ and the class $\mathcal{H}$. The m numbers $s(\frac{1}{m}), \ldots, s(\frac{m}{m})$ depend on both $\mathcal{H}$ and the data distribution. Hence the bound in (9) uses information about the distribution and hence can be tighter than classical VC bounds. A similar statement applies to the bound in Theorem (3.3) computed from the empirically observable numbers $\hat{s}(\frac{1}{m}), \ldots, \hat{s}(\frac{m}{m})$. In this case, the bound uses more information from the sample than just $\hat{e}(h)$. The optimality theorem given here also differs from the traditional lower bound results for VC dimension in that here the lower bounds match the upper bounds up to asymptotic equality.

The departure point for our optimality analysis is the following lemma from Cover and Thomas (1991).

Lemma 5.1 (Cover and Thomas) *If $\hat{p}$ is the fraction of heads out of m tosses of a coin where the true probability of heads is p then for $q \geq p$ we have*

$$Pr(\hat{p} \geq q) \geq \frac{1}{m+1} e^{-mD(q||p)}.$$

This lower bound on $Pr(\hat{p} \geq q)$ is very close to Chernoff's 1952 upper bound (1). The tightness of (9) is a direct reflection of the tightness (1). To exploit Lemma 5.1 we need to construct hypothesis classes and data distributions where distinct hypotheses have independent training errors. More specifically, we say that a set of hypotheses $\{h_1, \ldots, h_n\}$ has independent training errors if the random variables $\hat{e}(h_1), \ldots, \hat{e}(h_n)$ are independent.

By an argument similar to the derivation of (3) from (1) we can prove the from Lemma 5.1 that

$$Pr\left(D(\min(\hat{p},p)||p) \geq \frac{\ln(\frac{1}{\delta})-\ln(m+1)}{m}\right) \geq \delta. \tag{14}$$

Lemma 5.2 *Let X be any finite set, S a random variable, and $\Theta[S,x,\delta]$ a formula such that for every $x \in X$ and $\delta > 0$ we have $Pr(\Theta[S,x,\delta]) \geq \delta$, and $Pr(\forall x \in X\ \Theta[S,\ x,\ \delta]) = \prod_{x\in X} Pr(\Theta[S,\ x,\ \delta])$. We then have $\forall \delta > 0\ \forall^{\delta} S\ \ \exists x \in X\ \ \Theta[S,x,\frac{\ln(\frac{1}{\delta})}{|X|}]$.*

Proof

$$\begin{array}{rcl} Pr(\Theta[S,x,\frac{\ln(\frac{1}{\delta})}{|X|}]) & \geq & \frac{\ln(\frac{1}{\delta})}{|X|} \\ Pr(\neg\Theta[S,x,\frac{\ln(\frac{1}{\delta})}{|X|}]) & \leq & 1-\frac{\ln(\frac{1}{\delta})}{|X|} \\ & \leq & e^{-\frac{\ln(\frac{1}{\delta})}{|X|}} \\ Pr(\forall x \in X \neg\Theta[S,\ x,\ \frac{\ln(\frac{1}{\delta})}{|X|}]) & \leq & e^{-\ln(\frac{1}{\delta})} = \delta. \end{array}$$

■

Now define $h^*(\frac{k}{m})$ to be the hypothesis of minimal training error in the set $\mathcal{H}(\frac{k}{m})$. Let $\texttt{glb}\ \{x : \Phi[x]\}$ denote the greatest lower bound (the minimum) of the set $\{x :\ \Phi[x]\}$. We now have the following lemma.

Lemma 5.3 *If the hypotheses in the class $\mathcal{H}(\lceil\lceil q\rceil\rceil)$ are independent then $\forall\delta > 0$, $\forall^{\delta} S$, $\forall q \in \{\frac{1}{m},\ \ldots,\ \frac{m}{m}\}$,*

$$\hat{e}(h^*(q)) \leq \texttt{glb}\left\{\hat{e} :\ D(\min(\hat{e},\ q-\tfrac{1}{m})||q) \leq \tfrac{s(q)-\ln(m+1)-\ln(\ln(\frac{m}{\delta}))}{m}\ \right\}.$$

Proof To prove Lemma 5.3 let q be a fixed rational number of the form $\frac{k}{m}$. Assuming independent hypotheses we can applying Lemma 5.2 to (14) to get $\forall\delta > 0$, $\forall^{\delta} S$, $\exists h \in \mathcal{H}(\frac{k}{m})$,

$$D(\min(\hat{e}(h),e(h))||e(h)) \geq \frac{s(q)-\ln(m+1)-\ln(\ln(\frac{1}{\delta}))}{m}$$

Let w be the hypothesis in $\mathcal{H}(q)$ satisfying this formula. We now have $\hat{e}(h^*(q)) \leq \hat{e}(w)$ and $q-\frac{1}{m} \leq e(w) \leq q$. These two conditions imply $\forall\delta > 0$, $\forall^{\delta} S$,

$$D(\min(\hat{e}(h^*(q)),q-\tfrac{1}{m})||q) \geq \tfrac{s(q)-\ln(m+1)-\ln(\ln\frac{1}{\delta})}{m}.$$

This implies that

$$\hat{e}(h^*(q)) \leq \texttt{glb}\left\{\hat{e} :\ D(\min(\hat{e},\ q-\tfrac{1}{m})||q) \leq \tfrac{s(q)-\ln(m+1)-\ln(\ln(\frac{1}{\delta}))}{m}.\ \right\}$$

Lemma 5.3 now follows by quantification over $q \in \{\frac{1}{m},\ \ldots,\ \frac{m}{m}\}$.

■

For $q \in [0,1]$ we have that Lemma 3.1 implies that

$$\hat{e}(h^*(\lceil\lceil q\rceil\rceil)) \geq \texttt{glb}\left\{\hat{e}:\ D\left(\hat{e}||\lceil\lceil q\rceil\rceil - \tfrac{1}{m}\right) \leq \tfrac{s(\lceil\lceil q\rceil\rceil)+\ln\left(\frac{2m}{\delta}\right)}{m}.\right\}$$

We now have upper and lower bounds on the quantity $\hat{e}(h^*(\lceil\lceil q\rceil\rceil))$ which agree up to asymptotic equality — in a large m limit where $\frac{s_m(\lceil\lceil q\rceil\rceil)}{m}$ converges (pointwise) to a continuous function $\bar{s}(q)$ we have that the upper and lower bound on $\hat{e}(h^*(\lceil\lceil q\rceil\rceil))$ both converge (pointwise) to

$$\hat{e}(h^*(q)) = \texttt{glb}\,\{\hat{e} : D(\hat{e}||q) \leq \bar{s}(q)\}.$$

This asymptotic value of $\hat{e}(h^*(q))$ is a continuous function of q. Since q is held fixed in calculating the bounds on $\hat{e}(\lceil\lceil q\rceil\rceil)$, phase transitions are not an issue and uniform convergence of the functions $\frac{s_m(\lceil\lceil q\rceil\rceil)}{m}$ is not required. Note that for large m and independent hypotheses we get that $\hat{e}(h^*(q))$ is determined as a function of the true error rate q and $\frac{s(\lceil\lceil q\rceil\rceil)}{m}$.

The following lemma states that any limit function $\bar{s}(p)$ is consistent with the possibility that hypotheses are independent. This, together with Lemma 5.3 implies that no uniform bound on $e(h)$ as a function of $\hat{e}(h)$ and $|\mathcal{H}(\frac{1}{m})|$, ..., $|\mathcal{H}(\frac{m}{m})|$ can be asymptotically tighter than (9).

Theorem 5.4 *Let $\bar{s}(p)$ be any continuous function of $p \in [0,1]$. There exists an infinite sequence of hypothesis spaces $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, ..., and sequence of data distributions D_1, D_2, D_3, ... such that each class $\mathcal{H}_m$ has independent hypotheses for data distribution D_m and such that $\frac{s_m(\lceil\lceil p\rceil\rceil)}{m}$ converges (pointwise) to $\bar{s}(p)$.*

Proof First we show that if $|\mathcal{H}_m(\frac{i}{m})| = e^{m\bar{s}(\frac{i}{m})}$ then the functions $\frac{s_m(\lceil\lceil p\rceil\rceil)}{m}$ converge (pointwise) to $\bar{s}(p)$. Assume $|\mathcal{H}_m(\frac{i}{m})| = e^{m\bar{s}(\frac{i}{m})}$. In this case we have

$$\frac{s_m(\lceil\lceil p\rceil\rceil)}{m} = \bar{s}(\lceil\lceil p\rceil\rceil).$$

Since $\bar{s}(p)$ is continuous, for any fixed value of p we get that $\frac{s_m(\lceil\lceil p\rceil\rceil)}{m}$ converges to $\bar{s}(p)$.

Recall that D_m is a probability distribution on pairs $\langle x, y\rangle$ with $y \in \{0,1\}$ and $x \in X_m$ for some set X_m. We take $\mathcal{H}_m$ to be a disjoint union of sets $\mathcal{H}_m(\frac{k}{m})$ where $|\mathcal{H}_m(\frac{k}{m})|$ is selected as above. Let f_1, ..., f_N be the elements of $\mathcal{H}_m$ with $N = |\mathcal{H}_m|$. Let X_m be the set of all N-bit bit strings and define $f_i(x)$ to be the value of ith bit of the bit vector x. Now define the distribution D_m on pairs $\langle x, y\rangle$ by selecting y to be 1 with probability 1/2 and then selecting each bit of x independently where the ith bit is selected to disagree with y with probability $\frac{k}{m}$ where k is such that $f_i \in \mathcal{H}_m(\frac{k}{m})$. ■

6. Relating $\hat{s}$ and s

In this section we show that in large m limits of the type discussed in Section 4 the histogram of empirical errors need not converge to the histogram of true errors. So even in the large m asymptotic limit, the bound given by Theorem 3.3 is significantly weaker than the bound given by (9).

To show that $\hat{s}(\lceil\lceil q\rceil\rceil, \delta)$ can be asymptotically different from $s(\lceil\lceil q\rceil\rceil)$ we consider the case of independent hypotheses. More specifically, given a continuous function $\bar{s}(p)$ we construct an infinite

sequence of hypothesis spaces $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, ... and an infinite sequence of data distributions D_1, D_2, D_3, ... using the construction in the proof of Theorem 5.4. We note that if $\overline{s}(p)$ is differentiable with bounded derivative then the functions $\frac{s_m(\lceil\lceil p\rceil\rceil)}{m}$ converge uniformly to $\overline{s}(p)$.

For a given infinite sequence data distributions we generate an infinite sample sequence S_1, S_2, S_3, ..., by selecting S_m to consists of m pairs $\langle x, y\rangle$ drawn IID from distribution D_m. For a given sample sequence and $h \in \mathcal{H}_m$ we define $\hat{e}_m(h)$ and $\hat{s}_m(\frac{k}{m}, \delta)$ in a manner similar to $\hat{e}(h)$ and $\hat{s}(\frac{k}{m}, \delta)$ but for sample S_m. The main result of this section is the following.

Conjecture 6.1 *If each $\mathcal{H}_m$ has independent hypotheses under data distribution D_m, and the functions $\frac{s_m(\lceil\lceil p\rceil\rceil)}{m}$ converge uniformly to a continuous function $\overline{s}(p)$, then for any $\delta > 0$ and $p \in [0,1]$, we have, with probability 1 over the generation of the sample sequence, that*

$$\lim_{m\to\infty} \frac{\hat{s}_m(\lceil\lceil p\rceil\rceil, \delta)}{m} = \sup_{q\in[0,1]} \overline{s}(q) - D(p||q).$$

We call this a conjecture rather than a theorem because the proof has not been worked out to a high level of rigor. Nonetheless, we believe the proof sketch given below can be expanded to a fully rigorous argument.

Before giving the proof sketch we note that the limiting value of $\frac{\hat{s}_m(\lceil\lceil p\rceil\rceil, \delta)}{m}$ is independent of δ. This is consistent with Theorem 4.2. Define

$$\overline{\hat{s}}(p) \equiv \sup_{q\in[0,1]} \overline{s}(q) - D(p||q).$$

Note that $\overline{\hat{s}}(p) \geq \overline{s}(p)$. This gives an asymptotic version of Lemma 3.2. But since $D(p||q)$ can be locally approximated as $c(p-q)^2$ (up to its second order Taylor expansion), if $\overline{s}(p)$ is increasing at the point p then we also get that $\hat{s}(p)$ is strictly larger than $\overline{s}(p)$.

Proof Outline: To prove Statement 6.1 we first define $\mathcal{H}_m(p, q)$ for $p, q \in \{\frac{1}{m}, \ldots, \frac{m}{m}\}$ to be the set of all $h \in \mathcal{H}_m(q)$ such that $\hat{e}_m(h) = p$. Intuitively, $\mathcal{H}_m(p, q)$ is the set of concepts with true error rate near q that have empirical error rate p. Ignoring factors that are only polynomial in m, the probability of a hypothesis with true error rate q having empirical error rate p can be written as (approximately) $e^{-mD(p||q)}$. So the expected size of $\mathcal{H}_m(p, q)$ can be written as $|\mathcal{H}_m(q)|e^{-mD(p||q)}$, or alternatively, (approximately) as $e^{m\overline{s}(q)}e^{-mD(p||q)}$ or $e^{m(\overline{s}(q)-D(p||q))}$. More formally, we have, for any fixed value of p and q,

$$\lim_{m\to\infty} \frac{\ln(\max(1, \mathsf{E}(|\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)|)))}{m} = \max(0, \overline{s}(q) - D(p||q)).$$

We now show that the expectation can be eliminated from the above limit. First, consider distinct values of p and q such that $\overline{s}(q) - D(p||q) > 0$. Since p and q are distinct, the probability that a fixed hypothesis in $\mathcal{H}_m(\lceil\lceil q\rceil\rceil)$ is in $\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)$ declines exponentially in m. Since $\overline{s}(q) - D(p||q) > 0$ the expected size of $\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)$ grows exponentially in m. Since the hypotheses are independent, the distribution of possible values of $|\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)|$ becomes essentially a Poisson mass distribution with an expected number of arrivals growing exponentially in m. The probability that $|\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)|$ deviates from its expectation by as much as a factor of 2 declines exponentially in m. We say that a sample sequence is safe after k if for all $m > k$ we

have that $|\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)|$ is within a factor of 2 of its expectation. Since the probability of being unsafe at m declines exponentially in m, for any δ there exists a k such that with probability at least $1-\delta$ the sample sequence is safe after k. So for any $\delta > 0$ we have that with probability at least $1-\delta$ the sequence is safe after some k. But since this holds for all $\delta > 0$, with probability 1 such a k must exist:

$$\lim_{m\to\infty} \frac{\ln(\max(1, |\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)|))}{m} = \bar{s}(q) - D(p||q).$$

We now define

$$s_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil) \equiv \ln(\max(1, |\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)|)).$$

It is also possible to show for $p = q$ we have that with probability 1 we have that $\frac{s_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)}{m}$ approaches $\bar{s}(p)$ and that for distinct p and q with $\bar{s}(q) - D(p||q) \leq 0$ we have that $\frac{s_m(\lceil\lceil q\rceil\rceil, \lceil\lceil q\rceil\rceil)}{m}$ approaches 0. Putting these together yields that, with probability 1, we have

$$\lim_{m\to\infty} \frac{s_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q\rceil\rceil)}{m} = \max(0, \bar{s}(q) - D(p||q)). \tag{15}$$

Define $U_m(\frac{k}{m})$ and $u_m(\frac{k}{m})$ as in Section 4. We now have the following equality:

$$U_m(p) = \cup_{q\in\{\frac{1}{m}, \ldots, \frac{m}{m}\}} \mathcal{H}_m(p, q).$$

We now show that with probability 1, $\frac{u_m(p)}{m}$ approaches $\bar{\hat{s}}(p)$. First, consider a $p \in [0,1]$ such that $\bar{\hat{s}}(p) > 0$. Let Since $\bar{s}(q) - D(q||p)$ is a continuous function, and $[0,1]$ is a compact set, $\sup_{q\in[0,1]} \bar{s}(q) - D(p||q)$ must be realized at some value $\overset{*}{q} \in [0,1]$. Let q^* be such that $\bar{s}(\overset{*}{q}) - D(p||q^*)$ equals $\bar{\hat{s}}(p)$. We have that $u_m(\lceil\lceil p\rceil\rceil) \geq s_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q^*\rceil\rceil)$. This, together with (15), implies that

$$\liminf_{m\to\infty} \frac{u_m(\lceil\lceil p\rceil\rceil)}{m} \geq \bar{\hat{s}}(p).$$

The sample sequence is "safe" at m and $\frac{k}{m}$ if $|\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil \frac{k}{m}\rceil\rceil)|$ does not exceed twice the expectation of $|\mathcal{H}_m(\lceil\lceil p\rceil\rceil, \lceil\lceil q^*\rceil\rceil)|$. Assuming uniform convergence of $\frac{s_m(\lceil\lceil p\rceil\rceil)}{m}$, the probability of not being safe at m and $\frac{k}{m}$ declines exponentially in m at a rate at least as fast as the rate of decline of the probability of not being safe at m and $\lceil\lceil q^*\rceil\rceil$. By the union bound this implies that for a given m the probability that there exists an unsafe $\frac{k}{m}$ also declines exponentially. We say that the sequence is safe after N if it is safe for all m and $\frac{k}{m}$ with $m > N$. The probability of not being being safe after N also declines exponentially with N. By an argument similar to that given above, this implies that with probability 1 over the choice of the sequence there exists a N such that the sequence is safe after N. But if we are safe at m then $|U_m(\lceil\lceil p\rceil\rceil)| \leq 2mE|\mathcal{H}_m(p, \lceil\lceil q^*\rceil\rceil)|$. This implies that

$$\limsup_{m\to\infty} \frac{u_m(\lceil\lceil p\rceil\rceil)}{m} \leq \bar{\hat{s}}(p).$$

Putting the two bounds together we get

$$\lim_{m\to\infty} \frac{u_m(\lceil\lceil p\rceil\rceil)}{m} = \bar{\hat{s}}(p).$$

The above argument establishes (to some level of rigor) pointwise convergence of $\frac{u_m(\lceil\lceil p\rceil\rceil)}{m}$ to $\hat{s}(p)$. It is also possible to establish a convergence rate that is a continuous function of p. This implies that the convergence of $\frac{u_m(\lceil\lceil p\rceil\rceil)}{m}$ can be made locally uniform. Theorem 4.2 then implies the desired result. □

7. Improvements

Theorem 3.3 has been improved in various ways (Langford, 2002):

- Removing the discretization of true errors,
- Using one-sided bounds,
- Using nonuniform union bounds over discrete values of the form $\frac{k}{m}$,
- Tightening the Chernoff bound using direct calculation of Binomial coefficients, and
- Improving Lemma 3.4.

These improvements allow the removal of all but one $\ln(m)$ terms from the statement of the bound. However, they do not improve the asymptotic equations given by Theorem 4.1 and Statement 6.1.

A practical difficulty with the bound in Theorem 3.3 is that it is usually impossible to enumerate the elements of an exponentially large hypothesis class and hence impractical to compute the histogram of training errors for the hypotheses in the class. In practice the values of $s(\frac{k}{m})$ might be estimated using some form of Monte-Carlo Markov chain sampling over the hypotheses. For certain hypothesis spaces it might also be possible to directly calculate the empirical error distribution without evaluating every hypothesis. For example, this can be done with "partition rules" which, given a fixed partition of the input space, make predictions which are constant on each partition. If there are n elements in the partition then there are 2^n partition rules. For a fixed partition, the histogram of empirical errors for the 2^n partition rules can be computed in polynomial time. Note that the class of decision trees is a union of partition rules where the structure of a tree defines a partition and the labels at the leaves of the tree define a particular partition rule relative to that partition. Taking advantage of this, it is suprisingly easy to compute a shell bound for small decision trees (Langford, 2002).

8. Discussion and Conclusion

Traditional PAC bounds are stated in terms of the training error and class size or VC dimension. The computable bound given here is sometimes much tighter because it exploits the additional information in the histogram of training errors. The uncomputable bound uses the additional (unavailable) information in the distribution of true errors. Any distribution of true errors can be realized in a case with independent hypotheses. We have shown that in such cases this uncomputable bound is asymptotically equal to actual generalization error. Hence this is the tightest possible bound, up to asymptotic equality, over all bounds expressed as functions of $\hat{e}(h^*)$ and the distribution of true errors. We have also shown that the use of the histogram of empirical errors results in a bound that, while still tighter than traditional bounds, is looser than the uncomputable bound even in the large sample asymptotic limit.

One of the goals of learning theory is to give generalization guarantees that are predictive of actual generalization error. It is well known that the actual generalization error can exhibit phase transitions — as the sample size increases the expected generalization error can jump essentially discontinuously in sample size. So accurate true error bounds should also exhibit phase transitions. Shell bounds exhibit these phase transitions while other bounds such as VC dimension results do not.

The phase transitions can also be interpreted as a statement about the bound as a function of the confidence parameter δ. As the value of δ is varied the bound may shift essentially discontinuously. To put this another way, let h^* be the hypothesis of minimal training error on a large sample. Near a phase transition in true generalization error (as opposed to a phase transition in the bound) we may have that with probability $1-\delta$ the true error of h^* is near its training error but with probability $\delta/2$, say, the true error of h^* can be far from its training error. More traditional bounds do not exhibit this kind of sensitivity to δ. Bounds that exhibit phase transitions seem to bring the theoretical analysis of generalization closer to the actual phenomenon.

Acknowledgments

Yoav Freund, Avrim Blum, and Tobias Scheffer all provided useful discussion in forming this paper.

References

P. Bartlett, O. Bousquet and S. Mendelson. Localized Rademacher complexities. *Proceedings of the 15th Annual Conference on Computational Learning Theory*, pp. 44-58, (2002).

H. Chernoff. A measure of asymptotic efficiency for test of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493-507, 1952.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*, Wiley, 1991.

Y. Freund. Self bounding algorithms. *Computational Learning Theory (COLT)*, 1998.

D. Haussler, M. Kearns, H. S. Seung, and N. Tishby. (1996). Rigorous learning curve bounds from statistical mechanics. *Machine Learning* 25, pp. 195-236, 1996.

J. Langford. Practical prediction theory for classification, ICML 2003 tutorial, avaliable at http://hunch.net/ jl/projects/prediction_bounds/tutorial/tutorial.ps.

J. Langford. Quantitatively tight sample complexity bounds. Ph.D. Thesis, Carnegie Mellon, 2002.

J. Langford and A. Blum, Microchoice and self-bounding algorithms. *Computational Learning Theory (COLT)*, 1999.

Y. Mansour and D. McAllester. Generalization bounds for decision trees. *Computational Learning Theory (COLT)*, 2000.

A. Moore. VC dimension for characterizing classifiers. Tutorial at http://www-2.cs.cmu.edu/ awm/tutorials/vcdim08.pdf.

D. McAllester. PAC-Bayesian model averaging. *Computational Learning Theory (COLT)*, 1999.

D. McAllester and R. Schapire. On the convergence rate of good-Turing estimators. *Computational Learning Theory (COLT)*, 2000.

T. Scheffer and T. Joachims. Expected error analysis for model selection. *International Conference on Machine Learning (ICML)*, 1999.

S. van de Geer. *Empirical Process in M-Estimation*. Cambridge University Press, 1999.

Journal of Machine Learning Research 5 (2004) 549-573 Submitted 12/03; Revised 4/04; Published 5/04

Exact Bayesian Structure Discovery in Bayesian Networks

Mikko Koivisto MIKKO.KOIVISTO@CS.HELSINKI.FI
HIIT Basic Research Unit
Department of Computer Science
University of Helsinki
FIN-00014 Helsinki, Finland

Kismat Sood KISMAT.SOOD@KTL.FI
National Public Health Institute
Department of Molecular Medicine
FIN-00251 Helsinki, Finland

Editor: David Maxwell Chickering

Abstract

Learning a Bayesian network structure from data is a well-motivated but computationally hard task. We present an algorithm that computes the exact posterior probability of a subnetwork, e.g., a directed edge; a modified version of the algorithm finds one of the most probable network structures. This algorithm runs in time $O(n2^n + n^{k+1}C(m))$, where n is the number of network variables, k is a constant maximum in-degree, and $C(m)$ is the cost of computing a single local marginal conditional likelihood for m data instances. This is the first algorithm with less than super-exponential complexity with respect to n. Exact computation allows us to tackle complex cases where existing Monte Carlo methods and local search procedures potentially fail. We show that also in domains with a large number of variables, exact computation is feasible, given suitable a priori restrictions on the structures; combining exact and inexact methods is also possible. We demonstrate the applicability of the presented algorithm on four synthetic data sets with 17, 22, 37, and 100 variables.

Keywords: complex interactions, dynamic programming, layering, structure learning

1. Introduction

Structure discovery in Bayesian networks has attracted a great deal of research over the last decade. A Bayesian network specifies a joint probability distribution of a set of random variables in a structured fashion. A key component in this model is the network structure, a directed acyclic graph on the variables, encoding a set of conditional independence assertions. Learning unknown dependencies from data is motivated by a broad collection of applications in prediction and inference (Heckerman et al., 1995b).

Bayesian methods for structure learning concern the posterior distribution of network structures. Different applications require different types of posterior summaries—which are usually hard to compute. For example, when interest is in the prediction of future observations, one ought to integrate over the posterior distribution in the manner of model averaging. When the interest is purely inferential, then one may search for structures, or local structural features, that are highly probable. Since the seminal works of Buntine (1991) and Cooper and Herskovits (1992) numerous algorithms have been presented for these structure learning tasks. However, because the number of possible

structures grows super-exponentially with respect to the number of network variables, exact computations are often found infeasible. Indeed, it is known that finding an optimal Bayesian network structure is NP-hard even when the maximum in-degree is bounded by a constant greater than one (Chickering et al., 1995). Consequently, much of the research has focused on inexact methods.

To find a good or an optimal network structure, various generic heuristics, like stochastic local search and genetic algorithms, have been used (see, e.g., Heckerman et al., 1995a; Larrañaga et al., 1996). These methods can be also extended to find equivalence classes of network structures (for recent advances, see Chickering, 2002; Acid and de Campos, 2003; Castelo and Kočka, 2003). A central problem in all these algorithms is that one cannot guarantee the quality of the output. Also, the time requirement, though often practical, may be difficult to estimate beforehand.

Discovering high-probable structural features has not been studied so extensively. Madigan and York (1995) propose a Markov chain Monte Carlo (MCMC) method in the space of network structures. Friedman and Koller (2003) design a more efficient MCMC procedure in the space of variable orders. These algorithms output approximate posterior probabilities of structural features. The approximation quality is not guaranteed in finite runs.

Exact algorithms for structure learning have been presented for very restricted classes of Bayesian networks only. The algorithm by Cooper and Herskovits (1992) is polynomial in the number of variables, but a consistent ordering of the variables is assumed as an input. Chow and Liu (1968) give an efficient algorithm for learning tree structures (i.e., maximum in-degree is one). However, the most efficient exact algorithms, thus far, that allow for arbitrary network structures have super-exponential time complexity (Cooper and Herskovits, 1992; Friedman and Koller, 2003). Consequently, they can be applied only when the number of variables is very small (say, at most 10). Interestingly, however, Chickering (2002) shows that under certain monotonicity assumptions, a greedy search method will find a so called inclusion optimal network structure when the data size approaches infinity—but the time requirement can be exponential.

In this paper, we propose a novel *exact* algorithm for structure discovery in Bayesian networks of a moderate size (say, 25 variables or less). In fact, we consider two versions of the algorithm: one for computing posterior probabilities of structural features; another for finding an optimal structure. We also present a rigorous complexity analysis of the algorithm. This work is motivated by three serial observations summarized below.

First, we can expect that existing methods fail in cases where the posterior "landscape" of network structures is highly complex, including multiple "modes". This is the case especially when the underlying dependencies show no marginal signals. That is, the edges cannot be detected based on pairwise correlations (which may be all zero). Then it is necessary to consider all possible local dependency structures. This motivates the efforts to extend the scope of exact computation.

Second, it turns out that there are essentially two parallel contributions to the complexity of exact computation. One is due to consideration of all possible local dependency structures in light of the data. This part is unavoidable and may, in practice, actually dominate the overall complexity. Additively to this, another contribution is due to exploration of all graph structures. Although this seems to take more than polynomial time, it turns out that this part does not depend on the size of the data nor the complexity of local models. Thus, for a sufficiently small number of network variables, we really can afford exact exploration through all network structures.

The third observation is that the existing exact algorithms (Cooper and Herskovits, 1992; Friedman and Koller, 2003) can be improved significantly. In particular, we present the first algorithm

that averages (or alternatively maximizes) over all network structures in less than super-exponential time.

The remainder of this paper is organized as follows. In Section 2, we recall the ingredients of Bayesian networks and the task of structure discovery. Following the work of Buntine (1991), Cooper and Herskovits (1992), and Friedman and Koller (2003) we, in Section 3, describe the idea of conditioning by orders. Based on this, Section 4 presents a novel algorithm for averaging over all network structures. In Section 5, we modify this algorithm to handle the maximization problem. Possible extensions for large networks are sketched in Section 6. In Section 7, we provide experimental results on four synthetic data sets. Finally, Section 8 concludes with a brief discussion.

2. Preliminaries

We define a Bayesian network as a probability model for a vector $x = (x_1, \ldots, x_n)$ of random variables. Throughout this paper, we work on the index set $V = \{1, \ldots, n\}$ rather than on the set of the random variables. If $S = \{i_1, \ldots, i_r\}$ is a subset of V with $i_1 < \cdots < i_r$, we let x_S denote the vector $(x_{i_1}, \ldots, x_{i_r})$. A *graph structure* of a Bayesian network specifies conditional independencies among the variables. We represent the graph structure as a vector $G = (G_1, \ldots, G_n)$, where each G_i is a subset of V and specifies the *parents* x_{G_i} of x_i. Only *acyclic graphs* are valid, so that given a graph structure, the probability of x is composed by *local conditional distributions* $p(x_i \mid x_{G_i}, \theta)$ as

$$p(x \mid G, \theta) = \prod_{i=1}^{n} p(x_i \mid x_{G_i}, \theta). \tag{1}$$

Usually local distributions are of some common parametric form, such as Bernoulli or linear Gaussian. Therefore we here include the *parameters* θ explicitly in the conditional part. A *Bayesian network* is specified by a graph structure and a collection of associated conditional distributions.

Bayesian networks can be used to model multiple vectors $x[1], \ldots, x[m]$, henceforth called *data*. Throughout this paper we assume that the data is *complete*, i.e., there are no missing (unobserved) values. To incorporate the idea of learning from data, the vectors are judged to be *exchangeable*[1] (but not independent) so that the probability of the data, given the graph structure, can be expressed as

$$p(x[1], \ldots, x[m] \mid G) = \int_{\Theta} p(\theta \mid G) \prod_{t=1}^{m} p(x[t] \mid G, \theta)\, d\theta. \tag{2}$$

Here each term $p(x[t] \mid G, \theta)$ decomposes as in (1). Thus, when modeling multiple vectors in this way, a collection of Bayesian networks parametrized by $\theta \in \Theta$ is considered and a *prior* distribution $p(\theta \mid G)$ is assigned to the parameters. A natural application for (2) is the prediction of future events based on past observations.

Henceforth, we denote all the data briefly by x. Accordingly, for $i \in V$ we let x_i denote the vector $(x_i[1], \ldots, x_i[m])$.

When the graph structure is not known, it is subject to learning. After introducing a prior on the graph structures, we can write

$$p(x) = \sum_{G} p(G)\, p(x \mid G).$$

1. The data is judged to be part of an infinite exchangeable sequence.

This gives the distribution of data which can be used, e.g., in prediction tasks. Unfortunately, averaging over all graph structures is notoriously hard, as the number of possible graphs is found to be at least $2^{\binom{n}{2}}$ and thus super-exponential in the number of variables. Also, bounding the in-degree, i.e., the number of parents, per each variable by a constant k does not help much, a lower bound still being at least $2^{kn \log n}$ (for large enough n).

Sometimes, the interest is in the graph structure as such. Then it is appropriate to consider the posterior distribution of the graph structures, which by Bayes rule is given by

$$p(G \mid x) = \frac{p(G)\, p(x \mid G)}{p(x)}\,.$$

Various computational methods have been dedicated to the problem of finding a plausible graph structure. Unfortunately, finding a structure that maximizes the posterior probability,

$$\hat{G} \in \arg\max_G p(G \mid x)\,,$$

is known to be NP-hard in general (Chickering et al., 1995). But there are also statistical limits for this approach. Namely, searching for a single structure may not be most relevant, since exponentially many graph structures can be almost equally probable in light of modest amount of data.

An alternative approach is to compute local summaries of the posterior distribution of the graph structures. In restricted forms this idea appears already in the works of Buntine (1991) and Cooper and Herskovits (1992) but is significantly further developed by Friedman and Koller (2003). Friedman and Koller consider several types of local structural features, such as edges and Markov-blankets. More precisely, if f is the indicator of a structural feature, we are interested in the posterior

$$p(f \mid x) = \sum_G p(G \mid x)\, f(G)\,.$$

Here the indicator f is supposed to take value 1 if the feature is present and 0 otherwise. By turning to a clever decomposition of the space of possible graph structures, Friedman and Koller find an efficient MCMC method to estimate the above sum. We next review some key parts of their approach.

3. Conditioning on Orders

There is an efficient way to sum over an exponential number of graph structures that are consistent with a fixed order of variables. The key insight of Friedman and Koller (2003) is to use this result, originally due to Buntine (1991), to average over graph structures: they integrate over possible orders by MCMC. We next review the idea of conditioning on orders; MCMC methods will be only briefly mentioned in Section 6.

We define an *order* of variables as a total order on the index set V. We represent an order $\prec$ as a vector $(U_1, \ldots, U_n)$, where U_i gives the predecessors of i in the order, i.e.,

$$U_i = \{\, j \in V : j \prec i \,\}\,.$$

We say that a graph structure $(G_1, \ldots, G_n)$ is *consistent* with an order $(U_1, \ldots, U_n)$, denoted $G \subseteq \prec$, if $G_i \subseteq U_i$ for all i. Thus, the structures that are consistent with a fixed order form a subset of the

set of directed acyclic graphs. Note that for different orders, these subsets overlap. Actually, from this we get a fairly tight upper bound $n!\,2^{\binom{n}{2}}$ for the number of acyclic graphs. Asymptotically this is $o((2+\varepsilon)^{\binom{n}{2}})$ for any fixed $\varepsilon > 0$, and gives thus a much tighter characterization than the usual bounds $3^{\binom{n}{2}}$ or $2^{\Theta(n^2)}$ (see, e.g., Friedman and Koller, 2003).

Friedman and Koller (2003) observe that, from the computational point of view, it is advantageous to treat different variable orders as mutually exclusive events. While this is somewhat unnatural, since the corresponding sets of consistent graphs are overlapping, this approach is mathematically valid. Thus, in what follows, a graph structure alone does not determine whether an order is present or not. Therefore, we augment the prior of graph structures to a joint prior on orders and graphs. We also assume some fairly standard modularity assumptions stated below; related definitions are given, e.g., by Cooper and Herskovits (1992) and Friedman and Koller (2003).

Definition 1 *We say that a Bayesian network model p is* modular over $\prec, G, \theta$ and x, *or simply* order-modular *or* modular, *if the following properties hold:*

(M1) *If G is consistent with an order $\prec$, then*

$$p(\prec, G) = c \prod_{i=1}^{n} q_i(U_i)\, q'_i(G_i)\,,$$

where q_i and q'_i are probability distributions on the subsets of $V - \{i\}$ for each i, and c is a normalization constant. Otherwise, if $\prec$ is not a total order or if G is not consistent with $\prec$, then $p(\prec, G) = 0$.

(M2) *Given a structure G, the parameters θ decompose into $(\theta_{1,G_1}, \ldots, \theta_{n,G_n})$ such that*

$$p(\theta \mid G) = \prod_{i=1}^{n} p(\theta_{i,G_i} \mid G_i)\,,$$

and $p(x_i \mid x_{G_i}, \theta) = p(x_i \mid x_{G_i}, \theta_{i,G_i})$ for all i.

We note that while the above described augmentation is convenient, the semantics of the variable order becomes somewhat strange, thus making the elicitation of the prior distribution potentially difficult. However, one can avoid introducing a joint prior if one agrees with the resulting marginal prior distribution, $p(G)$, on graph structures. In that case the role of orders and associated probability distributions is technical. It is important to note that, in general, we do not have $p(G_i) = q'_i(G_i)$. Namely the prior $p(G_i)$ favors sets G_i that are small and thus consistent with many orders. Yet, it is easily seen that we have the following simple transform when conditioning by an order.

Proposition 2 *If p is modular and G is a graph structure consistent with an order $\prec$, then*

$$p(G \mid \prec) = \prod_{i=1}^{n} p(G_i \mid U_i)\,,$$

where $p(G_i \mid U_i) = q'_i(G_i) / \sum_{G'_i \subseteq U_i} q'_i(G'_i)$.

We give two examples to elucidate the relationship between orders and graphs. First, let each q_i and q'_i be uniform. Then it is not difficult to conclude that $p(G_i \mid U_i) = 2^{-|U_i|}$ and that $p(\prec) = 1/n!$. However, we note that, with this choice, the distribution of the number of parents is not uniform. In our second example q_i is again uniform, however, we set $q'_i(G_i)$ to be proportional to $1/\binom{n-1}{|G_i|}$ for parents G_i with cardinality at most the maximum in-degree. This assignment is natural when different cardinalities of parents are judged to be uniformly distributed. Note, however, that averaging over orders renders the marginal distribution of the cardinality $|G_i|$ slightly biased from uniform, since small cardinalities are favored as they are consistent with more orders.

While Proposition 2 above essentially follows from property (M1) of Definition 1, another appealing consequence of modularity is due to property (M2). Namely, the distribution of the data remains factorized when the parameters θ are marginalized out. A more precise statement is given below; the proof is simple and standard, and therefore omitted.

Proposition 3 *If p is modular, x is complete, and G is a graph structure, then*

$$p(x \mid G) = \prod_{i=1}^{n} p(x_i \mid x_{G_i}) \,,$$

where $p(x_i \mid x_{G_i}) = \int p(\theta_{i,G_i} \mid G_i)\, p(x_i \mid x_{G_i}, \theta_{i,G_i})\, d\theta_{i,G_i}$.

We proceed to consider probabilities of local features. Here we restrict our attention to modular features.

Definition 4 *A mapping f from graph structures onto $\{0,1\}$ is called* modular *if $f(G) = \prod_{i=1}^{n} f_i(G_i)$ where each f_i is a mapping from the subsets of $V - \{i\}$ onto $\{0,1\}$.*

For example, the indicator of a directed edge between two nodes is clearly modular. Also, the constant functions 1 and 0 are trivially modular. More generally, the indicator of any subgraph (a directed acyclic graph on a subset of V) is modular.

The key observation is that the summation over graph structures decomposes into a product of "local summations" (Buntine, 1991; Friedman and Koller, 2003).

Theorem 5 *If p and f are modular, x is complete, and $\prec = (U_1, \ldots, U_n)$ is a variable order, then*

$$p(x, f \mid \prec) = \prod_{i \in V} \sum_{G_i \subseteq U_i} p(G_i \mid U_i)\, p(x_i \mid x_{G_i})\, f_i(G_i) \,.$$

Proof Using first the marginalization and chain rules of probability, and then Proposition 2, Proposition 3, and Definition 4 to the three terms, respectively, we get

$$\begin{aligned} p(x, f \mid \prec) &= \sum_{G} p(G \mid \prec)\, p(x \mid G, \prec)\, p(f \mid x, G, \prec) \\ &= \sum_{G_1 \subseteq U_1} \cdots \sum_{G_n \subseteq U_n} \prod_{i=1}^{n} p(G_i \mid U_i)\, p(x_i \mid x_{G_i})\, f_i(G_i) \,, \end{aligned}$$

which factorizes into the desired product. ■

The posterior of the feature is obtained via Bayes rule:

$$p(f \mid x, \prec) = p(x, f \mid \prec) / p(x \mid \prec) \,.$$

Note that the probability of the data, $p(x \mid \prec)$ can be represented as $p(x, f' \mid \prec)$ where feature f' is the constant function 1.

Finally, once having the conditional probabilities for the feature, the unconditional posterior is obtained as

$$p(f \mid x) = \sum_{\prec} p(\prec \mid x)\, p(f \mid x, \prec)\,. \tag{3}$$

Here $\prec$ runs through all orders on the set V. There are $n!$ different orders, which is still super-exponential with respect to n. Yet, this is much less than the number of all possible graph structures.

Friedman and Koller (2003) propose an MCMC method to estimate sum (3) by drawing a sample of orders from the posterior $p(\prec \mid x) \propto p(\prec)\, p(x \mid \prec)$. They argue that this approach is more efficient than MCMC directly in the space of graph structures (Madigan and York, 1995).

It is important to note that, despite of the closed form expression of Theorem 5, the computations, given an order, may be quite expensive. Namely, one has to consider all possible sets of parents for each variable. For a fixed maximum number of parents, k, roughly n^{k+1} terms $p(x_i \mid x_{G_i})$, henceforth called *local conditional marginals*, need to be computed. Depending on the data size, on the functional forms of the local conditional distributions, and on the value k, this may contribute significantly to the total computational complexity. Henceforth we suppose that any local term $p(x_i \mid x_{G_i})$ can be computed in time $O(C(m))$, where C is a function of data size. For example, when a local conditional distributions is taken from the exponential family with appropriate conjugate priors, then the associated $C(m)$ is linear in m. For more structured models, e.g., decision graphs (Chickering et al., 1997), the complexity of computing local conditional marginals can be significantly greater, yet often linear in m.

4. Summation by Dynamic Programming

We next show how the summation over orders can be carried out in roughly $n2^n$ operations, which grows much slower than $n!$. This improved requirement should be contrasted with the lower bound n^{k+1} (with a potentially large constant factor). For moderate n (say, $n < 20$) and relatively large k (say, $k = 5$), the total complexity may be dominated by the polynomial term. Yet, the computations are feasible on modern computers.

We consider a summation that is slightly different from (3). We write

$$p(f \mid x) = p(f, x)/p(x)\,, \tag{4}$$

and continue by considering evaluation of

$$p(f, x) = \sum_{\prec} p(\prec)\, p(f, x \mid \prec)\,. \tag{5}$$

Note that $p(x)$ is of the same form.

In order to facilitate the forthcoming development, we define for each $i \in V$ a function α_i as follows. Let i be an element of V and let S be a subset of V that does not contain i. We define

$$\alpha_i(S) = \sum_{G_i \subseteq S} q_i'(G_i)\, p(x_i \mid x_{G_i})\, f_i(G_i)\,. \tag{6}$$

In essence, the function α_i gives the contribution of the ith local component (x_i and its unknown parents) to sum (5) above; notice the similarity to the terms in Theorem 5. The functions α_i serve

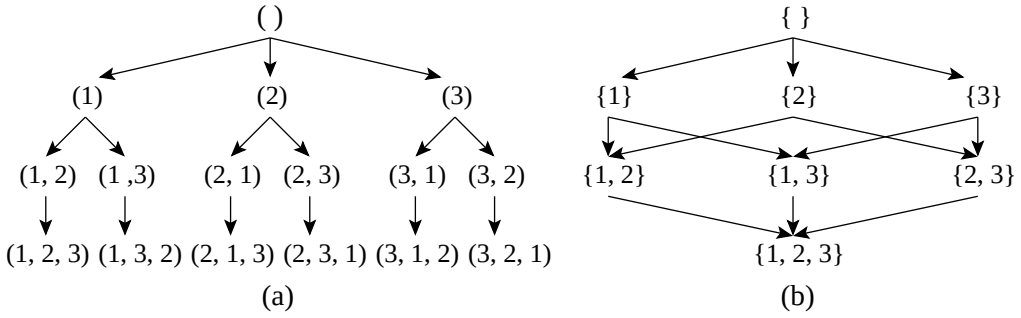


Figure 1: Illustrations of (a) the permutation tree and (b) the subset lattice of $\{1,2,3\}$. The nodes of the permutation tree are labeled by the corresponding paths from the root. (The labels of the edges are not shown but can be easily deduced.)

as a way to split the problem of evaluating sum (5) into two steps. The first is to compute these functions given a feature f and a data set x. The second task is to compute sum (5), given the functions α_i. We first consider the latter problem assuming that the functions α_i are precomputed; we will later come back to the former problem.

It turns out that the sum over orders is advantageous to compute in a manner of variable elimination. Here variables refer to the n elements $\sigma_1,\ldots,\sigma_n \in V$, where σ_j is the jth element in the order. A key observation is that when considering the parents of variable x_{σ_j} it is sufficient to know the unordered set $\{\sigma_1,\ldots,\sigma_{j-1}\}$ of the possible parents. In other words, the order of the possible parents is irrelevant.

One way to view this reduction from ordered sets to unordered sets is to consider a *permutation tree*. A permutation tree of $V=\{1,\ldots,n\}$ is a rooted (directed) tree with n levels. Any node at the hth level has $n-h$ children labeled by distinct elements from V, so that the labels on any (directed) path are all distinct. Thus, a path from the root to a leaf corresponds to a unique permutation on V (see Figure 1(a)). Evaluation of the sum over orders is carried out by a propagation algorithm, where each node obtains a value by summing up the values of its parents, each multiplied by a quantity that depends on the associated path (details will be given soon). However, apart from the last label of the path, only the unordered set of the labels matters. This means that computations over different paths can be merged. Graphically, the permutation tree collapses to a *subset lattice*. A subset lattice of V is a graph, where nodes corresponds to subsets of V and there is an edge from A to B if and only if $B=A\cup\{i\}$ for some $i\notin A$ (see Figure 1(b)).

We summarize the above discussion more formally:

Proposition 6 *If p and f are modular and x is complete, then*

$$p(f,x) = c\,g(V)\,,$$

where c is the normalizing constant and g is defined for all subsets S of V recursively by

$$g(S) = \sum_{i\in S} q_i(S-\{i\})\,\alpha_i(S-\{i\})\,g(S-\{i\}) \tag{7}$$

with boundary $g(\emptyset) = 1$.

Proof By simple induction we get that

$$g(V) = \sum_{\sigma_1,\ldots,\sigma_n} \prod_{j=1}^{n} q_i(S_j)\, \alpha_{\sigma_j}(S_j)\,,$$

where $(\sigma_1,\ldots,\sigma_n)$ runs through all permutations on V, and $S_j = \{\sigma_1,\ldots,\sigma_{j-1}\}$. Thus, each $(\sigma_1,\ldots,\sigma_n)$ is a realization of an order $\prec = (U_1,\ldots,U_n)$ with $\sigma_1 \prec \sigma_2 \prec \cdots \prec \sigma_n$ and $U_{\sigma_j} = \{\sigma_1,\ldots,\sigma_{j-1}\} = S_j$. By the modularity of p, we have

$$c \prod_{i=1}^{n} q_i(U_i)\, q'_i(G_i) = p(\prec)\, p(G \mid \prec)\,.$$

Thus, by (5), Proposition 2, and Theorem 5, we have $c\, g(V) = p(x, f)$. ■

This result gives us a relatively efficient way to sum over orders in the manner of dynamic programming (for a related algorithm, see Bellman, 1962). Provided that each value of the functions q_i, α_i, and g can be accessed in a constant (amortized) time, we have an $O(n2^n)$ time algorithm for computing the posterior $p(f \mid x)$. Note that the constant c cancels out in (4).

We now take a step back and consider the computation of the functions α_i. Since the representation of each function α_i already takes 2^{n-1} numbers, the best we can hope is to find an algorithm that runs in time $O(2^n)$. We next show how to achieve this optimal time complexity.[2]

Consider a fixed $i \in V$. It is convenient to define a new function, β_i, by

$$\beta_i(G_i) = q'_i(G_i)\, p(x_i \mid x_{G_i})\, f_i(G_i) \qquad \text{for all } G_i \subseteq V - \{i\}\,. \tag{8}$$

Hence, by the definition of α_i we have

$$\alpha_i(S) = \sum_{T \subseteq S} \beta_i(T) \qquad \text{for all } S \subseteq V - \{i\}\,.$$

The operator that in this way maps a function of subsets of $V - \{i\}$ to another such function is known as the *Möbius transform* (on the subset lattice of $V - \{i\}$). We say that α_i is the Möbius transform of β_i.

How fast can we evaluate the Möbius transform? In the following discussion we assume that the values of β_i are precomputed and can be accessed in a constant (amortized) time. The straightforward approach is to compute separately for each subset S the summation over its subsets. We see that one step takes $O(2^{|S|})$ time, and hence, the total time complexity is $O(3^n)$. To reduce this bound, we notice that $\beta_i(T)$ vanishes for all T with more than k elements. This yields the complexity $O(n^k 2^n)$. An alternative, and more efficient approach is to use the fast Möbius transform algorithm (see, e.g., Kennes and Smets, 1991). It takes advantage of the overlapping parts of the 2^n summations and runs in time $O(n2^n)$. However, it does not exploit the in-degree bound k. We next show that under this additional constraint we can still slightly reduce the time complexity, to the desired $O(2^n)$.

2. This bound is optimal up to a constant factor that may depend on the maximum in-degree k which is assumed to be constant.

FAST-TRUNCATED-MÖBIUS-TRANSFORM(h_0, k)
1 **for** $j \leftarrow 1$ **to** n **do**
2 **for** each $S \subseteq N$ with $|S \cap \{j+1,\ldots,n\}| \leq k$ **do**
3 $h_j(S) \leftarrow 0$
4 **if** $|S \cap \{j,\ldots,n\}| \leq k$ **then**
5 $h_j(S) \leftarrow h_{j-1}(S)$
6 **if** $j \in S$ **then**
7 $h_j(S) \leftarrow h_j(S) + h_{j-1}(S - \{j\})$
8 **return** h_n

Figure 2: An algorithm for evaluating the truncated Möbius transform on the subset lattice of $N = \{1,\ldots,n\}$. Input h_0 is the function to be transformed, and k is a number between 1 and n.

We extend the notion of Möbius transform by defining a truncated Möbius transform as a Möbius transform where the summation over subsets is restricted to subsets with at most k elements, where k is an additional input parameter. Figure 2 describes a general algorithm for computing truncated Möbius transforms on the subset lattice of $N = \{1,\ldots,n\}$. (When we apply this algorithm to compute a function α_i, we replace n by $n-1$.) The algorithm mainly follows the steps of the standard fast Möbius transform algorithm (see, e.g., Kennes and Smets, 1991) and splits the transform into n smaller transforms, each being a summation over the two subsets of a singleton $\{j\} \subseteq N$. These n transforms operate one after another, the jth transform to the result of the $(j-1)$th transform. This procedure can be also viewed as a variable elimination algorithm, where for each $\{j\}$ "its subset" is treated as a variable taking two values. In the fast truncated Möbius transform algorithm, described in Figure 2, this standard algorithm is modified so that each of the n transforms is evaluated only at as few subsets of N as needed. While the last transform has to be evaluated at all 2^n subsets, a polynomial number of evaluations is sufficient for the first transforms. Details of this discussion are given in the proof of the following result.

Proposition 7 *Algorithm* FAST-TRUNCATED-MÖBIUS-TRANSFORM *(Figure 2) with input* (h_0, k) *runs in time* $O(2^n)$ *for a constant* k *and computes the function* h_n*, given by*

$$h_n(S) = \sum_{T \subseteq S : |T| \leq k} h_0(T) \qquad \text{for all } S \subseteq N.$$

Proof We show by induction on j that $h_j(S)$ for $S \subseteq N$ with $|S \cap \{j+1,\ldots,n\}| \leq k$, computed at the jth iteration, is given by

$$h_j(S) = \sum_{T_1 \subseteq S_1} \cdots \sum_{T_j \subseteq S_j} 1(|T_{1,j} \cup S_{j+1,n}| \leq k)\, h_0(T_{1,j} \cup S_{j+1,n}), \tag{9}$$

where we denote $S_r = S \cap \{r\}$ and $S_{r,r'} = S_r \cup S_{r+1} \cup \cdots \cup S_{r'} = S \cap \{r, r+1, \ldots, r'\}$ (similarly for T_r and $T_{r,r'}$).

In the case $j = 0$ expression (9) trivially holds. We proceed by letting $j > 0$. Let $S \subseteq N$ with $|S \cap \{j+1,\ldots,n\}| \leq k$. We separate two cases, $S_j = \emptyset$ and $S_j = \{j\}$, and show that expression (9) holds in both cases.

First consider the case $S_j = \emptyset$. By the algorithm, after the jth step, we have

$$h_j(S) = h_{j-1}(S)\,.$$

Using the induction assumption (9) for h_{j-1}, it is easy to verify that $h_j(S)$ can be expressed as in the induction claim (9) since $T_j = S_j = \emptyset$.

Then consider the remaining case, $S_j = \{j\}$. After the jth step, we have

$$h_j(S) = 1(|S \cap \{j,\ldots,n\}| \leq k)\, h_{j-1}(S) + h_{j-1}(S - \{j\})\,.$$

The first term in the sum corresponds to the case $T_j = S_j = \{j\}$ in the summation (9). By the induction assumption (9) for h_{j-1}, it is sufficient to verify that

$$1(|S \cap \{j,\ldots,n\}| \leq k)\, 1(|T_{1,j-1} \cup S_{j,n}| \leq k) = 1(|T_{1,j} \cup S_{j+1,n}| \leq k)\,,$$

and that $T_{1,j-1} \cup S_{j,n} = T_{1,j} \cup S_{j+1,n}$. The latter is obvious since $T_j = S_j$. The former identity follows since $S \cap \{j,\ldots,n\} = S_{j,n}$. Note that the condition $|S \cap \{j,\ldots,n\}| \leq k$ ensures that h_j has been evaluated at S in the previous iteration.

The second term in the sum, $h_{j-1}(S - \{j\})$, corresponds to the case $T_j = \emptyset$. In this case, $|T_{1,j} \cup S_{j+1,n}| \leq k$ holds since we have assumed that $|S \cap \{j+1,\ldots,n\}| \leq k$. The argument of h_0 remains invariant because in expression (9) for $h_{j-1}(S - \{j\})$, we have $T_j = S_j = \emptyset$.

We have now shown that (9) holds for all $j = 1,\ldots,n$. Particularly, in the case $j = n$ we get the claimed transformation. This completes the first part of the proof.

We next show that the algorithm runs in time $O(2^n)$. For the steps $j = n-k+1,\ldots,n$ the required total number of operations is proportional to

$$\sum_{j=n-k+1}^{n} 2^j 2^{n-j} = k2^n = O(2^n)\,.$$

For the steps $j = 1,\ldots,n-k$ we get a bound

$$\sum_{j=1}^{n-k} 2^j \sum_{r=0}^{k} \binom{n-j}{r} \leq \sum_{j=1}^{n-k} 2^j (n-j)^k \leq 2^n \sum_{j=0}^{\infty} (1/2)^j j^k = O(2^n)\,,$$

since the infinite sum converges to a finite limit for a fixed k. Thus, the total running time is $O(2^n)$. This completes the second part of the proof. ■

It is possible to extend the above complexity result for k that is not a constant. In particular, for $k = n$ the presented algorithm obviously computes (ordinary) Möbius transform in time $O(n2^n)$. For general k, however, the presented complexity analysis is too rough to give a tight bound. We conjecture that the time complexity is $O(k2^n)$, but proving this would require a more detailed analysis beyond the scope of this discussion.

We finally summarize. By the fast truncated Möbius transform, the functions α_i for $i = 1,\ldots,n$ can be computed in total time $O(n2^n)$. This assumes that the functions β_i for $i = 1,\ldots,n$ are precomputed, which obviously can be done in $O(n^{k+1}C(m))$ time. Hence, we get our main result.

Theorem 8 *Let p and f be modular and let x be complete with m records. Assume that any local conditional marginal can be computed in time $O(C(m))$. Then for a constant in-degree bound k, the probability $p(f \mid x)$ can be evaluated in time $O(n2^n + n^{k+1}C(m))$.*

5. Finding One of the Most Probable Structures

We now turn to the problem of finding a single best graph structure. Since the algorithms of the previous section essentially exploit the distributive law of a sum-product semiring, it is not surprising that only slight modifications are needed for a max-product semiring. As various aspects of general semiring algorithms have been studied extensively in the last decade (e.g., Stearns and Hunt III, 1996; Lauritzen and Jensen, 1997; Dechter, 1999; Aji and McEliece, 2000), we here omit algorithmic details and focus on certain key points that are central from the modeling point of view, though algorithmically rather irrelevant.

We first observe that finding a maximizing pair

$$(\prec^*, G^*) \in \arg\max_{\prec, G} p(\prec, G \mid x)$$

would be rather straightforward based on certain modification in the summation algorithms described in the previous section. However, we are interested in finding a graph $\hat{G}$ that maximizes the marginal posterior $p(G \mid x)$. Note that this target function favors structures that are consistent with many (a priori probable) orders. It seems that we now have a somewhat harder task as we need to average over orders while maximizing over graphs.

Fortunately, there is a satisfactory solution to the computational problem stated above: we slightly change the problem. We specify a modular prior distribution directly on graph structures without augmenting the probability space by orders as was done in Definition 1. Actually, this has been a more common way to specify a prior on Bayesian network structures (Cooper and Herskovits, 1992; Heckerman et al., 1995a) than coupling with orders. The corresponding modularity property is as follows; related definitions are given, e.g., by Cooper and Herskovits (1992) and Friedman and Koller (2003).

Definition 9 *We say that a Bayesian network model p is* modular over G, θ and x, *or simply* graph-modular, *if (M2) of Definition 1 and (M1') below hold.*

(M1') *If G is acyclic, i.e., consistent with some order, then*

$$p(G) = c' \prod_{i=1}^{n} q_i''(G_i)\,,$$

where each q_i'' is a probability distribution over the subsets of $V - \{i\}$. Otherwise, if G is cyclic, then $p(G) = 0$. Here c' is a normalization constant.

We note that property (M1) in Definition 1 and property (M1') in Definition 9 imply slightly different forms of the structure prior $p(G)$. In particular, property (M1) is *not* a special case of property (M1'). Specifically, orders are now not treated as disjoint events, but instead, if a graph structure is consistent with two orders, then both are present. Thus, under (M1'), the probabilities $p(\prec)$ of different orders $\prec$, though well-defined, do not sum up to one. For a similar discussion, see Friedman and Koller (2003). In the maximization task we are considering, the advantage of (M1') over (M1) is that the former allows us to search for most probable graph structures in the joint space of orders and structures.

Proposition 10 *Let p be graph-modular and x complete. Let*

$$(\prec^*, G^*) \in \arg\max_{\prec, G} \left\{ \prod_{i=1}^{n} q_i''(G_i)\, p(x_i \mid x_{G_i}) \right\},$$

where $\prec$ runs through all total orders and G structures that are consistent with $\prec$. Then G^* maximizes the posterior $p(G \mid x)$.

Proof The product to be maximized is seen to be equal to $p(G \mid x)$ up to a constant factor. Since the joint space of orders and graph structures includes every directed acyclic graph, it also includes a graph that maximizes the posterior $p(G \mid x)$. ■

This simple observation gives us a way to find a graph structure that maximizes the posterior probability. Namely, it is rather straightforward to modify the summation algorithms given in the previous section to compute maximizing arguments instead; details are omitted. We have the following counterpart of Theorem 8.

Theorem 11 *Let p be graph-modular and let x be complete with m records. Assume that any local conditional marginal can be computed in time $O(C(m))$. Then for a constant in-degree bound k, a graph structure that maximizes the posterior probability can be found in time $O(n2^n + n^{k+1}C(m))$.*

We note that the idea of searching for an optimal order as a means for learning Bayesian networks is not new: Larrañaga et al. (1996) observe that this task resembles the Traveling Salesman Problem (TSP) and design a genetic algorithm for searching for good orders. However, they do not mention the exact dynamic programming solution that resembles the algorithm for the TSP due to Bellman (1962).

6. Discovering Larger Networks

We next sketch how the presented exact methods can be applied in cases where the number of variables is large, say $n > 30$. One possibility is to force additional restrictions on the space of structures. Another possibility is to resort to inexact techniques, such as MCMC and local search procedures. Though we in this section mainly consider the summation problem, modifications for the maximization problem are also possible in the same manner as discussed in Section 5.

6.1 Restrictions on Orders

It is fortunate that in some cases, where the number of variables is large, the modeler may have a strong prior on the graph structures. For example, some variables cannot have parents while some others cannot have children. Such knowledge may arise naturally when the direction of the edges are interpreted as the direction of causality (Heckerman et al., 1999; Pearl, 2000).

For a more general treatment, let $\{V_1, \ldots, V_\ell\}$ be a partition of the set $V = \{1, \ldots, n\}$. Recall that a total order on V corresponds to a permutation on V. Denote the set of all permutations on a set S by $\mathcal{S}(S)$. Now assume that prior knowledge justifies the restriction to the permutations in the Cartesian product

$$\mathcal{S}(V_1) \times \cdots \times \mathcal{S}(V_\ell) \subseteq \mathcal{S}(V)\,.$$

FEATURE-PROBABILITY-IN-LAYERED-NETWORK($V_1, \ldots, V_\ell$, q, β)
1 $g(\emptyset) \leftarrow 1$
2 **for** $h \leftarrow 1$ **to** ℓ **do**
3 **for** each $i \in V_h$ **do**
4 compute β'_i according to (10)
5 $\alpha'_i \leftarrow$FAST-TRUNCATED-MÖBIUS-TRANSFORM(β'_i, k)
6 **for** each nonempty $S \subseteq V_h$, in increasing order **do**
7 $g(S) \leftarrow \sum_{i \in S} q_i(S - \{i\})\, \alpha'_i(S - \{i\})\, g(S - \{i\})$
8 **return** $\prod_{h=1}^{\ell} g(V_h)$

Figure 3: An algorithm for computing the joint probability $p(f, x)$ up to a normalizing constant in a layered network.

That is, an edge from $i \in V_s$ to $j \in V_t$ is allowed only if $s \leq t$. This restricts us to a space of "layered" networks. On one extreme we get the set of all orders ($\ell = 1$), and on the other, we get a single order ($\ell = n$). A layering is a property induced by the prior on orders and graph structures. We say that a layering and a model p are *compatible* if the prior probability of any graph structure that violates the layering vanishes.

Fixing a layering structure may dramatically reduce the computational complexity of evaluating feature probabilities. We observe that it is sufficient to consider permutations on different layers V_h separately. Thus, the sum over orders on V factorizes into a product of sums of orders on each layer V_h. Perhaps a less immediate fact is that also the summations over different sets of parents can be handled efficiently. This is because the elements of $V_1 \cup \cdots \cup V_{h-1}$ are always possible parents of an $i \in V_h$ regardless of the order on V_h. To take advantage of this observation, we modify the mappings β_i defined in (8). For all subsets T of $V_h - \{i\}$ of size at most k we define

$$\beta'_i(T) = \sum_W \beta_i(T \cup W)\,, \tag{10}$$

where W runs through all subsets of the union $V_1 \cup \cdots \cup V_{h-1}$. Recall that $\beta_i(T \cup W) = 0$ when the size $|T \cup W|$ is larger than k.

Figure 3 displays an algorithm that exploits a layered network decomposition. The input consists of a partition $\{V_1, \ldots, V_\ell\}$ of $\{1, \ldots, n\}$, a modular prior on orders specified by $q = (q_1, \ldots, q_n)$, and a precomputed function $\beta = (\beta_1, \ldots, \beta_n)$ as defined in (8). Note that this β depends on the feature f. The algorithm outputs the proportional probability $p(f, x)/c$, where c is the normalizing constant (independent of f and x). Recall that this value is sufficient when computing posterior probabilities of features.

Theorem 12 *Let p and f be modular and let x be complete with m records. Assume p is compatible with a layering $\{V_1, \ldots, V_\ell\}$. Further assume that any local conditional marginal can be computed in time $O(C(m))$. Then for a constant in-degree bound k, the probability $p(f \mid x)$ can be evaluated in time $O(n2^{n_*} + n^{k+1}C(m))$, where n_* is the maximum of $|V_1|, \ldots, |V_\ell|$.*

Proof We first show that Algorithm FEATURE-PROBABILITY-IN-LAYERED-NETWORK, given in Figure 3, is correct, and then we analyze its time requirement.

Let i be an element of V_h and S a subset of $V_h - \{i\}$. By the definitions of the functions α_i, β_i, and β'_i, we have, after line 5,

$$\alpha'_i(S) = \sum_{T \subseteq S} \beta'_i(T) = \sum_{(T \cup W) \subseteq S \cup V_1 \cup \cdots \cup V_{h-1}} \beta_i(T \cup W) = \alpha_i(S \cup V_1 \cup \cdots \cup V_{h-1}) .$$

From this it is obvious that

$$\prod_{h=1}^{\ell} g(V_h) = \sum_{\sigma_1,\ldots,\sigma_n} \prod_{j=1}^{n} q_i(S_j)\, \alpha_{\sigma_j}(S_j) ,$$

where $(\sigma_1, \ldots, \sigma_n)$ runs through all permutations in the restricted set $\mathcal{S}(V_1) \times \cdots \times \mathcal{S}(V_\ell)$, and $S_j = \{\sigma_1, \ldots, \sigma_{j-1}\}$. Hence, by the arguments given in the proof of Proposition 6, we obtain $p(f, x) = c \prod_{h=1}^{\ell} g(V_h)$, as required.

We then analyze the complexity of the algorithm. Fix a level $h \in \{1, \ldots, \ell\}$. Denote $V' = V_1 \cup \cdots \cup V_h$. Line 4 can be performed in time $O(|V'|^k)$, since the summations (10) for different T include each subset $(T \cup W) \subseteq V'$ of size at most k just once. Line 5 can be performed in time $O(2^{|V_h|})$. Thus, the overall complexity of the for-loop starting in line 3 is $O(|V_h| 2^{|V_h|} + |V_h||V'|^k)$. The time complexity of lines 6–7 is clearly $O(|V_h| 2^{|V_h|})$.

Summing the complexities for $h = 1, \ldots, \ell$ gives the total time requirement of $O(n2^n + n^{k+1})$. The algorithm assumes that functions β_i are precomputed; this can be done in time $O(n^{k+1} C(m))$. Hence, combining these two bounds gives the claimed time complexity. ■

Sometimes domain knowledge may allow for further reduction in the problem complexity. As an example, consider the special case, where we know a priori that some variables cannot have parents while some others cannot have children. This can be expressed by a partition $\{V_1, V_2, V_3\}$ with three layers. We notice that, in this particular case, the orders on V_1 and V_3 are irrelevant, which reduces the computational complexity to $O(n2^{|V_2|} + n^{k+1} C(m))$. Thus, exact structure discovery remains practical even in cases where V_1 and V_3 are large.

6.2 Combining with Inexact Techniques

When prior knowledge cannot be used to restrict the space of graph structures, one has to resort to inexact methods. Currently the most efficient general method is perhaps the MCMC algorithm by Friedman and Koller (2003). It draws a sample from the posterior distribution of orders and estimates feature probabilities by sample averages. It can be fairly easily modified to search for most probable graphs, e.g., by using simulated annealing techniques. Moreover, as noted by Friedman and Koller, MCMC can be naturally extended to deal with data sets where some of the data is missing.

Based on the concept of layering, introduced in Section 6.1, we propose an extension of Friedman and Koller's method. Instead of sampling total orders on V, it might be advantageous to sample partitions $\{V_1, \ldots, V_\ell\}$, where the number of subsets ℓ is fixed and the subsets are of an equal size $r = n/\ell$. This reduces the size of the sample space by the factor $(r!)^{n/r}$. Still, for relatively small r, say $r = 10$, the computational cost per sampled partition is relatively low.

It is known that reducing the state space—sometimes called Rao-Blackwellization (Gelfand and Smith, 1990)—is generally a good idea to boost sampling methods, provided that the resulting

computational overhead is relatively small. Namely, merging states may result in a smoother, and thus easier, "landscape" with fewer local maxima (with respect to the neighborhood induced by the algorithm). Friedman and Koller (2003) showed that orders are preferred over graphs. We believe that, likewise, layerings are preferred over orders. However, validating this conjecture requires a dedicated study which is beyond the scope of this paper.

7. Experimental Results

We have implemented the algorithms described in this paper. Our implementation is written in the C++ programming language. The experiments to be described next were run under Linux on an ordinary desktop PC with a 2.4GHz Pentium processor and 1.0GB of memory.

The objectives of these experiments are threefold. First, we generate and analyze data sets that, we believe, might be hard to analyze by inexact methods. For comparison, some results for a small sample of existing heuristic algorithms are also presented. Second, we illustrate the summation and maximization tasks in structure discovery from the methodological point of view. This includes exemplification of the layering method described in the previous section. Third, we demonstrate that exact computations are feasible for relatively large networks. This includes measurements of exponential and polynomial contributions to the overall time requirement.

7.1 Data Sets

We have tested the summation and maximization algorithms on a small selection of data sets. The data sets contain discrete variables only and no values are missing. The *Zoo* data set is available from the UCI Machine Learning Repository (Blake and Merz, 1998, the data set contributed by Richard Forsyth). It contains 17 variables and 101 records. The *Alarm* data set built by (Herskovits, 1991) contains 37 variables and 20000 records, generated from the Alarm Monitoring System (Beinlich et al., 1989). In our experiments we used the first 100, 500, and 2500 records from the original *Alarm* data set.

We also generated two new data sets. Both employ the binary parity function. The *Parity1* data set contains 22 binary variables and 2500 records. We generated the data from a Bayesian network with structure as in Figure 4(a). The maximum in-degree is 4. The local distributions for each variable x_i given its parents x_{G_i} were specified as follows. For each variable we associate a fixed noise rate $\varepsilon_i \in [0,1]$. For each record $x[t]$ we set $x_i[t] = \mathrm{xor}(x_{G_i}[t])$ with probability $1-\varepsilon_i$ and $x_i[t] = 1 - \mathrm{xor}(x_{G_i}[t])$ with probability ε_i. Here the parity function $\mathrm{xor}(z_1,\ldots,z_h)$ returns 1 if the sum $z_1 + \cdots + z_h$ is odd and 0 if it is even. The idea behind using the parity function here is, of course, that no subset of the correct set of parent variables should give any hint about the state of the child. In structure learning, incremental construction of the parent set should be unstable if not impossible, thus requiring an exhaustive search. However, following this idea fully would require that value configurations of the parents of a node are uniformly distributed and that no node has just one parent. These conditions are not met in *Parity1* which makes learning somewhat easier—thus serving as a more realistic example. The *Parity2* data set extends this idea to 100 variables with a specific layering topology. We arranged 100 variables into two layers, one with 78 and the other with 22 variables, satisfying the following two constraints: (i) there are no edges within the first layer (78 variables); (ii) the edges between the layers are directed from the first layer to the second layer. Here we used the maximum in-degree 3. From the specified Bayesian network 2500 records

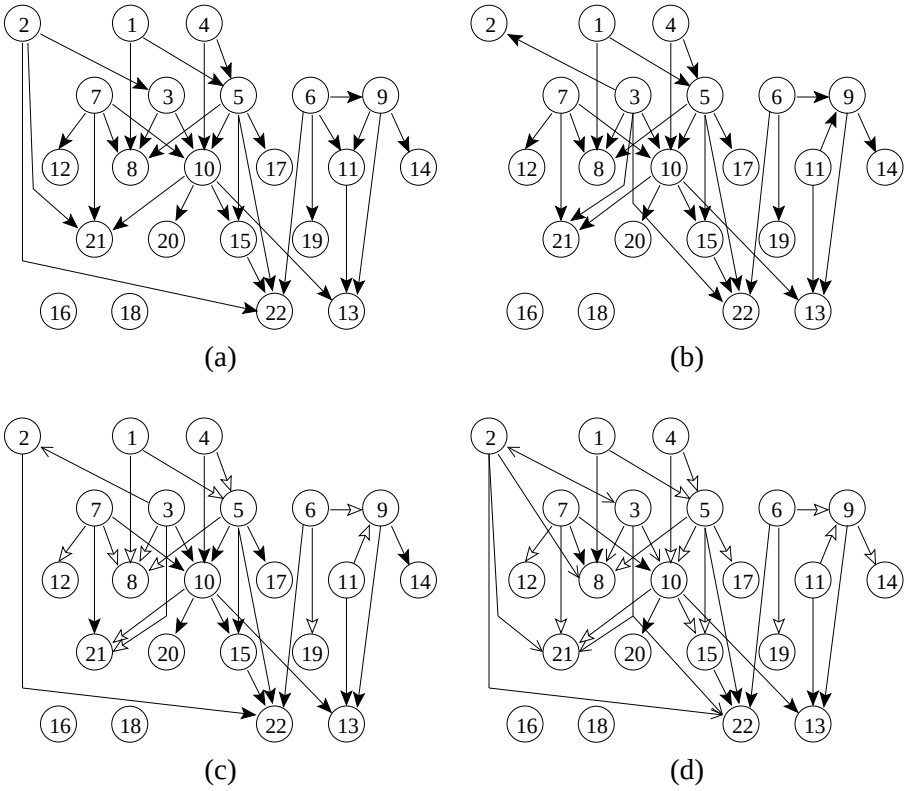


Figure 4: Discovering network structure for *Parity1*. (a) The correct structure. (b) A structure that maximizes the posterior probability with 500 records. Posterior probabilities of edges with (c) 2500, and (d) 500 records. In (c) and (d) arrow heads ▲, △, and ∧ correspond to intervals $[0.99, 1.00]$, $[0.67, 0.99)$, and $[0.33, 0.67)$, respectively.

were generated. In our experiments we also used prefixes of sizes 100 and 500 of the *Parity1* and *Parity2* data sets.

7.2 Model Specifications for Learning

In our experiments we used a simple generic Bayesian network model for structure discovery on the selected data sets. We used different values of the maximum in-degree for different data sets. For *Zoo*, *Alarm*, *Parity1*, and *Parity2* the values were 6, 4, 5, and 3, respectively. Note that in the *Alarm* network (Figure 5(a)) the in-degree of node 27 is 4. We set $q_i(U_i)$ to be uniform and $q'_i(G_i)$ to be proportional to $1/\binom{n-1}{|G_i|}$ (for $|G_i| \leq k$). In the case of maximization we used the same distribution on parents, that is, $q''_i(G_i) = q'_i(G_i)$. Since all variables are discrete, the local conditional distributions are multinomial. For the multinomial parameters we assigned a Dirichlet prior with hyperparameters set to 1. Note that a Dirichlet prior yields a closed-form solution to the local conditional marginal distributions. Hence, the complexity $C(m)$ of computing such a marginal is linear in the data size m (Cooper and Herskovits, 1992).

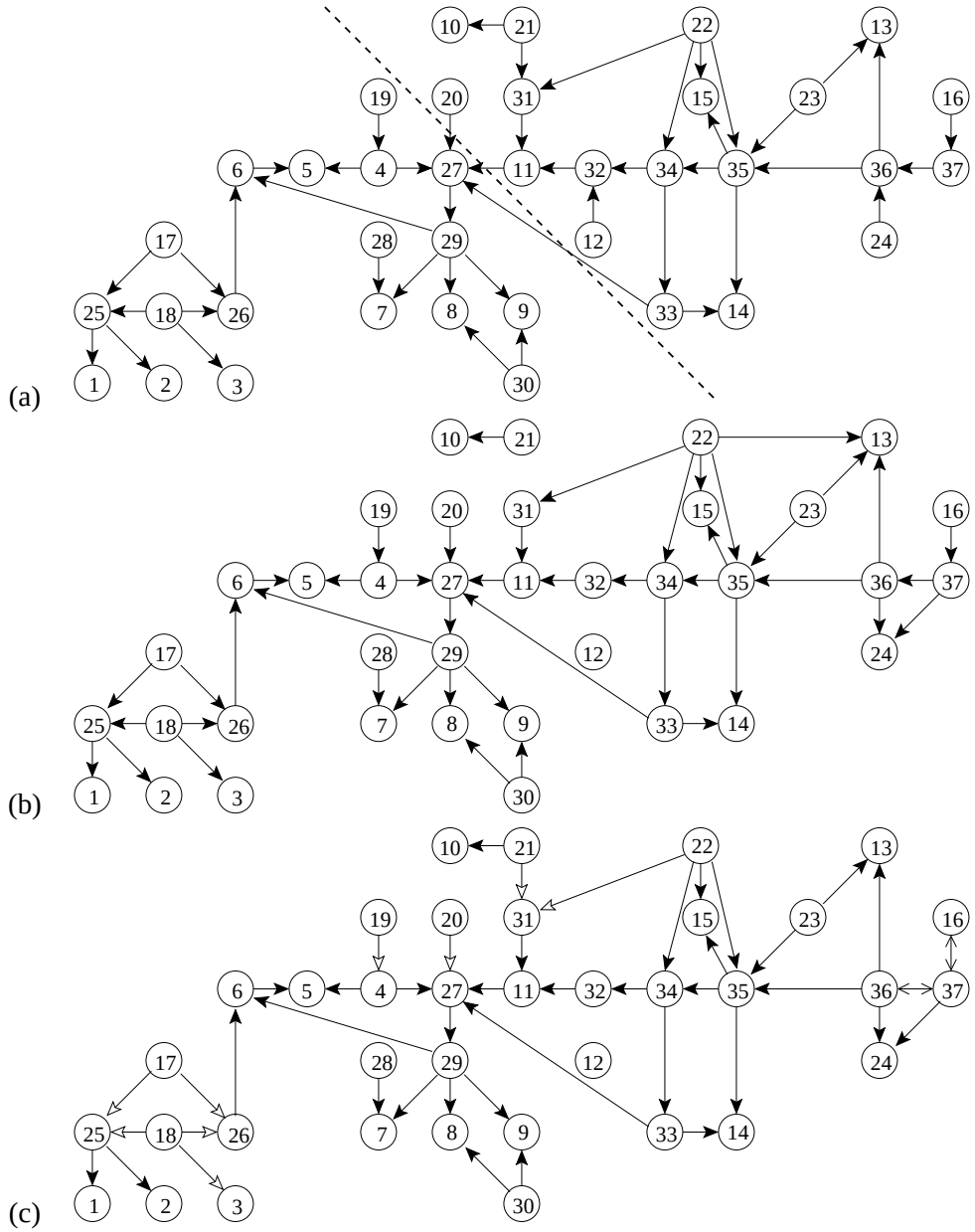


Figure 5: Discovering network structure for *Alarm*. (a) The correct structure. A dashed line separates two layers. For 2500 records, (b) a structure that maximizes the posterior probability, and (c) edge probabilities. Arrow heads ▲, △, and ∧ correspond to intervals $[0.99, 1.00]$, $[0.67, 0.99)$, and $[0.33, 0.67)$, respectively.

For the larger data sets, *Alarm* and *Parity2*, we resorted to a priori layering, as unconstrained exact computation is infeasible. For the *Alarm* data set, we divided the 37 variables into two layers of sizes 18 and 19 as illustrated in Figure 5(a). For the *Parity2* data set, we used the "correct" two layers of sizes 78 and 22.

7.3 Results on Structure Discovery

For each data set we computed a single network structure that maximizes the posterior probability. We also computed posterior probabilities of edges for every pair of variables. That is, although the presented methodology supports computation of the posterior probabilities of arbitrary subgraphs, here we only consider directed edges. Here we show results for the data sets *Parity1* and *Alarm* only. For the *Zoo* data set we are not aware of any correct structure, and for the *Parity2* data set the results are similar to the results for the *Parity1* data set.

Figure 4 illustrates possible end results of structure learning on the *Parity1* data sets of different sizes. We see that with the data size of 500 records the best structure found (Figure 4(b)) is almost identical with the underlying correct network (Figure 4(a)). However, the edge between x_2 and x_3 is reversed and x_3 has stolen the children of x_2. This can be explained by a strong correlation of x_2 and x_3. We also see that the underlying pattern of x_6, x_9, and x_{11} is just partly discovered. With 2500 records there is some improvement: the correct direction between x_2 and x_3 is identified, x_{22} is correctly detected as a child of x_2, and the incorrect edge from x_3 to x_{22} is removed (results not shown). This expected learning phenomenon can be observed in finer detail from the posterior probabilities of edges with the data sizes 500 and 2500 (Figure 4(c–d)). A small surprise is that with 2500 records the correct direction between x_2 and x_3 is less probable than the incorrect one.

Likewise, Figure 5 displays results of structure learning on the *Alarm* data sets of different sizes. We see that with 2500 records almost all correct edges were assigned a high probability and only few edges are added or are missing (meaning low posterior probabilities). An example of a missing edge is the one between the variables 12 and 37. That this edge is not supported by the data is in consistence with earlier studies on the *Alarm* data (e.g., Cooper and Herskovits, 1992).

To investigate whether exact search for an optimal structure can really outperform heuristic search methods, we ran two freely available programs on the *Parity1* data set with 2500 records and on its extended version with 10000 records. B-Course[3] (Myllymäki et al., 2002) is a web-based tool for Bayesian network modeling. Among other features it implements a search algorithm for finding plausible network structures. Because the program runs on a web server, the time of a single trial is limited to 15 minutes (in real time). The user cannot change the parameters of the algorithm which combines greedy and stochastic search heuristics. LibB[4] is a Bayesian network toolbox developed by Nir Friedman and Gal Elidan. It provides various search algorithms based on greedy and stochastic heuristics. Based on preliminary experiments we selected a greedy stochastic hill-climbing algorithm with 20 random restarts so that the time of a single run was about six minutes for the data set with 2500 records. (This is quite fair as our exact algorithm takes about nine minutes.) Since the algorithms implemented in B-Course and LibB are stochastic, we run both programs nine times. Each learned network structure was scored by comparing it against the correct structure. We counted the number of missing edges and the number of extra edges. A reversed edge does not contribute to these error counts if and only if the graph remains "locally equivalent", that is, the correct graph is equivalent[5] to a graph with the reversed edge. It is clear from the results, summarized in Figure 6, that finding an optimal structure is difficult for the tested heuristic methods. For these heuristic methods the number of errors is typically about four times greater than for the exact algorithm. That LibB performs better than B-Course might be explained by the fact that we

3. B-Course is available at http://b-course.cs.helsinki.fi/.
4. LibB is available at http://www.cs.huji.ac.il/labs/compbio/LibB/.
5. Two graphs are equivalent if they have the same skeleton (undirected structure) and the same sets of collapsing edges (also called v-structures).

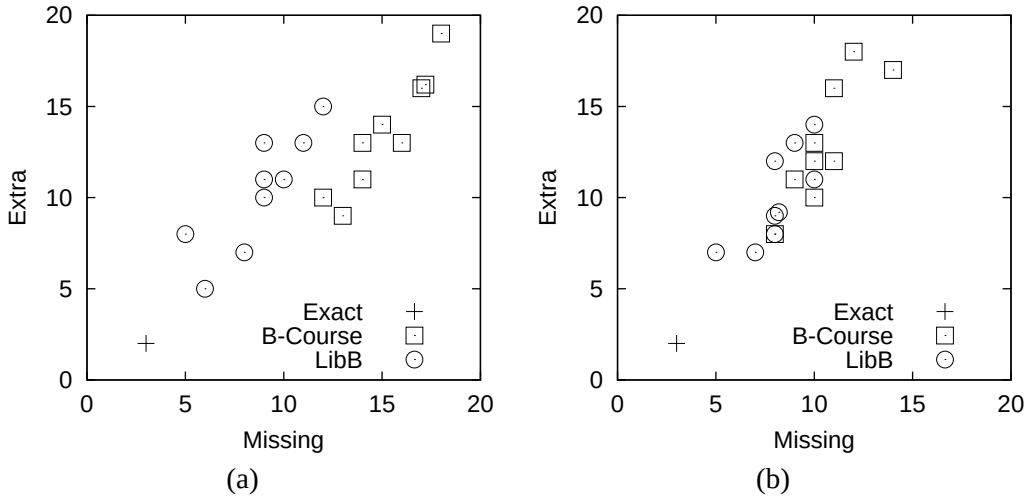


Figure 6: The number of missing and extra edges in graphs found by the exact algorithm and two heuristic methods with nine independent runs each. Structures were learned from the *Parity1* data set with (a) 2500 and (b) 10000 records. For visualization, coincident points are slightly perturbed. The correct structure has 31 edges.

made LibB to use many random restarts, whereas B-Course, perhaps, iterates over a single search chain; however, we do not know the actual search algorithm used by B-Course. Since the number of nodes, 22, is not very large and the optimization landscape is likely to be complex, using several random restarts may be a better strategy. Finally, it is worth noting that we did not set any in-degree bound for the heuristic methods. However, we compensated this by using the bound 5 for the exact algorithm, while the actual maximum in-degree is 4.

7.4 The Speed of Exact Structure Discovery

We also measured the speed of the implementation. To get a rather detailed picture of how the total time requirement is composed, separate measurements were carried out for a number of subroutines in the summation and maximization tasks. Table 1 summarizes the speed measurements over different data sets.

In the case of summation, Table 1 reports the time used for computing the functions α, β, and g as defined in Equations (6), (8), and (7), respectively. We denote these three time requirements by T_α, T_β, and T_g. Recall that for T_α and T_g we have an exponential bound $O(n2^n)$ whereas for T_β we have a polynomial bound $O(n^{k+1}C(m))$. When two layers were used (instead of one) the measurement was done for the function α' expressed in the algorithm in Figure 3 (though reported as T_α for convenience).

In the case of maximization, Table 1 displays the counterparts of these quantities. A difference, however, is that for the latter two quantities we only report their sum. This is because we implemented the maximization method so that it first finds an optimal order and then, based on the order, it finds an optimal graph. This reduces the memory requirement with the expense that we have to

				Summation Time (sec.)				Maximization Time (sec.)		
Name	n	k	m	T_α	T_β	T_g	Total	T_α	$T_{\beta+g}$	Total
Zoo	17	6	101	14	30	1	45	10	36	46
Parity1	22	5	100	536	17	30	583	379	35	414
			500	531	36	30	597	377	57	434
			2500	519	132	30	681	378	175	553
Alarm	$19+18$	4	100	66	28	4	98	47	42	89
			500	67	73	4	144	47	104	151
			2500	63	264	4	331	47	429	476
Parity2	$78+22$	3	100	345	235	32	612	253	448	701
			500	330	1042	31	1403	253	1925	2178
			2500	320	5517	31	5868	252	10070	10322

Table 1: The speed of summation and maximization on selected data sets.

recompute some values of the functions β. That said, in this case the polynomial and exponential contribution are not easily told apart.

The experimental results are in good agreement with the presented asymptotic bounds. We see that the total time requirements are about the same for the summation and maximization tasks. Yet, T_α is consistently less for maximization than for summation. This is because of certain special numeric routines used for adding small numbers. That this difference is not preserved in the total time requirement can be explained by the fact that in the maximization some quantities are computed twice. We also observe that the relationship of the data size m and T_β is close to linear, as expected. (The slight distortion from linearity can be explained by the fact that when processing first hundreds records some expensive memory allocation routines are used.) Note also that T_α is invariant with respect to m.

Regarding the number of seconds needed for computing the probability of a feature, or, for finding an optimal network structure, it is clear that none of the combinations of model and data parameters used in our experiments lies close to the boundaries of practical tractability: the longest time is about three hours, for finding an optimal structure for the *Parity2* data set with 2500 records. Yet, via the polynomial contribution, the time requirement is highly sensitive to the maximum in-degree k. For example, replacing $k=3$ by $k=4$ for the *Parity2* data set would result in about 25-fold increase in the time requirement.[6] Also, what is not explicit in the reported time requirements is that feature probabilities are usually computed for a large number of features; in our case, for every directed edge between two variables. Another important fact is that the space requirement of the implemented method is exponential: roughly $n2^n$ floating point values need to be stored. This limits the use of the current implementation to at most 25 variables.

8. Concluding Remarks

We have presented a novel algorithm for learning Bayesian network structures from data. Our algorithm computes the exact posterior probability of a queried local subnetwork, e.g., a directed edge between two nodes. A modified version of the algorithm finds a global network structure

6. The increase is from $\binom{100}{3}$ to $\binom{100}{4}$ rather than from 100^3 to 100^4 (as hinted by the asymptotic expression).

which maximizes the posterior probability. A major advantage of this method is that it explores all possible structures, still running "only" in an exponential time with respect to the number of network variables. We can expect that this feature makes it possible to successfully analyze cases where inexact methods may fail. Although we provided some bits of evidence for this hypothesis, more convincing validation would require a dedicated comparison study, not pursued in this paper. Here we, instead, reported a set of experiments to illustrate the presented methods and to investigate the actual speed of the implemented algorithms.

Our experiments demonstrate the applicability of the method. For moderate-size networks the running times were found to be feasible. The results also highlight the crucial role of the maximum in-degree and the size of the data (i.e., the polynomial contribution) in the overall complexity. We also showed that structure discovery on large networks is practical if one can judge a suitable layering constraint on the space of network structures. Another main conclusion is that the actual bottleneck in the current implementation is the space requirement, not the speed. For example, a simple extrapolation (not shown) based on the speed measurements (shown in Table 1) reveals that a posterior maximizing structure for the Alarm data (37 variables, 2500 examples, maximum in-degree 4) could be found in about three days, if enough memory is present.

Although the presented algorithm shows promise as an exact method for structure discovery in complex Bayesian networks, there remain many important open problems. From the practical point of view, reducing the space complexity is a question of great importance. An open problem is whether the space requirement can be significantly reduced with little or no computational overhead. Interesting, yet rather theoretic, is the question whether there exists an algorithm faster than the one presented in this paper (asymptotically with respect to the number of variables). Since the maximization and summation problems resemble much the TSP problem and the computation of matrix permanents, respectively—for which no faster algorithms is known—we believe that the answer is negative.

There are many directions in which the presented methods can be extended. A practically important issue is handling missing observations; in this paper we only considered the case of complete data. The full Bayesian solution involves integration over the missing data. Unfortunately, it seems that exact computation is not feasible in general, unless the number of missing values is very small (less than 10, depending on the number of possible values of the variables). MCMC methods provide a practical way to overcome this problem. It is worth noting, however, that the exact summation algorithm readily applies to (posterior) prediction of the values of some variables, given data on the rest of the variables. A different issue is relaxing the structure discovery by allowing for undirected edges. In particular, instead of finding a directed graph that maximizes the posterior probability, one might be interested in searching for an equivalence class of such optimal graphs, conveniently expressed by a partially directed acyclic graph (PDAG); see, e.g., a recent paper by Castelo and Kočka (2003) and references therein. We note that an optimal PDAG can be found indirectly by first finding an optimal directed graph and then turning to its equivalence class. However, it is not clear how the presented algorithms could be used for computing feature probabilities on PDAGs, or whether the techniques used in our work can be extended to handle PDAGs in some more direct way.

Finally, we think that the structural constraints exploited by the presented exact algorithms deserve discussion. The assumptions of order-modularity and graph-modularity are built on related previous work (Cooper and Herskovits, 1992; Heckerman et al., 1995a; Friedman and Koller, 2003). While the former essentially factorizes the conditional prior distribution $p(G \mid \prec)$ of structures given

a variable order, the latter factorizes the unconditional prior $p(G)$. These assumptions are not only vital for efficient computation, but also offer a convenient and often suitable form for expressing prior beliefs. Nevertheless, modular priors can be criticized as the posterior distribution typically will not remain modular. A novelty in the work of Friedman and Koller (2003) was the introduction of a prior distribution $p(\prec)$ on orders; we took a small step further assuming that this distribution factorizes. Accordingly, the modeler should be able to think of a "true order", which may be difficult if not unjustified. However, treating the order as a technical artifact without any interpretation removes the problem. Then the remaining problem is to justify the resulting form of the structure prior $p(G)$. How well this form in practice matches the modeler's prior is likely to be case-dependent.

In addition to the modularity assumption, we required that the number of parents of any variable is bounded by a constant. This is a convenient way to reduce the size of the search space. Unfortunately, only seldom is a small hard bound justified by background knowledge. However, under certain regularity assumptions one can argue that no variable should have more than about $\log_2 m$ parents, where m is the data size (Bouckaert, 1994). This is because sufficient amount of evidence for many parents is needed to compensate the cost of model complexity. A similar result is due to Höffgen (1993) who shows that about 2^k data records are sufficient (maybe also needed) to learn Boolean networks, where each variable has at most k parents. These results—though rather obvious when ignoring the issues of accuracy and confidence—recognize that if the modeler's background knowledge is vague, then a moderate in-degree bound is justified. On the other hand, it is worth noting that the algorithms presented in this paper work even if no in-degree bound is fixed. It is not difficult to see that then the time complexity is $O(n2^n C(m,n))$, where $C(m,n)$ is the time needed for computing a single local conditional marginal for m data records over at most n variables.

Acknowledgments

The authors thank Heikki Mannila for useful discussions and the anonymous reviewers for comments that helped to improve the manuscript. This work was partly supported and inspired by the Morgam (www.ktl.fi/morgam) and GenomeEUTwin (www.genomeutwin.org) projects.

References

S. Acid and L. de Campos. Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

I. Beinlich, G. Suermondt, R. Chavez, and G. F. Cooper. The ALARM monitoring system. In J. Hunter, editor, *Proceedings of the Second European Conference on Artificial Intelligence and Medicine*, pages 247–256. Springer-Verlag, Berlin, 1989.

R. Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9:61–63, 1962.

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

R. R. Bouckaert. Properties of Bayesian belief network learning algorithms. In R. Lopez de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 102–109, Seattle, WA, 1994. Morgan Kaufmann, San Francisco, CA.

W. Buntine. Theory refinement on Bayesian networks. In B. D'Ambrosio, P. Smets, and P. Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, Los Angeles, CA, 1991. Morgan Kaufmann, San Mateo, CA.

R. Castelo and T. Kočka. On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527–574, 2003.

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

D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth Conference on Artificial Intelligence and Statistics*, pages 112–128. Society for Artificial Intelligence and Statistics, Ft. Lauderdale, 1995.

D. M. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In D. Geiger and P. Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 80–89, Providence, RI, 1997. Morgan Kaufmann, San Francisco, CA.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

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

R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113 (1-2):41–85, 1999.

N. Friedman and D. Koller. Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1–2):95–125, 2003.

A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

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

D. Heckerman, A. Mamdani, and M. P. Wellman. Real-world applications of Bayesian networks. *Communications of the ACM*, 38(3):24–30, 1995b.

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

E. Herskovits. *Computer-Based Probabilistic Network Construction*. PhD thesis, Medical Information Sciences, Stanford University, 1991.

K.-U. Höffgen. Learning and robust learning of product distributions. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 77–83, Santa Cruz, CA, USA, 1993. ACM Press.

R. Kennes and P. Smets. Computational aspects of the Möbius transformation. In P. B. Bonissone, M. Henrion, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pages 401–416. North-Holland, Amsterdam, 1991.

P. Larrañaga, C. Kuijpers, R. Murga, and Y. Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(4):487–493, 1996.

S. L. Lauritzen and F. V. Jensen. Local computation with valuations from a commutative semigroup. *Annals of Mathematics and Artificial Intelligence*, 21(1):51–69, 1997.

D. Madigan and J. York. Bayesian graphical models for discrete data. *International Statistical Review*, 63:215–232, 1995.

P. Myllymäki, T. Silander, H. Tirri, and P. Uronen. B-Course: A web-based tool for Bayesian and causal data analysis. *International Journal on Artificial Intelligence Tools*, 11(3):369–387, 2002.

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

R. E. Stearns and H. B. Hunt III. An algebraic model for combinatorial problems. *SIAM Journal on Computing*, 25(2):448–476, 1996.

Journal of Machine Learning Research 5 (2004) 575-604 Submitted 1/04; Published 6/04

A Universal Well-Calibrated Algorithm for On-line Classification

Vladimir Vovk VOVK@CS.RHUL.AC.UK
Computer Learning Research Centre
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK

Editors: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

We study the problem of on-line classification in which the prediction algorithm, for each "significance level" δ, is required to output as its prediction a range of labels (intuitively, those labels deemed compatible with the available data at the level δ) rather than just one label; as usual, the examples are assumed to be generated independently from the same probability distribution P. The prediction algorithm is said to be "well-calibrated" for P and δ if the long-run relative frequency of errors does not exceed δ almost surely w.r. to P. For well-calibrated algorithms we take the number of "uncertain" predictions (i.e., those containing more than one label) as the principal measure of predictive performance. The main result of this paper is the construction of a prediction algorithm which, for any (unknown) P and any δ: (a) makes errors independently and with probability δ at every trial (in particular, is well-calibrated for P and δ); (b) makes in the long run no more uncertain predictions than any other prediction algorithm that is well-calibrated for P and δ; (c) processes example n in time $O(\log n)$.

Keywords: Transductive Confidence Machine, on-line prediction

1. Introduction

Typical machine learning algorithms output a point prediction for the label of an unknown object. This paper continues study of an algorithm called the Transductive Confidence Machine (TCM), introduced by Saunders et al. (1999) and Vovk et al. (1999), that complements its predictions with some measures of confidence. There are different ways of presenting TCM's output; in this paper (as in the related Vovk, 2002a,b) we use TCM as a "region predictor", in the sense that it outputs a nested family of prediction regions (indexed by the significance level δ) rather than a point prediction.

Any TCM is well-calibrated when used in the on-line mode: for any significance level δ the long-run relative frequency of erroneous predictions does not exceed δ. What makes this feature of TCM especially appealing is that it is far from being just an asymptotic phenomenon: a slight modification of TCM called randomized[1] TCM (rTCM) makes errors independently at different trials and with probability δ at each trial. The property of being well-calibrated then immediately follows by the Borel strong law of large numbers. Figure 1 shows the cumulative numbers of errors at the significance levels 1%–5% made on the well-known USPS data set of hand-written digits (randomly permuted); as expected, these are straight lines with the slope approximately equal to the significance level. For proofs and further information, see Vovk (2002a).

1. Randomization is needed to break ties and deal efficiently with borderline cases.

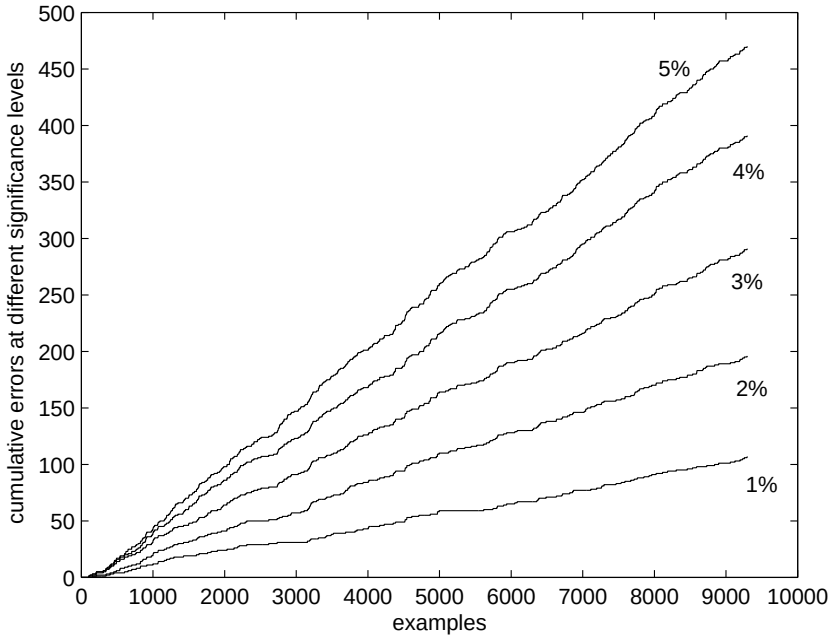


Figure 1: TCM's cumulative errors at the significance levels 1%–5% on the USPS data set

The justification of the study of TCM given by Vovk (2002a) was its good performance on real-world and standard benchmark data sets. For example, Figure 2 shows that for the significance levels between 1% and 5% most examples in the USPS data set can be predicted categorically (by a simple 1-Nearest Neighbour TCM, used in all experiments reported in this paper): the prediction region contains only one label.

This paper presents theoretical results about TCM's performance in the problem of classification, where the number of possible labels is finite; we show that there exists a *universal* rTCM, which, for any significance level δ and without knowing the true distribution P generating the examples:

- produces, asymptotically, no more uncertain predictions than any other prediction algorithm that is well-calibrated for P and δ;
- produces, asymptotically, at least as many empty predictions as any other prediction algorithm that is well-calibrated for P and δ and whose percentage of uncertain predictions is optimal (in the sense of the previous item).

The importance of the first item is obvious: we want to minimize the number of uncertain predictions. This asymptotic criterion ceases to work, however, when the number of uncertain predictions stabilizes, as in Figure 2 for significance levels 3%–5%. In such cases the number of empty predictions becomes important: empty predictions (automatically leading to an error) provide a warning that the object is atypical (looks very different from the previous objects), and one would like to be warned as often as possible, taking into account that the relative frequency of errors (including empty predictions) is guaranteed not to exceed δ in the long run. Remember that TCM outputs a whole family of prediction regions, so the fact that at some significance level the prediction region

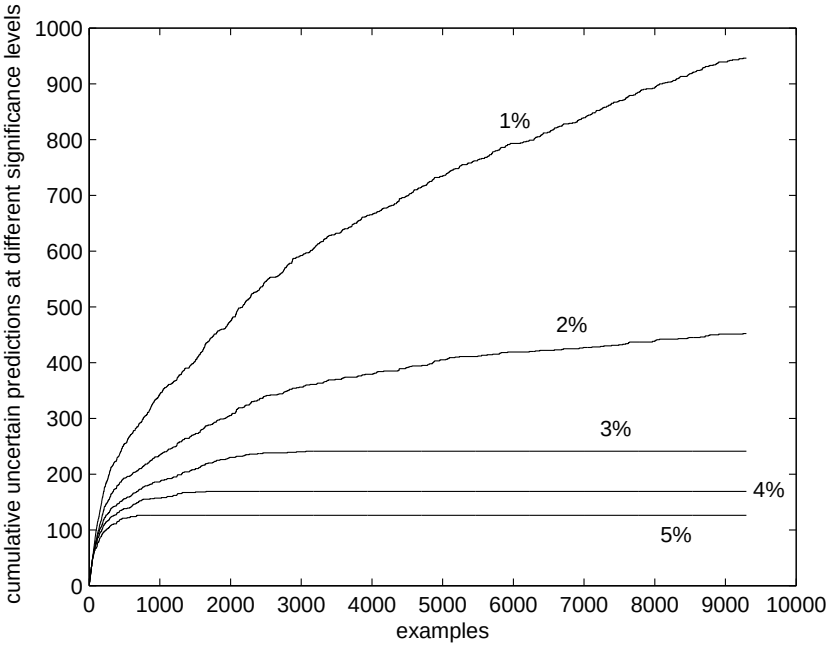


Figure 2: Cumulative number of "uncertain" predictions (i.e., prediction regions containing more than one label) made by the 1-Nearest Neighbour TCM at the significance levels 1%–5% on the USPS data set

becomes empty does not mean that all potential labels for a new object become equally likely: we should just shift our attention to other significance levels. Figure 3 shows the cumulative numbers of empty predictions for the USPS data set.

The full prediction output by a TCM is a complicated mathematical object: for each significance level δ we have a prediction region. In practice, a good starting point might be first to look at the prediction regions corresponding to two or three conventional significance levels, such as 1% and 5% (afterwards, of course, the prediction regions at other significance levels should be looked at). For example, denoting Γ^δ the prediction region at significance level δ, we could say that the prediction is "highly certain" if $|\Gamma^{1\%}| \le 1$ and "certain" if $|\Gamma^{5\%}| \le 1$; similarly, we could say that the new object (whose label is being predicted) is "highly atypical" if $|\Gamma^{1\%}| = 0$ and "atypical" if $|\Gamma^{5\%}| = 0$. In the case of classification, the family of prediction regions Γ^δ can be summarized by reporting the *confidence*

$$\sup\{1-\delta : |\Gamma^\delta| \le 1\},$$

the *credibility*

$$\inf\{\delta : |\Gamma^\delta| = 0\},$$

and the *prediction* Γ^δ, where $1-\delta$ is the confidence (in the case of TCM, $|\Gamma^\delta| \le 1$ and usually $|\Gamma^\delta| = 1$ when $1-\delta$ is the confidence). Reporting the prediction, confidence, and credibility, as in Saunders et al. (1999) and Vovk et al. (1999), is analogous to reporting the observed level of significance (Cox and Hinkley, 1974, p. 66) in statistics.

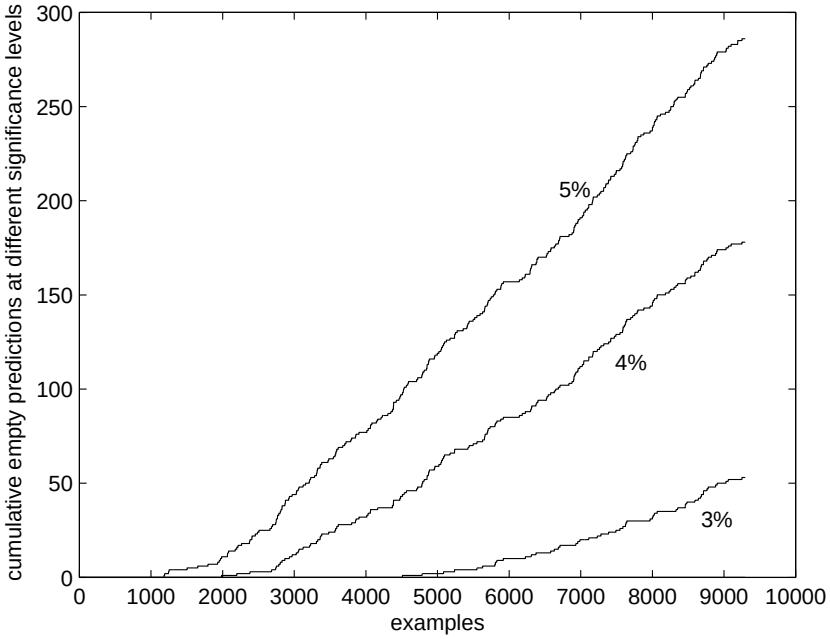


Figure 3: Cumulative number of empty predictions made by the 1-Nearest Neighbour TCM at the significance levels 1%–5% on the USPS data set (there are no empty predictions for 1% and 2%)

This paper's result elaborates on Vovk (2002b), where it was shown that an optimal randomized TCM exists when the distribution P generating the examples is known. In the rest of this paper we consider only randomized TCM, so we drop the adjective "randomized".

The two areas of mainstream machine learning that are most closely connected with this paper are PAC learning theory and Bayesian learning theory. Whereas we often use the rich arsenal of mathematical tools developed in these fields, they do not provide the same kind of guarantees (the right probability of error at each significance level, with errors at different trials independent) under unknown P; for more details, see Vovk (2002a) and references therein. Several papers (such as Rivest and Sloan, 1988; Freund et al., 2004) extend the standard PAC framework by allowing the prediction algorithm to abstain from making a prediction at some trials. Our results show that for any significance level δ there exists a prediction algorithm that: (a) makes a wrong prediction with relative frequency at most δ; (b) has an optimal frequency of abstentions among the prediction algorithms that satisfy property (a) (for details, see Remark 2 on p. 580). The paper by Freund et al. (2004) is especially close to the approach of this paper, defining a very natural TCM in the situation where a hypothesis class is given (the "empirical log ratio" of Freund et al. (2004), taken with appropriate sign, can be used as an "individual strangeness measure", as defined in §3).

2. Main Result

In our learning protocol, Reality outputs pairs $(x_1,y_1),(x_2,y_2),\ldots$ called *examples*. Each example (x_i,y_i) consists of an *object* x_i and its *label* y_i. The objects are chosen from a measurable space $\mathbf{X}$

called the *object space* and the labels are elements of a measurable space $\mathbf{Y}$ called the *label space*. In this paper we assume that $\mathbf{Y}$ is finite (and endowed with the σ-algebra of all subsets). The protocol includes variables Err_n^δ (the total number of errors made up to and including trial n at significance level δ) and err_n^δ (the binary variable showing whether an error is made at trial n). It also includes analogous variables Unc_n^δ, unc_n^δ, Emp_n^δ, emp_n^δ for uncertain and empty predictions:

$\mathrm{Err}_0^\delta := 0$, $\mathrm{Unc}_0^\delta := 0$, $\mathrm{Emp}_0^\delta := 0$ for all $\delta \in (0,1)$;
FOR $n = 1,2,\ldots$:
  Reality outputs $x_n \in \mathbf{X}$;
  Predictor outputs $\Gamma_n^\delta \subseteq \mathbf{Y}$ for all $\delta \in (0,1)$;
  Reality outputs $y_n \in \mathbf{Y}$;
  $\mathrm{err}_n^\delta := \begin{cases} 1 & \text{if } y_n \notin \Gamma_n^\delta \\ 0 & \text{otherwise} \end{cases}$, $\mathrm{Err}_n^\delta := \mathrm{Err}_{n-1}^\delta + \mathrm{err}_n^\delta$ for all $\delta \in (0,1)$;
  $\mathrm{unc}_n^\delta := \begin{cases} 1 & \text{if } |\Gamma_n^\delta| > 1 \\ 0 & \text{otherwise} \end{cases}$, $\mathrm{Unc}_n^\delta := \mathrm{Unc}_{n-1}^\delta + \mathrm{unc}_n^\delta$ for all $\delta \in (0,1)$;
  $\mathrm{emp}_n^\delta := \begin{cases} 1 & \text{if } |\Gamma_n^\delta| = 0 \\ 0 & \text{otherwise} \end{cases}$, $\mathrm{Emp}_n^\delta := \mathrm{Emp}_{n-1}^\delta + \mathrm{emp}_n^\delta$ for all $\delta \in (0,1)$
END FOR.

We will use the notation $\mathbf{Z} := \mathbf{X} \times \mathbf{Y}$ for the *example space*; Γ_n^δ will be called the *prediction region* (or just *prediction*).

We will assume that each example $z_n = (x_n, y_n)$, $n = 1,2,\ldots$, is output according to a probability distribution P in $\mathbf{Z}$ and the examples are independent of each other (so the sequence $z_1 z_2 \ldots$ is output by the power distribution P^∞). This is Reality's randomized strategy.

A *region predictor* is a measurable function

$$\Gamma^\delta(x_1, \tau_1, y_1, \ldots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n), \tag{1}$$

where $\delta \in (0,1)$, $n = 1,2,\ldots$, the $(x_i, y_i) \in \mathbf{Z}$, $i = 1,\ldots,n-1$, are examples, $x_n \in \mathbf{X}$ is an object, and $\tau_i \in [0,1]$ ($i = 1,\ldots,n$), which satisfies

$$\Gamma^{\delta_1}(x_1, \tau_1, y_1, \ldots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n) \subseteq \Gamma^{\delta_2}(x_1, \tau_1, y_1, \ldots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n)$$

whenever $\delta_1 \geq \delta_2$. The measurability of (1) means that for each n the set

$$\left\{(\delta, x_1, \tau_1, y_1, \ldots, x_n, \tau_n, y_n) : y_n \in \Gamma^\delta(x_1, \tau_1, y_1, \ldots, x_{n-1}, \tau_{n-1}, y_{n-1}, x_n, \tau_n)\right\} \subseteq (0,1) \times (\mathbf{X} \times [0,1] \times \mathbf{Y})^n$$

is measurable.

Since we are interested in prediction with confidence, the region predictor (1) is given an extra input $\delta \in (0,1)$, which we call the *significance level* (typically it is close to 0, standard values being 1% and 5%); the complementary value $1 - \delta$ is called the *confidence level*. We will always assume that τ_n are independent random variables uniformly distributed in $[0,1]$. This makes a region predictor a family (indexed by $\delta \in (0,1)$) of Predictor's randomized strategies.

We will often use the notation err_n^δ, unc_n^δ, etc., in the case where Reality and Predictor are using given randomized strategies. For example, $\mathrm{err}_n^\delta(P^\infty, \Gamma)$ is the random variable equal to 0 if Predictor

is right at trial n and at significance level δ and equal to 1 otherwise. It is always assumed that the random numbers τ_n used by Γ and the random examples z_n chosen by Reality are independent.

We say that a region predictor Γ is (conservatively) *well-calibrated* for a probability distribution P in $\mathbf{Z}$ and a significance level $\delta \in (0,1)$ if

$$\limsup_{n\to\infty} \frac{\mathrm{Err}_n^\delta(P^\infty,\Gamma)}{n} \le \delta \quad \text{a.s.}$$

We say (as in Vovk, 2002b) that Γ is *optimal* for P and δ if, for any region predictor $\Gamma^\dagger$ which is well-calibrated for P and δ,

$$\limsup_{n\to\infty} \frac{\mathrm{Unc}_n^\delta(P^\infty,\Gamma)}{n} \le \liminf_{n\to\infty} \frac{\mathrm{Unc}_n^\delta(P^\infty,\Gamma^\dagger)}{n} \quad \text{a.s.} \tag{2}$$

(It is natural to assume in this and other similar definitions that the random numbers used by Γ and $\Gamma^\dagger$ are independent, but this assumption is not needed for our mathematical results and we do not make it.) Of course, the definition of optimality is natural only for well-calibrated Γ.

A region predictor Γ is *universal well-calibrated* if:

- it is well-calibrated for any P and δ;
- it is optimal for any P and δ;
- for any P, any δ, and any region predictor $\Gamma^\dagger$ which is well-calibrated and optimal for P and δ,
$$\liminf_{n\to\infty} \frac{\mathrm{Emp}_n^\delta(P^\infty,\Gamma)}{n} \ge \limsup_{n\to\infty} \frac{\mathrm{Emp}_n^\delta(P^\infty,\Gamma^\dagger)}{n} \quad \text{a.s.}$$

Recall that a measurable space $\mathbf{X}$ is *Borel* if it is isomorphic to a measurable subset of the interval $[0,1]$. The class of Borel spaces is very rich; for example, all Polish spaces (such as finite-dimensional Euclidean spaces $\mathbb{R}^n$, $\mathbb{R}^\infty$, functional spaces C and D) are Borel.

Theorem 1 *Suppose the object space $\mathbf{X}$ is Borel. There exists a universal well-calibrated region predictor.*

This is the main result of the paper. In §3 we construct a universal well-calibrated region predictor (processing example n in time $O(\log n)$) and in §4 outline the idea of the proof that it indeed satisfies the required properties. Technical details will be given in §5.

Remark The protocol of Rivest and Sloan (1988) and Freund et al. (2004) is in fact a restriction of our protocol, in which Predictor is only allowed to output a one-element set or the whole of $\mathbf{Y}$; the latter is interpreted as abstention. (And in the situation where the numbers of errors and uncertain predictions are of primary interest, as in this paper, the difference between these two protocols is not significant.) The universal well-calibrated region predictor can be adapted to the restricted protocol by replacing an uncertain prediction with $\mathbf{Y}$ and replacing an empty prediction with a randomly chosen label. In this way we obtain a prediction algorithm in the restricted protocol which is well-calibrated and has an optimal frequency of abstentions, in the sense of (2), among the well-calibrated algorithms.

3. Construction of a Universal Well-Calibrated Region Predictor

In this section we first define the general notion of Transductive Confidence Machine, and then we specialize it using a nearest neighbours procedure to obtain a universal well-calibrated region predictor.

3.1 Preliminaries

If τ is a number in $[0,1]$, we split it into two numbers $\tau', \tau'' \in [0,1]$ as follows: if the binary expansion of τ is $0.a_1a_2\ldots$ (redefine the binary expansion of 1 to be $0.11\ldots$), set $\tau' := 0.a_1a_3a_5\ldots$ and $\tau'' := 0.a_2a_4a_6\ldots$. If τ is distributed uniformly in $[0,1]$, then both τ' and τ'' are, and they are independent of each other.

We will often apply our procedures (e.g., the "individual strangeness measure" in §3.2, the Nearest Neighbours rule in §3.3) not to the original objects $x \in \mathbf{X}$ but to *extended objects* $(x,\sigma) \in \tilde{\mathbf{X}} := \mathbf{X} \times [0,1]$, where x is complemented by a random number σ (to be extracted from one of the τ_n). In other words, along with examples (x,y) we will also consider *extended examples* $(x,\sigma,y) \in \tilde{\mathbf{Z}} := \mathbf{X} \times [0,1] \times \mathbf{Y}$.

Let us set $\mathbf{X} := [0,1]$; we can do this without loss of generality since $\mathbf{X}$ is Borel. This makes the extended object space $\tilde{\mathbf{X}} = [0,1]^2$ a linearly ordered set with the *lexicographic order*: $(x_1,\sigma_1) < (x_2,\sigma_2)$ means that either $x_1 = x_2$ and $\sigma_1 < \sigma_2$ or $x_1 < x_2$. We say that (x_1,σ_1) is *nearer* to (x_3,σ_3) than (x_2,σ_2) is if

$$|x_1 - x_3, \sigma_1 - \sigma_3| < |x_2 - x_3, \sigma_2 - \sigma_3|, \tag{3}$$

where

$$|x,\sigma| := \begin{cases} (x,\sigma) & \text{if } (x,\sigma) \geq (0,0) \\ (-x,-\sigma) & \text{otherwise.} \end{cases} \tag{4}$$

The value $|x_1 - x_2, \sigma_1 - \sigma_2|$ plays the role of the distance between extended objects (x_1,σ_1) and (x_2,σ_2). Despite such distances being two-dimensional, they are still always comparable using the lexicographic order.

Our construction will be based on the Nearest Neighbours algorithm, which is known to be strongly universally consistent in the traditional theory of pattern recognition (see, e.g., Devroye et al., 1996, Chapter 11); the random components σ are needed for tie-breaking.

3.2 Transductive Confidence Machines

Transductive Confidence Machine, or TCM, is a way of transition from what we call an "individual strangeness measure" to a region predictor. A family of measurable functions $\{A_n : n = 1,2,\ldots\}$, where $A_n : \tilde{\mathbf{Z}}^n \to \mathbb{R}^n$ for all n, is called an *individual strangeness measure* if, for any $n = 1,2,\ldots$, each α_i in

$$A_n : (w_1,\ldots,w_n) \mapsto (\alpha_1,\ldots,\alpha_n) \tag{5}$$

is determined by w_i and the multiset $\{\!|w_1,\ldots,w_n|\!\}$. (The difference between a multiset $\{\!|w_1,\ldots,w_n|\!\}$ and a set $\{w_1,\ldots,w_n\}$ is that the former can contain several copies of the same element.)

The *TCM associated with an individual strangeness measure* A_n is the following region predictor $\Gamma^\delta(x_1,\tau_1,y_1,\ldots,x_{n-1},\tau_{n-1},y_{n-1},x_n,\tau_n)$: at any trial n and for any label $y \in \mathbf{Y}$, define

$$(\alpha_1,\ldots,\alpha_n) := A_n((x_1,\tau'_1,y_1),\ldots,(x_{n-1},\tau'_{n-1},y_{n-1}),(x_n,\tau'_n,y)),$$

and include y in Γ^δ if and only if

$$\tau''_n < \frac{\#\{i=1,\dots,n : \alpha_i \geq \alpha_n\} - n\delta}{\#\{i=1,\dots,n : \alpha_i = \alpha_n\}} \tag{6}$$

(in particular, include y in Γ^δ if $\#\{i=1,\dots,n : \alpha_i > \alpha_n\}/n > \delta$ and do not include y in Γ^δ if $\#\{i=1,\dots,n : \alpha_i \geq \alpha_n\}/n \leq \delta$).

A *TCM* is the TCM associated with some individual strangeness measure. It was shown in Vovk (2002a) that

Proposition 2 *Every TCM is well-calibrated for every P and δ.*

The definition of TCM can be illustrated by the following simple example of an individual strangeness measure, the one used in producing Figures 1–3: mapping (5) can be defined, in the spirit of the 1-Nearest Neighbour Algorithm, as (assuming the objects are vectors in a Euclidean space)

$$\alpha_i := \frac{\min_{j \neq i : y_j = y_i} d(x_i, x_j)}{\min_{j \neq i : y_j \neq y_i} d(x_i, x_j)},$$

where d is the Euclidean distance (i.e., an object is considered strange if it is in the middle of objects labelled in a different way and is far from the objects labelled in the same way).

3.3 Universal TCM

Fix a monotonically non-decreasing sequence of integer numbers K_n, $n = 1, 2, \dots$, such that

$$K_n \to \infty,\ K_n = o\left(\sqrt{n/\ln n}\right) \tag{7}$$

as $n \to \infty$. The *Nearest Neighbours TCM* is defined as follows. Let $w_1, \dots, w_n$ be a sequence of extended examples $w_i = (x_i, \sigma_i, y_i)$. To define the corresponding αs , as seen in (5), we first define Nearest Neighbours approximations $P_n^{\neq}(y \mid x_i, \sigma_i)$ to the true (but unknown) conditional probabilities $P(y \mid x_i)$: for every extended example (x_i, σ_i, y_i) in the sequence,

$$P_n^{\neq}(y \mid x_i, \sigma_i) := N^{\neq}(x_i, \sigma_i, y)/K_n, \tag{8}$$

where $N^{\neq}(x_i, \sigma_i, y)$ is the number of $j = 1, \dots, n$ such that $y_j = y$ and (x_j, σ_j) is one of the K_n nearest neighbours, in the sense of (3), of (x_i, σ_i) in the sequence

$$((x_1, \sigma_1), \dots, (x_{i-1}, \sigma_{i-1}), (x_{i+1}, \sigma_{i+1}), \dots, (x_n, \sigma_n)).$$

(The upper index $\neq$ reminds us of the fact that (x_i, σ_i) is not counted as one of its own nearest neighbours in this definition.) If $K_n \geq n$ or $K_n \leq 0$, this definition does not work, so set, e.g., $P_n^{\neq}(y \mid x_i, \sigma_i) := 1/|\mathbf{Y}|$ for all y and i (this particular convention is not essential since, by (7), $0 < K_n < n$ from some n on). If the expression "K_n nearest neighbours" is not defined because of distance ties, we again set $P_n^{\neq}(y \mid x_i, \sigma_i) := 1/|\mathbf{Y}|$ for all y and i (this convention is not essential since distance ties happen with probability zero).

Define the "empirical predictability function" $f_n^{\neq}$ by

$$f_n^{\neq}(x_i, \sigma_i) := \max_{y \in \mathbf{Y}} P_n^{\neq}(y \mid x_i, \sigma_i). \tag{9}$$

For each (x_i, σ_i) fix some

$$\hat{y}_n(x_i, \sigma_i) \in \arg\max_y P_n^{\neq}(y \mid x_i, \sigma_i) \tag{10}$$

(e.g., take the first element of $\arg\max_y P_n^{\neq}(y \mid x_i, \sigma_i)$ in a fixed ordering of $\mathbf{Y}$) and define the mapping (5) (where $w_i = (x_i, \sigma_i, y_i)$, $i = 1, \dots, n$) setting

$$\alpha_i := \begin{cases} -f_n^{\neq}(x_i, \sigma_i) & \text{if } y_i = \hat{y}_n(x_i, \sigma_i) \\ f_n^{\neq}(x_i, \sigma_i) & \text{otherwise.} \end{cases} \tag{11}$$

This completes the definition of the Nearest Neighbours TCM, which will later be shown to be universal.

Proposition 3 *Let $\Delta \subseteq (0,1)$ be finite. If $\mathbf{X} = [0,1]$ and $K_n \to \infty$ sufficiently slowly, the Nearest Neighbours TCM can be implemented for significance levels $\delta \in \Delta$ so that the computations at trial n are performed in time $O(\log n)$.*

Proposition 3 assumes a computational model that allows operations (such as comparison) with real numbers. If $\mathbf{X}$ is an arbitrary Borel space, for this proposition to be applicable $\mathbf{X}$ should be embedded in $[0,1]$ first; e.g., if $\mathbf{X} \subseteq [0,1]^n$, an $x = (x_1, \dots, x_n) \in \mathbf{X}$ can be represented as

$$(x_{1,1}, x_{2,1}, \dots, x_{n,1}, x_{1,2}, x_{2,2}, \dots, x_{n,2}, \dots) \in [0,1],$$

where $0.x_{i,1}x_{i,2}\dots$ is the binary expansion of x_i. We use the expression "can be implemented" in a wide sense, only requiring that the implementation should give the correct results almost surely.

4. Fine Details of Region Prediction

In this section we make first steps towards the proof of Theorem 1. Let P be the true distribution in $\mathbf{Z}$ generating the examples. We denote by $P_{\mathbf{X}}$ the marginal distribution of P in $\mathbf{X}$ (i.e., $P_{\mathbf{X}}(E) := P(E \times \mathbf{Y})$) and by $P_{\mathbf{Y}|\mathbf{X}}(y|x)$ the conditional probability that, for a random example (X,Y) chosen from P, $Y = y$ provided $X = x$ (we fix arbitrarily a regular version of this conditional probability). We will often omit lower indices $_{\mathbf{X}}$ and $_{\mathbf{Y}|\mathbf{X}}$ and P itself from our notation.

The *predictability* of an object $x \in \mathbf{X}$ is

$$f(x) := \max_{y \in \mathbf{Y}} P(y \mid x)$$

and the *predictability distribution function* is the function $F : [0,1] \to [0,1]$ defined by

$$F(\beta) := P\{x : f(x) \le \beta\}.$$

An example of such a function F is given in Figure 4 (left), where the graph of F is the thick line.

The *success curve* $\mathbf{S}_P$ of P is defined by the equality

$$\mathbf{S}_P(\delta) = \inf\left\{ B \in [0,1] : \int_0^1 (F(\beta) - B)^+ d\beta \le \delta \right\}, \tag{12}$$

where t^+ stands for $\max(t, 0)$; the function $\mathbf{S}_P$ is also of the type $[0,1] \to [0,1]$. Geometrically, $\mathbf{S}_P(\delta)$ is defined from the graph of F as follows (see Figure 4, left; we often drop the lower index $_P$):

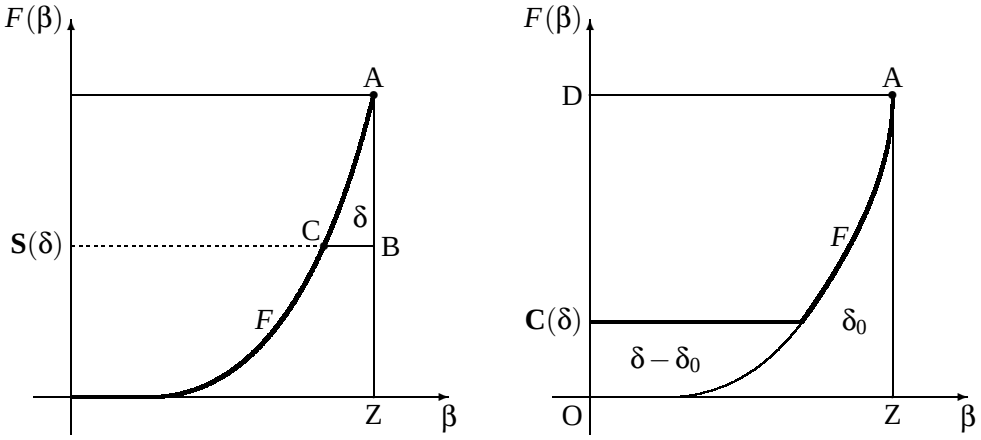


Figure 4: The predictability distribution function F and the success curve $\mathbf{S}(\delta)$ (left); the complementary success curve $\mathbf{C}(\delta)$ (right)

move the point B from A to Z until the area of the curvilinear triangle ABC becomes δ or B reaches Z; the ordinate of B is then $\mathbf{S}(\delta)$.

The *complementary success curve* $\mathbf{C}_P$ of P is defined by

$$\mathbf{C}_P(\delta) = \sup\left\{B \in [0,1] : B + \int_0^1 (F(\beta) - B)^+ d\beta \le \delta\right\}, \tag{13}$$

where $\sup\emptyset$ is interpreted as 0. Similarly to the case of $\mathbf{S}(\delta)$, $\mathbf{C}(\delta)$ is defined as the value such that the area of the part of the box AZOD below the thick line in Figure 4 (right) is δ ($\mathbf{C}(\delta) = 0$ if such a value does not exist).

Define the *critical significance level* δ_0 as

$$\delta_0 := \int_0^1 F(\beta) d\beta. \tag{14}$$

It is clear that

$$\delta \le \delta_0 \Longrightarrow \int_0^1 (F(\beta) - \mathbf{S}(\delta))^+ d\beta = \delta \,\&\, \mathbf{C}(\delta) = 0$$
$$\delta \ge \delta_0 \Longrightarrow \mathbf{S}(\delta) = 0 \,\&\, \mathbf{C}(\delta) + \int_0^1 (F(\beta) - \mathbf{C}(\delta))^+ d\beta = \delta.$$

The following result is proved in Vovk (2002b).

Proposition 4 *Let P be a probability distribution in $\mathbf{Z}$ and $\delta \in (0,1)$ be a significance level. If a region predictor Γ is well-calibrated for P and δ, then*

$$\liminf_{n\to\infty} \frac{\mathrm{Unc}_n^\delta(P^\infty, \Gamma)}{n} \ge \mathbf{S}_P(\delta) \quad \textit{a.s.} \tag{15}$$

In this paper we complement Proposition 4 with

Proposition 5 *Let P be a probability distribution in $\mathbf{Z}$ and $\delta \in (0,1)$ be a significance level. If a region predictor Γ is well-calibrated for P and δ and satisfies*

$$\limsup_{n\to\infty} \frac{\mathrm{Unc}_n^\delta(P^\infty,\Gamma)}{n} \le \mathbf{S}_P(\delta) \quad a.s., \tag{16}$$

then

$$\limsup_{n\to\infty} \frac{\mathrm{Emp}_n^\delta(P^\infty,\Gamma)}{n} \le \mathbf{C}_P(\delta) \quad a.s.$$

Theorem 1 immediately follows from Propositions 2, 4, 5 and the following proposition.

Proposition 6 *Suppose $\mathbf{X}$ is Borel. The Nearest Neighbours TCM constructed in §3.3 satisfies, for any P and any significance level δ,*

$$\limsup_{n\to\infty} \frac{\mathrm{Unc}_n^\delta(P^\infty,\Gamma)}{n} \le \mathbf{S}_P(\delta) \quad a.s. \tag{17}$$

and

$$\liminf_{n\to\infty} \frac{\mathrm{Emp}_n^\delta(P^\infty,\Gamma)}{n} \ge \mathbf{C}_P(\delta) \quad a.s. \tag{18}$$

5. Proofs

In this section we will assume that all extended objects $(x_i,\tau_i') \in [0,1]^2$, where x_i are output by Reality and τ_i are the random numbers used, are different and that all pairwise distances between them are also different (this is true with probability one, since τ_i' are independent random numbers uniformly distributed in $[0,1]$).

5.1 Proof Sketch of Proposition 3

Without loss of generality we assume that Δ contains only one significance level δ, which will be omitted from our notation. Our computational model has an operation of splitting $\tau \in [0,1]$ into τ' and τ'' (or is allowed to generate both τ_n' and τ_n'' at every trial n).

We will use two main data structures in our implementation of the Nearest Neighbours TCM:

- a red-black binary *search tree*;[2]
- a growing *array* of nonnegative integers indexed by $k \in \{-K_n, -K_n+1, \dots, K_n\}$ (where n is the ordinal number of the example being processed).

Immediately after processing the nth extended example (x_n,τ_n,y_n) the contents of these data structures are as follows:

- The search tree contains n vertices, corresponding to the extended examples (x_i,τ_i,y_i) seen so far. The key of vertex i is the extended object $(x_i,\tau_i') \in [0,1]^2$; the linear order on the keys is the lexicographic order. The other information contained in vertex i is the random number τ_i'',

2. See, e.g., Cormen et al. (2001), Chapters 12–14. The only two operations on red-black trees we need in this paper are the query SEARCH and the modifying operation INSERT.

the label y_i, the set $\{P_n^{\neq}(y\,|\,x_i,\tau_i') : y \in \mathbf{Y}\}$ of conditional probability estimates (8), the pointer to the following vertex (i.e., the vertex that has the smallest key greater than (x_i,τ_i'); if there is no greater key, the pointer is NIL), and the pointer to the previous vertex (i.e., the vertex that has the greatest key smaller than (x_i,τ_i'); if (x_i,τ_i') is the smallest key, the pointer is NIL).

- The array contains the numbers

$$N(k) := \#\{i = 1,\dots,n : \alpha_i = k/K_n\}$$

(α_i are defined by (11) with $\sigma_i := \tau_i'$).

Notice that the information contained in vertex i of the search tree is sufficient to find $\hat{y}_n(x_i,\tau_i')$ and α_i in time $O(1)$.

We will say that an extended object (x_j,τ_j') is in the *vicinity* of an extended object (x_i,τ_i'), $i \neq j$, if there are less than K_n extended objects (x_k,τ_k') (strictly) between (x_i,τ_i') and (x_j,τ_j').

When a new object x_n becomes known, the algorithm does the following:

- Generates τ_n' and τ_n''.
- Locates the successor and predecessor of (x_n,τ_n') in the search tree (using the query SEARCH and the pointers to the following and previous vertices); this requires time $O(\log n)$.
- Computes the estimated conditional probabilities $\{P_n^{\neq}(y\,|\,x_n,\tau_n') : y \in \mathbf{Y}\}$; this also gives $\hat{y}_n(x_n,\tau_n')$. This involves scanning the vicinity of (x_n,τ_n') for the K_n nearest neighbours of (x_n,τ_n'), which can be done in time $O(K_n)$: the K_n nearest neighbours can be extracted from the vicinity of (x_n,τ_n') sorted in the order of increasing distances from (x_n,τ_n'); since initially the vicinity consists of two sorted lists (to the left and to the right of (x_n,τ_n')), the procedure MERGE used in the merge sort algorithm (see, e.g., Cormen et al. 2001, §2.3.1) will sort the whole vicinity in time $O(K_n)$. Therefore, the required time is $O(K_n) = O(\log n)$.
- For each $y \in \mathbf{Y}$ looks at what happens if the nth example is $(x_n,\tau_n,y_n) = (x_n,\tau_n,y)$: computes α_n and updates (if necessary) α_i for (x_i,τ_i') in the vicinity of (x_n,τ_n'); using the array and τ_n'', finds whether $y \in \Gamma_n$. This requires time $O(K_n^2) = O(\log n)$, since there are $O(K_n)$ α_i's in the vicinity of (x_n,τ_n') and each of them can be computed in time $O(K_n)$.
- Outputs the prediction region Γ_n (time $O(1)$).

When the label y_n arrives, the algorithm:

- Inserts the new vertex $(x_n,\tau_n',\tau_n'',y_n,\{P_n^{\neq}(y\,|\,x_n,\tau_n') : y \in \mathbf{Y}\})$ in the search tree, repairs the pointers to the following and previous elements for (x_n,τ_n')'s left and right neighbours, initializes the pointers to the following and previous elements for (x_n,τ_n') itself, and rebalances the tree (time $O(\log n)$).
- Updates (if necessary) the conditional probabilities

$$\{P_{n-1}^{\neq}(y\,|\,x_i,\tau_i') : y \in \mathbf{Y}\} \mapsto \{P_n^{\neq}(y\,|\,x_i,\tau_i') : y \in \mathbf{Y}\}$$

for the $2K_n$ existing vertices (x_i,τ_i') in the vicinity of (x_n,τ_n'); this requires time $O(K_n^2) = O(\log n)$. The conditional probabilities for other (x_i,τ_i'), $i = 1,\dots,n-1$, do not change.

- Updates the array, changing $N(K_n\alpha_i)$ for the $(x_i,\tau'_i) \neq (x_n,\tau'_n)$ in the vicinity of (x_n,τ'_n) and for both old and new values of α_i and changing $N(K_n\alpha_n)$ (time $O(K_n) = O(\log n)$).

In conclusion we discuss how to do the updates required when K_n changes. At the critical trials n when K_n changes the array and the estimated conditional probabilities $P_n^{\neq}(y|x_i,\tau'_i)$ have to be recomputed, which, if done naively, would require time $\Theta(nK_n)$.

The assumption we have made about K_n so far is that $K_n = O(\sqrt{\log n})$. We now also assume that K_n is monotonic non-decreasing and

$$\#\{n : K_n < c\} = O(\#\{n : K_n = c\}) \tag{19}$$

as $c \to \infty$. This is the full explication of the "$K_n \to \infty$ sufficiently slowly" in the statement of the lemma, as used in this proof.

An *epoch* is defined to be a maximal sequence of ns with the same K_n. Since the changes that need to be done when a new epoch starts are substantial, they will be spread over the whole preceding epoch; we will only discuss updating the estimated conditional probabilities $P_n^{\neq}(y|x_i,\tau'_i)$: the array is treated similarly. An epoch is *odd* if the corresponding K_n is odd and *even* if K_n is even. At every step in an epoch we prepare the ground for the next epoch. By the end of epoch $n = A+1, A+2, \dots, B$ we need to change B sets $\{P_n^{\neq}(y|x_i,\tau'_i) : y \in \mathbf{Y}\}$ in $B-A$ steps (the duration of the epoch). Therefore, each vertex of the search tree should contain not only $\{P_n^{\neq}(y|x_i,\tau'_i)\}$ for the current epoch but also $\{P_n^{\neq}(y|x_i,\tau'_i)\}$ for the next epoch (two structures for holding $\{P_n^{\neq}(y|x_i,\tau'_i)\}$ will suffice, one for even epochs and one for odd epochs). Our assumptions of the slow growth of K_n, as seen in 19), imply that $B = O(B-A)$. This means that at each step $O(1)$ sets $\{P_n^{\neq}(y|x_i,\tau'_i)\}$ for the next epoch should be added. This will take time $O(K_n) = O(\log n)$. As soon as a set $\{P_n^{\neq}(y|x_i,\tau'_i) : y \in \mathbf{Y}\}$ for the next epoch is added at some trial, both sets (for the current and next epoch) will have to be updated for each new example.

5.2 Proof Sketch of Proposition 5

The proof of Proposition 5 is similar to (but more complicated than) the proof of Theorems 1 and 1r in Vovk (2002b); this proof sketch can be made rigorous using the Neyman–Pearson lemma, as in Vovk (2002b).

We will use the notations g'_{left} and g'_{right} for the left and right derivatives, respectively, of a function g. The following lemma parallels Lemma 2 in Vovk (2002b), which deals with $\mathbf{S}(\delta)$.

Lemma 7 *The complementary success curve $\mathbf{C} : [0,1] \to [0,1]$ always satisfies these properties:*

1. *There is a point $\delta_0 \in [0,1]$ (namely, the critical significance level) such that $\mathbf{C}(\delta) = 0$ for $\delta \le \delta_0$ and $\mathbf{C}(\delta)$ is concave for $\delta \ge \delta_0$.*

2. *$\mathbf{C}'_{\text{right}}(\delta_0) < \infty$ and $\mathbf{C}'_{\text{left}}(1) \ge 1$; therefore, for $\delta \in (\delta_0, 1)$, $1 \le \mathbf{C}'_{\text{right}}(\delta) \le \mathbf{C}'_{\text{left}}(\delta) < \infty$ and the function $\mathbf{C}(\delta)$ is increasing.*

3. *$\mathbf{C}(\delta)$ is continuous at $\delta = \delta_0$; therefore, it is continuous everywhere in $[0,1]$.*

If a function $\mathbf{C} : [0,1] \to [0,1]$ satisfies these properties, there exist a measurable space $\mathbf{X}$, a finite set $\mathbf{Y}$, and a probability distribution P in $\mathbf{X} \times \mathbf{Y}$ for which $\mathbf{C}$ is the complementary success curve.

Proof sketch The statement of the lemma follows from the fact that the complementary success curve $\mathbf{C}$ can be obtained from the predictability distribution function F using these steps (labelling the horizontal and vertical axes as x and y respectively):

1. Invert F: $F_1 := F^{-1}$.
2. Integrate F_1: $F_2(x) := \int_0^x F_1(t)dt$.
3. Increase F_2: $F_3(x) := F_2(x) + \delta_0$, where $\delta_0 := \int_0^1 F(x)dx$.
4. Invert F_3: $F_4 := F_3^{-1}$.

It can be shown that $\mathbf{C} = F_4$, if we define $g^{-1}(y) := \sup\{x : g(x) \le y\}$ for non-decreasing g (so that g^{-1} is continuous on the right). ■

Complement the protocol of §2 in which Reality plays P^∞ and Predictor plays Γ with the following variables:

$$\overline{\mathrm{err}}_n := (P \times \mathbf{U})\left\{(x,y,\tau) : y \notin \Gamma^\delta(x_1,\tau_1,y_1,\dots,x_{n-1},\tau_{n-1},y_{n-1},x,\tau)\right\},$$
$$\overline{\mathrm{unc}}_n := (P_{\mathbf{X}} \times \mathbf{U})\left\{(x,\tau) : |\Gamma^\delta(x_1,\tau_1,y_1,\dots,x_{n-1},\tau_{n-1},y_{n-1},x,\tau)| > 1\right\},$$
$$\overline{\mathrm{emp}}_n := (P_{\mathbf{X}} \times \mathbf{U})\left\{(x,\tau) : |\Gamma^\delta(x_1,\tau_1,y_1,\dots,x_{n-1},\tau_{n-1},y_{n-1},x,\tau)| = 0\right\},$$

δ being fixed and $\mathbf{U}$ standing for the uniform distribution in $[0,1]$, and

$$\overline{\mathrm{Err}}_n := \sum_{i=1}^n \overline{\mathrm{err}}_i, \quad \overline{\mathrm{Unc}}_n := \sum_{i=1}^n \overline{\mathrm{unc}}_i, \quad \overline{\mathrm{Emp}}_n := \sum_{i=1}^n \overline{\mathrm{emp}}_i.$$

By the martingale strong law of large numbers, to prove the proposition it suffices to consider only these "predictable" versions of Err_n, Unc_n, and Emp_n: indeed, since $\mathrm{Err}_n - \overline{\mathrm{Err}}_n$, $\mathrm{Unc}_n - \overline{\mathrm{Unc}}_n$, and $\mathrm{Emp}_n - \overline{\mathrm{Emp}}_n$ are martingales (with increments bounded by 1 in absolute value) with respect to the filtration $\mathcal{F}_n$, $n = 0,1,\dots$, where each $\mathcal{F}_n$ is generated by $(x_1,\tau_1,y_1),\dots,(x_n,\tau_n,y_n)$, we have

$$\lim_{n\to\infty} \frac{\mathrm{Err}_n - \overline{\mathrm{Err}}_n}{n} = 0 \qquad \text{a.s.},$$
$$\lim_{n\to\infty} \frac{\mathrm{Unc}_n - \overline{\mathrm{Unc}}_n}{n} = 0 \qquad \text{a.s.},$$

and

$$\lim_{n\to\infty} \frac{\mathrm{Emp}_n - \overline{\mathrm{Emp}}_n}{n} = 0 \qquad \text{a.s.}$$

(See, e.g., Shiryaev, 1996, Theorem VII.5.4.)

Without loss of generality we can assume that Predictor's move Γ_n at trial n is $\{\hat{y}(x_n)\}$ (where $x \mapsto \hat{y}(x) \in \arg\max_y P(y|x)$ is a fixed "choice function") or the empty set $\emptyset$ or the whole label space $\mathbf{Y}$. Furthermore, we can assume that, at every trial, the predictions are certain for the new objects

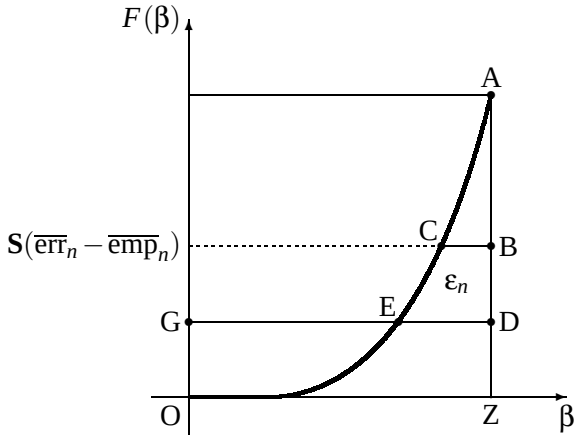


Figure 5: An admissible region predictor. The thick line is the predictability distribution function F; the area of the curvilinear triangle ABC is $\overline{\mathrm{err}}_n - \overline{\mathrm{emp}}_n$; the area of the rectangle DZOG is $\overline{\mathrm{emp}}_n$; the (non-negative) area of the curvilinear quadrangle BDEC is denoted ε_n

above the straight line BC in Figure 5,[3] and that the predictions are empty for the objects below the straight line DG in Figure 5.[4] It is clear that for the region predictor to satisfy (16) it must hold that

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (\varepsilon_i \wedge \overline{\mathrm{emp}}_i) = 0$$

(otherwise $\overline{\mathrm{Unc}}_n$ can be decreased substantially, which contradicts (15); ε_i are defined in the caption of Figure 5), and so we can assume, without loss of generality, that either $\varepsilon_n = 0$ or $\overline{\mathrm{emp}}_n = 0$ at every trial n, i.e., that

$$\overline{\mathrm{unc}}_n = \mathbf{S}(\overline{\mathrm{err}}_n), \quad \overline{\mathrm{emp}}_n = \mathbf{C}(\overline{\mathrm{err}}_n)$$

at every trial.

Let us check that to achieve (16) the region predictor must satisfy

$$\delta < \delta_0 \Longrightarrow \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (\overline{\mathrm{err}}_i - \delta_0)^+ = 0 \tag{20}$$

$$\delta \geq \delta_0 \Longrightarrow \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (\delta_0 - \overline{\mathrm{err}}_i)^+ = 0, \tag{21}$$

where the convergence is, as usual, almost certain. It was shown in Vovk (2002b) (Lemma 2) that the success curve $\mathbf{S}$ is convex, non-increasing, continuous, and has slope at most -1 before it hits

3. More formally, predictions are certain for new extended objects (x,τ) satisfying

$$F(x,\tau) := F(f(x)-) + \tau(F(f(x)+) - F(f(x)-)) \geq \mathbf{S}(\overline{\mathrm{err}}_n - \overline{\mathrm{emp}}_n).$$

Intuitively, considering extended objects makes the vertical axis "infinitely divisible".

4. Indeed, predictions of this kind are admissible in the sense that we cannot improve $\overline{\mathrm{unc}}_n$ and $\overline{\mathrm{emp}}_n$ simultaneously, and all admissible predictions are equivalent to predictions of this kind. A formal argument for the case where emp_n are omitted is given in Vovk (2002b).

the x axis at $\delta = \delta_0$. The second implication, (21), now immediately follows from the fact that, under $\delta \geq \delta_0$ and (16),

$$0 = \limsup_{n\to\infty} \frac{\overline{\mathrm{Unc}}_n}{n} = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbf{S}(\overline{\mathrm{err}}_i) \geq \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (\delta_0 - \overline{\mathrm{err}}_i)^+ .$$

The first implication, (20), can be extracted from the chain

$$\frac{\overline{\mathrm{Unc}}_n}{n} = \frac{1}{n} \sum_{i=1}^{n} \overline{\mathrm{unc}}_i = \frac{1}{n} \sum_{i=1}^{n} \mathbf{S}(\overline{\mathrm{err}}_i) \geq \mathbf{S}\left(\frac{1}{n} \sum_{i=1}^{n} \overline{\mathrm{err}}_i \right) = \mathbf{S}\left(\frac{\overline{\mathrm{Err}}_n}{n} \right) \geq \mathbf{S}(\delta) - \varepsilon \tag{22}$$

(with the last inequality holding almost surely for an arbitrary $\varepsilon > 0$ from some n on) used by Vovk (2002b, in the proof of Theorems 1 and 1r). Indeed, it can be seen from (22) that, assuming the predictor is well-calibrated and optimal and $\delta < \delta_0$,

$$\overline{\mathrm{Err}}_n / n \to \delta \quad \text{a.s.}$$

and, therefore,

$$\begin{aligned}
\mathbf{S}(\delta) &\geq \limsup_{n\to\infty} \frac{\overline{\mathrm{Unc}}_n}{n} = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbf{S}(\overline{\mathrm{err}}_i) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbf{S}(\overline{\mathrm{err}}_i \wedge \delta_0) \\
&\geq \limsup_{n\to\infty} \mathbf{S}\left(\frac{1}{n} \sum_{i=1}^{n} (\overline{\mathrm{err}}_i \wedge \delta_0) \right) = \limsup_{n\to\infty} \mathbf{S}\left(\frac{\overline{\mathrm{Err}}_n}{n} - \frac{1}{n} \sum_{i=1}^{n} (\overline{\mathrm{err}}_i - \delta_0)^+ \right) \\
&= \limsup_{n\to\infty} \mathbf{S}\left(\delta - \frac{1}{n} \sum_{i=1}^{n} (\overline{\mathrm{err}}_i - \delta_0)^+ \right) = \mathbf{S}\left(\delta - \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} (\overline{\mathrm{err}}_i - \delta_0)^+ \right)
\end{aligned}$$

almost surely. This proves (20).

Using (20), (21), and the fact that the complementary success curve $\mathbf{C}$ is concave, increasing, and (uniformly) continuous for $\delta \geq \delta_0$ (see Lemma 7), we obtain: if $\delta < \delta_0$,

$$\begin{aligned}
\frac{\overline{\mathrm{Emp}}_n}{n} &= \frac{1}{n} \sum_{i=1}^{n} \overline{\mathrm{emp}}_i = \frac{1}{n} \sum_{i=1}^{n} \mathbf{C}(\overline{\mathrm{err}}_i) \\
&\leq \frac{1}{n} \mathbf{C}'_{\mathrm{right}}(\delta_0) \sum_{i=1}^{n} (\overline{\mathrm{err}}_i - \delta_0)^+ \to 0 \quad (n \to \infty);
\end{aligned}$$

if $\delta \geq \delta_0$,

$$\begin{aligned}
\frac{\overline{\mathrm{Emp}}_n}{n} &= \frac{1}{n} \sum_{i=1}^{n} \mathbf{C}(\overline{\mathrm{err}}_i) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{C}(\overline{\mathrm{err}}_i \vee \delta_0) \\
&\leq \mathbf{C}\left(\frac{1}{n} \sum_{i=1}^{n} (\overline{\mathrm{err}}_i \vee \delta_0) \right) = \mathbf{C}\left(\frac{1}{n} \sum_{i=1}^{n} \overline{\mathrm{err}}_i + \frac{1}{n} \sum_{i=1}^{n} (\delta_0 - \overline{\mathrm{err}}_i)^+ \right) \\
&\leq \mathbf{C}\left(\frac{1}{n} \sum_{i=1}^{n} \overline{\mathrm{err}}_i \right) + o(1) \leq \mathbf{C}(\delta) + \varepsilon,
\end{aligned}$$

the last inequality holding almost surely for an arbitrary $\varepsilon > 0$ from some n on and δ being the significance level used.

5.3 Proof Sketch of Proposition 6

Let us first modify and extend the notation $P_n^{\neq}(y\,|\,x_i,\sigma_i)$ introduced in (8). Consider the sequence of extended examples $w_i=(x_i,\tau_i',y_i)$, $i=1,\dots,n$ ((x_i,y_i) are the first n examples chosen by Reality and τ_i are the random numbers used by Predictor). We define the Nearest Neighbours approximations $P_n(y\,|\,x,\sigma)$ to the conditional probabilities $P(y\,|\,x)$ as follows: for every $(x,\sigma,y)\in\tilde{\mathbf{Z}}$,

$$P_n(y\,|\,x,\sigma):=N(x,\sigma,y)/K_n, \tag{23}$$

where $N(x,\sigma,y)$ is the number of $i=1,\dots,n$ such that (x_i,τ_i') is among the K_n nearest neighbours of (x,σ) and $y_i=y$ (this time (x_i,τ_i') is not prevented from being counted as one of the K_n nearest neighbours of (x,σ) if $(x_i,\tau_i')=(x,\sigma)$). We define the empirical predictability function f_n by

$$f_n(x,\sigma):=\max_{y\in\mathbf{Y}}P_n(y\,|\,x,\sigma). \tag{24}$$

The proof will be based on the following version of a well-known fundamental result.

Lemma 8 *Suppose $K_n\to\infty$, $K_n=o(n)$, and $\mathbf{Y}=\{0,1\}$. For any $\varepsilon>0$ and large enough n,*

$$\mathbb{P}\left\{\int|P(1\,|\,x)-P_n(1\,|\,x,\sigma)|\,P_{\mathbf{X}}(dx)\mathbf{U}(d\sigma)>\varepsilon\right\}\le e^{-n\varepsilon^2/40},$$

where the outermost probability distribution $\mathbb{P}$ (essentially $(P\times\mathbf{U})^\infty$) generates the extended examples (x_i,τ_i,y_i), which determine the empirical distributions P_n.

Proof This is almost a special case of Devroye et al.'s (1994) Theorem 1. There is, however, an important difference between the way we break distance ties and the way Devroye et al. (1994) do this. In that work, instead of our (3),

$$(|x_1-x_3|,|\sigma_1-\sigma_3|)<(|x_2-x_3|,|\sigma_2-\sigma_3|)$$

is used. (Our way of breaking ties better agrees with the lexicographic order on $[0,1]^2$, which is useful in the proof of Proposition 3 and, less importantly, in the proof of Lemma 10.) It is easy to check that the proof given by Devroye et al. (1994) also works (and becomes simpler) for our way of breaking distance ties. ■

Lemma 9 *Suppose $K_n\to\infty$ and $K_n=o(n)$. For any $\varepsilon>0$ there exists an $\varepsilon^*>0$ such that, for large enough n,*

$$\mathbb{P}\left\{(P_{\mathbf{X}}\times\mathbf{U})\left\{(x,\sigma):\max_{y\in\mathbf{Y}}|P_n(y\,|\,x,\sigma)-P(y\,|\,x)|>\varepsilon\right\}>\varepsilon\right\}\le e^{-\varepsilon^* n};$$

in particular,

$$\mathbb{P}\big\{(P_{\mathbf{X}}\times\mathbf{U})\{(x,\sigma):|f_n(x,\sigma)-f(x)|>\varepsilon\}>\varepsilon\big\}\le e^{-\varepsilon^* n}.$$

Proof We apply Lemma 8 to the binary classification problem obtained from our classification problem by replacing label $y\in\mathbf{Y}$ with 1 and replacing all other labels with 0:

$$\mathbb{P}\left\{\int|P(y\,|\,x)-P_n(y\,|\,x,\sigma)|\,P_{\mathbf{X}}(dx)\mathbf{U}(d\sigma)>\varepsilon\right\}\le e^{-n\varepsilon^2/40}.$$

By Markov's inequality this implies

$$\mathbb{P}\left\{(P_{\mathbf{X}}\times\mathbf{U})\{|P(y\,|\,x)-P_n(y\,|\,x,\sigma)|>\sqrt{\varepsilon}\}>\sqrt{\varepsilon}\right\}\le e^{-n\varepsilon^2/40},$$

which, in turn, implies

$$\mathbb{P}\left\{(P_{\mathbf{X}}\times\mathbf{U})\left\{\max_{y\in\mathbf{Y}}|P(y\,|\,x)-P_n(y\,|\,x,\sigma)|>\sqrt{\varepsilon}\right\}>|\mathbf{Y}|\sqrt{\varepsilon}\right\}\le e^{-n\varepsilon^2/40}.$$

This completes the proof, since we can take the ε in the last equation arbitrarily small as compared to the ε in the statement of the lemma. ■

We will use the shorthand "$\forall^\infty n$" for "from some n on".

Lemma 10 *Suppose $K_n\to\infty$ and $K_n=o(n)$. For any $\varepsilon>0$ there exists an $\varepsilon^*>0$ such that, for large enough n,*

$$\mathbb{P}\left\{\frac{\#\left\{i:\max_y\left|P(y\,|\,x_i)-P_n^{\neq}(y\,|\,x_i,\tau'_i)\right|>\varepsilon\right\}}{n}>\varepsilon\right\}\le e^{-\varepsilon^* n}.$$

In particular,

$$\forall^\infty n:\mathbb{P}\left\{\frac{\#\left\{i:\left|f(x_i)-f_n^{\neq}(x_i,\tau'_i)\right|>\varepsilon\right\}}{n}>\varepsilon\right\}\le e^{-\varepsilon^* n}.$$

Proof Since

$$\left|P_n^{\neq}(y\,|\,x_i,\tau'_i)-P_n(y\,|\,x_i,\tau'_i)\right|\le\frac{1}{K_n}=o(1),$$

we can, and will, ignore the upper indices $^{\neq}$ in the statement of the lemma.

Define

$$I_n(x,\sigma):=\begin{cases}0 & \text{if } \max_y|P(y\,|\,x)-P_n(y\,|\,x,\sigma)|\le\varepsilon\\ 1 & \text{if } \max_y|P(y\,|\,x)-P_n(y\,|\,x,\sigma)|\ge 2\varepsilon\\ (\max_y|P(y\,|\,x)-P_n(y\,|\,x,\sigma)|-\varepsilon)/\varepsilon & \text{otherwise}\end{cases}$$

(intuitively, $I_n(x,\sigma)$ is a "soft version" of $\mathbb{I}_{\{\max_y|P(y|x)-P_n(y|x,\sigma)|>\varepsilon\}}$).

The main tool in this proof (and several other proofs in this section) will be McDiarmid's theorem (see, e.g., Devroye et al., 1996, Theorem 9.2). First we check the possibility of its application. If we replace an extended object (x_j,τ'_j) by another extended object (x_j^*,τ_j^*), the expression

$$\sum_{i=1}^n I_n(x_i,\tau'_i)$$

will change as follows:

- the addend $I_n(x_i,\tau'_i)$ for $i=j$ changes by 1 at most;
- the addends $I_n(x_i,\tau'_i)$ for $i\ne j$ such that neither (x_j,τ'_j) nor (x_j^*,τ_j^*) are among the K_n nearest neighbours of (x_i,τ'_i) do not change at all;

- the sum over the at most $4K_n$ (see below) addends $I_n(x_i,\tau_i')$ for $i\neq j$ such that either (x_j,τ_j') or (x_j^*,τ_j^*) (or both) are among the K_n nearest neighbours of (x_i,τ_i') can change by at most

$$4K_n\frac{1}{\varepsilon}\frac{1}{K_n}=\frac{4}{\varepsilon}. \tag{25}$$

The left-hand side of (25) reflects the following facts: the change in $P_n(y\,|\,x_i,\tau_i')$ for $i\neq j$ is at most $1/K_n$; the number of $i\neq j$ such that (x_j,τ_j') is among the K_n nearest neighbours of (x_i,τ_i') does not exceed $2K_n$ (since the extended objects are linearly ordered and (3) is used for breaking distance ties); analogously, the number of $i\neq j$ such that (x_j^*,τ_j^*) is among the K_n nearest neighbours of (x_i,τ_i') does not exceed $2K_n$.

Therefore, by McDiarmid's theorem,

$$\mathbb{P}\left\{\frac{1}{n}\sum_{i=1}^n I_n(x_i,\tau_i')-\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^n I_n(x_i,\tau_i')\right)>\varepsilon\right\}$$
$$\le\exp\left(-2\varepsilon^2 n/\left(1+4/\varepsilon\right)^2\right)=\exp\left(-\frac{2\varepsilon^4}{(4+\varepsilon)^2}n\right). \tag{26}$$

Next we find:

$$\begin{aligned}&\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^n I_n(x_i,\tau_i')\right)=\mathbb{E}\left(I_n(x_n,\tau_n')\right)\le\mathbb{E}\left(I_{n-1}(x_n,\tau_n')\right)+o(1)\\&\le\mathbb{E}(P_{\mathbf{X}}\times\mathbf{U})\{(x,\sigma):\max_y|P(y\,|\,x)-P_{n-1}(y\,|\,x,\sigma)|>\varepsilon\}+o(1)\\&\le e^{-\varepsilon^* n}+\varepsilon+o(1)\le 2\varepsilon\end{aligned}$$

(the penultimate inequality follows from Lemma 9) from some n on. In combination with (26) this implies

$$\forall^\infty n:\mathbb{P}\left\{\frac{1}{n}\sum_{i=1}^n I_n(x_i,\tau_i')>3\varepsilon\right\}\le\exp\left(-\frac{2\varepsilon^4}{(4+\varepsilon)^2}n\right),$$

in particular

$$\mathbb{P}\left\{\frac{\#\{i:\max_y|P(y\,|\,x_i)-P_n(y\,|\,x_i,\tau_i')|\ge 2\varepsilon\}}{n}>3\varepsilon\right\}\le\exp\left(-\frac{2\varepsilon^4}{(4+\varepsilon)^2}n\right).$$

Replacing 3ε by ε, we obtain that, from some n on,

$$\mathbb{P}\left\{\frac{\#\{i:\max_y|P(y\,|\,x_i)-P_n(y\,|\,x_i,\tau_i')|>\varepsilon\}}{n}>\varepsilon\right\}\le\exp\left(-\frac{2(\varepsilon/3)^4}{(4+\varepsilon/3)^2}n\right),$$

which completes the proof. ■

We say that an extended example (x_i,τ_i,y_i), $i=1,\dots,n$, is *n-strange* if $y_i\neq\hat{y}_n(x_i,\tau_i')$; otherwise, (x_i,τ_i,y_i) will be called *n-ordinary*. We will assume that $(f_n^{\neq}(x_i,\tau_i'),\tau_i'')$, $i=1,\dots,n$, are all different for all n; even more than that, we will assume that τ_i'' are all different (we can do so since the probability of this event is one).

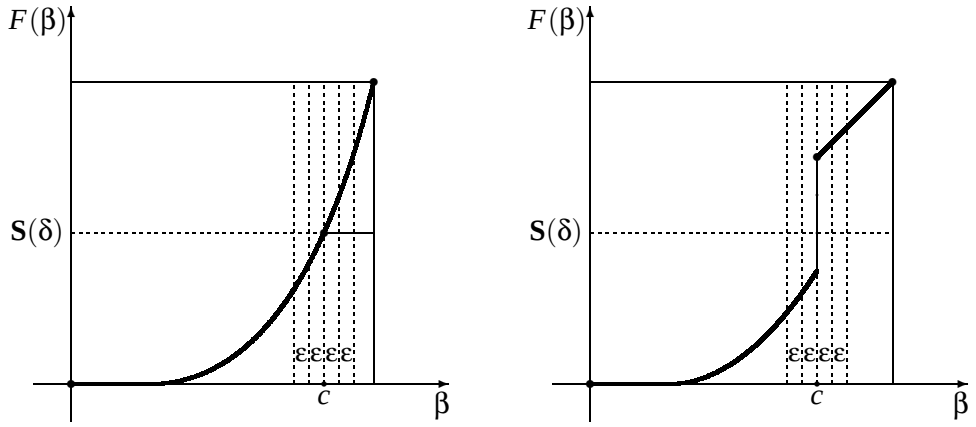


Figure 6: Cases $F(c) = \mathbf{S}(\delta)$ (left) and $F(c) > \mathbf{S}(\delta)$ (right). The vertical bands of width ε determine the division of the first n extended examples into five classes

Lemma 11 *Suppose (7) is satisfied and $\delta \le \delta_0$. With probability one, the $\lfloor (1-\mathbf{S}(\delta))n \rfloor$ extended examples with the largest (in the sense of the lexicographic order) $(f_n^{\neq}(x_i,\tau_i'),\tau_i'')$ among $(x_1,\tau_1,y_1),\dots,(x_n,\tau_n,y_n)$ contain at most $n\delta + o(n)$ n-strange extended examples as $n \to \infty$.*

Proof Define

$$c := \sup\{\beta : F(\beta) \le \mathbf{S}(\delta)\}.$$

It is clear that $0 < c < 1$. Our proof will work both in the case where $F(c) = \mathbf{S}(\delta)$ and in the case where $F(c) > \mathbf{S}(\delta)$, as illustrated in Figure 6.

Let $\varepsilon > 0$ be a small constant (we will let $\varepsilon \to 0$ eventually). Define a "threshold" $(c_n', c_n'') \in [0,1]^2$ requiring that

$$\mathbb{P}\left\{ f(x_n) = c, (f_{n-1}(x_n,\tau_n'),\tau_n'') > (c_n',c_n'') \right\} = F(c) - \mathbf{S}(\delta) - \varepsilon \tag{27}$$

if $F(c) > \mathbf{S}(\delta)$; we assume that ε is small enough for

$$2\varepsilon < F(c) - \mathbf{S}(\delta) \tag{28}$$

to hold . Among other things this will ensure the validity of the definition (27). If $F(c) = \mathbf{S}(\delta)$, we set $(c_n', c_n'') := (c+\varepsilon, 0)$; in any case, we will have

$$\mathbb{P}\left\{ f(x_n) = c, (f_{n-1}(x_n,\tau_n'),\tau_n'') > (c_n',c_n'') \right\} \ge F(c) - \mathbf{S}(\delta) - \varepsilon. \tag{29}$$

Let us say that an extended example (x_i,τ_i,y_i) is *above the threshold* if

$$(f_n^{\neq}(x_i,\tau_i'),\tau_i'') > (c_n',c_n'');$$

otherwise, we say it is *below the threshold*. Divide the first n extended examples (x_i,τ_i,y_i), $i = 1,\dots,n$, into five classes:

Class I: Those satisfying $f(x_i) \le c - 2\varepsilon$.

Class II: Those that satisfy $f(x_i) = c$ and are below the threshold.

Class III: Those satisfying $c - 2\varepsilon < f(x_i) \le c + 2\varepsilon$ but not $f(x_i) = c$.

Class IV: Those that satisfy $f(x_i) = c$ and are above the threshold.

Class V: Those satisfying $f(x_i) > c + 2\varepsilon$.

First we explain the general idea of the proof. The threshold (c', c'') was chosen so that approximately $\lfloor (1 - \mathbf{S}(\delta))n \rfloor$ of the available extended examples will be above the threshold. Because of this, the extended examples above the threshold will essentially be the $\lfloor (1 - \mathbf{S}(\delta))n \rfloor$ extended examples with the largest $(f_n^{\neq}(x_i, \tau_i'), \tau_i'')$ referred to in the statement of the lemma. For each of the five classes we will be interested in the following questions:

- How many extended examples are there in the class?
- How many of those are above the threshold?
- How many of those above the threshold are n-strange?

If the sum of the answers to the last question does not exceed $n\delta$ by too much, we are done.

With this plan in mind, we start the formal proof. (Of course, we will not be following the plan literally: for example, if a class is very small, we do not need to answer the second and third questions.) The first step is to show that

$$c - \varepsilon \le c'_n \le c + \varepsilon \tag{30}$$

from some n on; this will ensure that the classes are conveniently separated from each other. We only need to consider the case $F(c) > \mathbf{S}(\delta)$. The inequality $c'_n \le c + \varepsilon$ follows from

$$\forall^\infty n : \mathbb{P}\left\{ f(x_n) = c, f_{n-1}(x_n, \tau'_n) > c + \varepsilon \right\} < \varepsilon < F(c) - \mathbf{S}(\delta) - \varepsilon$$

Simply combine Lemma 9 with (28). The inequality $c - \varepsilon \le c'_n$ follows in a similar way from

$$\begin{aligned} \forall^\infty n : \quad & \mathbb{P}\left\{ f(x_n) = c, f_{n-1}(x_n, \tau'_n) \ge c - \varepsilon \right\} \\ & = \mathbb{P}\{ f(x_n) = c \} - \mathbb{P}\left\{ f(x_n) = c, f_{n-1}(x_n, \tau'_n) < c - \varepsilon \right\} \\ & > F(c) - F(c-) - \varepsilon \ge F(c) - \mathbf{S}(\delta) - \varepsilon. \end{aligned}$$

Now we are ready to analyze the composition of our five classes. Among the Class I extended examples at most

$$\varepsilon n \tag{31}$$

will be above the threshold from some n on almost surely (by Lemma 10 and the Borel–Cantelli lemma). None of the Class II extended examples will be above the threshold, by definition. The fraction of Class III extended examples among the first n extended examples will tend to

$$F(c + 2\varepsilon) - F(c) + F(c-) - F(c - 2\varepsilon) \tag{32}$$

as $n \to \infty$ almost surely.

To estimate the number N_n^{IV} of Class IV extended examples among the first n extended examples, we use McDiarmid's theorem. If one extended example is replaced by another, N_n^{IV} will change by at most $2K_n+1$ (since this extended example can affect $f_n^{\neq}(x_i,\tau_i')$ for at most $2K_n$ other extended examples (x_i,τ_i,y_i)). Therefore,

$$\mathbb{P}\left\{\left|\frac{1}{n}N_n^{\mathrm{IV}}-\frac{1}{n}\mathbb{E}N_n^{\mathrm{IV}}\right|\ge\varepsilon\right\}\le 2e^{-2\varepsilon^2 n/(2K_n+1)^2};$$

the assumption $K_n=o\left(\sqrt{n/\ln n}\right)$ and the Borel–Cantelli lemma imply that

$$\left|\frac{1}{n}N_n^{\mathrm{IV}}-\frac{1}{n}\mathbb{E}N_n^{\mathrm{IV}}\right|<\varepsilon$$

from some n on almost surely. Since

$$\frac{1}{n}\mathbb{E}N_n^{\mathrm{IV}}=\mathbb{P}\left\{f(x_n)=c,(f_{n-1}(x_n,\tau_n'),\tau_n'')>(c_n',c_n'')\right\}\ge F(c)-\mathbf{S}(\delta)-\varepsilon,$$

as in (29), we have

$$N_n^{\mathrm{IV}}>(F(c)-\mathbf{S}(\delta)-2\varepsilon)n \tag{33}$$

from some n on almost surely. Of course, all these examples are above the threshold.

Now we estimate the number $N_n^{\mathrm{IV,str}}$ of n-strange extended examples of Class IV. Again McDiarmid's theorem implies that

$$\left|\frac{1}{n}N_n^{\mathrm{IV,str}}-\frac{1}{n}\mathbb{E}N_n^{\mathrm{IV,str}}\right|<\varepsilon$$

from some n on almost surely. Now, from some n on,

$$\begin{aligned}
\frac{1}{n}\mathbb{E}N_n^{\mathrm{IV,str}}&=\mathbb{P}\left\{f(x_n)=c,(f_{n-1}(x_n,\tau_n'),\tau_n'')>(c_n',c_n''),\hat{y}_n(x_n,\tau_n')\ne y_n\right\}\\
&=\mathbb{E}\left(\left(1-P_{\mathbf{Y}\mid\mathbf{X}}\left(\hat{y}_n(x_n,\tau_n')\mid x_n\right)\right)\mathbb{I}_{\{f(x_n)=c,(f_{n-1}(x_n,\tau_n'),\tau_n'')>(c_n',c_n'')\}}\right)\\
&\le e^{-\varepsilon^* n}+\varepsilon+(1-c+2\varepsilon)\\
&\quad\times\mathbb{P}\{f(x_n)=c,(f_{n-1}(x_n,\tau_n'),\tau_n'')>(c_n',c_n'')\}\\
&=e^{-\varepsilon^* n}+\varepsilon+(1-c+2\varepsilon)(F(c)-\mathbf{S}(\delta)-\varepsilon) \qquad (34)\\
&\le(F(c)-\mathbf{S}(\delta))(1-c)+4\varepsilon \qquad (35)
\end{aligned}$$

in the case $F(c)>\mathbf{S}(\delta)$; the first inequality in this chain follows from Lemma 9: indeed, this lemma implies that, unless an event of the small probability $e^{-\varepsilon^* n}+\varepsilon$ happens,

$$P\left(\hat{y}_n(x_n,\tau_n')\mid x_n\right)\ge P_{n-1}\left(\hat{y}_n(x_n,\tau_n')\mid x_n,\tau_n'\right)-\varepsilon=f_{n-1}\left(x_n,\tau_n'\right)-\varepsilon\ge f(x_n)-2\varepsilon. \tag{36}$$

If $F(c)=\mathbf{S}(\delta)$, the lines (34) and (35) of that chain have to be changed to

$$\begin{aligned}
&\le e^{-\varepsilon^* n}+\varepsilon+(1-c+2\varepsilon)\,\mathbb{P}\{f(x_n)=c,f_{n-1}(x_n,\tau_n')\ge c+\varepsilon\}\\
&\le e^{-\varepsilon^* n}+\varepsilon+(1-c+2\varepsilon)\left(e^{-\varepsilon^* n}+\varepsilon\right)<4\varepsilon
\end{aligned}$$

(where the obvious modification of Lemma 9 with all "$> \varepsilon$" changed to "$\geq \varepsilon$" is used), but the inequality between the extreme terms of the chain still holds. Therefore, the number of n-strange Class IV extended examples does not exceed

$$((F(c)-\mathbf{S}(\delta))(1-c)+5\varepsilon)\,n \tag{37}$$

from some n on almost surely.

By the Borel strong law of large numbers, the fraction of Class V extended examples among the first n extended examples will tend to

$$1-F(c+2\varepsilon) \tag{38}$$

as $n\to\infty$ almost surely. By Lemma 10, the Borel–Cantelli lemma, and (30), almost surely from some n on at least

$$(1-F(c+2\varepsilon)-2\varepsilon)n \tag{39}$$

extended examples in Class V will be above the threshold.

Finally, we estimate the number $N_n^{\mathrm{V,str}}$ of n-strange extended examples of Class V among the first n extended examples. By McDiarmid's theorem,

$$\left|\frac{1}{n}N_n^{\mathrm{V,str}}-\frac{1}{n}\,\mathbb{E}\,N_n^{\mathrm{V,str}}\right|<\varepsilon$$

from some n on almost surely. Now

$$\begin{aligned}
\frac{1}{n}\,\mathbb{E}\,N_n^{\mathrm{V,str}} &= \mathbb{P}\left\{f(x_n)>c+2\varepsilon, \hat{y}_n(x_n,\tau_n')\neq y_n\right\}\\
&= \mathbb{E}\left(\left(1-P_{\mathbf{Y}|\mathbf{X}}\left(\hat{y}_n(x_n,\tau_n')\,|\,x_n\right)\right)\mathbb{I}_{\{f(x_n)>c+2\varepsilon\}}\right)\\
&\leq e^{-\varepsilon^* n}+\varepsilon+\mathbb{E}\left((1-f(x_n)+2\varepsilon)\,\mathbb{I}_{\{f(x_n)>c+2\varepsilon\}}\right)\\
&\leq e^{-\varepsilon^* n}+3\varepsilon+\mathbb{E}\left((1-f(x_n))\,\mathbb{I}_{\{f(x_n)>c+2\varepsilon\}}\right)\\
&= e^{-\varepsilon^* n}+3\varepsilon+\int_0^1 (F(\beta)-F(c+2\varepsilon))^+ d\beta\\
&< \int_0^1 (F(\beta)-F(c))^+ d\beta+4\varepsilon
\end{aligned}$$

from some n on. The first inequality follows from Lemma 9, as in (36). Therefore,

$$\frac{1}{n}N_n^{\mathrm{V,str}}<\int_0^1 (F(\beta)-F(c))^+ d\beta+5\varepsilon \tag{40}$$

from some n on almost surely.

Summarizing, we can see that the total number of extended examples above the threshold among the first n extended examples will be at least

$$(F(c)-\mathbf{S}(\delta)-2\varepsilon+1-F(c+2\varepsilon)-2\varepsilon)\,n=(1-\mathbf{S}(\delta)+F(c)-F(c+2\varepsilon)-4\varepsilon)\,n \tag{41}$$

(see (33) and (39)) from some n on almost surely. The number of n-strange extended examples among them will not exceed

$$\begin{aligned}\Bigg(&\varepsilon + F(c+2\varepsilon) - F(c) + F(c-) - F(c-2\varepsilon) + \varepsilon \\ &\quad + (F(c) - \mathbf{S}(\delta))(1-c) + 5\varepsilon + \int_0^1 (F(\beta) - F(c))^+ d\beta + 5\varepsilon\Bigg) n \\ &= \Bigg(F(c+2\varepsilon) - F(c) + F(c-) - F(c-2\varepsilon) \\ &\quad + (F(c) - \mathbf{S}(\delta))(1-c) + \int_0^1 (F(\beta) - F(c))^+ d\beta + 12\varepsilon\Bigg) n \quad (42)\end{aligned}$$

(see (31), (32), (37), and (40)) from some n on almost surely. Combining (41) and (42), we can see that the number of n-strange extended examples among the $\lfloor (1 - \mathbf{S}(\delta))n \rfloor$ extended examples with the largest $(f_n^{\neq}(x_i, \tau_i'), \tau_i'')$ does not exceed

$$\begin{aligned}\Bigg(&F(c+2\varepsilon) - F(c) + F(c-) - F(c-2\varepsilon) + (F(c) - \mathbf{S}(\delta))(1-c) \\ &\quad + \int_0^1 (F(\beta) - F(c))^+ d\beta + 12\varepsilon\Bigg) n + (F(c+2\varepsilon) - F(c) + 4\varepsilon)\, n \\ &= \Bigg(2(F(c+2\varepsilon) - F(c)) + (F(c-) - F(c-2\varepsilon)) + (F(c) - \mathbf{S}(\delta))(1-c) \\ &\quad + \int_0^1 (F(\beta) - F(c))^+ d\beta + 16\varepsilon\Bigg) n\end{aligned}$$

from some n on almost surely. Since ε can be arbitrarily small, the coefficient in front of n in the last expression can be made arbitrarily close to

$$(F(c) - \mathbf{S}(\delta))(1-c) + \int_0^1 (F(\beta) - F(c))^+ d\beta = \int_0^1 (F(\beta) - \mathbf{S}(\delta))^+ d\beta = \delta,$$

which completes the proof. ■

Lemma 12 *Suppose (7) is satisfied. The fraction of n-strange extended examples among the first n extended examples (x_i, τ_i, y_i) approaches δ_0 asymptotically with probability one.*

Proof sketch The lemma is not difficult to prove using McDiarmid's theorem and the fact that, by Lemma 10, $P(\hat{y}_n(x_i, \tau_i') \,|\, x_i)$ will typically differ little from $f(x_i)$. Notice, however, that the part that we really need in this paper (that the fraction of n-strange extended examples does not exceed $\delta_0 + o(1)$ as $n \to \infty$ with probability one) is just a special case of Lemma 11, corresponding to $\delta = \delta_0$. ■

Lemma 13 *Suppose (7) is satisfied and $\delta > \delta_0$. The fraction of n-ordinary extended examples among the $\lfloor \mathbf{C}(\delta) n \rfloor$ extended examples (x_i, τ_i, y_i), $i = 1, \ldots, n$, with the lowest $(f_n^{\neq}(x_i, \tau_i'), \tau_i'')$ does not exceed $\delta - \delta_0 + o(1)$ as $n \to \infty$ with probability one.*

Lemma 13 can be proved analogously to Lemma 11.

Lemma 14 *Let $\mathcal{F}_1 \supseteq \mathcal{F}_2 \supseteq \cdots$ be a decreasing sequence of σ-algebras and $\xi_1, \xi_2 \ldots$ be a bounded adapted (in the sense that ξ_n is $\mathcal{F}_n$-measurable for all n) sequence of random variables such that*

$$\limsup_{n\to\infty} \mathbb{E}(\xi_n \mid \mathcal{F}_{n+1}) \le 0 \quad a.s.$$

Then

$$\limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \xi_i \le 0 \quad a.s.$$

Proof Replacing, if necessary, ξ_n by $\xi_n - \mathbb{E}(\xi_n \mid \mathcal{F}_{n+1})$, we reduce our task to the following special case (a reverse Borel strong law of large numbers): if $\xi_1, \xi_2, \ldots$ is a bounded *reverse martingale difference*, in the sense of being adapted and satisfying $\forall n : \mathbb{E}(\xi_n \mid \mathcal{F}_{n+1}) = 0$, then

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \xi_i = 0 \quad \text{a.s.} \tag{43}$$

Fix a bounded reverse martingale difference $\xi_1, \xi_2, \ldots$; our goal is to prove (43). By the martingale version of Hoeffding's inequality (Devroye et al., 1996, Theorem 9.1) applied to the martingale difference $(\xi_i, \mathcal{F}_i)$, $i = n, \ldots, 1$,

$$\mathbb{P}\left\{ \left| \frac{1}{n} \sum_{i=1}^{n} \xi_i \right| \ge \varepsilon \right\} \le 2e^{-2\varepsilon^2 n/(2C)^2}, \tag{44}$$

where C is an upper bound on $\sup_n |\xi_n|$. Combined with the Borel–Cantelli–Lévy lemma, (44) implies (43). ∎

Now we can sketch the proof of Proposition 6. Define $\mathcal{F}_n$, $n = 1, 2, \ldots$, to be the σ-algebra on $\tilde{\mathbf{Z}}^\infty$ generated by the multiset of the first $n-1$ extended examples (x_i, τ_i, y_i), $i = 1, \ldots, n-1$, and the sequence of extended examples (x_i, τ_i, y_i), $i = n, n+1, \ldots$ (starting from the nth extended example).

Suppose first that $\delta < \delta_0$. Consider the $\lfloor (1 - \mathbf{S}(\delta - \varepsilon))n \rfloor$ extended examples with the largest $(f_n^{\neq}(x_i, \tau_i'), \tau_i'')$ among $(x_1, \tau_1, y_1), \ldots, (x_n, \tau_n, y_n)$, where $\varepsilon \in (0, \delta)$ is a small constant. Let us show that each of these examples will be predicted with certainty from the other extended examples in the sequence $(x_1, \tau_1, y_1), \ldots, (x_n, \tau_n, y_n)$, from some n on. We will be assuming n large enough.

Let (x_k, τ_k, y_k) be the extended example with the $(\lfloor (\delta - \varepsilon/2)n \rfloor + 1)$th largest (in the sense of the lexicographic order) $(f_n^{\neq}(x_i, \tau_i'), \tau_i'')$ among all n-strange extended examples (x_i, τ_i, y_i), $i = 1, \ldots, n$. (Remember that all τ_i'' are assumed to be different.) Let (x_j, τ_j, y_j) be one of the $\lfloor (1 - \mathbf{S}(\delta - \varepsilon))n \rfloor$ extended examples with the largest $(f_n^{\neq}(x_i, \tau_i'), \tau_i'')$ and let $y \in \mathbf{Y}$ be a label different from $\hat{y}_n(x_j, \tau_j')$. It suffices to prove that

$$\tau_j'' \ge \frac{\#\{i = 1, \ldots, n : \alpha_i^y \ge \alpha_j^y\} - n\delta}{\#\{i = 1, \ldots, n : \alpha_i^y = \alpha_j^y\}} \tag{45}$$

(cf. (6) on p. 582), where all α^y are computed as α in (11) from the sequence

$$(x_1, \tau_1, y_1), \ldots, (x_n, \tau_n, y_n)$$

with y_j replaced by y. It will be more convenient to write (45) in the form

$$\#\{i : \alpha_i^y > \alpha_j^y\} + (1-\tau_j'')\#\{i : \alpha_i^y = \alpha_j^y\} \le n\delta.$$

Since $\alpha_j^y = f_n^{\neq}(x_j, \tau_j')$ and $\alpha_i^y \neq \alpha_i$ for at most $2K_n + 1$ values of i (indeed, changing y_j will affect at most $2K_n + 1$ αs), it suffices to prove

$$\#\{i : \alpha_i > f_n^{\neq}(x_j, \tau_j')\} + (1-\tau_j'')\#\{i : \alpha_i = f_n^{\neq}(x_j, \tau_j')\} \le n(\delta - \varepsilon^*), \tag{46}$$

where $\varepsilon^* \ll \varepsilon$ is a positive constant.

Since $(f_n^{\neq}(x_j, \tau_j'), \tau_j'') \ge (\alpha_k, \tau_k'')$ (indeed, by Lemma 11, there are less than $(\delta - \varepsilon/2)n$ n-strange extended examples among the $\lfloor (1 - \mathbf{S}(\delta - \varepsilon))n \rfloor$ extended examples with the largest $(f_n^{\neq}(x_i, \tau_i'), \tau_i'')$), (46) will follow from

$$\#\{i : \alpha_i > \alpha_k\} + (1-\tau_k'')\#\{i : \alpha_i = \alpha_k\} \le n(\delta - \varepsilon^*). \tag{47}$$

If $\#\{i : \alpha_i = \alpha_k\} \le \frac{\varepsilon}{3} n$, the left-hand side of (47) does not exceed

$$\left(\delta - \frac{\varepsilon}{2}\right) n + \frac{\varepsilon}{3} n < n(\delta - \varepsilon^*),$$

so we can, and will, assume without loss of generality that

$$\#\{i : \alpha_i = \alpha_k\} > \frac{\varepsilon}{3} n. \tag{48}$$

Since τ_i'' for the extended examples satisfying $\alpha_i = \alpha_k$ are output according to the uniform distribution $\mathbf{U}$, the expected value of $1 - \tau_k''$ is about

$$\frac{(\delta - \varepsilon/2)n - \#\{i : \alpha_i > \alpha_k\}}{\#\{i : \alpha_i = \alpha_k\}},$$

and so by Hoeffding's inequality and the Borel–Cantelli lemma we will have (from some n on)

$$1 - \tau_k'' \le \frac{(\delta - \varepsilon/2)n - \#\{i : \alpha_i > \alpha_k\}}{\#\{i : \alpha_i = \alpha_k\}} + \varepsilon^*, \tag{49}$$

remembering (48). Equation (47) will hold because its left-hand side can be transformed using (49) as

$$\begin{aligned} \#\{i : \alpha_i > \alpha_k\} + (1-\tau_k'')\#\{i : \alpha_i = \alpha_k\} &\le (\delta - \varepsilon/2)n + \varepsilon^* \#\{i : \alpha_i = \alpha_k\} \\ &\le (\delta - \varepsilon/2 + \varepsilon^*)n \le (\delta - \varepsilon^*)n. \end{aligned}$$

The assertion we have just proved means that, almost surely from some n on,

$$\mathbb{P}(\{\mathrm{unc}_n = 0\} \mid \mathcal{F}_{n+1}) \ge \frac{\lfloor (1 - \mathbf{S}(\delta - \varepsilon))n \rfloor}{n} \ge 1 - \mathbf{S}(\delta - \varepsilon) - \frac{1}{n}.$$

Since ε can be arbitrarily small and $\mathbf{S}$ is continuous (Vovk, 2002b, Lemma 2), this implies

$$\limsup_{n \to \infty} \mathbb{E}(\mathrm{unc}_n \mid \mathcal{F}_{n+1}) \le \mathbf{S}(\delta) \quad \text{a.s.}$$

By Lemma 14 this implies, in turn,

$$\limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathrm{unc}_i \le \mathbf{S}(\delta) \quad \text{a.s.},$$

which coincides with (17).

If $\delta \ge \delta_0$, Lemma 12 implies that

$$\lim_{n\to\infty} \mathbb{E}(\mathrm{unc}_n \mid \mathcal{F}_{n+1}) = 0 \quad \text{a.s.}$$

(and actually $\mathbb{E}(\mathrm{unc}_n \mid \mathcal{F}_{n+1}) = 0$ from some n on if $\delta > \delta_0$); in combination with Lemma 14 this again implies (17).

Inequality (18) is treated in a similar way to (17). Lemmas 12 and 13 imply that

$$\liminf_{n\to\infty} \mathbb{E}(\mathrm{emp}_n \mid \mathcal{F}_{n+1}) \ge \mathbf{C}(\delta) \quad \text{a.s.} \tag{50}$$

(this inequality is vacuously true when $\delta \le \delta_0$). Another application of Lemma 14 gives

$$\liminf_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \mathrm{emp}_i \ge \mathbf{C}(\delta) \quad \text{a.s.},$$

i.e., (18).

Remark The derivation of Proposition 6 from Lemmas 11–14 would be very simple if we defined the individual strangeness measure by, say,

$$\alpha_i := \begin{cases} (-f_n^{\neq}(x_i,\sigma_i),\sigma_i) & \text{if } y_i = \hat{y}_n(x_i,\sigma_i) \\ (f_n^{\neq}(x_i,\sigma_i),\sigma_i) & \text{otherwise} \end{cases}$$

(with the lexicographic order on the α's) instead of (11) (in which case the denominator of (6) would be 1 almost surely). Our definition (11), however, is simpler and, most importantly, facilitates the proof of Proposition 3. Another simplification would be to use Lemma 11 (applied to $\delta := \delta - \mathbf{C}(\delta)$) instead of Lemma 13 in the derivation of (50); we preferred a more symmetric picture.

6. Conclusion

We have shown that there exist universal well-calibrated region predictors, thus satisfying, to some degree, the desiderata mentioned in §1: well-calibratedness and optimal performance. Notice, however, that the ways in which these two desiderata are satisfied are very different: the well-calibratedness holds in a very specific finitary sense, since the errors have probability δ and are independent, whereas the optimal performance is achieved only asymptotically.

An important direction of further research is to obtain non-asymptotic results about TCM's optimality. A natural setting is where we have a Bayesian model for Reality's strategy, $\{P_\theta : \theta \in \Theta\}$ with a prior $\mu(d\theta)$ on Θ, and our goal is to minimize Unc_n^δ under this model. The intuition behind this setting is that we do not really believe that the data is generated from our model and so prefer a predictor that is well-calibrated regardless the correctness of the model; but if the model is correct, we would like to have an optimal performance. A special case of this setting, with $\mu(d\theta)$

concentrated at one point, was considered in Vovk (2002b); however, all results in that paper are asymptotic.

Acknowledgments

This is a full version of the conference paper (Vovk, 2003); I am grateful to both sets of anonymous referees (for the conference and journal versions), whose comments and suggestions helped to improve the quality of presentation and correct several mistakes. This work was partially supported by EPSRC (grant GR/R46670/01), BBSRC (grant 111/BIO14428), and EU (grant IST-1999-10226).

Appendix A. Notation

The following table contains, strictly speaking, not only the notation used in this paper but also the preferred use of symbols.

$\mathbf{X}$	object space
$\mathbf{Y}$	label space
$\mathbf{Z}$	example space ($\mathbf{Z} = \mathbf{X} \times \mathbf{Y}$)
P	the probability distribution in $\mathbf{Z}$ generating individual examples $z_1 = (x_1, y_1), z_2 = (x_2, y_2), \ldots$
δ	significance level
Γ_n^δ	prediction region
err_n^δ	indicator of error at trial n
unc_n^δ	indicator of uncertain prediction at trial n
emp_n^δ	indicator of empty prediction at trial n
Err_n^δ	cumulative number of errors up to trial n
Unc_n^δ	cumulative number of uncertain predictions up to trial n
Emp_n^δ	cumulative number of empty predictions up to trial n
τ_n	the nth random number used by a region predictor
τ'_n, τ''_n	two components of τ_n, as defined in §3.1
$\tilde{\mathbf{X}}$	the extended object space $\mathbf{X} \times [0,1]$
$\tilde{\mathbf{Z}}$	the extended example space $\mathbf{X} \times [0,1] \times \mathbf{Y}$
$<$, $\leq$	may refer to the lexicographic order on $[0,1]^2$, as defined on p. 581
$\lvert x, \sigma \rvert$	the absolute value of $(x, \sigma) \in [0,1]^2$, as defined in (4)
A_n	individual strangeness measure
α_i	values taken by an individual strangeness measure
$\#E$	the size of set E
K_n	the number of nearest neighbours taken into account at trial n
$P_n^{\neq}(y \mid x_i, \sigma_i)$	empirical estimate of $P(y \mid x_i)$ without taking y_i into account, as defined in (8)
$f_n^{\neq}(x_i, \sigma_i)$	corresponding empirical predictability function, (9)
$\hat{y}_n(x_i, \sigma_i)$	"choice function", as defined in (10)
Δ	finite set of significance levels
$P_{\mathbf{X}}$	the marginal distribution of P in $\mathbf{X}$

$P_{\mathbf{Y}\mid\mathbf{X}}$	the regular conditional distribution of $y \in \mathbf{Y}$ given $x \in \mathbf{X}$, where (x,y) is distributed as P
$f(x)$	predictability of object x
$F(\beta)$	predictability distribution function
$\mathbf{S}(\delta)$	success curve, defined in (12)
$\mathbf{C}(\delta)$	complementary success curve, defined in (13)
δ_0	critical significance level, defined in (14)
$\overline{\text{err}}$, $\overline{\text{unc}}$, $\overline{\text{emp}}$	"predictable" versions of err, unc, emp, as defined on p. 588
$\overline{\text{Err}}$, $\overline{\text{Unc}}$, $\overline{\text{Emp}}$	"predictable" versions of Err, Unc, Emp
$F(t-)$	the limit of $F(u)$ as u approaches t from below
$F(t+)$	the limit of $F(u)$ as u approaches t from above
$u \vee v$	the maximum of u and v, also denoted $\max(u,v)$
$u \wedge v$	the minimum of u and v, also denoted $\min(u,v)$
t^+	$t \vee 0$
t^-	$(-t) \vee 0$
$\mathbf{U}$	the uniform probability distribution in $[0,1]$
$P_n(y\mid x,\sigma)$	empirical estimate of $P(y\mid x)$, defined by (23)
$f_n(x,\sigma)$	corresponding empirical predictability function, defined by (24)
$\mathbb{P}$	probability
$\mathbb{E}$	expectation
$\forall^\infty n$	from some n on
$\mathbb{I}_E$	the indicator function of set E

References

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

David R. Cox and David V. Hinkley. *Theoretical Statistics*. Chapman and Hall, London, 1974.

Luc Devroye, László Györfi, Adam Krzyżak, and Gábor Lugosi. On the strong universal consistency of nearest neighbor regression function estimates. *Annals of Statistics*, 22:1371–1385, 1994.

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

Yoav Freund, Yishay Mansour, and Robert E. Schapire. Generalization bounds for averaged classifiers. *Annals of Statistics*, 32(4), 2004.

Ronald L. Rivest and Robert H. Sloan. Learning complicated concepts reliably and usefully. In *Proceedings of the First Annual Conference on Computational Learning Theory*, pages 69–79, San Mateo, CA, 1988. Morgan Kaufmann.

Craig Saunders, Alex Gammerman, and Vladimir Vovk. Transduction with confidence and credibility. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, volume 2, pages 722–726. Morgan Kaufmann, 1999.

Albert N. Shiryaev. *Probability*. Springer, New York, second edition, 1996.

Vladimir Vovk. On-line Confidence Machines are well-calibrated. In *Proceedings of the Forty Third Annual Symposium on Foundations of Computer Science*, pages 187–196, Los Alamitos, CA, 2002a. IEEE Computer Society.

Vladimir Vovk. Asymptotic optimality of Transductive Confidence Machine. In *Proceedings of the Thirteenth International Conference on Algorithmic Learning Theory*, volume 2533 of *Lecture Notes in Artificial Intelligence*, pages 336–350, Berlin, 2002b. Springer.

Vladimir Vovk. Universal well-calibrated algorithm for on-line classification. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Learning Theory and Kernel Machines: Sixteenth Annual Conference on Learning Theory and Seventh Kernel Workshop*, volume 2777 of *Lecture Notes in Artificial Intelligence*, pages 358–372, Berlin, 2003. Springer.

Vladimir Vovk, Alex Gammerman, and Craig Saunders. Machine-learning applications of algorithmic randomness. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 444–453, San Francisco, CA, 1999. Morgan Kaufmann.

Journal of Machine Learning Research 5 (2004) 605-621 Submitted 3/03; Revised 10/03; Published 6/04

New Techniques for Disambiguation in Natural Language and Their Application to Biological Text

Filip Ginter FILIP.GINTER@IT.UTU.FI
Jorma Boberg JORMA.BOBERG@IT.UTU.FI
Jouni Järvinen JOUNI.JARVINEN@IT.UTU.FI
Tapio Salakoski TAPIO.SALAKOSKI@IT.UTU.FI
Department of Information Technology, University of Turku, and
The Turku Centre for Computer Science (TUCS)
Lemminkäisenkatu 14A
20520 Turku, Finland

Editor: William W. Cohen

Abstract

We study the problems of disambiguation in natural language, focusing on the problem of gene vs. protein name disambiguation in biological text and also considering the problem of context-sensitive spelling error correction. We introduce a new family of classifiers based on ordering and weighting the feature vectors obtained from word counts and word co-occurrence in the text, and inspect several concrete classifiers from this family. We obtain the most accurate prediction when weighting by positions of the words in the context. On the gene/protein name disambiguation problem, this classifier outperforms both the Naive Bayes and SNoW baseline classifiers. We also study the effect of the smoothing techniques with the Naive Bayes classifier, the collocation features, and the context length on the classification accuracy and show that correct setting of the context length is important and also problem-dependent.

Keywords: biological text, gene vs. protein name disambiguation, textual data mining, word sense disambiguation, context-sensitive spelling error correction

1. Introduction

Disambiguation in natural language is a general problem of resolving the ambiguity present in natural language. The problems are, for example, word sense disambiguation, context-sensitive spelling error correction, the more special problem of gene/protein name disambiguation, and many other related problems.

Word sense disambiguation (WSD) is a long studied problem in the natural language processing community and it is important especially in the areas of information extraction and text understanding research. Given an ambiguous word in a text, the task of word sense disambiguation is to decide which of the several possible senses the word takes in this given instance. An often used example is the word “bank”. Bank can be a river bank, it can be a financial institution, or it can be the house in which the financial institution resides.

One common and closely related problem to WSD is context-sensitive spelling error correction such that the misspelled variant of the original word belongs to the language. Consider, for example, misspelling the word *dessert* as *desert*. Because *desert* belongs to the English lexicon, the traditional

lexicon-based spell-checkers will fail to discover the spelling error. A set of similar and correct English words, which are commonly misspelled, is called *confusion set*, for example, {"dessert", "desert"}. The task of context-sensitive spelling correction is to choose, for an instance of a word in text, its correct spelling from its confusion set. Thus, for example, whenever the word *dessert* is encountered in the text, the context-sensitive spell-checker decides, whether the correct spelling is *dessert* or *desert* in this case. It is easy to cast this problem as WSD: each word of the confusion set is considered as a "sense".

In this paper, we concentrate on another related disambiguation problem arising from the area of biological text, where very often a protein carries the same name as the gene which codes the protein. The application of WSD here is to decide whether the given gene/protein name in the given context refers to the sense "gene" or to the sense "protein". Hatzivassiloglou et al. (2001) give as an example the following two sentences: "By UV cross-linking and immunoprecipitation, we show that SBP2 specifically binds selenoprotein mRNAs both in vitro and in vivo." "The SBP2 clone used in this study generates a 3173 nt transcript (2541 nt of coding sequence plus a 632 nt 3′ UTR truncated at the polyadenylation site)." In the first sentence the occurrence of SBP2 is a protein, while the occurrence of SBP2 in the second sentence is a gene. Often the gene/protein name ambiguity is also an issue for human readers, as evidenced by the occasional inclusion of disambiguating information (for example "the SBP2 gene") by the authors of an article, and by the establishment of typographic conventions involving capitalization or italicizing by some journals. However, the authors do not follow these conventions faithfully, and therefore they cannot serve as a disambiguation technique for genes and proteins. Further, the italicizing is not preserved in the plain text format in which the PubMed abstracts are accessible.

Lexical disambiguation problems other than WSD can also be used as alternatives for the purpose of evaluating WSD systems. For example, Yarowsky (1994) uses the problem of restoring accents in Spanish and French texts as a substitute for the WSD problem. Similarly, we cast context-sensitive spelling error correction and gene/protein name disambiguation as alternatives to the WSD problem and use WSD terminology. We will refer to the work of Hatzivassiloglou et al. (2001), which offers a basic insight into the various existing methods used to solve the WSD problem for gene/protein name disambiguation. A more exhaustive review of existing methods is presented, for example, by Manning and Schütze (1999). Hatzivassiloglou et al. (2001) compare three existing state-of-the-art machine learning methods used for the gene/protein disambiguation problem: the C4.5 implementation of decision tree learning (Quinlan, 1993), the RIPPER implementation of inductive rule learning (Cohen, 1996), and the Naive Bayes classifier (e.g., Gale et al., 1992; Manning and Schütze, 1999). They experimentally show that the Naive Bayes classifier and the C4.5 classifier perform essentially with the same accuracy, whereas the RIPPER classifier gains accuracy of about 2% lower than the other two classifiers. However, the Naive Bayes classifier is considerably faster than the C4.5 classifier, both in training and in prediction. Therefore, they decided to use the Naive Bayes classifier in their subsequent experiments.

Here we propose a family of classifiers for the word sense disambiguation task. The classifiers are based on ordering and weighting of the feature vectors obtained from word counts and word co-occurrence in the text. We experimentally evaluate several classifiers from the family and compare their performance with the Naive Bayes classifier. As a second baseline method, we employ the SNoW learning architecture. Although SNoW is not a commonly used approach to the WSD problem, it is an efficient method shown to perform well on WSD by Golding and Roth (1999).

The performance of the proposed methods, Naive Bayes classifier with six smoothing techniques, and the SNoW classifier are studied carefully on the gene/protein name disambiguation task. We consider the effect of the context length and the use of collocations on the accuracy of these methods. The proposed method that incorporates the information about positions of words in the context of the word to be disambiguated into the decision function proves to be the best, demonstrating that the positional information improves the accuracy of the classification. In addition to the gene/protein name disambiguation problem, we test the performance of all the classifiers on the context-sensitive spelling error correction problem.

The paper is organized in the following way. In Section 2 we present the new classifiers for the WSD problem and in Section 3 we consider collocation features. Section 4 is devoted to a brief introduction to the baseline methods. In Section 5 we describe the experimental setting, the data used and the cross-validation method, and present the results for gene/protein name disambiguation and context-sensitive spelling error correction. We discuss the results and propose possible directions for the future research in Section 6.

2. The Proposed Method

Let w be the term whose sense we are disambiguating and let $s_1,\ldots,s_K$ be the K possible senses the term w can take. Let further $\overline{w}=(w_1,\ldots,w_n)$ be the context of the term w, where n is a positive even integer. The context words $w_1,\ldots,w_{\frac{n}{2}}$ are the words immediately preceding the term w in the text and the context words $w_{\frac{n}{2}+1},\ldots,w_n$ are the words immediately following the term w in the text. The context words appear in the vector $\overline{w}$ in the same order as in the text. The term w itself is not included to its context unless another instance of w belongs to the context $\overline{w}$. Let T be a training text, where the occurrences of the term w are labeled with their correct sense. We say that a *sense* s_k *has occurred* in the training text T, when a term w labeled with the sense s_k has occurred in T. Similarly, we say that a word w_i *belongs to a context of a sense* s_k, if it belongs to a context of a term w labeled with the sense s_k. Further, let $\mathrm{count}(w_i,s_k)$ be the number of occurrences of the word w_i in contexts of the sense s_k in the training text T (every occurrence of w_i is counted, also in case w_i appears more than once in the same context) and let $\mathrm{count}(s_k)$ be the number of occurrences of the sense s_k in the training corpus T.

For each context word w_i and sense s_k, we want to be able to measure to what extent the occurrence of the word w_i in the context of w suggests that the term w takes the sense s_k in this context. From the training text T we can count how many times the word w_i appeared in the context of the sense s_k. We call this number *evidence* and define it by

$$\mathrm{ev}(w_i,s_k)=\mathrm{count}(w_i,s_k)\ .$$

In order to estimate to what extent the word w_i is positively (or negatively) tied to the sense s_k, we define *expectation* which measures how many times the word w_i would be expected to appear in a context of the sense s_k, if w_i and s_k were independent, that is, if w_i was neither positively nor negatively tied to s_k. The expectation is defined by

$$\mathrm{ex}(w_i,s_k)=\frac{\mathrm{count}(w_i)}{|T|}\cdot\mathrm{total}(s_k)\ , \tag{1}$$

where $\mathrm{count}(w_i)$ is the number of times the word w_i occurred in the training text T, $|T|$ is the number of words in T, and $\mathrm{total}(s_k)$ is the number of the words in all the contexts of the sense s_k in T so that

in overlapping contexts the common words are counted only once. Multiplying total(s_k) with the relative frequency of w_i, we get how many of the occurrences of w_i would be expected to belong to a context of the sense s_k, if both w_i and s_k were evenly distributed in T and conditionally independent.

The more the evidence and the expectation differ, the more the word w_i is either positively or negatively tied to the sense s_k. We define a function

$$f(w_i,s_k) = \begin{cases} \frac{\text{ev}(w_i,s_k)-\text{ex}(w_i,s_k)}{\text{count}(w_i)\cdot\text{count}(s_k)} & \text{if count}(w_i) \neq 0, \\ 0 & \text{if count}(w_i) = 0, \end{cases}$$

which computes the *feature value* of the word w_i with respect to the sense s_k. The nominator measures the difference between the evidence (i.e., the really observed co-occurrence) and the expected co-occurrence. The denominator normalizes the value. The normalization step is important because the same value of $\text{ev}(w_i,s_k) - \text{ex}(w_i,s_k)$ has different significance for differently represented words and senses. For example, a difference of 10 for a word which has occurred 1000 times is less significant than a difference of 10 for a word which has occurred 100 times. In the former case the difference makes only 1% of the occurrences of the word, whereas in the latter case the difference makes 10% of the occurrences of the word. For similar reasons we also normalize by count(s_k) to account for different frequencies of senses. The normalization step assures that the feature values are of the same magnitude and comparable with each other.

A positive value of $f(w_i,s_k)$ corresponds the situation when the word w_i appears in the context of the sense s_k more often than would be by random and thus the word w_i is positively tied to the sense s_k. Similarly, a negative value of $f(w_i,s_k)$ means that w_i is negatively tied to s_k. If the value of $f(w_i,s_k)$ equals to zero, then w_i is in no way tied to s_k and brings no information as to what sense the given context belongs to.

The problem of zero-counts needs to be addressed since the value of count(w_i) becomes zero for words which do not appear in the training text T. It can be done by setting $f(w_i,s_k) = 0$ whenever count$(w_i) = 0$. Note that count(s_k) never becomes zero, as it is assumed that at least one training case for each sense exists. The reason why $f(w_i,s_k)$ is defined to be zero for unknown words is intuitive. The value zero means that there is no dependence between w_i and s_k, and thus the word w_i does not provide any information for the disambiguation.

In order to disambiguate between different possible senses of the term w, we consider its context $\bar{w} = (w_1,\ldots,w_n)$. Let us define a vector $f^k = (f_1^k,\ldots,f_n^k)$, where

$$f_i^k = f(w_i,s_k) \text{ for all } 1 \leq i \leq n.$$

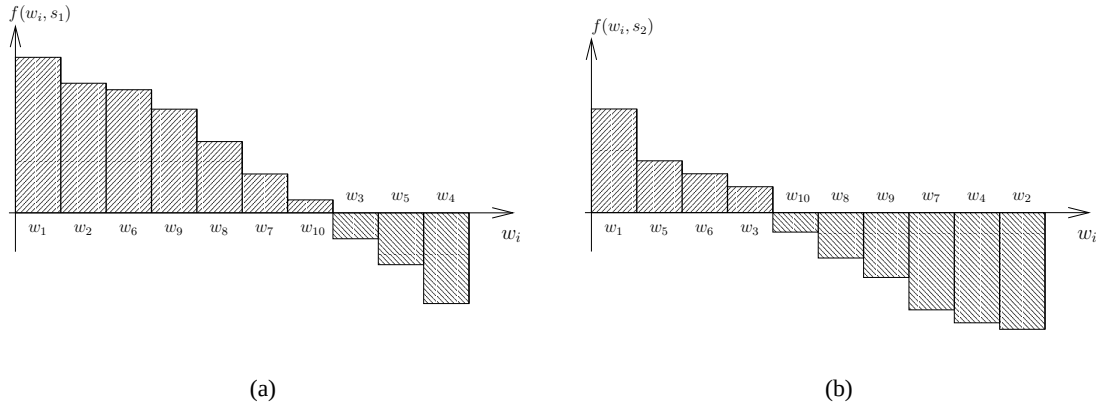
The vector f^k expresses how much the context words $w_1,\ldots,w_n$ are positively or negatively tied to the sense s_k. For each context $\bar{w}$, we construct the vectors $f^1,\ldots,f^K$, corresponding the senses $s_1,\ldots,s_K$. Let us call these vectors *feature vectors* of the context $\bar{w}$.

2.1 Unweighted Method: Sum of the Feature Values

In contrast with the Naive Bayes classifier (Section 4.1), which calculates a product of conditional probabilities, we define an additive decision function

$$s^\star = \arg\max_k \sum_{i=1}^{n} f_i^k \,. \tag{2}$$

In Figure 1 is depicted an example case for two senses ($K = 2$). For illustrative reasons, the components of the vectors f^1 and f^2 have been ordered in a descending order by value. Notice that even for $K = 2$, some words have positive feature values for more than one sense. For example, the word w_1 is positively tied to both senses s_1 and s_2. The figure also provides a graphical explanation for the decision function in Equation 2. It is easy to see that the decision function compares the total areas delimited by the feature values for each sense (where total area is the area delimited by positive feature values minus the area delimited by negative feature values). In the case of Figure 1 the sense s_1 is chosen.



(a) (b)

Figure 1: Example of feature vectors for two senses ($K = 2$). The components of the feature vectors have been ordered by value for visualization purposes.

2.2 Generalized Method: Weighting the Feature Values

In this section, we generalize the method described in Section 2.1. We introduce a weighting scheme and consider different orderings of the feature vectors. The decision function defined in Equation 2 performs unweighted summing of the components of the feature vectors, and thus the feature values of all context words w_i influence the final decision with equal strength. A straightforward generalization is to introduce a weighting scheme to allow different influences of the individual context words. The weighting scheme is carried out by ordering the feature vectors, and thereby the ordering is a tool to assign appropriate weights for the words in the context.

Let $M(\pi, v, n)$ be a classifier, where $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$ is a weight vector, n is an even context length, and $\pi : \mathbb{R}^n \to \mathbb{R}^n$ is a function that orders the components of $x \in \mathbb{R}^n$, that is, if $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, then $\pi(x) = (x_{j_1}, \ldots, x_{j_n})$ for some permutation $(j_1, \ldots, j_n)$ of $(1, \ldots, n)$. The classifiers $M(\pi, v, n)$ form a family, where each member of this family is distinguished by the ordering function π, the associated weight vector v, and the context length n.

A classifier $M(\pi, v, n)$ decides the sense of the word with K associated feature vectors $f^1, \ldots, f^K$ using the decision function

$$s^\star = \arg\max_k \sum_{i=1}^{n} v_i \cdot \pi(f^k)_i \,. \tag{3}$$

Hence, the classifier performs a weighted sum of the ordered feature vector using the associated weights and chooses such sense s_k for which the weighted sum is maximal.

In the following sections, we inspect more closely two subfamilies of classifiers distinguished by their associated ordering function π.

2.2.1 WEIGHTING BY FEATURE VALUES

Let us define a subfamily of $M(\pi, v, n)$ in which the ordering function π orders the vectors f^k in a descending order of feature values (recall Figure 1). If we further restrict the weights by the following conditions: $v_i \in [0,1]$, $1 \leq i \leq n$, and $\sum_{i=1}^{n} v_i = 1$, then the weighted sum $\sum_{i=1}^{n} v_i \cdot \pi(f^k)_i$ is an ordered weighted averaging (OWA) operator, well known in the theory of decision making (Yager, 1988). Furthermore, the Equation 3 can be interpreted as multicriteria decision making (Yager, 1997), where the aggregation function is an OWA operator and the individual criteria are the context words. The word sense disambiguation problem is thus cast as a multicriteria decision making process.

The weight vector v for this kind of ordering must be set with a knowledge of the given problem (Yager, 1997). However, in our extensive experiments we were not able to find such a feature-value-based weight vector v that the classifier introduced in Equation 3 would outperform the unweighted classifier introduced in Section 2.1. Instead, we found that better results are obtained when weighting the feature values by the positions of their corresponding words in the context.

2.2.2 WEIGHTING BY POSITION

Next we consider classifiers whose ordering function is the identity function $\pi(x) = x$, which preserves the contextual order in the vector f^k, that is, the order in which the words appear in the text. The meaning of the weight vector v here is to assess the relative importance of a context word w_i with respect to its distance from the term w (the term being disambiguated). Assuming that words closer to w are more significant for the decision, we can define the weight vector v, for example, as

$$v_i = \frac{1}{\mathrm{dist}(w_i)^{\alpha}} + \beta \text{ for all } 1 \leq i \leq n, \tag{4}$$

where $\alpha, \beta \in \mathbb{R}$, $\alpha > 0$, $\beta \geq 0$, and $\mathrm{dist}(w_i)$ is the distance between the positions of the words w and w_i, that is,

$$\mathrm{dist}(w_i) = \left\lceil \left| \frac{n+1}{2} - i \right| \right\rceil .$$

The values of the weights adopt a hyperbolic shape with highest values at the center of the vector v (see Figure 2). The parameter α determines how steeply the weight values grow towards the center of the vector, and the parameter β is an offset of the values. The role of the parameter β is to reduce the ratio between the weights of the words which are close to the term w and the weights of the words which are far from the term w.

3. Collocation Features

In order to introduce a local syntax information to the classifier, the *collocation* features are commonly used. The collocation features test for the presence of a pattern of up to l contiguous words around the target term w. We set $l = 2$ in our experiments, and thus in the context $\bar{w}$ of the term w we

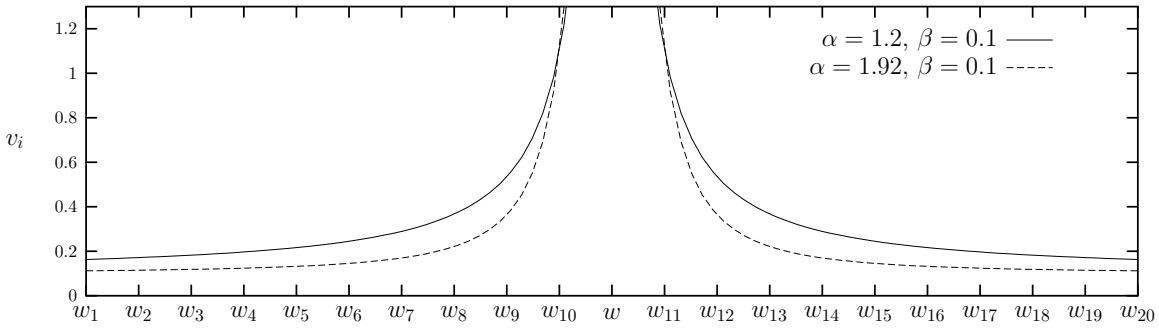


Figure 2: An example of the weight values defined in Equation 4 for different parameters α and β with $n = 20$.

have the five collocation features c_1 :="w_i _", c_2 :="_ w_i", c_3 :="w_i w_{i+1} _", c_4 :="_ w_i w_{i+1}", and c_5 :="w_i _ w_{i+1}", where the symbol _ matches the term w, regardless of its sense.

The collocations can be incorporated into the proposed classifiers by considering them as pseudowords. The evidence ev is defined as in Section 2. However, the expectation ex must be redefined. Since some collocation c_i can appear only in one place in the context of some sense s_k, and hence in count(s_k) places in T, we define

$$\mathrm{ex}(c_i, s_k) = \frac{\mathrm{count}(c_i)}{|T|} \cdot \mathrm{count}(s_k).$$

For the purpose of calculating the value of count(c_i), the symbol _ in the collocation matches any word in the text.

We will also introduce for each collocation c_i a pseudodistance dist(c_i), which will be used in Equation 4 to determine the weight for c_i. The pseudodistances we used in our experimental studies are described below, in Section 5.1.2.

4. Baseline Methods

In this section, we briefly introduce the Naive Bayes classifier and the SNoW classification architecture, the two baseline methods used in the experimental evaluation in Section 5.

4.1 The Naive Bayes Classifier

We introduce the Naive Bayes classifier as it is applied to the problem of word sense disambiguation. The specific formalization we describe is due to Gale et al. (1992) and Manning and Schütze (1999). The decision rule of the Naive Bayes classifier is

$$s^\star = \arg\max_k P(s_k|\overline{w}) = \arg\max_k (P(s_k) \cdot \prod_{i=1}^{n} P(w_i|s_k)), \tag{5}$$

where $P(s_k|\overline{w})$ is the conditional probability of the term w taking the sense s_k, given the context $\overline{w}$. The decision rule thus picks the sense s_k that is most probable for the context $\overline{w}$. The decision

rule assumes that the context words $w_1, \ldots, w_n$ are conditionally independent (the Naive Bayes assumption). The Maximum-Likelihood estimates

$$P(w_i|s_k) = \frac{\text{count}(w_i, s_k)}{\text{count}(s_k)} \qquad \text{and} \qquad P(s_k) = \frac{\text{count}(s_k)}{\sum_{j=1}^{K} \text{count}(s_j)} \tag{6}$$

are computed from the labeled training corpus T.

When disambiguating an occurrence of the term w, we face the problem of small values of $\text{count}(w_i, s_k)$, which make the Maximum-Likelihood estimate of $P(w_i|s_k)$ unreliable. In the extreme case when $\text{count}(w_i, s_k) = 0$, we get $P(w_i|s_k) = 0$ by Equation 6 and then $P(s_k|\bar{w}) = 0$ by Equation 5. To address this problem, various smoothing techniques have been derived, which redistribute some of the probability mass to the rare and unseen events. For the definitions and explanations of the smoothing techniques used in this paper, please refer to Chen and Goodman (1998) for Add-1, Kneser-Ney and Katz smoothing, Ng (1997) for Ng's smoothing, Kohavi et al. (1997) for No-matches-0.01 smoothing, and Golding and Roth (1999) for the Interpolative smoothing. Chen and Goodman (1998) provide a very valuable insight into various smoothing techniques and their performance on the language modeling problem.

4.2 The SNoW Classification Architecture

As a baseline alternative to the Naive Bayes classifier, we employ the SNoW[1] (Sparse Network of Winnows) classification architecture. Next we provide a brief introduction into the Winnow classifier. For a more detailed introduction refer to Golding and Roth (1999).

Let $\mathcal{F}$ be a space of features and let $\mathcal{F}_A \subseteq \mathcal{F}$ be a set of active features of an example. In our setting, $\mathcal{F}$ represents the set of all words in T and $\mathcal{F}_A$ represents the set of words present in the context $\bar{w}$. Further let $v_f \in \mathbb{R}_+$ be the weight of a feature $f \in \mathcal{F}$. The Winnow classifier then returns a positive classification 1 if

$$\sum_{f \in \mathcal{F}_A} v_f > \theta \,,$$

where $\theta \in \mathbb{R}$ is a suitable threshold, and a negative classification 0 otherwise.

The online mistake-driven training algorithm of the Winnow classifier is governed by three parameters: the promotion parameter $\alpha \in \mathbb{R}$, $\alpha > 1$, the demotion parameter $\beta \in \mathbb{R}$, $0 < \beta < 1$, and the default weight $v_{\text{def}} \in \mathbb{R}_+$. The weights v_f for all features $f \in \mathcal{F}$ are initialized to $v_f = 0$. When the classifier is presented an example, its prediction is computed. In case the prediction was correct, no changes are made to the classifier. In case the prediction was incorrect, the weights of all features $f \in \mathcal{F}_A$ are updated by

$$v_f \leftarrow \begin{cases} \alpha \cdot v_f & \text{if the example belongs to the positive class,} \\ \beta \cdot v_f & \text{if the example belongs to the negative class.} \end{cases}$$

In the former case (the misclassified example belongs to the positive class), all weights v_f such that $v_f = 0$ are set to $v_f = v_{\text{def}}$ before updating. Note that the weights of features which only occurred in negative examples always remain zero.

In the SNoW architecture several classifiers representing the same positive class are grouped into one *cloud*. The same examples are presented to each classifier in the cloud both in training and

1. http://l2r.cs.uiuc.edu/~cogcomp/

in prediction; the classifiers differ only by their training algorithm parameters. The decision of the classifiers within one cloud is combined as a weighted majority, where the weights depend on the performance of the individual classifiers during the training. The final prediction of SNoW is the class whose cloud had the highest weighted majority prediction. For details, see Golding and Roth (1999) or Littlestone and Warmuth (1994).

5. Evaluation of the Methods

We evaluate the unweighted classifier and the positionally weighted classifier. As baseline methods, we evaluate the Naive Bayes classifier with various smoothing techniques and the SNoW classification architecture. We evaluate the performance of the classifiers on the gene/protein name disambiguation problem and on the context-sensitive spelling error correction problem.

5.1 The Gene/Protein Name Disambiguation Task

Let us first consider the experimental setup and results for the gene/protein name disambiguation task.

5.1.1 DATA AND ITS PREPROCESSING

A common issue when using statistical methods is to obtain a large enough training set, which allows the calculation of the word frequencies on a sufficiently representative corpus of text. For the gene/protein disambiguation task, it would demand an enormous amount of expert work to obtain a large enough manually annotated corpus of text. Hatzivassiloglou et al. (2001) propose a simple approach to obtain the necessary annotated corpus in a fully automatic way. The automatic annotation method is based on the fact that sometimes the author disambiguates the term by explicitly following it with the word "gene" or "protein". These instances are clearly disambiguated by the author and can be used as training cases. The training text T is thus formed by a text where the instances readily disambiguated by the authors are tagged to be a gene or a protein, and the term "gene" or "protein" immediately following the disambiguated occurrences is removed. The document boundaries are preserved—the contexts $\bar{w}$ may not span between two different documents. The reported value of n, the context length, is thus a maximum value. Since the authors of the documents may explicitly disambiguate primarily the most difficult instances, the data may suffer of a certain bias. However, there is no practical way to obtain a large-enough set of certainly unbiased examples.

As the corpus we use 560093 documents which are article abstracts from years 1998–2002 downloaded from the PubMed database.[2] In order to identify the protein/gene names in the text, we use a list of names derived from 92849 records in the Swissprot database.[3] The actual number of searched terms is larger because many names have several synonyms. A small number (less than 1%) of gene/protein names are English words, which would lead to many false positives in gene/protein name identification. Thus, we remove from the list of gene/protein names all the words that occur in the English lexicon of the Brill's part-of-speech tagger.[4] In the corpus we have identified 65068 instances of gene/protein names, out of which 30768 were readily disambiguated

2. http://www.ncbi.nlm.nih.gov/PubMed/
3. http://www.expasy.org/sprot/
4. http://www.cs.jhu.edu/~brill/

as proteins and 34300 were readily disambiguated as genes. Any gene/protein name found in the text is replaced with an artificial word.

We further perform the following modifications of the text, as proposed by Hatzivassiloglou et al. (2001):

Case mapping All words are capitalized.

Removing stopwords Extremely common function words (for example: "is", "are", "the", "a") are removed from the text.

Stemming All words are stemmed using the Porter stemming algorithm (Porter, 1980).[5] Stemming maps related words to their stem so that, for example, "activate", "activating", "activated" all become one word "activ".

5.1.2 EXPERIMENTAL RESULTS

In order to estimate the parameters of the weighted classifier, we use random 200523 of the available documents and 5-fold cross-validation. For computational reasons, the parameter estimation is done in two phases. First we search for the parameters α, β, and n without collocation features. In this experiment, we perform an exhaustive grid search for the parameters trying all possible combinations of $\alpha \in [0, 3.5]$ with step 0.1, $\beta \in [0, 1]$ with step 0.1, and $n \in [2, 352]$ with step 10. We then select the best-performing combination of parameters $\alpha = 1.2$, $\beta = 0.1$, and $n = 252$. With these values of α, β, and n, we search for the distances of the collocations. Because collocations c_1 and c_2 are "symmetric", we fix $\mathrm{dist}(c_1) = \mathrm{dist}(c_2)$. Similarly, we set $\mathrm{dist}(c_3) = \mathrm{dist}(c_4)$. Thus, it suffices to search for $\mathrm{dist}(c_1)$, $\mathrm{dist}(c_3)$, and $\mathrm{dist}(c_5)$ only. We perform exhaustive grid search where $\mathrm{dist}(c_i) \in [0.05, 1]$, $i \in \{1, 3, 5\}$, with step 0.05. The best values found were $\mathrm{dist}(c_1) = 0.1$, $\mathrm{dist}(c_3) = 0.15$, and $\mathrm{dist}(c_5) = 0.15$.

Using the remaining 359570 documents, which have not been used for estimating the parameters, we then perform a series of 10-fold experiments for various settings of the context length n in order to study its effect on the accuracy.

The setting for the SNoW classifier is that of Golding and Roth (1999). For each sense, we define a cloud of five Winnows, differing the demotion parameter from $\beta = 0.5$ to $\beta = 0.9$. The promotion parameter $\alpha = 1.5$ and the default weight is set to 0.1.

Each classifier is evaluated for $n = 2, 4, \ldots, 160$ and the best accuracy value of each classifier and its corresponding context length is reported in Table 1. Note that the weighted classifier uses $n = 252$ obtained during the parameter estimation. The weighted classifier outperforms both the Naive Bayes classifier and the SNoW classifier, both with and without collocation features. Further, we see that the information about position of words in the context notably increases the accuracy of the weighted classifier when compared to the unweighted classifier that uses no positional information. The collocation features further substantially increase the accuracy of all the tested classifiers. It is interesting to note that the optimal value of n is generally bigger with collocation features.

The accuracy values for the various context lengths are presented in Figure 3. For the Naive Bayes classifier, we observe a steadily descending curve meaning that the Naive Bayes classifier performs best for short context lengths around $n = 10$ without collocations and $n = 24$ with collocations. On the contrary, both the weighted classifier and SNoW perform best for very long contexts.

5. http://www.tartarus.org/~martin/PorterStemmer/

Classifier		Without collocations		With collocations	
		n	Accuracy	*n*	Accuracy
New method	Weighted	252	82.37	252	86.12
	Unweighted	252	81.21	16	82.47
SNoW		98	77.87	102	84.54
Naive Bayes	Ng	10	78.54	24	84.44
	Add-1	14	78.79	24	84.40
	Kneser-Ney	10	78.42	24	83.97
	Interpolative	10	78.22	22	83.74
	No-matches-0.01	8	77.38	38	83.08
	Katz	28	77.87	30	82.67

Table 1: The maximum accuracy achieved by the compared methods together with the respective value of *n* for the gene/protein name disambiguation problem.

In order to test for statistical significance, we perform a 10-10-fold experiment (with collocations) for the weighted classifier, the Naive Bayes classifier with the Ng smoothing, and the SNoW classifier, in addition to the experiments described so far. In this experiment we repeat a 10-fold cross-validated experiment ten times, every time with a different split of the data. We then average the results of the ten 10-fold experiments and test for statistical significance using the paired Student's *t*-Test on the ten 10-fold results. The results are presented in Table 2. The test shows that all differences between the classifiers are significant ($p < 0.01$).

Classifier	*n*	Accuracy	Standard deviation
Weighted	252	86.12	0.05
Ng	24	84.29	0.06
SNoW	102	83.47	0.69

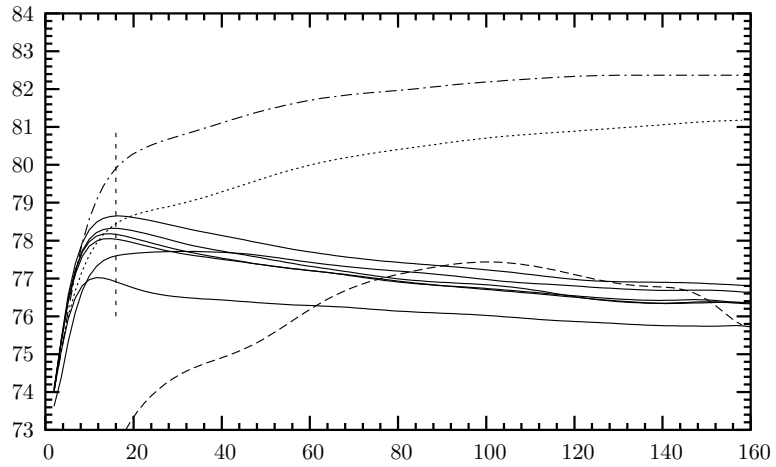
Table 2: Average results of the ten 10-fold experiments and standard deviation of the 10-fold experiment results for the gene/protein name disambiguation problem.

5.2 Context Sensitive Spelling Error Correction

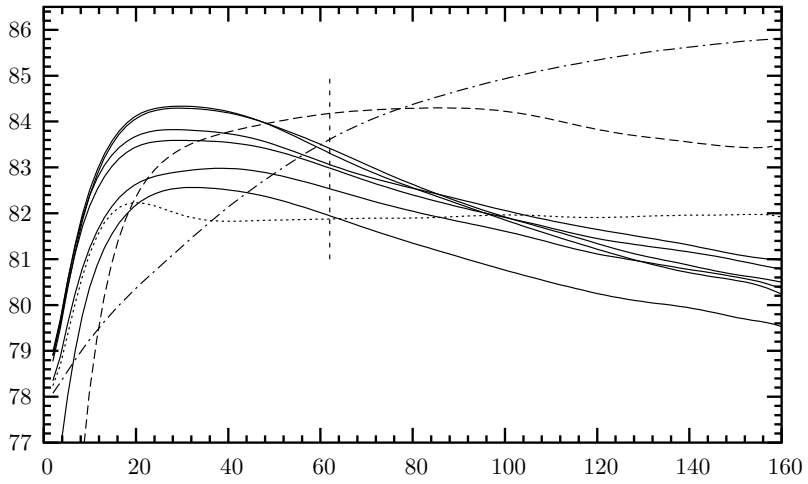
We perform further experiments on the context-sensitive spelling error correction problem. We evaluate the weighted classifier, the Naive Bayes classifier with various smoothing techniques, and the SNoW classification architecture. We only consider the case with collocations.

5.2.1 DATA AND ITS PREPROCESSING

We use first 473877 articles from the Reuters corpus (Rose et al., 2002). The confusion sets are those used in the experiments of Golding and Roth (1999). Once the words of the 21 confusion sets



(a) Without collocations. The full lines cross the dashed vertical marker line, from top to bottom in the following order: Add-1, Ng, Kneser-Ney, Interpolative, Katz, No-matches-0.01.



(b) With collocations. The full lines cross the dashed vertical marker line, from top to bottom in the following order: Add-1, Ng, Kneser-Ney, Interpolative, No-matches-0.01, Katz.

Figure 3: The relationship between context length and accuracy, measured on the complete gene/protein test data using 10-fold cross-validation. The curves are smoothed (Bezier curves). The full line represents Naive Bayes with various smoothing methods, the dotted line represents the unweighted classifier, the dash-dot line represents the weighted classifier, and the dashed line represents the SNoW classifier.

	α	β	dist(c_1)	dist(c_3)	dist(c_5)
Average	1.92	0.10	0.33	0.52	0.25
St. dev.	0.78	0.14	0.25	0.22	0.28
Min.	0.30	0.00	0.10	0.10	0.10
Max.	3.00	0.45	1.00	0.90	1.00

Table 3: Summary of the best performing parameter values found for the 21 confusion sets.

have been identified (case-insensitive except for the word "I") and labeled as training examples, we perform case mapping and stemming. The article boundaries are preserved.

5.2.2 EXPERIMENTAL RESULTS

After preliminary experiments we observed that the value $n = 252$ is too large for the weighted classifier on the context-sensitive spelling error correction problem. We thus set $n = 20$ for all the tested classifiers, which is also the value of *n* used by Golding and Roth (1999) when applying the SNoW and Naive Bayes classifiers to context-sensitive spelling error correction problem. In order to estimate the parameters, we use random 158105 of the available articles and 5-fold cross-validation. We then estimate the parameters α, β, dist(c_1), dist(c_3), and dist(c_5) separately for each of the 21 confusion sets. We use a similar two-phase protocol as for the gene/protein name disambiguation problem. First we perform a grid search of $\alpha \in [0,3]$ with step 0.1 and $\beta \in [0,0.5]$ with step 0.025. Then, using the best performing combination of α and β for each confusion set, we perform the grid search dist(c_i) $\in [0,1]$, $i \in \{1,3,5\}$, with step 0.1. A summary of the parameter values found is presented in Table 3. The weights induced by the average values of α and β are presented in Figure 2.

Using the remaining 315772 articles we then perform 10-fold cross-validated experiment for each of the 21 confusion sets. The results for the weighted classifier, SNoW, and Naive Bayes with the best smoothing are presented in Table 4. In Table 5 we present the average results for the various smoothing techniques of the Naive Bayes classifier. The *majority baseline* is the accuracy obtained by always selecting the most common member of each confusion set. We measure statistical significance of the average difference of the classifiers by the paired Student's *t*-Test on the 21 10-fold results of the confusion sets. The difference between the weighted classifier and the SNoW and Naive Bayes classifiers is statistically significant ($p \approx 0.01$ and $p \approx 0.03$). The difference between SNoW and Naive Bayes is not significant ($p \approx 0.77$).

6. Conclusions and Discussion

We propose for the problems of disambiguation in natural language a new family of classifiers characterized by additive decision function and weighting of word co-occurrence features. The proposed classifiers perform weighted combination of features, where the weights are assigned based on an ordering of the features. The concrete member of the proposed family on which we focus in this work assigns the weights of the features based on their distance from the word to be disambiguated.

Of the problems of disambiguation in natural language, we focus on gene/protein name disambiguation and also consider context-sensitive spelling error correction. For the gene/protein name

Confusion set	Num. of examples	Majority	SNoW	Naive Bayes	Weighted
accept, except	16828	71.76	98.26	94.75	98.38
affect, effect	14063	71.37	96.14	96.74	96.95
among, between	80731	74.46	94.29	94.77	94.54
amount, number	43408	63.19	93.02	90.56	90.83
begin, being	43243	79.86	98.39	98.44	97.45
cite, sight, site	6633	79.48	94.13	95.64	94.20
country, county	48838	80.16	98.36	97.97	97.77
fewer, less	18670	89.99	93.63	94.01	90.90
I, me	93579	93.60	99.41	99.40	99.23
its, it's	288962	91.01	98.91	99.03	97.00
lead, led	36655	56.41	95.90	95.64	95.05
maybe, may be	12269	82.03	95.36	95.54	91.45
passed, past	24589	79.27	97.68	97.90	97.17
peace, piece	22661	95.14	99.02	99.36	99.02
principal, principle	4720	53.05	92.12	93.23	93.15
quiet, quite	12946	53.77	96.60	96.87	96.73
raise, rise	56435	76.23	98.21	97.97	94.12
than, then	115760	80.26	98.37	97.84	97.97
their, there, they're	219097	54.46	98.85	98.50	95.96
weather, whether	29064	68.75	99.20	99.20	99.05
your, you're	5855	74.39	93.43	94.42	92.97
Average		74.70	96.63	96.56	95.71
Standard deviation		12.84	2.36	2.35	2.70

Table 4: Results for the 21 confusion sets.

Smoothing	Accuracy
No-matches-0.01	96.56
Ng	96.43
Kneser-Ney	96.10
Add-1	95.80
Katz	95.02
Interpolative	94.58

Table 5: Average accuracy on the 21 confusion sets for various smoothing techniques.

disambiguation problem, we perform a study of the effect of the context length *n* and show that each of the tested classifiers generally performs best for different context lengths on the gene/protein name disambiguation problem. While the Naive Bayes classifier performs best for short context lengths, SNoW and the proposed weighted classifier perform best for very long contexts. This is, however, problem-dependent, because shorter context length gave better results for all classifiers on the context-sensitive spelling error correction problem. We also evaluate the effect of collocation features which provide the classifiers with local syntax information. As expected, the collocation

features increase the accuracy of all the evaluated classifiers on both problems. The increase is bigger for the baseline methods than for the weighted classifier, which is consistent with understanding the positional weights as an alternative approach of introducing the local syntax to the classifier.

On our main task, that is, the gene/protein name disambiguation task, the proposed weighted classifier is shown to outperform the baselines, thus meeting our objective of improving the classification accuracy on this problem. On context-sensitive spelling error correction the baselines outperform the new method. A feature of context-sensitive spelling error correction to consider is the context length, where in context-sensitive spelling error correction, a short context length performs better. This might indicate that the proposed classifier performs better than the two baselines on problems which allow combining of features from a very long context.

Considering the per-confusion-set parameters presented in Table 3, we observe that the parameter α is relatively high in most of the cases, verifying the intuition that close features are more important for the disambiguation. This is also true for the gene/protein problem. Further, we find that the parameters dist(c_i) set the weight of collocation features higher than that of context words, verifying the intuition that collocation features are more important for the disambiguation. Further, the values of the parameters dist(c_i) suggest that, on average, the features c_3 and c_4 (that is, collocations of the types "$w_i\ w_{i+1}\ \_$" and "$\_\ w_i\ w_{i+1}$") are, somewhat surprisingly, least important among the collocation features. The variance of the parameters is relatively high, suggesting that their optimal values are data-dependent.

The new method is comparable to both the Naive Bayes and SNoW classifiers in its computational complexity, as it performs a simple word count statistics similar to that of the two baselines. However, the new method is more demanding in terms of space, since it stores a dictionary of words appearing in the whole text (due to the term count(w_i) in Equation 1) rather than words appearing only in the contexts $\overline{w}$.

An advantage of the new method is the simple way it deals with the zero-count problem. The proposed method permits zero feature values and does not require any special smoothing technique. For the Naive Bayes classifier, it is not always obvious which of the smoothing techniques should be used. This is demonstrated also in this study, where the relative accuracy of the various smoothing techniques differs between the two problems/corpora.

The new method exploits the information about the position of the words in the context, which has not been successfully accomplished with the Naive Bayes classifier for the same task by Hatzivassiloglou et al. (2001), who made the position a part of the feature and consequently the classifier apparently suffered from sparse data. In this paper, we show that the positional information can be incorporated in the form of weights and it substantially improves the accuracy of the classifier, as shown in the experiments.

Hatzivassiloglou et al. (2001) report classification accuracy for Naive Bayes on the gene/protein disambiguation task to be 84.48%. We have achieved a comparable accuracy of 84.44% for the Naive Bayes classifier with the collocation features. We are not sure whether Hatzivassiloglou et al. used the collocation features, what smoothing for Naive Bayes they used, and what was the context length in their experiments. Further, the two studies differ by the corpus used. Thus, it is impossible to compare the results directly.

We introduce the weighting scheme via ordering of the feature vectors. We perform our experiments on the context order, that is, the natural order of the words in the sentence. As a future work, we find interesting to study other possible orderings, that is, other possible models of relative importance of the individual features. For example, a word of biological relevance in the context may

be very important regardless its position in the text, and an ordering based on biological relevance of the context words could be considered.

Acknowledgments

We would like to acknowledge the people of the MediCel company, particularly Meelis Kolmer, Ph.D., who have kindly answered our numerous questions in the field of biology and advanced our understanding of the specific problems which biological texts bring to the natural language processing. This work uses the Reuters corpus volume 1 distributed by Reuters. This work has been supported by Tekes, the Finnish National Technology Agency.

References

Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Centre for Research in Computing Technology, Harvard University, Cambridge, Massachusetts, 1998.

William W. Cohen. Learning trees and rules with set-valued features. In William J. Clancey and Dan Weld, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 709–716. AAAI Press, Menlo Park, California, 1996.

William A. Gale, Kenneth W. Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439, 1992.

Andrew R. Golding and Dan Roth. A Winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34:107–130, 1999.

Vasileios Hatzivassiloglou, Pablo A. Duboué, and Andrey Rzhetsky. Disambiguating proteins, genes, and RNA in text: A machine learning approach. *Bioinformatics*, 17:97–106, 2001.

Ron Kohavi, Barry Becker, and Dan Sommerfield. Improving Simple Bayes. In Maarten van Someren and Gerhard Widmer, editors, *Proceedings of the 9th European Conference on Machine Learning*, pages 78–87. Springer Verlag, Heidelberg, 1997.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

Hwee Tou Ng. Exemplar-based word sense disambiguation: Some recent improvements. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 208–213. Association for Computational Linguistics, Somerset, New Jersey, 1997.

Martin F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, California, 1993.

Tony G. Rose, Mark Stevenson, and Miles Whitehead. The Reuters Corpus Volume 1: From yesterday's news to tomorrow's language resources. In Manuel Gonzales Rodriguez and Carmen Paz Suarez Araujo, editors, *Proceedings of the Third International Conference on Language Resources and Evaluation*. ELRA, Paris, 2002.

Ronald R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190, 1988.

Ronald R. Yager. On the inclusion of importances in OWA aggregations. In Ronald R. Yager and Janusz Kacprzyk, editors, *The Ordered Weighted Averaging Operators, Theory and Applications*, pages 41–59. Kluwer Academic Publishers, Norwell, Massachusetts, 1997.

David Yarowsky. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95. Association for Computational Linguistics, Somerset, New Jersey, 1994.

Journal of Machine Learning Research 5 (2004) 623-648 Submitted 1/04; Published 6/04

The Sample Complexity of Exploration in the Multi-Armed Bandit Problem

Shie Mannor SHIE@MIT.EDU
John N. Tsitsiklis JNT@MIT.EDU
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Editors: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

We consider the multi-armed bandit problem under the PAC ("probably approximately correct") model. It was shown by Even-Dar et al. (2002) that given n arms, a total of $O\big((n/\varepsilon^2)\log(1/\delta)\big)$ trials suffices in order to find an ε-optimal arm with probability at least $1-\delta$. We establish a matching lower bound on the expected number of trials under any sampling policy. We furthermore generalize the lower bound, and show an explicit dependence on the (unknown) statistics of the arms. We also provide a similar bound within a Bayesian setting. The case where the statistics of the arms are known but the identities of the arms are not, is also discussed. For this case, we provide a lower bound of $\Theta\big((1/\varepsilon^2)(n+\log(1/\delta))\big)$ on the expected number of trials, as well as a sampling policy with a matching upper bound. If instead of the expected number of trials, we consider the maximum (over all sample paths) number of trials, we establish a matching upper and lower bound of the form $\Theta\big((n/\varepsilon^2)\log(1/\delta)\big)$. Finally, we derive lower bounds on the expected regret, in the spirit of Lai and Robbins.

1. Introduction

The multi-armed bandit problem is a classical problem in decision theory. There is a number of alternative arms, each with a stochastic reward whose probability distribution is initially unknown. We try these arms in some order, which may depend on the sequence of rewards that have been observed so far. A common objective in this context is to find a policy for choosing the next arm to be tried, under which the sum of the expected rewards comes as close as possible to the ideal reward, i.e., the expected reward that would be obtained if we were to try the "best" arm at all times. One of the attractive features of the multi-armed bandit problem is that despite its simplicity, it encompasses many important decision theoretic issues, such as the tradeoff between exploration and exploitation.

The multi-armed bandit problem has been widely studied in a variety of setups. The problem was first considered in the 50's, in the seminal work of Robbins (1952), which derives policies that asymptotically attain an average reward that converges in the limit to the reward of the best arm. The multi-armed bandit problem was later studied in discounted, Bayesian, Markovian, expected reward, and adversarial setups. See Berry and Fristedt (1985) for a review of the classical results on the multi-armed bandit problem.

Lower bounds for different variants of the multi-armed bandit have been studied by several authors. For the expected regret model, where the regret is defined as the difference between the ideal reward (if the best arm were known) and the reward under an online policy, the seminal work of Lai and Robbins (1985) provides asymptotically tight bounds in terms of the Kullback-Leibler divergence between the distributions of the rewards of the different arms. These bounds grow logarithmically with the number of steps. The adversarial multi-armed bandit problem (i.e., without any probabilistic assumptions) was considered in Auer et al. (1995, 2002b), where it was shown that the expected regret grows proportionally to the square root of the number of steps. Of related interest is the work of Kulkarni and Lugosi (2000) which shows that for any specific time t, one can choose the reward distributions so that the expected regret is linear in t.

The focus of this paper is the classical multi-armed bandit problem, but rather than looking at the expected regret, we are concerned with PAC-type bounds on the number of steps needed to identify a near-optimal arm. In particular, we are interested in the expected number of steps that are required in order to identify with high probability (at least $1-\delta$) an arm whose expected reward is within ε from the expected reward of the best arm. This naturally abstracts the case where one must eventually commit to one specific arm, and quantifies the amount of exploration necessary. This is in contrast to most of the results for the multi-armed bandit problem, where the main aim is to maximize the expected cumulative reward while both exploring and exploiting. In Even-Dar et al. (2002), a policy, called the median elimination algorithm, was provided which requires $O((n/\varepsilon^2)\log(1/\delta))$ trials, and which finds an ε-optimal arm with probability at least $1-\delta$. A matching lower bound was also derived in Even-Dar et al. (2002), but it only applied to the case where $\delta > 1/n$, and therefore did not capture the case where high confidence (small δ) is desired. In this paper, we derive a matching lower bound which also applies when $\delta > 0$ is arbitrarily small.

Our main result can be viewed as a generalization of a $O((1/\varepsilon^2)\log(1/\delta))$ lower bound provided in Anthony and Bartlett (1999), and Chernoff (1972), for the case of two bandits. The proof in Anthony and Bartlett (1999) is based on a hypothesis interchange argument, and relies critically on the fact there are only two underlying hypotheses. Furthermore, it is limited to "nonadaptive" policies, for which the number of trials is fixed a priori. The technique we use is based on a likelihood ratio argument and a tight martingale bound, and applies to general policies.

A different type of lower bound was derived in Auer et al. (2002b) for the expected regret in an adversarial setup. The bounds derived there can also be used to derive a lower bound for our problem, but do not appear to be tight enough to capture the $\log(1/\delta)$ dependence on δ. Our work also provides fundamental lower bounds in the context of sequential analysis (see, e.g., Chernoff, 1972; Jennison et al., 1982; Siegmund, 1985). In the language of Siegmund (1985), we provide a lower bound on the expected length of a sequential sampling policy under any adaptive allocation scheme. For the case of two arms, it was shown in Siegmund (1985) (p. 148) that if one restricts to sampling policies that only take into account the empirical average rewards from the different arms, then the problems of inference and arm selection can be treated separately. As a consequence, and under this restriction, Siegmund (1985) shows that an optimal allocation cannot be much better than a uniform one. Our results are different in a number of ways. First, we consider multiple hypotheses (multiple arms). Second, we allow the allocation rule to be completely general and to depend on the whole history. Third, unlike most of the sequential analysis literature (see, e.g., Jennison et al., 1982), we do not restrict ourselves to the limiting case where the probability of error converges to zero. Finally, we consider finite time bounds, rather than asymptotic ones. We further comment that

our results extend those of Jennison et al. (1982), in that we consider the case where the reward is not Gaussian.

Paper Outline

The paper is organized as follows. In Section 2, we set up our framework, and since we are mainly interested in lower bounds, we restrict to the special case where each arm is a "coin," i.e., the rewards are Bernoulli random variables, but with unknown parameters ("biases"). In Section 3, we provide a $O((n/\varepsilon^2)\log(1/\delta))$ lower bound on the expected number of trials under any policy that finds an ε-optimal coin with probability at least $1-\delta$. In Section 4, we provide a refined lower bound that depends explicitly on the specific (though unknown) biases of the coins. This lower bound has the same $\log(1/\delta)$ dependence on δ; furthermore, every coin roughly contributes a factor inversely proportional to the square difference between its bias and the bias of a best coin, but no more that $1/\varepsilon^2$. In Section 5, we derive a lower bound similar to the one in Section 3, but within a Bayesian setting, under a prior distribution on the set of biases of the different coins.

In Section 6 we provide a bound on the expected regret which is similar in spirit to the bound in Lai and Robbins (1985). The constants in our bounds are slightly worse than the ones in Lai and Robbins (1985), but the different derivation, which links the PAC model to regret bounds, may be of independent interest. Our bound holds for any finite time, as opposed to the asymptotic result provided in Lai and Robbins (1985).

The case where the coin biases are known in advance, but the identities of the coins are not, is discussed in Section 7. We provide a policy that finds an ε-optimal coin with probability at least $1-\delta$, under which the expected number of trials is $O\left((1/\varepsilon^2)(n+\log(1/\delta))\right)$. We show that this bound is tight up to a multiplicative constant. If instead of the expected number of trials, we consider the maximum (over all sample paths) number of trials, we establish a matching upper and lower bounds of the form $\Theta((n/\varepsilon^2)\log(1/\delta))$. Finally, Section 8 contains some brief concluding remarks.

2. Problem Definition

The exploration problem for multi-armed bandits is defined as follows. We are given n arms. Each arm ℓ is associated with a sequence of identically distributed Bernoulli (i.e., taking values in $\{0,1\}$) random variables X_k^ℓ, $k=1,2,\ldots$, with unknown mean p_ℓ. Here, X_k^ℓ corresponds to the reward obtained the kth time that arm ℓ is tried. We assume that the random variables X_k^ℓ, for $\ell=1,\ldots,n$, $k=1,2,\ldots$, are independent, and we define $p=(p_1,\ldots,p_n)$. Given that we restrict to the Bernoulli case, we will use in the sequel the term "coin" instead of "arm."

A *policy* is a mapping that given a history, chooses a particular coin to be tried next, or selects a particular coin and stops. We allow a policy to use randomization when choosing the next coin to be tried or when making a final selection. However, we only consider policies that are guaranteed to stop with probability 1, for every possible vector p. (Otherwise, the expected number of steps would be infinite.) Given a particular policy, we let $\mathbf{P}_p$ be the corresponding probability measure (on the natural probability space for this model). This probability space captures both the randomness in the coins (according to the vector p), as well as any additional randomization carried out by the policy. We introduce the following random variables, which are well defined, except possibly on the set of measure zero where the policy does not stop. We let T_ℓ be the total number of times that

coin ℓ is tried, and let $T = T_1 + \cdots + T_n$ be the total number of trials. We also let I be the coin which is selected when the policy decides to stop.

We say that a policy is (ε,δ)-*correct* if

$$\mathbf{P}_p\Big(p_I > \max_\ell p_\ell - \varepsilon\Big) \geq 1-\delta,$$

for *every* $p \in [0,1]^n$. It was shown in Even-Dar et al. (2002) that there exist constants c_1 and c_2 such that for every n, $\varepsilon > 0$, and $\delta > 0$, there exists an (ε,δ)-correct policy under which

$$\mathbf{E}_p[T] \leq c_1 \frac{n}{\varepsilon^2} \log \frac{c_2}{\delta}, \qquad \forall\ p \in [0,1]^n.$$

A matching lower bound was also established in Even-Dar et al. (2002), but only for "large" values of δ, namely, for $\delta > 1/n$. In contrast, we aim at deriving bounds that capture the dependence of the sample-complexity on δ, as δ becomes small.

3. A Lower Bound on the Sample Complexity

We start with our central result, which can be viewed as an extension of Lemma 5.1 from Anthony and Bartlett (1999), as well as a special case of Theorem 5. We present it here because it admits a simpler proof, but also because parts of the proof will be used later. Throughout the rest of the paper, log will stand for the natural logarithm.

Theorem 1 *There exist positive constants c_1, c_2, ε_0, and δ_0, such that for every $n \geq 2$, $\varepsilon \in (0,\varepsilon_0)$, and $\delta \in (0,\delta_0)$, and for every (ε,δ)-correct policy, there exists some $p \in [0,1]^n$ such that*

$$\mathbf{E}_p[T] \geq c_1 \frac{n}{\varepsilon^2} \log \frac{c_2}{\delta}.$$

In particular, ε_0 and δ_0 can be taken equal to 1/8 and $e^{-4}/4$, respectively.

Proof Let us consider a multi-armed bandit problem with $n+1$ coins, which we number from 0 to n. We consider a finite set of $n+1$ possible parameter vectors p, which we will refer to as "hypotheses." Under any one of the hypotheses, coin 0 has a known bias $p_0 = (1+\varepsilon)/2$. Under one hypothesis, denoted by H_0, all the coins other than zero have a bias of 1/2,

$$H_0:\ p_0 = \frac{1}{2} + \frac{\varepsilon}{2}, \qquad p_i = \frac{1}{2},\ \text{for } i \neq 0\,,$$

which makes coin 0 the best coin. Furthermore, for $\ell = 1,\ldots,n$, there is a hypothesis

$$H_\ell:\ p_0 = \frac{1}{2} + \frac{\varepsilon}{2}, \qquad p_\ell = \frac{1}{2} + \varepsilon, \qquad p_i = \frac{1}{2},\ \text{for } i \neq 0,\ell\,,$$

which makes coin ℓ the best coin.

We define $\varepsilon_0 = 1/8$ and $\delta_0 = e^{-4}/4$. From now on, we fix some $\varepsilon \in (0,\varepsilon_0)$ and $\delta \in (0,\delta_0)$, and a policy, which we assume to be ($\varepsilon/2$,δ)-correct . If H_0 is true, the policy must have probability at least $1-\delta$ of eventually stopping and selecting coin 0. If H_ℓ is true, for some $\ell \neq 0$, the policy must have probability at least $1-\delta$ of eventually stopping and selecting coin ℓ. We denote by $\mathbf{E}_\ell$ and $\mathbf{P}_\ell$ the expectation and probability, respectively, under hypothesis H_ℓ.

We define t^* by

$$t^* = \frac{1}{c\varepsilon^2}\log\frac{1}{4\delta} = \frac{1}{c\varepsilon^2}\log\frac{1}{\theta}, \tag{1}$$

where $\theta = 4\delta$, and where c is an absolute constant whose value will be specified later.[1] Note that $\theta < e^{-4}$ and $\varepsilon < 1/4$.

Recall that T_ℓ stands for the number of times that coin ℓ is tried. We assume that for some coin $\ell \neq 0$, we have $\mathbf{E}_0[T_\ell] \leq t^*$. We will eventually show that under this assumption, the probability of selecting H_0 under H_ℓ exceeds δ, and violates $(\varepsilon/2,\delta)$-correctness. It will then follow that we must have $\mathbf{E}_0[T_\ell] > t^*$ for all $\ell \neq 0$. Without loss of generality, we can and will assume that the above condition holds for $\ell = 1$, so that $\mathbf{E}_0[T_1] \leq t^*$.

We will now introduce some special events A and C under which various random variables of interest do not deviate significantly from their expected values. We define

$$A = \{T_1 \leq 4t^*\},$$

and obtain

$$t^* \geq \mathbf{E}_0[T_1] \geq 4t^*\mathbf{P}_0(T_1 > 4t^*) = 4t^*(1-\mathbf{P}_0(T_1 \leq 4t^*)),$$

from which it follows that

$$\mathbf{P}_0(A) \geq 3/4.$$

We define $K_t = X_1^1 + \cdots + X_t^1$, which is the number of unit rewards ("heads") if the first coin is tried a total of t (not necessarily consecutive) times. We let C be the event defined by

$$C = \left\{\max_{1\leq t\leq 4t^*}\left|K_t - \frac{1}{2}t\right| < \sqrt{t^*\log(1/\theta)}\right\}.$$

We now establish two lemmas that will be used in the sequel.

Lemma 2 *We have* $\mathbf{P}_0(C) > 3/4$.

Proof We will prove a more general result:[2] we assume that coin i has bias p_i under hypothesis H_ℓ, define K_t^i as the number of unit rewards ("heads") if coin i is tested for t (not necessarily consecutive) times, and let

$$C_i = \left\{\max_{1\leq t\leq 4t^*}\left|K_t^i - p_i t\right| < \sqrt{t^*\log(1/\theta)}\right\}.$$

First, note that $K_t^i - p_i t$ is a $\mathbf{P}_\ell$-martingale (in the context of Theorem 1, $p_i = 1/2$ is the bias of coin $i = 1$ under hypothesis H_0). Using Kolmogorov's inequality (Corollary 7.66, in p. 244 of Ross, 1983), the probability of the complement of C_i can be bounded as follows:

$$\mathbf{P}_\ell\left(\max_{1\leq t\leq 4t^*}\left|K_t^i - p_i t\right| \geq \sqrt{t^*\log(1/\theta)}\right) \leq \frac{\mathbf{E}_\ell[(K_{4t^*}^i - 4p_i t^*)^2]}{t^*\log(1/\theta)}.$$

Since $\mathbf{E}_\ell[(K_{4t^*}^i - 4p_i t^*)^2] = 4p_i(1-p_i)t^*$, we obtain

$$\mathbf{P}_\ell(C_i) \geq 1 - \frac{4p_i(1-p_i)}{\log(1/\theta)} > \frac{3}{4}, \tag{2}$$

where the last inequality follows because $\theta < e^{-4}$ and $4p_i(1-p_i) \leq 1$. □

1. In this and subsequent proofs, and in order to avoid repeated use of truncation symbols, we treat t^* as if it were integer.
2. The proof for a general p_i will be useful later.

Lemma 3 *If $0 \leq x \leq 1/\sqrt{2}$ and $y \geq 0$, then*

$$(1-x)^y \geq e^{-dxy},$$

where $d = 1.78$.

Proof A straightforward calculation shows that $\log(1-x)+dx \geq 0$ for $0 \leq x \leq 1/\sqrt{2}$. Therefore, $y(\log(1-x)+dx) \geq 0$ for every $y \geq 0$. Rearranging and exponentiating, leads to $(1-x)^y \geq e^{-dxy}$. □

We now let B be the event that $I = 0$, i.e., that the policy eventually selects coin 0. Since the policy is $(\varepsilon/2,\delta)$-correct for $\delta < e^{-4}/4 < 1/4$, we have $\mathbf{P}_0(B) > 3/4$. We have already shown that $\mathbf{P}_0(A) \geq 3/4$ and $\mathbf{P}_0(C) > 3/4$. Let S be the event that A, B, and C occur, that is $S = A \cap B \cap C$. We then have $\mathbf{P}_0(S) > 1/4$.

Lemma 4 *If $\mathbf{E}_0[T_1] \leq t^*$ and $c \geq 100$, then $\mathbf{P}_1(B) > \delta$.*

Proof We let W be the history of the process (the sequence of coins chosen at each time, and the sequence of observed coin rewards) until the policy terminates. We define the likelihood function L_ℓ by letting

$$L_\ell(w) = \mathbf{P}_\ell(W = w),$$

for every possible history w. Note that this function can be used to define a random variable $L_\ell(W)$. We also let K be a shorthand notation for K_{T_1}, the total number of unit rewards ("heads") obtained from coin 1. Given the history up to time $t-1$, the coin choice at time t has the same probability distribution under either hypothesis H_0 and H_1; similarly, the coin reward at time t has the same probability distribution, under either hypothesis, unless the chosen coin was coin 1. For this reason, the likelihood ratio $L_1(W)/L_0(W)$ is given by

$$\begin{aligned} \frac{L_1(W)}{L_0(W)} &= \frac{(\frac{1}{2}+\varepsilon)^K(\frac{1}{2}-\varepsilon)^{T_1-K}}{(\frac{1}{2})^{T_1}} \\ &= (1+2\varepsilon)^K(1-2\varepsilon)^K(1-2\varepsilon)^{T_1-2K} \\ &= (1-4\varepsilon^2)^K(1-2\varepsilon)^{T_1-2K}. \end{aligned} \tag{3}$$

We will now proceed to lower bound the terms in the right-hand side of Eq. (3) when event S occurs.

If event S has occurred, then A has occurred, and we have $K \leq T_1 \leq 4t^*$, so that

$$\begin{aligned} (1-4\varepsilon^2)^K \geq (1-4\varepsilon^2)^{4t^*} &= (1-4\varepsilon^2)^{(4/(c\varepsilon^2))\log(1/\theta)} \\ &\geq e^{-(16d/c)\log(1/\theta)} \\ &= \theta^{16d/c}. \end{aligned}$$

We have used here Lemma 3, which applies because $4\varepsilon^2 < 4/4^2 < 1/\sqrt{2}$.

Similarly, if event S has occurred, then $A \cap C$ has occurred, which implies,

$$T_1 - 2K \leq 2\sqrt{t^*\log(1/\theta)} = (2/\varepsilon\sqrt{c})\log(1/\theta),$$

where the equality above made use of the definition of t^*. Therefore,

$$\begin{aligned}(1-2\varepsilon)^{T_1-2K} &\geq (1-2\varepsilon)^{\left(2/\varepsilon\sqrt{c}\right)\log(1/\theta)} \\ &\geq e^{-(4d/\sqrt{c})\log(1/\theta)} \\ &= \theta^{4d/\sqrt{c}}.\end{aligned}$$

Substituting the above in Eq. (3), we obtain

$$\frac{L_1(W)}{L_0(W)} \geq \theta^{(16d/c)+(4d/\sqrt{c})}.$$

By picking c large enough ($c = 100$ suffices), we obtain that $L_1(W)/L_0(W)$ is larger than $\theta = 4\delta$ whenever the event S occurs. More precisely, we have

$$\frac{L_1(W)}{L_0(W)} 1_S \geq 4\delta 1_S,$$

where 1_S is the indicator function of the event S. Then,

$$\mathbf{P}_1(B) \geq \mathbf{P}_1(S) = \mathbf{E}_1[1_S] = \mathbf{E}_0\left[\frac{L_1(W)}{L_0(W)} 1_S\right] \geq \mathbf{E}_0[4\delta 1_S] = 4\delta \mathbf{P}_0(S) > \delta,$$

where we used the fact that $\mathbf{P}_0(S) > 1/4$. □

To summarize, we have shown that when $c \geq 100$, if $\mathbf{E}_0[T_1] \leq (1/c\varepsilon^2)\log(1/(4\delta))$, then $\mathbf{P}_1(B) > \delta$. Therefore, if we have an $(\varepsilon/2, \delta)$-correct policy, we must have $\mathbf{E}_0[T_\ell] > (1/c\varepsilon^2)\log(1/(4\delta))$, for every $\ell > 0$. Equivalently, if we have an (ε, δ)-correct policy, we must have $\mathbf{E}_0[T] > (n/(4c\varepsilon^2))\log(1/(4\delta))$, which is of the desired form. □

4. A Lower Bound on the Sample Complexity - General Probabilities

In Theorem 1, we worked with a particular unfavorable vector p (the one corresponding to hypothesis H_0), under which a lot of exploration is necessary. This leaves open the possibility that for other, more favorable choices of p, less exploration might suffice. In this section, we refine Theorem 1 by developing a lower bound that explicitly depends on the actual (though unknown) vector p. Of course, for any given vector p, there is an "optimal" policy, which selects the best coin without any exploration: e.g., if $p_1 \geq p_\ell$ for all ℓ, the policy that immediately selects coin 1 is "optimal." However, such a policy will not be (ε, δ)-correct for *all* possible vectors p.

We start with a lower bound that applies when all coin biases p_i lie in the range $[0, 1/2]$. We will later use a reduction technique to extend the result to a generic range of biases. In the rest of the paper, we use the notational convention $(x)^+ = \max\{0, x\}$.

Theorem 5 *Fix some $\underline{p} \in (0, 1/2)$. There exists a positive constant δ_0, and a positive constant c_1 that depends only on $\underline{p}$, such that for every $\varepsilon \in (0, 1/2)$, every $\delta \in (0, \delta_0)$, every $p \in [0, 1/2]^n$, and every (ε,δ)-correct policy, we have*

$$\mathbf{E}_p[T] \geq c_1 \left\{ \frac{(|M(p,\varepsilon)| - 1)^+}{\varepsilon^2} + \sum_{\ell \in N(p,\varepsilon)} \frac{1}{(p_* - p_\ell)^2} \right\} \log \frac{1}{8\delta},$$

where $p_* = \max_i p_i$,

$$M(p,\varepsilon) = \left\{ \ell : \ p_\ell > p_* - \varepsilon, \text{ and } p_\ell > \underline{p}, \text{ and } p_\ell \geq \frac{\varepsilon + p_*}{1+\sqrt{1/2}} \right\}, \tag{4}$$

and

$$N(p,\varepsilon) = \left\{ \ell : p_\ell \leq p_* - \varepsilon, \text{ and } p_\ell > \underline{p}, \text{ and } p_\ell \geq \frac{\varepsilon + p_*}{1+\sqrt{1/2}} \right\}. \tag{5}$$

In particular, δ_0 *can be taken equal to* $e^{-8}/8$.

Remarks:

(a) The lower bound involves two sets of coins whose biases are not too far from the best bias p_*. The first set $M(p,\varepsilon)$ contains coins that are within ε from the best and would therefore be legitimate selections. In the presence of multiple such coins, a certain amount of exploration is needed to obtain the required confidence that none of these coins is significantly better than the others. The second set $N(p,\varepsilon)$ contains coins whose bias is more than ε away from p_*; they come into the lower bound because again some exploration is needed in order to obtain the required confidence that none of these coins is significantly better than the best coin in $M(p,\varepsilon)$.

(b) The expression $(\varepsilon + p_*)/(1+\sqrt{1/2})$ in Eqs. (4) and (5) can be replaced by $(\varepsilon + p_*)/(2-\alpha)$ for any positive constant α, by changing some of the constants in the proof.

(c) This result actually provides a family of lower bounds, one for every possible choice of $\underline{p}$. A tighter bound can be obtained by optimizing the choice of $\underline{p}$, while also taking into account the dependence of the constant c_1 on $\underline{p}$. This is not hard (the dependence of c_1 on $\underline{p}$ is described in Remark 7), but does not provide any new insights.

Proof Let us fix $\delta_0 = e^{-8}/8$, some $\underline{p} \in (0,1/2)$, $\varepsilon \in (0,1/2)$, $\delta \in (0,\delta_0)$, an (ε,δ)-correct policy, and some $p \in [0,1/2]^n$. Without loss of generality, we assume that $p_* = p_1$. Let us denote the true (unknown) bias of each coin by q_i. We consider the following hypotheses:

$$H_0 : \ q_i = p_i, \text{ for } i = 1,\ldots,n,$$

and for $\ell = 1,\ldots,n$,

$$H_\ell : q_\ell = p_1 + \varepsilon, \qquad q_i = p_i, \text{ for } i \neq \ell.$$

If hypothesis H_ℓ is true, the policy must select coin ℓ. We will bound from below the expected number of times the coins in the sets $N(p,\varepsilon)$ and $M(p,\varepsilon)$ must be tried, when hypothesis H_0 is true. As in Section 3, we use $\mathbf{E}_\ell$ and $\mathbf{P}_\ell$ to denote the expectation and probability, respectively, under the policy being considered and under hypothesis H_ℓ.

We define $\theta = 8\delta$, and note that $\theta < e^{-8}$. Let

$$t_\ell^* = \begin{cases} \dfrac{1}{c\varepsilon^2}\log\dfrac{1}{\theta}, & \text{if } \ \ell \in M(p,\varepsilon), \\ \dfrac{1}{c(p_1-p_\ell)^2}\log\dfrac{1}{\theta}, & \text{if } \ell \in N(p,\varepsilon), \end{cases}$$

where c is a constant that only depends on $\underline{p}$, and whose value will be chosen later. Recall that T_ℓ stands for the total number of times that coin ℓ is tried. We define the event

$$A_\ell = \{T_\ell \le 4t_\ell^*\}.$$

As in the proof of Theorem 1, if $\mathbf{E}_0[T_\ell] \le t_\ell^*$, then $\mathbf{P}_0(A_\ell) \ge 3/4$.

We define $K_t^\ell = X_1^\ell + \cdots + X_t^\ell$, which is the number of unit rewards ("heads") if the ℓ-th coin is tried a total of t (not necessarily consecutive) times. We let C_ℓ be the event defined by

$$C_\ell = \Big\{ \max_{1\le t\le 4t_\ell^*} |K_t^\ell - p_\ell t| < \sqrt{t_\ell^* \log(1/\theta)} \Big\}.$$

Similar to Lemma 2, and since $\theta = 8\delta < e^{-8}$, we have[3]

$$\mathbf{P}_0(C_\ell) > 7/8.$$

Let B_ℓ be the event $\{I = \ell\}$, i.e., that the policy eventually selects coin ℓ, and let B_ℓ^c be its complement. Since the policy is (ε,δ)-correct with $\delta < \delta_0 < 1/2$, we must have

$$\mathbf{P}_0(B_\ell^c) > 1/2, \qquad \forall\, \ell \in N(p,\varepsilon).$$

We also have $\sum_{\ell\in M(p,\varepsilon)} \mathbf{P}_0(B_\ell) \le 1$, so that the inequality $\mathbf{P}_0(B_\ell) > 1/2$ can hold for at most one element of $M(p,\varepsilon)$. Equivalently, the inequality $\mathbf{P}_0(B_\ell^c) \le 1/2$ can hold for at most one element of $M(p,\varepsilon)$. Let

$$M_0(p,\varepsilon) = \Big\{ \ell \in M(p,\varepsilon) \text{ and } \mathbf{P}_0(B_\ell^c) > \frac{1}{2} \Big\}.$$

It follows that $|M_0(p,\varepsilon)| \ge (|M(p,\varepsilon)| - 1)^+$.

The following lemma is an analog of Lemma 4.

Lemma 6 *Suppose that $\ell \in M_0(p,\varepsilon) \cup N(p,\varepsilon)$ and that $\mathbf{E}_0[T_\ell] \le t_\ell^*$. If the constant c in the definition of t^* is chosen large enough (possibly depending on $\underline{p}$), then $\mathbf{P}_\ell(B_\ell^c) > \delta$.*

Proof Fix some $\ell \in M_0(p,\varepsilon) \cup N(p,\varepsilon)$. For future reference, we note that the definitions of $M(p,\varepsilon)$ and $N(p,\varepsilon)$ include the condition $p_\ell \ge (\varepsilon + p_*)/(1+\sqrt{1/2})$. Recalling that $p_* = p_1$, $p_\ell \le 1/2$, and using the definition $\Delta_\ell = p_1 - p_\ell \ge 0$, some easy algebra leads to the conditions

$$\frac{\varepsilon + \Delta_\ell}{1 - p_\ell} \le \frac{\varepsilon + \Delta_\ell}{p_\ell} \le \frac{1}{\sqrt{2}}. \tag{6}$$

We define the event S_ℓ by

$$S_\ell = A_\ell \cap B_\ell^c \cap C_\ell.$$

Since $\mathbf{P}_0(A_\ell) \ge 3/4$, $\mathbf{P}_0(B_\ell^c) > 1/2$, and $\mathbf{P}_0(C_\ell) > 7/8$, we have

$$\mathbf{P}_0(S_\ell) > \frac{1}{8}, \qquad \forall\, \ell \in M_0(p,\varepsilon) \cup N(p,\varepsilon).$$

3. The derivation is identical to Lemma 2 except for Eq. (2), where one should replace the assumption that $\theta < e^{-4}$ with the stricter assumption that $\theta < e^{-8}$ used here.

As in the proof of Lemma 4, we define the likelihood function L_ℓ by letting

$$L_\ell(w) = \mathbf{P}_\ell(W = w),$$

for every possible history w, and use again $L_\ell(W)$ to define the corresponding random variable.

Let K be a shorthand notation for $K_{T_\ell}^\ell$, the total number of unit rewards ("heads") obtained from coin ℓ. We have

$$\begin{aligned}
\frac{L_\ell(W)}{L_0(W)} &= \frac{(p_1+\varepsilon)^K(1-p_1-\varepsilon)^{T_\ell-K}}{p_\ell^K(1-p_\ell)^{T_\ell-K}} \\
&= \left(\frac{p_1}{p_\ell}+\frac{\varepsilon}{p_\ell}\right)^K\left(\frac{1-p_1}{1-p_\ell}-\frac{\varepsilon}{1-p_\ell}\right)^{T_\ell-K} \\
&= \left(1+\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^K\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{T_\ell-K},
\end{aligned}$$

where we have used the definition $\Delta_\ell = p_1 - p_\ell$. It follows that

$$\begin{aligned}
\frac{L_\ell(W)}{L_0(W)} &= \left(1+\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^K\left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^K\left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^{-K}\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{T_\ell-K} \\
&= \left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^K\left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^{-K}\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{T_\ell-K} \\
&= \left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^K\left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^{-K}\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{K(1-p_\ell)/p_\ell} \\
&\qquad \cdot\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{(p_\ell T_\ell-K)/p_\ell}. \qquad (7)
\end{aligned}$$

We will now proceed to lower bound the right-hand side of Eq. (7) for histories under which event S_ℓ occurs. If event S_ℓ has occurred, then A_ℓ has occurred, and we have $K \le T_\ell \le 4t^*$, so that for every $\ell \in N(\varepsilon, p)$, we have

$$\begin{aligned}
\left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^K &\ge \left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^{4t_\ell^*} \\
&= \left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^{(4/c\Delta_\ell^2)\log(1/\theta)} \\
&\overset{a}{\ge} \exp\left\{-d\frac{4}{c}\left(\frac{(\varepsilon/\Delta_\ell)+1}{p_\ell}\right)^2\log(1/\theta)\right\} \\
&\overset{b}{\ge} \exp\left\{-d\frac{16}{cp_\ell{}^2}\log(1/\theta)\right\} \\
&= \theta^{16d/p_\ell^2 c}.
\end{aligned}$$

In step (a), we have used Lemma 3 which applies because of Eq. (6); in step (b), we used the fact $\varepsilon/\Delta_\ell \le 1$, which holds because $\ell \in N(\varepsilon, p)$.

Similarly, for $\ell \in M(\varepsilon, p)$, we have

$$\begin{aligned}
\left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^K &\geq \left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^{4t_\ell^*} \\
&= \left(1-\left(\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^2\right)^{(4/c\varepsilon^2)\log(1/\theta)} \\
&\stackrel{a}{\geq} \exp\left\{-d\frac{4}{c}\Big(\frac{1+(\Delta_\ell/\varepsilon)}{p_\ell}\Big)^2\log(1/\theta)\right\} \\
&\stackrel{b}{\geq} \exp\left\{-d\frac{16}{c{p_\ell}^2}\log(1/\theta)\right\} \\
&= \theta^{16d/p_\ell^2 c}.
\end{aligned}$$

In step (a), we have again used Lemma 3; in step (b), we used the fact $\Delta_\ell/\varepsilon \leq 1$, which holds because $\ell \in M(\varepsilon, p)$.

We now bound the product of the second and third terms in Eq. (7).

If $b \geq 1$, then the mapping $y \mapsto (1-y)^b$ is convex for $y \in [0,1]$. Thus, $(1-y)^b \geq 1-by$, which implies that

$$\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{(1-p_\ell)/p_\ell} \geq \left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right),$$

so that the product of the second and third terms can be lower bounded by

$$\left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^{-K}\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{K(1-p_\ell)/p_\ell} \geq \left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^{-K}\left(1-\frac{\varepsilon+\Delta_\ell}{p_\ell}\right)^{K} = 1.$$

We still need to bound the fourth term of Eq. (7). We start with the case where $\ell \in N(p,\varepsilon)$. We have

$$\begin{aligned}
\left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{(p_\ell T_\ell - K)/p_\ell} &\stackrel{a}{\geq} \left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{(1/p_\ell)\sqrt{t_\ell^*\log(1/\theta)}} && (8) \\
&\stackrel{b}{=} \left(1-\frac{\varepsilon+\Delta_\ell}{1-p_\ell}\right)^{(1/p_\ell\sqrt{c}\Delta_\ell)\log(1/\theta)} \\
&\stackrel{c}{\geq} \exp\left\{-\frac{d}{\sqrt{c}}\cdot\frac{\varepsilon+\Delta_\ell}{\Delta_\ell(1-p_\ell)p_\ell}\log(1/\theta)\right\} && (9) \\
&\stackrel{d}{\geq} \exp\left\{-\frac{2d}{\sqrt{c}(1-p_\ell)p_\ell}\log(1/\theta)\right\} && (10) \\
&\stackrel{e}{\geq} \exp\left\{-\frac{4d}{\sqrt{c}p_\ell}\log(1/\theta)\right\} \\
&= \theta^{4d/(p_\ell\sqrt{c})}.
\end{aligned}$$

Here, (a) holds because we are assuming that the events A_ℓ and C_ℓ occurred; (b) uses the definition of t_ℓ^* for $\ell \in N(p,\varepsilon)$; (c) follows from Eq. (6) and Lemma 3; (d) follows because $\Delta_\ell > \varepsilon$; and (e) holds because $0 \leq p_\ell \leq 1/2$, which implies that $1/(1-p_\ell) \leq 2$.

Consider now the case where $\ell \in M_0(p,\varepsilon)$. Equation (8) holds for the same reasons as when $\ell \in N(p,\varepsilon)$. The only difference from the above calculation is in step (b), where t_ℓ^* should be replaced with $(1/c\varepsilon^2)\log(1/\theta)$. Then, the right-hand side in Eq. (9) becomes

$$\exp\left\{-\frac{d}{\sqrt{c}}\cdot\frac{\varepsilon+\Delta_\ell}{\varepsilon(1-p_\ell)p_\ell}\log(1/\theta)\right\}.$$

For $\ell \in M_0(p,\varepsilon)$, we have $\Delta_\ell \leq \varepsilon$, which implies that $(\varepsilon+\Delta_\ell)/\varepsilon \leq 2$, which then leads to the same expression as in Eq. (10). The rest of the derivation is identical. Summarizing the above, we have shown that if $\ell \in M_0(p,\varepsilon)\cup N(p,\varepsilon)$, and event S_ℓ has occurred, then

$$\frac{L_\ell(W)}{L_0(W)} \geq \theta^{(4d/p_\ell\sqrt{c})+(16d/p_\ell^2 c)}.$$

For $\ell \in M_0(p,\varepsilon)\cup N(p,\varepsilon)$, we have $\underline{p} < p_\ell$. We can choose c large enough so that $L_\ell(W)/L_0(W) \geq \theta = 8\delta$; the value of c depends only on the constant $\underline{p}$. Similar to the proof of Theorem 1, we have

$$\frac{L_\ell(W)}{L_0(W)}1_{S_\ell} \geq 8\delta 1_{S_\ell},$$

where 1_{S_ℓ} is the indicator function of the event S_ℓ. It follows that

$$\mathbf{P}_\ell(B_\ell^c) \geq \mathbf{P}_\ell(S_\ell) = \mathbf{E}_\ell[1_{S_\ell}] = \mathbf{E}_0\left[\frac{L_\ell(W)}{L_0(W)}1_{S_\ell}\right] \geq \mathbf{E}_0[8\delta 1_{S_\ell}] = 8\delta\mathbf{P}_0(S_\ell) > \delta,$$

where the last inequality relies on the already established fact $\mathbf{P}_0(S_\ell) > 1/8$. □

Since the policy is (ε,δ)-correct, we must have $\mathbf{P}_\ell(B_\ell^c) \leq \delta$, for every ℓ. Lemma 6 then implies that $\mathbf{E}_0[T_\ell] > t_\ell^*$ for every $\ell \in M_0(p,\varepsilon)\cup N(p,\varepsilon)$. We sum over all $\ell \in M_0(p,\varepsilon)\cup N(p,\varepsilon)$, use the definition of t_ℓ^*, together with the fact $|M_0(p,\varepsilon)| \geq (|M(p,\varepsilon)|-1)^+$, to conclude the proof of the theorem. □

Remark 7 A close examination of the proof reveals that the dependence of c_1 on $\underline{p}$ is captured by a requirement of the form $c_1 \leq c_2\underline{p}^2$, for some absolute constant c_2. This suggests that there is a tradeoff in the choice of $\underline{p}$. By choosing a large $\underline{p}$, the constant c_1 is made larger, but the sets M and N become smaller, and vice versa.

The preceding result may give the impression that the sample complexity is high only when the p_i are bounded by 1/2. The next result shows that similar lower bounds hold (with a different constant) whenever the p_i can be assumed to be bounded away from 1. However, the lower bound becomes weaker (i.e., the constant c_1 is smaller) when the upper bound on the p_i approaches 1. In fact, the dependence of a lower bound on ε cannot be $\Theta(1/\varepsilon^2)$ when $\max_i p_i = 1$. To see this, consider the following policy π. Try each coin $O((1/\varepsilon)\log(n/\delta))$ times. If one of the coins always resulted in heads, select it. Otherwise, use some (ε,δ)-correct policy $\tilde{\pi}$. It can be shown that the policy π is (ε,δ)-correct (for every $p \in [0,1]^n$), and that if $\max_i p_i = 1$, then $\mathbf{E}_p[T] = O((n/\varepsilon)\log(n/\delta))$.

Theorem 8 *Fix an integer $s \geq 2$, and some $\underline{p} \in (0,1/2)$. There exists a positive constant c_1 that depends only on $\underline{p}$ such that for every $\varepsilon \in (0, 2^{-(s+2)})$, every $\delta \in (0, e^{-8}/8)$, every $p \in [0, 1-2^{-s}]^n$, and every (ε,δ)-correct policy, we have*

$$\mathbf{E}_p[T] \geq \frac{c_1}{s\eta^2} \left\{ \frac{(|M(\tilde{p}, \varepsilon\eta)| - 1)^+}{\varepsilon^2} + \sum_{\ell \in N(\tilde{p}, \eta\varepsilon)} \frac{1}{(p_* - p_\ell)^2} \right\} \log \frac{1}{8\delta},$$

where $p_* = \max_i p_i$, $\eta = 2^{s+1}/s$, $\tilde{p}$ is the vector with components $\tilde{p}_i = 1 - (1-p_i)^{1/s}$ (for $i = 1, 2, \ldots, n$), and M and N are as defined in Theorem 5.

Proof Let us fix $s \geq 2$, $\underline{p} \in (0,1/2)$, $\varepsilon \in (0, 2^{-(s+2)})$, and $\delta \in (0, e^{-8}/8)$. Suppose that we have an (ε,δ)-correct policy π whose expected time to termination is $\mathbf{E}_p[T]$, whenever the vector of coin biases happens to be p. We will use the policy π to construct a new policy $\tilde{\pi}$ such that

$$\mathbf{P}_{\tilde{p}}\Big(\tilde{p}_I > \max_i \tilde{p}_i - \eta\varepsilon\Big) \geq 1 - \delta, \qquad \forall\ \tilde{p} \in [0, (1/2) + \eta\varepsilon]^n;$$

(we will then say that $\tilde{\pi}$ is $(\eta\varepsilon, \delta)$-correct on $[0, (1/2) + \eta\varepsilon]^n$). Finally, we will use the lower bounds from Theorem 5, applied to $\tilde{\pi}$, to obtain a lower bound on the sample complexity of π.

The new policy $\tilde{\pi}$ is specified as follows. Run the original policy π. Whenever π chooses to try a certain coin i once, policy $\tilde{\pi}$ tries coin i for s consecutive times. Policy $\tilde{\pi}$ then "feeds" π with 0 if all s trials resulted in 0, and "feeds" π with 1 otherwise. If $\tilde{p}$ is the true vector of coin biases faced by policy $\tilde{\pi}$, and if policy π chooses to sample coin i, then policy π "sees" an outcome which equals 1 with probability $p_i = 1 - (1 - \tilde{p}_i)^s$. Let us define two mappings $f, g : [0,1] \mapsto [0,1]$, which are inverses of each other, by

$$f(p_i) = 1 - (1-p_i)^{1/s}, \qquad g(\tilde{p}_i) = 1 - (1 - \tilde{p}_i)^s,$$

and with a slight abuse of notation, let $f(p) = (f(p_1), \ldots, f(p_n))$, and similarly for $g(\tilde{p})$. With our construction, when policy $\tilde{\pi}$ is faced with a bias vector $\tilde{p}$, it evolves in an identical manner as the policy π faced with a bias vector $p = g(\tilde{p})$. But under policy $\tilde{\pi}$, there are s trials associated with every trial under policy π, which implies that $\tilde{T} = sT$ ($\tilde{T}$ is the number of trials under policy $\tilde{\pi}$) and therefore

$$\mathbf{E}^{\tilde{\pi}}_{\tilde{p}}[\tilde{T}] = s\mathbf{E}^{\pi}_{g(\tilde{p})}[T], \qquad \mathbf{E}^{\tilde{\pi}}_{f(p)}[\tilde{T}] = s\mathbf{E}^{\pi}_{p}[T], \tag{11}$$

where the superscript in the expectation operator indicates the policy being used.

We will now determine the "correctness" guarantees of policy $\tilde{\pi}$. We first need some algebraic preliminaries. Let us fix some $\tilde{p} \in [0, (1/2) + \eta\varepsilon]^n$ and a corresponding vector p, related by $\tilde{p} = f(p)$ and $p = g(\tilde{p})$. Let also $p_* = \max_i p_i$ and $\tilde{p}_* = \max_i \tilde{p}_i$. Using the definition $\eta = 2^{s+1}/s$ and the assumption $\varepsilon < 2^{-(s+2)}$, we have $\tilde{p}_* \leq (1/2) + (1/2s)$, from which it follows that

$$p_* \leq 1 - \left(\frac{1}{2} - \frac{1}{2s}\right)^s = 1 - \frac{1}{2^s}\left(1 - \frac{1}{s}\right)^s \leq 1 - \frac{1}{2^s} \cdot \frac{1}{4} = 1 - 2^{-(s+2)}.$$

The derivative f' of f is monotonically increasing on $[0,1)$. Therefore,

$$\begin{aligned} f'(p_*) &\leq f'(1 - 2^{-(s+2)}) = \frac{1}{s}\left(2^{-(s+2)}\right)^{(1/s)-1} = \frac{1}{s} 2^{-(s+2)(1-s)/s} \\ &= \frac{1}{s} 2^{s+1-(2/s)} \leq \frac{1}{s} 2^{s+1} = \eta. \end{aligned}$$

Thus, the derivative of the inverse mapping g satisfies

$$g'(\tilde{p}_*) \geq \frac{1}{\eta},$$

which implies, using the concavity of g, that

$$g(\tilde{p}_* - \eta\varepsilon) \leq g(\tilde{p}_*) - g'(\tilde{p}_*)\varepsilon\eta \leq g(\tilde{p}_*) - \varepsilon.$$

Let I be the coin index finally selected by policy $\tilde{\pi}$ when faced with $\tilde{p}$, which is the same as the index chosen by π when faced with p. We have (the superscript in the probability indicates the policy being used)

$$\begin{aligned}
\mathbf{P}_{\tilde{p}}^{\tilde{\pi}}(\tilde{p}_I \leq \tilde{p}_* - \eta\varepsilon) &= \mathbf{P}_{\tilde{p}}^{\tilde{\pi}}(g(\tilde{p}_I) \leq g(\tilde{p}_* - \eta\varepsilon)) \\
&\leq \mathbf{P}_{\tilde{p}}^{\tilde{\pi}}(g(\tilde{p}_I) \leq g(\tilde{p}_*) - \varepsilon) \\
&= \mathbf{P}_{p}^{\pi}(p_I \leq p_* - \varepsilon) \\
&\leq 1 - \delta,
\end{aligned}$$

where the last inequality follows because policy π was assumed to be (ε,δ)-correct. We have therefore established that $\tilde{\pi}$ is $(\eta\varepsilon, \delta)$-correct on $[0, (1/2) + \eta\varepsilon]^n$. We now apply Theorem 5, with $\eta\varepsilon$ instead of ε. Even though that theorem is stated for a policy which is (ε,δ)-correct for all possible p, the proof only requires the policy to be (ε,δ)-correct for $p \in [0, (1/2) + \varepsilon]^n$. This gives a lower bound on $\mathbf{E}_{\tilde{p}}^{\tilde{\pi}}[\tilde{T}]$ which, using Eq. (11), translates to the claimed lower bound on $\mathbf{E}_{p}^{\pi}[T]$. This lower bound applies whenever $p = g(\tilde{p})$, for some $\tilde{p} \in [0, 1/2]^n$, and therefore whenever $p \in [0, 1 - 2^{-s}]^n$. □

5. The Bayesian Setting

There is another variant of the problem which is of interest. In this variant, the parameters p_i associated with each arm are not unknown constants, but random variables described by a given prior. In this case, there is a single underlying probability measure which we denote by $\mathbf{P}$, and which is the average of the measures $\mathbf{P}_p$ over the prior distribution of p. We also use $\mathbf{E}$ to denote the expectation with respect to $\mathbf{P}$. We then define a policy to be (ε,δ)-correct, for a particular prior and associated measure $\mathbf{P}$, if

$$\mathbf{P}\Big(p_I > \max_i p_i - \varepsilon\Big) \geq 1 - \delta.$$

We then have the following result.

Theorem 9 *There exist positive constants c_1, c_2, ε_0, and δ_0, such that for every $n \geq 2$ and $\varepsilon \in (0, \varepsilon_0)$, there exists a prior for the n-bandit problem such that for every $\delta \in (0, \delta_0)$, and (ε,δ)-correct policy for this prior, we have*

$$\mathbf{E}[T] \geq c_1 \frac{n}{\varepsilon^2} \log \frac{c_2}{\delta}.$$

In particular, ε_0 and δ_0 can be taken equal to $1/8$ and $e^{-4}/12$, respectively.

Proof Let $\varepsilon_0 = 1/8$ and $\delta_0 = e^{-4}/12$, and let us fix $\varepsilon \in (0, \varepsilon_0)$ and $\delta \in (0, \delta_0)$. Consider the hypotheses $H_0, \ldots, H_n$, introduced in the proof of Theorem 1. Let the prior probability of H_0 be $1/2$, and the prior probability of H_ℓ be $1/2n$, for $\ell = 1, \ldots, n$. Fix an $(\varepsilon/2, \delta)$-correct policy with respect to this prior, and note that it satisfies

$$\mathbf{E}[T] \geq \frac{1}{2}\mathbf{E}_0[T] \geq \frac{1}{2}\sum_{\ell=1}^{n} \mathbf{E}_0[T_\ell]. \tag{12}$$

Since the policy is $(\varepsilon/2, \delta)$-correct, we have $\mathbf{P}(p_I > \max_\ell p_\ell - (\varepsilon/2)) \geq 1 - \delta$.

As in the proof of Theorem 5, let B_ℓ be the event that the policy eventually selects coin ℓ. We have

$$\frac{1}{2}\mathbf{P}_0(B_0) + \frac{1}{2n}\sum_{\ell=1}^{n} \mathbf{P}_\ell(B_\ell) \geq 1 - \delta,$$

which implies that

$$\frac{1}{2n}\sum_{\ell=1}^{n} \mathbf{P}_\ell(B_0) \leq \delta. \tag{13}$$

Let G be the set of hypotheses $\ell \neq 0$ under which the probability of selecting coin 0 is at most 3δ, i.e.,

$$G = \{\ell : 1 \leq \ell \leq n,\ \mathbf{P}_\ell(B_0) \leq 3\delta\}.$$

From Eq. (13), we obtain

$$\frac{1}{2n}(n - |G|)3\delta < \delta,$$

which implies that $|G| > n/3$. Following the same argument as in the proof of Lemma 4, we obtain that there exists a constant c such that if $\delta' \in (0, e^{-4}/4)$ and $\mathbf{E}_0[T_\ell] \leq (1/c\varepsilon^2)\log(1/4\delta')$, then $\mathbf{P}_\ell(B_0) > \delta'$. By taking $\delta' = 3\delta$ and requiring that $\delta \in (0, e^{-4}/12)$, we see that the inequality $\mathbf{E}_0[T_\ell] \leq (1/c\varepsilon^2)\log(1/12\delta)$ implies that $\mathbf{P}_\ell(B_0) > 3\delta$ (here, c is the same constant as in Lemma 4). But for every $\ell \in G$ we have $\mathbf{P}_\ell(B_0) \leq 3\delta$, and therefore $\mathbf{E}_0[T_\ell] \geq (1/c\varepsilon^2)\log(1/12\delta)$. Then, Eq. (12) implies that

$$\mathbf{E}[T] \geq \frac{1}{2}\sum_{\ell \in G} \mathbf{E}_0[T_\ell] \geq |G|\frac{1}{c\varepsilon^2}\log\frac{1}{12\delta} \geq c_1'\frac{n}{\varepsilon^2}\log\frac{c_2}{\delta},$$

where we have used the fact $|G| > n/3$ in the last inequality.

To conclude, we have shown that there exists constants c_1' and c_2 and a prior for a problem with $n+1$ coins, such that any $(\varepsilon/2, \delta)$-correct policy satisfies $\mathbf{E}[T] \geq (c_1' n/\varepsilon^2)\log(c_2/\delta)$. The result follows by taking a larger constant c_1' (to account for having $n+1$ and not n coins, and ε instead of $\varepsilon/2$). □

6. Regret Bounds

In this section we consider lower bounds on the regret of *any* policy, and show that one can derive the $\Theta(\log t)$ regret bound of Lai and Robbins (1985) using the techniques in this paper. The results of Lai and Robbins (1985) are asymptotic as $t \to \infty$, whereas ours deal with finite times t. Our lower bound has similar dependence in t as the upper bounds given by Auer et al. (2002a) for some

natural sampling algorithms. As in Lai and Robbins (1985) and Auer et al. (2002a), we also show that when t is large, the regret depends linearly on the number of coins.

Given a policy, let S_t be the total number of unit rewards ("heads") obtained in the first t time steps. The regret by time t is denoted by R_t, and is defined by

$$R_t = t \max_i p_i - S_t \,.$$

Note that the regret is a random variable that depends on the results of the coin tosses as well as of the randomization carried out by the policy.

Theorem 10 *There exist positive constants c_1, c_2, c_3, c_4, and a constant c_5, such that for every $n \geq 2$, and for every policy, there exists some $p \in [0,1]^n$ such that for all $t \geq 1$,*

$$\mathbf{E}_p[R_t] \geq \min\{c_1 t,\ c_2 n + c_3 t,\ c_4 n(\log t - \log n + c_5)\}. \tag{14}$$

The inequality (14) suggests that there are essentially two regimes for the expected regret. When n is large compared to t, the expected regret is linear in t. When t is large compared to n, the regret behaves like $\log t$, but depends linearly on n.

Proof We will prove a stronger result, by considering the regret in a Bayesian setting. By proving that the expectation with respect to the prior is lower bounded by the right-hand side in Eq. (14), it will follow that the bound also holds for at least one of the hypotheses. Consider the same scenario as in Theorem 1, where we have $n+1$ coins and $n+1$ hypotheses $H_0, H_1, \ldots, H_n$. The prior assigns a probability of $1/2$ to H_0, and a probability of $1/2n$ to each of the hypotheses $H_1, H_2, \ldots, H_n$. Similar to Theorem 1, we will use the notation $\mathbf{E}_\ell$ and $\mathbf{P}_\ell$ to denote expectation and probability when the ℓth hypothesis is true, and $\mathbf{E}$ to denote expectation with respect to the prior.

Let us fix t for the rest of the proof. We define T_ℓ as the number of times coin ℓ is tried in the first t time steps. The expected regret when H_0 is true is

$$\mathbf{E}_0[R_t] = \frac{\varepsilon}{2} \sum_{\ell=1}^{n} \mathbf{E}_0[T_\ell],$$

and the expected regret when H_ℓ ($\ell = 1, \ldots, n$) is true is

$$\mathbf{E}_\ell[R_t] = \frac{\varepsilon}{2}\mathbf{E}_\ell[T_0] + \varepsilon \sum_{i \neq 0,\ell} \mathbf{E}_\ell[T_i],$$

so that the expected (Bayesian) regret is

$$\mathbf{E}[R_t] = \frac{1}{2} \cdot \frac{\varepsilon}{2} \sum_{\ell=1}^{n} \mathbf{E}_0[T_\ell] + \frac{\varepsilon}{2} \cdot \frac{1}{2n} \sum_{\ell=1}^{n} \mathbf{E}_\ell[T_0] + \frac{\varepsilon}{2n} \sum_{\ell=1}^{n} \sum_{i \neq 0,\ell} \mathbf{E}_\ell[T_i]. \tag{15}$$

Let D be the event that coin 0 is tried at least $t/2$ times, i.e.,

$$D = \{T_0 \geq t/2\}\,.$$

We consider separately the two cases $\mathbf{P}_0(D) < 3/4$ and $\mathbf{P}_0(D) \geq 3/4$. Suppose first that $\mathbf{P}_0(D) < 3/4$. In that case, $\mathbf{E}_0[T_0] < 7t/8$, so that $\sum_{\ell=1}^{n} \mathbf{E}_0[T_\ell] \geq t/8$. Substituting in Eq. (15), we obtain $\mathbf{E}[R_t] \geq \varepsilon t/32$. This gives the first term in the right-hand side of Eq. (14), with $c_1 = \varepsilon/32$.

We assume from now on that $\mathbf{P}_0(D) \geq 3/4$. Rearranging Eq. (15), and omitting the third term, we have

$$\mathbf{E}[R_t] \geq \frac{\varepsilon}{4} \sum_{\ell=1}^{n} \left(\mathbf{E}_0[T_\ell] + \frac{1}{n} \mathbf{E}_\ell[T_0] \right).$$

Since $\mathbf{E}_\ell[T_0] \geq (t/2)\mathbf{P}_\ell(D)$, we have

$$\mathbf{E}[R_t] \geq \frac{\varepsilon}{4} \sum_{\ell=1}^{n} \left(\mathbf{E}_0[T_\ell] + \frac{t}{2n} \mathbf{P}_\ell(D) \right). \tag{16}$$

For every $\ell \neq 0$, let us define δ_ℓ by

$$\mathbf{E}_0[T_\ell] = \frac{1}{c\varepsilon^2} \log \frac{1}{4\delta_\ell}.$$

(Such a δ_ℓ exists because of the monotonicity of the mapping $x \mapsto \log(1/x)$.) Let $\delta_0 = e^{-4}/4$. If $\delta_\ell < \delta_0$, we argue exactly as in Lemma 4, except that the event B in that lemma is replaced by event D. Since $\mathbf{P}_0(D) \geq 3/4$, the same proof applies and shows that $\mathbf{P}_\ell(D) \geq \delta_\ell$, so that

$$\mathbf{E}_0[T_\ell] + \frac{t}{2n} \mathbf{P}_\ell(D) \geq \frac{1}{c\varepsilon^2} \log \frac{1}{4\delta_\ell} + \frac{t}{2n} \delta_\ell.$$

If on the other hand, $\delta_\ell \geq \delta_0$, then $\mathbf{E}_0[T_\ell] \leq (1/c\varepsilon^2)\log(1/4\delta_0)$, which implies (by the earlier analogy with Lemma 4) that $\mathbf{P}_\ell(D) \geq \delta_0$, and

$$\mathbf{E}_0[T_\ell] + \frac{t}{2n} \mathbf{P}_\ell(D) \geq \frac{1}{c\varepsilon^2} \log \frac{1}{4\delta_\ell} + \frac{t}{2n} \delta_0.$$

Using the above bounds in Eq. (16), we obtain

$$\mathbf{E}[R_t] \geq \frac{\varepsilon}{4} \sum_{\ell=1}^{n} \left(\frac{1}{c\varepsilon^2} \log \frac{1}{4\delta_\ell} + h(\delta_\ell) \frac{t}{2n} \right), \tag{17}$$

where $h(\delta) = \delta$ if $\delta < \delta_0$, and $h(\delta) = \delta_0$ otherwise. We can now view the δ_ℓ as free parameters, and conclude that $\mathbf{E}[R_t]$ is lower bounded by the minimum of the right-hand side of Eq. (17), over all δ_ℓ. When optimizing, all the δ_ℓ will be set to the same value. The minimizing value can be δ_0, in which case we have

$$\mathbf{E}[R_t] \geq \frac{n}{4c\varepsilon} \log \frac{1}{4\delta_0} + \delta_0 \frac{\varepsilon}{8} t.$$

Otherwise, the minimizing value is $\delta_\ell = n/2ct\varepsilon^2$, in which case we have

$$\mathbf{E}[R_t] \geq \left(\frac{1}{16c\varepsilon} + \frac{1}{4c\varepsilon} \log(c\varepsilon^2/2) \right) n + \frac{1}{4c\varepsilon} n \log(1/n) + \frac{n}{4c\varepsilon} \log t.$$

Thus, the theorem holds with $c_2 = (1/4c\varepsilon)\log(1/4\delta_0)$, $c_3 = \delta_0\varepsilon/8$, $c_4 = 1/4c\varepsilon$, and $c_5 = (1/4) + \log(c\varepsilon^2/2)$. □

7. Permutations

We now consider the case where the coin biases p_i are known up to a permutation. More specifically, we are given a vector $q \in [0,1]^n$, and we are told that the true vector p of coin biases is of the form $p = q \circ \sigma$, where σ is an unknown permutation of the set $\{1,\ldots,n\}$, and where $q \circ \sigma$ stands for permuting the components of the vector q according to σ, i.e., $(q \circ \sigma)_\ell = q_{\sigma(\ell)}$. We say that a policy is (q,ε,δ)-correct if the coin I eventually selected satisfies

$$\mathbf{P}_{q\circ\sigma}\Big(p_I > \max_\ell q_\ell - \varepsilon\Big) \geq 1-\delta,$$

for every permutation σ of the set $\{1,\ldots,n\}$. We start with a $O\left((n+\log(1/\delta))/\varepsilon^2\right)$ upper bound on the expected number of trials, which is significantly smaller than the bound obtained when the coin biases are completely unknown (cf. Sections 3 and 4). We also provide a lower bound which is within a constant factor of our upper bound.

We then consider a different measure of sample complexity: instead of the expected number of trials, we consider the maximum (over all sample paths) number of trials. We show that for every (q,ε,δ)-correct policy, there is a $\Theta((n/\varepsilon^2)\log(1/\delta))$ lower bound on the maximum number of trials. We note that in the median elimination algorithm of Even-Dar et al. (2002), the length of all sample paths is the same and within a constant factor from our lower bound. Hence our bound is again tight.

We therefore see that for the permutation case, the sample complexity depends critically on whether our criterion involves the expected or maximum number of trials. This is in contrast to the general case considered in Section 3: the lower bound in that section applies under both criteria, as does the matching upper bound from Even-Dar et al. (2002).

7.1 An Upper Bound on the Expected Number of Trials

Suppose we are given a vector $q \in [0,1]^n$, and we are told that the true vector p of coin biases is a permutation of q. The policy in Table 1 takes as input the accuracy ε, the confidence parameter δ, and the vector q. In fact the policy only needs to know the bias of the best coin, which we denote by $q^* = \max_\ell q_\ell$. The policy also uses an additional parameter $\delta' \in (0,1/2]$.

The following theorem establishes the correctness of the policy, and provides an upper bound on the expected number of trials.

Theorem 11 *For every $\delta' \in (0,1/2]$, $\varepsilon \in (0,1)$, and $\delta \in (0,1)$, the policy in Table 1 is guaranteed to terminate after a finite number of steps, with probability 1, and is (q,ε,δ)-correct. For every permutation σ, the expected number of trials satisfies*

$$\mathbf{E}_{q\circ\sigma}[T] \leq \frac{1}{\varepsilon^2}\left(c_1 n + c_2 \log\frac{1}{\delta}\right),$$

for some positive constants c_1 and c_2 that depend only on δ'.

Proof We start with a useful calculation. Suppose that at iteration k, the median elimination algorithm selects a coin I_k whose true bias is p_{I_k}. Then, using the Hoeffding inequality, we have

$$\mathbf{P}(|\hat{p}_k - p_{I_k}| \geq \varepsilon/3) \leq \exp\{-2(\varepsilon/3)^2 m_k\} \leq \frac{\delta}{2^k}. \tag{18}$$

Input: Accuracy and confidence parameters $\varepsilon \in (0,1)$ and $\delta \in (0,1)$; the bias of the best coin q^*.
Parameter: $\delta' \leq 1/2$.
0. $k = 1$; 1. Run the median elimination algorithm to find a coin I_k whose bias is within $\varepsilon/3$ of q^*, with probability at least $1-\delta'$. 2. Try coin I_k for $m_k = \lceil (9/2\varepsilon^2)\log(2^k/\delta) \rceil$ times. Let $\hat{p}_k$ be the fraction of these trials that result in "heads." 3. If $\hat{p}_k \geq q^* - 2\varepsilon/3$ declare that coin I_k is an ε-optimal coin and terminate. 4. Set $k := k+1$ and go back to Step 1.

Table 1: A policy for finding an ε-optimal coin when the bias of the best coin is known.

Let K be the number of iterations until the policy terminates. Given that $K > k-1$ (i.e., the policy did not terminate in the first $k-1$ iterations), there is probability at least $1-\delta' \geq 1/2$ that $p_{I_k} \geq q^* - (\varepsilon/3)$, in which case, from Eq. (18), there is probability at least $1-(\delta/2^k) \geq 1/2$ that $\hat{p}_k \geq q^* - (2\varepsilon/3)$. Thus, $\mathbf{P}(K > k \mid K > k-1) \leq 1-\eta$, with $\eta = 1/4$. Consequently, the probability that the policy does not terminate by the kth iteration, $\mathbf{P}(K > k)$, is bounded by $(1-\eta)^k$. Thus, the probability that the policy never terminates is bounded above by $(3/4)^k$ for all k, and is therefore 0.

We now bound the expected number of trials. Let c be such that the number of trials in one execution of the median elimination algorithm is bounded by $(cn/\varepsilon^2)\log(1/\delta')$. Then, the number of trials, $t(k)$, during the kth iteration is bounded by $(cn/\varepsilon^2)\log(1/\delta') + m_k$. It follows that the expected total number of trials under our policy is bounded by

$$\begin{aligned}
\sum_{k=1}^{\infty} \mathbf{P}(K \geq k) t(k) &\leq \frac{1}{\varepsilon^2} \sum_{k=1}^{\infty} (1-\eta)^{k-1} \left(cn\log(1/\delta') + (9/2)\log(2^k/\delta) + 1 \right) \\
&= \frac{1}{\varepsilon^2} \sum_{k=1}^{\infty} (1-\eta)^{k-1} \left(cn\log(1/\delta') + (9/2)\log(1/\delta) + (9k/2)\log 2 + 1 \right) \\
&\leq \frac{1}{\varepsilon^2}(c_1 n + c_2 \log(1/\delta)),
\end{aligned}$$

for some positive constants c_1 and c_2.

We finally argue that the policy is (q,ε,δ)-correct. For the policy to select a coin I with bias $p_I \leq q^* - \varepsilon$, it must be that at some iteration k, a coin I_k with $p_{I_k} \leq q^* - \varepsilon$ was obtained, but $\hat{p}_k$ came out larger than $q^* - 2\varepsilon/3$. From Eq. (18), for any fixed k, the probability of this occurring is bounded by $\delta/2^k$. By the union bound, the probability that $p_I \leq q^* - \varepsilon$ is bounded by $\sum_{k=1}^{\infty} \delta/2^k = \delta$. □

Remark 12 The knowledge of q^* turns out to be significant: it enables the policy to terminate as soon as there is high confidence that a coin has been found whose bias is larger than $q^* - \varepsilon$, without having to check the other coins. A policy of this type would not work for the hypotheses

considered in the proofs of Theorems 1 and 5: under those hypotheses, the value of q^* is not a priori known. We note that Theorem 11 disagrees with a lower bound in a preliminary version (Mannor and Tsitsiklis, 2003) of this paper. It turns out that the latter lower bound is only valid under an additional restriction on the set of policies, which will be the subject of Section 7.3.

7.2 A Lower Bound

We now prove that the upper bound in Theorem 11 is tight, within a constant.

Theorem 13 *There exist positive constants c_1, c_2, ε_0, and δ_1, such that for every $n \geq 2$ and $\varepsilon \in (0, \varepsilon_0)$, there exists some $q \in [0,1]^n$, such that for every $\delta \in (0, \delta_1)$ and every (q, ε, δ)-correct policy, there exists some permutation σ such that*

$$\mathbf{E}_{q \circ \sigma}[T] \geq \frac{1}{\varepsilon^2}\left(c_1 n + c_2 \log \frac{1}{\delta}\right).$$

Proof Let $\varepsilon_0 = 1/4$ and let $\delta_1 = \delta_0/5$, where δ_0 is the same constant as in Theorem 5. Let us fix some $n \geq 2$ and $\varepsilon \in (0, \varepsilon_0)$. We will establish the claimed lower bound for

$$q = (0.5 + \varepsilon,\ 0.5 - \varepsilon, \ldots, 0.5 - \varepsilon), \tag{19}$$

and for every $\delta \in (0, \delta_1)$. In fact, it is sufficient to establish a lower bound of the form $(c_2/\varepsilon^2)\log(1/\delta)$ and a lower bound of the form $c_1 n/\varepsilon^2$. We start with the former.

Part I. Let us consider the following three hypothesis testing problems. For each problem, we are interested in a δ-correct policy, i.e., a policy whose probability of error is less than δ under any hypothesis. We will show that a δ-correct policy for the first problem can be used to construct a δ-correct policy for the third problem, with the same sample complexity, and then apply Theorem 5 to obtain a lower bound.

Π_1: We have two coins and the bias vector is either $(0.5 - \varepsilon,\ 0.5 + \varepsilon)$ or $(0.5 + \varepsilon,\ 0.5 - \varepsilon)$. We wish to determine the best coin. This is a special case of our permutation problem, with $n = 2$.

Π_2: We have a single coin whose bias is either $0.5 - \varepsilon$ or $0.5 + \varepsilon$, and we wish to determine the bias of the coin.[4]

Π_3: We have two coins and the bias vector can be $(0.5, 0.5 - \varepsilon)$, $(0.5 + \varepsilon, 0.5 - \varepsilon)$, or $(0.5, 0.5 + \varepsilon)$. We wish to determine the best coin.

Consider a δ-correct policy for problem Π_1 except that the coin outcomes are encoded as follows. Whenever coin 1 is tried, record the outcome unchanged; whenever coin 2 is tried, record the opposite of the outcome (i.e., record a 0 outcome as a 1, and vice versa). Under the first hypothesis in problem Π_1, every trial (no matter which coin was tried) has probability $0.5 - \varepsilon$ of being equal to 1, and under the second hypothesis has probability $0.5 + \varepsilon$ of being equal to 1. With this encoding, it is apparent that the information provided by a trial of either coin in problem Π_1 is the same as the

4. A lower bound for this problem was provided in Lemma 5.1 from Anthony and Bartlett (1999). However, that bound is only established for policies with an a priori fixed number of trials, whereas our policies allow the number of trials to be determined adaptively, based on observed outcomes.

information provided by a trial of the single coin in problem Π_2. Thus, a δ-correct policy for Π_1 translates to a δ-correct policy for Π_2, with the same sample complexity.

In problem Π_3, note that coin 2 is the best coin if and only if its bias is equal to $0.5+\varepsilon$ (as opposed to $0.5-\varepsilon$). Thus, any δ-correct policy for Π_2 can be applied to the second coin in Π_3, to yield a δ-correct policy for Π_3 with the same sample complexity.

We now observe that problem Π_3 involves a set of three hypotheses, of the form considered in the proof of Theorem 5, for the case of two coins. More specifically, in terms of the notation used in that proof, we have $p=(0.5,0.5-\varepsilon)$, and $N(p,\varepsilon)=\{2\}$. It follows that the sample complexity of any δ-correct policy for Π_3 is lower bounded by $(c_1/\varepsilon^2)\log(1/8\delta)$, where c_1 is the constant in Theorem 5.[5] Because of the relation between the three problems established earlier, the same lower bound applies to any δ-correct policy for problem Π_1, which is the permutation problem of interest.

We have so far established a lower bound proportional to $(1/\varepsilon^2)/\log(1/\delta)$ for problem Π_1, which is the permutation problem we are interested in, with a q vector of the form (19), for the case $n=2$. Consider now the permutation problem for the q vector in (19), but for general n. If we are given the information that the best coin can only be one of the first two coins, we obtain problem Π_1. In the absence of this side-information, the permutation problem cannot be any easier. This shows that the same lower bound holds for every $n\geq 2$.

Part II: We now continue with the second part of the proof. We will establish a lower bound of the form $c_1 n/\varepsilon^2$ for the permutation problem associated with the bias vector q introduced in Eq. (19), to be referred to as problem Π. The proof involves a reduction of Π to a problem $\tilde{\Pi}$ of the form considered in the proof of Theorem 5.

The problem $\tilde{\Pi}$ involves $n+1$ coins (coins $0,1,\ldots,n$) and the following $n+2$ hypotheses:

$$H_0':\ p_0=0.5, \qquad p_i=0.5-\varepsilon,\ \text{for } i\neq 0,$$

$$H_0'':\ p_0=0.5+\varepsilon, \qquad p_i=0.5-\varepsilon,\ \text{for } i\neq 0,$$

and

$$H_\ell:\ p_0=0.5, \qquad p_\ell=0.5+\varepsilon, \qquad p_i=0.5-\varepsilon,\ \text{for } i\neq 0,\ell.$$

Note that the best coin is coin 0 under either hypothesis H_0' or H_0'', and the best coin is coin ℓ under H_ℓ, for $\ell\geq 1$. This leads us to define H_0 as the hypothesis that either H_0' or H_0'' is true.

We say that a policy for $\tilde{\Pi}$ is δ-correct if it selects the best coin with probability at least $1-\delta$. We will show that if we have a (q,ε,δ)-correct policy π for Π, with a certain sample complexity, we can construct an $(\varepsilon,5\delta)$-correct policy $\tilde{\pi}$ with a related sample complexity. We will then apply Theorem 5 to lower bound the sample complexity of $\tilde{\pi}$, and finally translate to a lower bound for π.

The idea of the reduction is as follows. In problem $\tilde{\Pi}$, if we knew that H_0 is not true, we would be left with a permutation problem with n coins, to which π could be applied. However, if H_0 is true, the behavior of π is unpredictable. (In particular, π might not terminate, or it might terminate with an arbitrary decision: this is because we are only assuming that π behaves properly when faced with the permutation problem Π.) If H_0 is true, we can replace coin 1 with a coin whose bias is $0.5+\varepsilon$, resulting in the bias vector q, in which case π is guaranteed to work properly. But what if we replace coin 1 as above, but some H_ℓ, $\ell\neq 0,1$, happens to be true? In that case, there will be two coins with bias $0.5+\varepsilon$ and π may misbehave. The solution is to run two processes in parallel, one

5. Although Theorem 5 is stated for (ε,δ)-correct policies, it is clear from the proof that the lower bound applies to any policy that has the desired behavior under all of the hypotheses considered in the proof.

with and one without this modification, in which case one of the two will have certain performance guarantees that we can exploit.

Consider the (q,ε,δ)-correct policy π for problem Π. Let t_π be the maximum (over all permutations σ) expected time until π terminates when the true coin bias vector is $q\circ\sigma$.

We define two more bias vectors that will be used below:

$$q_- = (0.5-\varepsilon,\ldots,0.5-\varepsilon),$$

and

$$q_+ = (0.5+\varepsilon,\ 0.5+\varepsilon,\ 0.5-\varepsilon,\ldots,0.5-\varepsilon).$$

Note that if H_0 is true in problem $\tilde{\Pi}$, and π is applied to coins $1,\ldots,n$, then π will be faced with the bias vector q_-. Also, if H_ℓ, is true in problem $\tilde{\Pi}$, for some $\ell\neq 0,1$, and we modify the bias of coin 1 to $0.5+\varepsilon$, then policy π will be faced with the bias vector q_+.

Let us note for future reference that, as in Eq. (18), if we sample a coin with bias $0.5+\varepsilon$ for $m \stackrel{\triangle}{=} \lceil (1/\varepsilon^2)\log(1/\delta)\rceil$ times, the empirical mean reward is larger than 0.5 with probability at least $1-\delta$. Similarly, if we sample a coin with bias $0.5-\varepsilon$ for m times, the empirical mean reward is smaller than 0.5 with probability at least $1-\delta$. Sampling a specific coin that many times, and comparing the empirical mean to 0.5, will be referred to as "validating" the coin.

We now describe policy $\tilde{\pi}$ for problem $\tilde{\Pi}$. The policy involves two parallel processes $\mathcal{A}$ and $\mathcal{B}$: it alternates between the two processes, and each process samples one coin in each one of its turns. The processes continue to sample the coins alternately until one of them terminates and declares one of the coins as the best coin (or equivalently selects a hypothesis). The parameter k below is set to $k=\lceil 18\log(1/\delta)\rceil$.

$\mathcal{A}$: Apply policy π to coins $1,2,\ldots,n$. If π terminates and selects coin ℓ, validate coin ℓ by sampling it m times. If the empirical mean reward is more than 0.5, then $\mathcal{A}$ terminates and declares coin ℓ as the best coin. If the empirical mean reward is less than or equal to 0.5, then $\mathcal{A}$ terminates and declares coin 0 as the best coin. If π has carried out $\lceil t_\pi/\delta\rceil$ trials, then $\mathcal{A}$ terminates and declares coin 0 as the best coin.

$\mathcal{B}$: Sample coin 1 for m times. If the empirical mean reward is more than 0.5, then $\mathcal{B}$ terminates and declares coin 1 as the best coin. Otherwise, replace coin 1 with another coin whose bias is $0.5+\varepsilon$. Initialize a counter N with $N=0$.

Repeat k times the following:

(a) Pick a random permutation τ (uniformly over the set of permutations).

(b) Run a τ-permuted version of π, to be referred to as $\tau\circ\pi$; that is, whenever π is supposed to sample coin i, $\tau\circ\pi$ samples coin $\tau(i)$ instead.

(c) If $\tau\circ\pi$ terminates and selects coin 1 as the best coin, set $N := N+1$.

If $N > 2k/3$, then $\mathcal{B}$ terminates and declares coin 0 as the best coin. Otherwise, wait until process $\mathcal{A}$ terminates.

We first address the issue of correctness of policy $\tilde{\pi}$. Note that $\tilde{\pi}$ is guaranteed to terminate in finite time, because process $\mathcal{A}$ can only run for a bounded number of steps. We consider separately the following cases:

1. Process $\mathcal{A}$ terminates first, H_0 is true. In this case the true bias vector faced by π is q_- rather than q. An error can happen only if π terminates and the coin erroneously passes the validation test. The probability of this occurring is at most δ. In all other cases (validation fails or the running time exceeds the time limit $\lceil t_\pi/\delta \rceil$), process $\mathcal{A}$ correctly declares H_0 to be the true hypothesis.

2. Process $\mathcal{A}$ terminates first, H_ℓ is true for some $\ell \neq 0$. Process $\mathcal{A}$ does not declare the correct H_ℓ if one of the following events occurs: $\mathcal{A}$ fails to select the best coin (probability at most δ, since π is (q,ε,δ)-correct); or the validation makes an error (probability at most δ); or the time limit is exceeded. By Markov's inequality, the probability that the running time T_π of policy π exceeds the time limit satisfies

$$\mathbf{P}_q(T_\pi \geq t_\pi/\delta) \leq \mathbf{E}_q[T_\pi]\delta/t_\pi \leq t_\pi\delta/t_\pi = \delta.$$

   So, the total the probability of errors that fall within this case is bounded by 3δ.

3. Process $\mathcal{B}$ terminates first, H_0 is true. Note that under H_0, process $\mathcal{B}$ is faced with n coins whose bias vector is q. Process $\mathcal{B}$ does not declare H_0 if one of the following events occurs: the initial validation of coin 1 makes an error (probability at most δ); or in k runs, the permuted versions of policy π make at least $k/3$ errors. Each such run has probability at most δ of making an error (since π is (q,ε,δ)-correct), independently for each run (because we use a random permutation before each run). Using Hoeffding's inequality, the probability of at least $k/3$ errors is bounded by $\exp\{-2k(1/3-\delta)^2\}$. Since $\delta < 1/12$, this probability is at most $e^{-k/8}$. So, the total the probability of errors that fall within this case is bounded by $\delta + e^{-k/8}$.

4. Process $\mathcal{B}$ terminates first, H_ℓ is true for some $\ell > 1$. Process $\mathcal{B}$ does not declare H_ℓ if one of the following events occurs: the initial validation of coin 1 makes an error (probability at most δ); or in k runs, the permuted versions of policy π select coin 1 for $N > 2k/3$ times. Since in each of the k runs the policy is faced with the bias vector q_+, there are no guarantees on its behavior. However, since we use a random permutation before each run, and since there are two coins with bias $0.5+\varepsilon$, namely coins 1 and ℓ, the probability that the permuted version of π selects coin 1 at any given run is bounded by 1/2. Using Hoeffding's inequality, the probability of selecting coin 1 more than $2k/3$ times is bounded by $\exp\{-2k((2/3)-(1/2))^2\} = e^{-k/18}$. So, the total the probability of errors that fall within this case is bounded by $\delta + e^{-k/18}$.

5. Process $\mathcal{B}$ terminates first, H_1 is true. Process $\mathcal{B}$ fails to declare coin 1 only if an error is made in the initial validation step, which happens with probability bounded by δ.

To conclude, using the union bound, when H_0 is true, the probability of error is bounded by $2\delta + e^{-k/8}$; when H_1 is true, it is bounded by 4δ; and when H_ℓ, $\ell > 1$ is true, it is bounded by $4\delta + e^{-k/18}$. Since $k = 18\log(1/\delta)$, we see that the probability of error, under any hypothesis, is bounded by 5δ.

We now examine the sample complexity of policy $\tilde{\pi}$. We consider two cases, depending on which hypothesis is true.

1. H_0 is true. In process $\mathcal{A}$, policy π is faced with the bias vector q_- and has no termination guarantees, unless it reaches the time limit $\lceil t_\pi/\delta \rceil$. As established earlier (case 3), Process $\mathcal{B}$

will terminate after the initial validation of coin 1 (m trials), plus possibly k runs of policy π (expected number of trials kt_π), with probability at least $1-e^{-k/8}$. Otherwise, $\mathcal{B}$ waits until $\mathcal{A}$ terminates (at most $1+t_\pi/\delta$ time). Multiplying everything by a factor of 2 (because the two processes alternate), the expected time until $\tilde{\pi}$ terminates is bounded by

$$2(m+kt_\pi)+2e^{-k/8}(m+t_\pi/\delta+1).$$

2. H_ℓ is true for some $\ell \neq 0$. In this case, process $\mathcal{A}$ terminates after the validation time m and the time it takes for π to run. Thus, the expected time until termination is bounded by $2(m+1+t_\pi)$.

We have constructed an $(\varepsilon, 5\delta)$-correct policy $\tilde{\pi}$ for problem $\tilde{\Pi}$. Using the above derived time bounds, and the definitions of k and m, the expected number of trials, under any hypothesis H_ℓ, is bounded from above by

$$4m+36\left(\log\left(\frac{1}{\delta}\right)+3\right)t_\pi.$$

On the other hand, problem $\tilde{\Pi}$ involves hypotheses of the form considered in the proof of Theorem 5, with $p=(0.5,\ 0.5-\varepsilon,\ldots,0.5-\varepsilon)$, and with $N(p,\varepsilon)=\{1,2,\ldots,n\}$. Thus, the expected number of trials under some hypothesis is bounded below by $(c_1 n/\varepsilon^2)\log(c_2/\delta)$, for some positive constants c_1 and c_2, leading to

$$c_1\frac{n}{\varepsilon^2}\log\frac{c_2}{\delta} \leq 4m+36\left(\log\left(\frac{1}{\delta}\right)+3\right)t_\pi.$$

This translates to a lower bound of the form $t_\pi \geq c_2 n/\varepsilon^2$, for some new constant c_2, and for all n larger than some n_0. But for $n \leq n_0$, we can use the lower bound $(c_2/\varepsilon^2)\log(1/\delta)$ that was derived in the first part of the proof. □

7.3 Pathwise Sample Complexity

The sample complexity of the policy presented in Section 7.1 was measured in term of the *expected* number of trials. Suppose, however, that we are interested in a policy for which the number of trials is always low. Let us say that a policy has a *pathwise sample complexity* of t, if the policy terminates after at most t trials, with probability 1. We note that the median elimination algorithm of Even-Dar et al. (2002) is a (q,ε,δ)-correct policy whose pathwise sample complexity is of the form $(c_1 n/\varepsilon^2)\log(c_2/\delta)$. In this section, we show that at least for a certain q, there is a matching lower bound on the pathwise sample complexity of any (q,ε,δ)-correct policy.

Theorem 14 *There exist positive constants c_1, c_2, ε_0, and δ_1 such that for every $n \geq 2$ and $\varepsilon \in (0,\varepsilon_0)$, there exists some $q \in [0,1]^n$, such that for every $\delta \in (0,\delta_1)$ and every (q,ε,δ)-correct policy π, there exists some permutation σ under which the pathwise sample complexity of π is at least*

$$c_1\frac{n}{\varepsilon^2}\log\left(\frac{c_2}{\delta}\right).$$

Proof The proof uses a reduction similar to the one in the proof of Theorem 13. Let $\varepsilon_0 = 1/8$ and $\delta_1 = \delta_0/2$, where $\delta_0 = e^{-8}/8$ is the constant in Theorem 5. Let q be the same as in the proof of Theorem 13 (cf. Eq. (19)), and consider the associated permutation problem, referred to as problem

Π. Fix some $\delta \in (0, \delta_1)$ and suppose that we have a (q, ε, δ)-correct policy π for problem Π whose pathwise sample complexity is bounded by t_π for every permutation σ. Consider also the problem $\tilde{\Pi}$ introduced in the proof of Theorem 13, involving the hypotheses H_0, $H_1, \ldots, H_n$. We will now use the policy π to construct a policy $\tilde{\pi}$ for problem $\tilde{\Pi}$.

We run π on the coins $1, 2, \ldots, n$. If π terminates at or before time t_π and selects some coin ℓ, we sample coin ℓ for $\lceil (1/\varepsilon^2) \log(1/\delta) \rceil$ times. If the empirical mean reward is larger than 0.5 we declare H_ℓ as the true hypothesis. If the empirical mean reward of coin ℓ is less than or equal to 0.5, or if π does not terminate by time t_π, we declare H_0 as the true hypothesis.

We start by showing correctness. Suppose first that H_0 is true. For the policy $\tilde{\pi}$ to make an incorrect decision, it must be the case that policy π selected some coin and the empirical mean reward of this coin was larger than 1/2; using Hoeffding's inequality, the probability of this event is bounded by δ. Suppose instead that H_ℓ is true for some $\ell \geq 1$. In this case, policy π is guaranteed to terminate within t_π steps. Policy $\tilde{\pi}$ will make an incorrect decision if either policy π makes an incorrect decision (probability at most δ), or if policy π makes a correct decision but the selected coin fails to validate (probability at most δ). It follows that policy $\tilde{\pi}$ is $(q, \varepsilon, 2\delta)$-correct.

The number of trials under policy $\tilde{\pi}$ is bounded by $t' = t_\pi + \lceil (1/\varepsilon^2) \log(1/\delta) \rceil$, under any hypothesis. On the other hand, using Theorem 5, the expected number of trials under some hypothesis is bounded below by $(c_1 n/\varepsilon^2) \log(c_2/2\delta)$, leading to

$$c_1 \frac{n}{\varepsilon^2} \log \frac{c_2}{2\delta} \leq t_\pi + \frac{1}{\varepsilon^2} \log \left(\frac{1}{\delta} \right).$$

this translates to a lower bound of the form $t \geq (c_1 n/\varepsilon^2) \log(c_2/\delta)$, for some new constants c_1 and c_2. □

8. Concluding Remarks

We have provided bounds on the number of trials required to identify a near-optimal arm in a multi-armed bandit problem, with high probability. For the problem formulations studied in Sections 3 and 5, the lower bounds match the existing $O((n/\varepsilon^2) \log(1/\delta))$ upper bounds. For the case where the values of the biases are known but the identities of the coins are not, we provided two different tight bounds, depending on the particular criterion being used (expected versus maximum number of trials). Our results have been derived under the assumption of Bernoulli rewards. Clearly, the lower bounds also apply to more general problem formulations, as long as they include Bernoulli rewards as a special case. It would be of some interest to derive similar lower bounds for other special cases of reward distributions. It is reasonable to expect that essentially the same results will carry over, as long as the Kullback-Leibler divergence between the reward distributions associated with different arms is finite (as in Lai and Robbins, 1985).

Acknowledgments

We would like to thank Susan Murphy and David Siegmund for pointing out some relevant references from the sequential analysis literature. We thank two anonymous reviewers for their comments. This research was supported by the MIT-Merrill Lynch partnership, the ARO under grant DAAD10-00-1-0466, and the National Science Foundation under grant ECS-0312921.

References

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

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002a.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proc. 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, 1995.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002b.

D.A. Berry and B. Fristedt. *Bandit Problems*. Chapman and Hall, 1985.

H. Chernoff. *Sequential Analysis and Optimal Design*. Society for Industrial & Applied Mathematics, 1972.

E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In J. Kivinen and R. H. Sloan, editors, *Fifteenth Annual Conference on Computational Learning Theory*, pages 255–270. Springer, 2002.

C. Jennison, I. M. Johnstone, and B. W. Turnbull. Asymptotically optimal procedures for sequential adaptive selection of the best of several normal means. In S. S. Gupta and J. Berger, editors, *Statistical decision theory and related topics III*, volume 3, pages 55–86. Academic Press, 1982.

S. R. Kulkarni and G. Lugosi. Finite-time lower bounds for the two-armed bandit problem. *IEEE Trans. Aut. Control*, 45(4):711–714, 2000.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

S. Mannor and J.N. Tsitsiklis. Lower bounds on the sample complexity of exploration in the multi-armed bandit problem. In B. Schölkopf and M. K. Warmuth, editors, *Sixteenth Annual Conference on Computational Learning Theory*, pages 418–432. Springer, 2003.

H. Robbins. Some aspects of sequential design of experiments. *Bulletin of the American Mathematical Society*, 55:527–535, 1952.

S. M. Ross. *Stochastic Processes*. Wiley, 1983.

D. Siegmund. *Sequential analysis: Tests and Confidence Intervals*. Springer Verlag, 1985.

Journal of Machine Learning Research 5 (2004) 649-667 Submitted 1/04; Published 6/04

Preference Elicitation and Query Learning

Avrim Blum AVRIM@CS.CMU.EDU
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891, USA

Jeffrey Jackson JACKSON@MATHCS.DUQ.EDU
Department of Mathematics and Computer Science
Duquesne University
Pittsburgh, PA 15282-0001, USA

Tuomas Sandholm SANDHOLM@CS.CMU.EDU
Martin Zinkevich MAZ@CS.CMU.EDU
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891, USA

Editors: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

In this paper we explore the relationship between "preference elicitation", a learning-style problem that arises in combinatorial auctions, and the problem of learning via queries studied in computational learning theory. Preference elicitation is the process of asking questions about the preferences of bidders so as to best divide some set of goods. As a learning problem, it can be thought of as a setting in which there are multiple target concepts that can each be queried separately, but where the goal is not so much to learn each concept as it is to produce an "optimal example". In this work, we prove a number of similarities and differences between two-bidder preference elicitation and query learning, giving both separation results and proving some connections between these problems.

Keywords: exact learning, query learning, combinatorial auctions, preference elicitation.

1. Introduction

In a combinatorial auction, an entity (the "auctioneer") has a set S of n items that he would like to partition among a set of k bidders. What makes an auction *combinatorial* is that the valuations of the bidders — how much they would be willing to pay for different subsets of items — may not necessarily be linear functions over the items. For instance, if item a is a left shoe and item b is a right shoe, then a bidder might be willing to pay a reasonable amount for the bundle $\{a,b\}$ but very little for just $\{a\}$ or just $\{b\}$. In the other direction, if a and b are each *pairs* of shoes, then a bidder might value $\{a,b\}$ less than the sum of his valuations on $\{a\}$ and $\{b\}$ (especially if he just needs one pair of shoes right now). A standard goal for the auctioneer in such a setting is to determine the allocation of goods that maximizes *social welfare*: this is the sum, over all bidders, of the value that each bidder places on the set of items that he receives. This goal is perhaps most natural if one thinks of the auctioneer as not having a financial interest of its own but simply as an agent acting to help divide up a given set of items in a way that maximizes overall happiness or value. For

example, the auctioneer might be a government agency, or a manager dividing up resources among different projects. The case of $k = 2$ can also be thought of as a situation in which one of the bidders represents a buyer (with various preferences over bundles of items) and the other bidder represents a marketplace (with various discounts and package-deals), and the auctioneer is helping the buyer decide what subset of items to purchase from the marketplace.[1]

There are a number of issues that arise in the combinatorial auction setting. For example, there is much work on designing protocols (mechanisms) so that bidders will be truthful in reporting their valuations and not want to "game" the system (Sandholm, 2002b; Nisan and Ronen, 2000; Lehmann et al., 2002). But another issue is that even if we can get bidders to be truthful, their valuation functions can be quite complicated. Because of this, bidding in a traditional manner may require an exponential amount of communication. This has led researchers to study the notion of preference elicitation, in which the auctioneer asks questions of the bidders in order to learn (elicit) enough information about their preferences to determine the best, or approximately best, allocation of the items (Conen and Sandholm, 2001). Because the issues of truthfulness can be handled in this setting via known mechanisms (discussed later in Section 6), the problem then becomes one of determining which questions to ask in order to extract the information needed for allocation, and to understand when this can be done quickly.

1.1 Preference Elicitation and Query Learning

We can think of preference elicitation in the context of query learning by thinking of each item as a boolean feature, thinking of a subset of the n items (a "bundle") as an example $x \in \{0,1\}^n$ indicating which items are in the subset, and thinking of the bidder's valuation function as a target function. The standard assumption of "free disposal" — bidders can throw away items for free — means we can assume that these valuation functions are *monotone*, though they typically will not be boolean-valued (we might have $f(x) = \$100$, for instance). Furthermore, one of the natural types of queries studied in preference elicitation, the *value query* (where the auctioneer asks the bidder how much he values some bundle), corresponds exactly with the learning-theoretic notion of a membership query.

On the other hand, a key difference between preference elicitation and query learning is in the goals. In learning, the objective is typically to recover the target function, either exactly or approximately. In preference elicitation, however, the goal is more one of finding the "best example". For instance, if there are just two bidders with preference functions f and g, then the goal is to find a partition (S', S'') of the n items to maximize $f(S') + g(S'')$. Thinking in terms of functions over $\{0,1\}^n$, the goal is to find $x \in \{0,1\}^n$ to maximize $f(x) + g(\overline{x})$.

Notice that one of the immediate differences between preference elicitation and query learning is that preference elicitation makes sense even if the target functions do not have short descriptions, or even short approximations. We will see some interesting examples later, but as a simple case, if we learn that bidder A will pay \$100 for the entire set of n items but no more than \$50 for any subset of size $n-1$ (she is a collector and wants the whole set), and B will pay a maximum of \$50 even for the whole lot, then we know we might as well give all items to A, and we do not need to know exactly how much each bidder would have paid for different subsets. On the other hand, it is quite possible for allocation of items to be *computationally* hard, even if the preferences of all the bidders are known. For example, even if each bidder's preferences can be expressed as a simple conjunction

1. To think of this as a combinatorial auction, it is easiest to imagine that the auctioneer has pre-purchased *all* the items, and then is deciding which the buyer should keep and which should be returned for a refund.

(these are called "single-minded" bidders), then if there are many bidders, allocation is equivalent to the NP-hard set-packing problem. Even for two bidders, allocation can be NP-hard for somewhat more complicated preference functions, such as read-once formulas (Zinkevich et al., 2003).

Another difference concerns the types of queries that are most natural in each setting. While value/membership queries are common to both, equivalence queries are quite *un*natural in the context of preference elicitation. On the other hand, the *demand query*, a powerful type of query for preference elicitation introduced by Nisan and Segal (2003), does not seem to have been studied in query learning.[2]

In this paper, we discuss similarities and differences between the three objectives of exact learning, approximate learning, and preference elicitation. We then give a number of upper and lower bounds for preference elicitation of natural preference (concept) classes. We focus primarily on the case of $k = 2$ bidders, because even this case is quite interesting, both practically (since it models a buyer and a marketplace as mentioned above) and technically. We first consider monotone DNF formulas, which have long been known to be hard to learn exactly from membership queries alone but easy to learn approximately (in a strong, distribution-free PAC sense) given membership queries and polynomially many random examples of the target formula (Angluin, 1988). We show that monotone DNF formulas are *hard* for preference elicitation, even with demand queries. However, the hardness we show is $2^{\Omega(\sqrt{n})}$-hard rather than the $2^{\Omega(n)}$-hardness one gets for exact learning of monotone DNF. On the other hand, we show $\log(n)$-DNF formulas are *easy* for preference-elicitation, even if the functions have more than polynomially many terms. We also give a number of general statements about when the ability to succeed for one of these goals implies being able to succeed in the others. After comparing elicitation with boolean-valued and real-valued preference functions, we briefly consider the issue of truthfulness in elicitation. We then summarize subsequent related work that has been done since the conference (COLT-03) version of this paper appeared. Finally, we end with two open problems.

1.2 Related Work on Combinatorial Auctions

There is a substantial literature of work on combinatorial auctions. The standard view of these auctions (without the interactive notion of preference elicitation) is that bidders submit bids on bundles in some appropriate bidding language, and then the auctioneer determines who wins what based on these bids. The issue of determining the winners in such auctions, given the bids, is a complex optimization problem that has received considerable attention (Rothkopf et al., 1998; Sandholm, 2002a; Fujishima et al., 1999; Nisan, 2000; Andersson et al., 2000; Sandholm et al., 2001). Communication complexity issues have received substantial attention as well. There are $2^n - 1$ bundles, and each agent may need to bid on all of them to fully express its preferences. Appropriate bidding languages (Sandholm, 2002a,b; Fujishima et al., 1999; Nisan, 2000; Hoos and Boutilier, 2001; Sandholm and Suri, 2001) can address the communication overhead in some cases where the bidder's utility function is compressible. However, they still require the agents to completely determine and transmit their valuation functions and as such do not solve all the issues. So in practice, when the number of items for sale is even moderate, the bidders cannot bid on all bundles. Instead, they may bid on bundles which they will not win, and they may fail to bid on bundles they

2. In a demand query, the auctioneer proposes a price for each item and then asks the bidder to specify a single subset of the items that is optimal for the bidder given those prices. Demand queries will be discussed further in Section 2.

would have won. The former problem leads to wasted effort, and the latter problem leads to reduced economic efficiency of the resulting allocation of items to bidders.

Preference elicitation, the subject of this paper, was recently proposed to address these problems (Conen and Sandholm, 2001), and several papers have studied different types of elicitors (Conen and Sandholm, 2002b,a; Hudson and Sandholm, 2004; Nisan and Segal, 2003; Smith et al., 2002). On the negative side, if valuations are arbitrary monotone functions, then the worst-case communication complexity to find an (even approximately) optimal allocation is exponential in the number of items, no matter what query types are used (Nisan and Segal, 2003). Nonetheless, empirically, preference elicitation can provide a substantial savings compared to explicitly bidding on all $2^n - 1$ bundles (Hudson and Sandholm, 2004).

An issue that arises in the study of auctions is that bidders may lie about their preferences (e.g., bidder f when queried with x may not truthfully report $f(x)$) if they believe it may help in the final allocation. However, *Vickrey-Clarke-Groves (VCG)* schemes (Vickrey, 1961; Clarke, 1971; Groves, 1973) provide a method for charging bidders so that each is motivated to tell the truth about its valuations. Briefly, in this scheme the elicitor first finds the optimal allocation OPT under the assumption that bidders are behaving truthfully. Then, for each bidder i, the elicitor finds the optimal allocation OPT_i without bidder i, again assuming truthful behavior. Finally, bidder i is charged a fee based on the difference between the utility of the other agents in OPT and OPT_i. It seems perhaps circular at first glance, but one can show that in such a scheme, each bidder will in fact be motivated to be truthful throughout the whole process (Conen and Sandholm, 2001). Formally, bidding truthfully is an *ex-post equilibrium*. We discuss these issues in more detail in Section 6, but the conclusion is that if one can elicit the optimal allocation exactly assuming that agents tell the truth, one can determine VCG payments that make truth-telling the best strategy for the bidders. Because of this, for the remainder of the paper (except Section 6) we will assume that the bidders are truthful.

Driven by the same concerns as preference elicitation in combinatorial auctions, there has also been significant recent work on ascending combinatorial auctions (Parkes, 1999b,a; Ausubel and Milgrom, 2002; Wurman and Wellman, 2000; Bikhchandani et al., 2001; Bikhchandani and Ostroy, 2001). These are multistage mechanisms. At each stage the auctioneer announces prices (on items or in some cases on bundles of items), and each bidder states which bundle of items he would prefer (that is, which bundle would maximize his valuation minus the price he would have to pay for the bundle) at those prices. The auctioneer increases the prices between stages, and the auction usually ends when the optimal allocation is found. Ascending auctions can be viewed as a special case of preference elicitation where the queries are demand queries ("If these were the prices, what bundle would you buy from the auction?") and the query policy is constrained to increasing the prices in the queries over time. Recently it was shown that if *per-item* prices suffice to support an optimal allocation (i.e., a *Walrasian equilibrium* exists), then the optimal allocation can be found with a polynomial number of queries, where each query and answer is of polynomial size (Nisan and Segal, 2003).

Recently, some of us (Zinkevich et al., 2003), noticing the connection to query learning, showed how the algorithm of Angluin et al. (1993) for learning read-once formulas over standard boolean gates could be adapted to elicit preferences expressible as read-once-formulas over gates that are especially natural in the context of combinatorial auctions. This work goes on to discuss computational considerations, showing on the negative side that allocation can be NP-hard even for two

bidders with read-once-formula preferences, but on the other hand, allocation can be done in polynomial time if one of the two bidders has a linear value function.

2. Notation and Definitions

Because subset notation is most natural from the point of view of preference elicitation, we will use both subset notation and bit-vector notation in this paper. That is, we will think of the instance space X both as elements of $\{0,1\}^n$ and as the power set of some set S of *n items*. We will also interchangeably call a subset of S a "bundle" or an "example". When discussing preference elicitation, we assume there are k bidders with monotone real-valued preference functions over the instance space. The objective of preference elicitation is to determine a k-way partition $(S_1,\ldots,S_k)$ of S to maximize $f_1(S_1)+f_2(S_2)+\ldots+f_k(S_k)$, where $f_1,\ldots,f_k$ are the k real-valued preference functions. Unless otherwise noted, we will assume $k=2$; Section 1.1 provides some justification for this focus.

Let C be a class of monotone functions. We will be interested in the learnability of various C — that is, in various subsets of the class of monotone functions — in the exact learning, approximate learning, and preference elicitation models given the ability to make various types of queries. While learning algorithms are typically considered efficient if they run in time polynomial in the number of items n and in the length of the representation of the target (and possibly other parameters), we will at times explicitly require run time bounds independent of description length in order to demonstrate a fundamental advantage of preference elicitation for problems involving complex targets. The hardness observations for learning problems when this restriction is in place are therefore not hardness results in the standard learning-theoretic sense.

Query types: A *membership query* or *value query* is a request $x \in \{0,1\}^n$ to an oracle for a target f. The oracle responds with the value $f(x)$ corresponding to x. We can think of these queries as asking the following question of a bidder: "How much are you willing to pay for this bundle of items?"

A *demand query* is a request $w \in (\mathbf{R}^+)^n$ ($\mathbf{R}^+$ here represents non-negative real values) to an oracle for a target f. The oracle responds with an example $x \in \{0,1\}^n$ that maximizes $f(x)-w\cdot x$. We can think of a demand query w as asking the following question of a bidder: "If you were in a store in which item i had cost w_i, what subset of items would you choose to buy?"

We can illustrate the power of demand queries with the following observation due to Nisan. If one of the bidders has a linear valuation function, and the other is arbitrary, then preference elicitation can be done with $n+1$ queries: n value queries and one demand query. Specifically, we simply ask the linear bidder n value queries to determine his value on each item, and then send the other bidder these values as prices and ask him what he would like to buy. Thus it is interesting that our main lower bounds, for elicitation of monotone DNF formulas, hold for demand queries as well.[3]

Natural function/representation classes: One of the most natural representation classes of monotone functions in machine learning is that of monotone DNF formulas. In combinatorial auctions,

3. Lahaie and Parkes (2004) recently studied a more powerful notion of demand query in which one can propose an arbitrary polynomial-size function $h(x)$, and receive the x that maximizes $f(x)-h(x)$. With this type of query, one *can* elicit monotone DNF formulas, since one can now directly apply Angluin's algorithm (Angluin, 1988), making a demand query of this kind whenever Angluin's algorithm would make an equivalence query. Lahaie and Parkes go on to explore further the power of this query class.

the analog of this representation is called the "XOR bidding language" (Sandholm, 2002a).[4] A preference in this representation is a set of bundles (terms) $T = \{T_1, T_2, \ldots, T_m\}$ along with a set of values $v = \{v_1, v_2, \ldots, v_m\}$, one value v_i for each bundle T_i. The value of this preference for all $S' \subseteq S$ is

$$f_{T,v}(S') = \max_{T_i \subseteq S'} v_i.$$

In other words, the value of a set of items S' is the maximum value of any of the "desired bundles" in T that are contained in S'. We will call this the *DNF representation* of preferences, or "DNF preferences" for short. Our hardness results for this class will all go through for the boolean case (all v_i are equal to 1), while two of our positive results will hold for general v_i.

3. DNF Preferences

Angluin (1988) shows that monotone DNF formulas are hard to exactly learn from value (membership) queries alone, but are easy to learn approximately given membership queries plus random examples (in the PAC distribution-free, strong learning model). Angluin's example showing hardness of exact learning can be thought of as follows: imagine the n items are really $n/2$ pairs of shoes. The buyer would be happy with any bundle containing at least one pair of shoes (any such bundle is worth \$1). But then we add one final term to the DNF: a bundle of size $n/2$ containing exactly one shoe from each pair, where for each pair we flip a coin to decide whether to include the left or right shoe. Since the learning algorithm already knows the answer will be positive to any query containing a pair of shoes, the only interesting queries are those that contain no such pair, and therefore it has to match the last term *exactly* to provide any information. Thus even for a randomized algorithm, an expected $2^{n/2-1}$ value queries are needed for exact learning of monotone DNF formulas.

We now consider the two-bidder preference elicitation problem when one or both of the preferences are represented as monotone DNF expressions, beginning with a few simple observations.

Observation 1 *If f is a* known *DNF preference function with m terms, and g is an* arbitrary *unknown monotone preference function, then preference elicitation can be performed using $m+1$ value queries.*

Proof Because g is monotone, the optimal allocation will either be of the form $(T_i, S - T_i)$, for some term T_i in f, or else all of the items in S will go to the bidder with preference g. So, we simply need to query g once for S and once for each set $S - T_i$ and then pick the best of these $m+1$ partitions. ■

Observation 2 *If f and g are boolean DNF preferences each containing at most one term with more than two literals (the hard case in Angluin's construction) then preference elicitation can be performed using poly(n) value queries.*

Proof Since f and g are boolean (they each assign every bundle a value of either 0 or 1), the problem is simply to find terms T_f in f and T_g in g which have no items in common, if such a pair of

4. This terminology is to indicate that the bidder wants only one of his listed bundles and will not pay more for a set of items that contains multiple bundles inside it. This usage is very different from the standard definition of XOR as a sum modulo 2. Therefore, to avoid confusion, we will not use the XOR terminology here.

terms exists. We begin by finding all terms in f of size ≤ 2 by asking n^2 queries. Suppose two of these terms T_1 and T_2 are disjoint. In that case, we query g on $S-T_1$ and $S-T_2$. If one answer is 1 then we are done. If both answers are 0 then this means all of g's terms intersect both T_1 and T_2. In particular, g can have only a constant number of terms, and therefore *exactly* learning g is easy, after which we can then apply Observation 1 (swapping f and g). On the other hand, if f does not have two disjoint terms of size ≤ 2, then the only way f can have more than three such terms is if they all share some common item x_i. It is thus now easy to learn the large term in f: if $f(S-\{x_i\})=1$, we can find this term by "walking downward" from the example $S-\{x_i\}$, that is, by iteratively removing as many items as possible from this bundle subject to keeping f's value 1. On the other hand, if $f(S-\{x_i\})=0$, meaning that the large term contains x_i as well, we can walk downward from the example in which all the *other* items in the small terms have been removed. Once f has been learned, we can again apply Observation 1. ■

We now show that even though Angluin's specific example is no longer hard in the preference elicitation model, monotone DNF formulas (defining boolean preference functions) remain hard for preference elicitation using value queries, even when the preference functions are quite small. We then extend this result to demand queries as well.

Theorem 1 *Preference elicitation of monotone DNF formulas requires $2^{\Omega(\sqrt{n})}$ value queries. This holds even if each bidder's preference function has only $O(\sqrt{n})$ terms.*

Proof We construct a hard example as follows. There will be $n=m^2$ items, arranged in an m-by-m matrix. Let us label the items x_{ij} for $1\leq i,j\leq m$. We will call the two preference functions f_R and f_C. Both will be boolean functions. Bidder f_R is happy with any row: that is, $f_R = x_{11}x_{12}\cdots x_{1m} \vee x_{21}x_{22}\cdots x_{2m} \vee \ldots \vee x_{m1}x_{m2}\cdots x_{mm}$. Bidder f_C is happy with any column: that is, $f_C = x_{11}x_{21}\cdots x_{m1} \vee x_{21}x_{22}\cdots x_{m2} \vee \ldots \vee x_{1m}x_{2m}\cdots x_{mm}$. Thus, at this point, it is impossible to make both bidders happy. However, we now add one additional term to each preference function. We flip a coin for each of the n items in S, labeling the item as heads or tails. Let H be the set of all items labeled heads, and T be the set of all items labeled tails. We now add the conjunction of all items in H as one additional term to f_R, and the conjunction of all items in T as one additional term to f_C. Thus now it *is* possible to make both bidders happy, and the optimal allocation will be to give the items in H to the "row bidder" and the items in T to the "column bidder".

We now argue that no query algorithm can find this allocation in less than $\frac{1}{2}2^{\sqrt{n}}-2$ queries in expectation. Let us enforce that the last two questions of the query protocol are the values of the actual allocation. That is, if the elicitor assigns the items in H to the row agent and T to the column agent, it must ask the row agent the value of H and the column agent the value of T. This constraint only increases the length of the protocol by at most 2 questions.

Let us assume that the elicitor knows in advance the structure of the problem, the row sets and the column sets, and the only information the elicitor does not know are the sets H and T. In this case, we can assume without loss of generality that the elicitor never asks the row bidder about any bundle containing a row (because he already knows the answer will be "yes") and similarly never asks the column bidder about any bundle containing a column.

We now argue as follows. If the elicitor asks a query of the row bidder, the query must be missing at least one item in each row, and if the elicitor ask a query of the column bidder, it must be missing at least one item in each column. However, notice that in the first case, the answer will be

positive only if all missing items are in T, and in the second case, the answer will be positive only if all missing items are in H. Therefore, for any given such query, the probability that the answer will be positive taken over the random coin flips is at most $2^{-\sqrt{n}}$. Thus, for any elicitation strategy, the probability the elicitor gets a positive response in the first q queries is at most $q2^{-\sqrt{n}}$ and therefore the expected number of queries is at least $\frac{1}{2}2^{\sqrt{n}}$. ■

We now show that preference elicitation remains hard for DNF preferences even if we allow demand queries.

Theorem 2 *Even if both demand queries and value queries are allowed, preference elicitation of monotone DNF formulas requires $2^{\Omega(\sqrt{n})}$ queries. This holds even if each bidder's preference function has only $O(\sqrt{n})$ terms.*

Proof We use the same example as in the proof of Theorem 1. As in that proof, we can insist that the last question be a demand query where the agent responds with the set H or T respectively. Let us without loss of generality consider a sequence of demand queries to the "row bidder". What we need to calculate now is the probability, for any given cost vector w, that the set H happens to be the cheapest term in his DNF formula. The intuition is that this is highly unlikely because H is so much larger than the other terms.

Specifically, for a given query cost vector w, let w_i be the total cost of the ith row. Thus, the cheapest row has cost $\min(w_1,\ldots,w_m)$ and the *expected* cost of H is $\frac{1}{2}(w_1+\ldots+w_m)$. One simple observation that helps in the analysis is that if we define h_i as the cost of the items in H that are in the ith row, then $\Pr(h_i \geq w_i/2) \geq 1/2$. That is because if any particular subset of the ith row has cost less than $w_i/2$, its complement in the ith row must have cost greater than $w_i/2$. Furthermore, these events are independent over the different rows.

So, we can reduce the problem to the following: we have m independent events each of probability at least $1/2$. If at least two of these events occur, the elicitor gets no information (H is not the cheapest bundle because it is not cheaper than the cheapest row). Thus, the probability the elicitor *does* get some information is at most $(m+1)2^{-m}$ and the expected number of queries is at least $\frac{1}{2(m+1)}2^m$. ■

Open Problem 1 *Can preferences expressible as polynomial-size DNF formulas be elicited in $2^{O(\sqrt{n})}$ value queries or demand queries? (This is open even for the boolean preference case.)*

3.1 log(n)-DNF Preferences

In the previous problem, even though there were only $O(\sqrt{n})$ terms in each preference function, the terms themselves were fairly large. What if all of the terms are small, of size no more than $\log n$? Observe that there are $\binom{n}{\log n}$ possible terms of size $\log n$, so some members of this class cannot be represented in $\text{poly}(n)$ bits.

Theorem 3 *If f and g are DNF-preferences where all terms are of $O(\log n)$, then preference elicitation can be performed in a number of value queries polynomial in n.*

Proof We begin by giving a randomized construction and then show a derandomization.

For convenience let us put an empty term T_0 of value 0 into both f and g. With this convention we can assume the optimal allocation satisfies some term $T' \in f$ and some term $T'' \in g$.

We now simply notice that since T' and T'' are both of size $O(\log n)$, a random partition (S', S'') has probability at least $1/\text{poly}(n)$ of having the property that $S' \supseteq T'$ and $S'' \supseteq T''$. So, we simply need to try $\text{poly}(n, \log \frac{1}{\delta})$ random partitions and take the best one, and with probability at least $1-\delta$ we will have found the optimal allocation.

We can now derandomize this algorithm using the (n,k)-universal sets of Naor and Naor (1990). A set of assignments to n boolean variables is (n,k)-universal if for every subset of k variables, the induced assignments to those variables covers all 2^k possible settings. Naor and Naor (1990) give efficient explicit constructions of such sets using only $2^{O(k)} \log n$ assignments. In our case, we can use the case of $k = O(\log n)$, so the construction is polynomial time and size. Each of these assignments corresponds to a partition of the items, and we simply ask f and g for their valuations on each one and take the best. ■

4. General Relationships

In this section we describe a number of general relationships between query learning and preference elicitation. We will solely concern ourselves here with communication/query complexity issues, and not with the issue of computation time.

To begin with some simple relationships, it is clear that preference elicitation is no harder than exact learning, since one way to perform elicitation is to simply learn each bidder's preferences exactly. On the other hand, the results from the previous section show that this is not true for approximate learning. Occam's razor theorems (Blumer et al., 1987) imply that *any* function can be approximately learned with a number of queries polynomial in the description length of the function (ignoring issues of computation time), and thus Theorems 1 and 2 imply a super-polynomial gap between the number of value queries needed for preference elicitation and approximate learning for the case of monotone DNF formulas. In the other direction, we have seen several examples of cases where preference elicitation is *easy* and yet it is hard to perform exact learning (Theorem 3) or even approximate learning (example in Section 1.1) because the target function cannot be written down, even approximately, in a small number of bits.

These last examples are something of a "cheat" because the function cannot even be described compactly. We now describe a case in which preference elicitation is easy but exact learning is hard, even though the function has a small description. We then show that in certain circumstances, however, the ability to elicit does imply the ability to learn with queries.

4.1 Almost-Threshold Preferences

We now define a class of boolean preference functions that we call *almost-threshold*. This class will be used to show that, even if all of the functions in a class have representations of size polynomial in n, we can still separate exact learning and preference elicitation with respect to value queries. In fact, the gap is super-exponential.

An "almost threshold" preference function is defined by specifying a single set S'. This set in turn defines a preference function that is 1 for any set of size greater than or equal to $|S'|$, except for

S' itself, and is 0 otherwise. Formally, for any $S' \neq \emptyset$, define

$$h_{S'}(S'') = \begin{cases} 1 & \text{if } S'' \neq S' \text{ and } |S''| \geq |S'| \\ 0 & \text{otherwise} \end{cases}$$

The class H_{AT} of almost-threshold preference functions is then $H_{AT} = \{h_{S'}\}$.

Observation 3 *It requires at least* $\binom{n}{\lceil n/2 \rceil - 1}$ *value queries to exactly learn the class* H_{AT}.

Theorem 4 *If* $f, g \in H_{AT}$ *then the optimal allocation can be elicited in* $4 + \log_2 n$ *value queries.*

Proof Recall that we use S to represent the set of all items, and assume $|S| > 2$. Also suppose $f = h_{S'}$. The first step is to determine $|S'|$. We can do this in $\log_2 n + 1$ queries using binary search. We next find two sets T, T' of size $|S'|$ such that $f(T) = f(T') = 1$. This can be done by picking three arbitrary sets of size $|S'|$ and querying the first two: at most one of these two sets can have the value 0, and if one of the two sets does have this value then the third set must have the value 1. Our final query is for the value of $g(S - T)$. If this query returns 1, then $T, S - T$ is an optimal allocation (and has value 2). Otherwise, $T', S - T'$, regardless of its value, is an optimal allocation. Thus, we can find the optimal allocation in $4 + \log_2 n$ value queries, although we may need one more query if we wish to determine the value of this allocation. ■

4.2 When Easy Elicitation Implies Easy Learning

We now show that in certain circumstances, however, the ability to elicit with a polynomial number of value queries does imply the ability to exactly learn with a similar number of queries. In particular, we will show that if preference elicitation can be performed query-efficiently for preferences drawn from certain boolean classes then a superset query can be efficiently simulated using value queries.

Definition 5 *A* **superset oracle** *for a target* f^* *in a boolean concept class H takes a function* $f \in H$ *as input. If* f *is a superset of* f^**, that is,* $\{x : f(x) = 1\} \supseteq \{x : f^*(x) = 1\}$*, then the query returns "true". Otherwise the query produces a counterexample: an x such that* $f(x) = 0$ *but* $f^*(x) = 1$.

Recall that Angluin's algorithm (Angluin, 1988) for learning monotone DNF uses value (membership) and equivalence queries, but that the equivalence oracle is always queried with a hypothesis that is a subset of the target function (it is an improper subset, the target itself, on the final query). Therefore, the same algorithm can be used to learn monotone DNF from a value and superset oracle. In fact, it can be seen that any subclass of monotone DNF that is closed under removal of terms can be learned from value and superset queries by the same algorithm.

What makes this interesting is the following relationship between preference elicitation and superset queries. First, for any boolean function f, let us define its "dual"

$$\hat{f}(S') = 1 - f(S - S').$$

Or, in other words, $\hat{f}(x) = \overline{f}(\overline{x})$. Given a boolean hypothesis class H, define $\hat{H} = \{\hat{f} : f \in H\}$. For example, the dual of the class of monotone $\log(n)$-DNF formulas is the class of monotone $\log(n)$-CNF formulas. The set of monotone functions is closed under dual.

Theorem 6 *Let H be a boolean concept class with dual $\hat{H}$. If, given value oracles for any $f \in H$ and $g \in \hat{H}$, the optimal allocation S', S'' can be elicited using M value queries, then a superset query for a target $f^* \in H$ can be simulated using $M+2$ queries to a value oracle for f^*.*

Proof Suppose that one wants to perform a superset query with $g \in H$. First, compute $\hat{g} \in \hat{H}$. Then, perform preference elicitation on $f^*, \hat{g}$. If this procedure returns an allocation satisfying both agents, this means we have an x such that $f^*(x) = 1$ and $\hat{g}(\overline{x}) = 1$. But, $\hat{g}(\overline{x}) = \overline{g}(x)$ so this means that x is a counterexample to the superset query. On the other hand, if the elicitation procedure fails to satisfy both agents then no such x exists, so the superset query can return "true" in this case. ■

Corollary 7 *If H is a subclass of monotone DNF that is closed under removal of terms, and if preference elicitation for $(H, \hat{H})$ can be performed in M queries, then H is exactly learnable from a number of value queries that is polynomial in n, M, and the number of terms in the target monotone DNF.*

5. Boolean-Valued Versus Real-Valued Elicitation

It might seem that eliciting real-valued preference functions would generally be much more difficult than eliciting boolean preferences. In this section, we show that—under certain conditions—the number of value queries necessary for elicitation of real-valued preferences is not that much greater than the number of queries required for eliciting boolean preferences.

First of all, for any real-valued function class H we define a related boolean-valued class as follows. Let $\succ$ be a variable representing either the $>$ or $\geq$ relational operator and let $\mathbf{P}(S)$ represent the power set of S. Given a function $f : \mathbf{P}(S) \rightarrow \mathbf{R}^+$ and an $a \in \mathbf{R}$, define $thresh^{\succ}_{f,a} : \mathbf{P}(S) \rightarrow \{0,1\}$ to be a function such that

$$thresh^{\succ}_{f,a}(S') = \begin{cases} 1 & \text{if } f(S') \succ a \\ 0 & \text{otherwise.} \end{cases}$$

A function f is said to project onto a set of boolean-valued functions H' if for all $a \in \mathbf{R}$, $thresh^{\geq}_{f,a}, thresh^{>}_{f,a} \in H'$. A set of functions H is said to project onto a set H' if each $f \in H$ projects onto H'. The **boolean projection** of H is the smallest set of functions that H projects onto.

Now, imagine that f and g, the preferences for two agents, are drawn from H. This problem is closely related to the case where preferences f' and g' are drawn from H', the boolean projection of H. By considering the ranges of the functions in H, it is sometimes clear that the above two problems are equally hard. We begin with the observations that when the functions in H all share a small known domain, it is easy to see that these problems are of similar difficulty. We will show in the first theorem below that if the ranges of the functions in H are small sets, then these two problems are still of similar difficulty. On the other hand, in the second theorem of this section, we will show that there exists a real-valued H with a corresponding boolean projection H' such that exponentially more value queries are required to perform preference elicitation on H than are required to exactly learn H' from value queries.

5.1 Single Small Co-Domain

First of all, suppose that H is such that there exists a small set R_H which is a co-domain for every function $f \in H$. That is, for any $f \in H$, for every $S' \subseteq S$, $f(S') \in R_H$. Observe that if the elicitor

knows H, then it knows R_H. In this scenario, it is easy to prove that the elicitation of an optimal allocation for $f,g \in H$ and the elicitation of an optimal allocation for $f',g' \in H'$, where H' is the boolean projection of H, can be performed using similar numbers of value queries.

Before we describe the proof, we define a **threshold query**. Let both $\succ$ and $\succ'$ be variables taking on values in $\{>,\geq\}$. Then a $(a,b,\succ\succ')$ threshold query is defined as follows: Does there exist a set $S' \subseteq S$ such that $f(S') \succ a$ and $g(S-S') \succ' b$? If so, return $(f(S'),g(S-S'),S')$, otherwise return false.

Observation 4 *Suppose class H has boolean projection H' such that the optimal allocation for any $f',g' \in H'$ can be elicited in k value queries. Then a threshold query on any $f,g \in H$ can be performed using at most $k+2$ value queries.*

Proof Suppose that we are attempting to perform an $(a,b,>\geq)$ threshold query on f and g, and let $f' = thresh^{>}_{f,a}$ and $g' = thresh^{\geq}_{g,b}$. Then the threshold query should return false if and only if no allocation can satisfy both f' and g'. So, we simply perform elicitation on f' and g', using value queries to f and g to simulate value queries to f' and g' respectively, and see if both can be satisfied. If so, we perform two more queries (one each to f and g) to determine the actual value of this allocation. ■

Observation 5 *Suppose class H has boolean projection H', and suppose that R_H is a co-domain for every function $f \in H$. If $|R_H| = m$, and the optimal allocation for two preferences $f',g' \in H'$ can be elicited in k value queries, then the optimal allocation for any two preferences $f,g \in H$ can be elicited in $(k+2)m^2$ value queries.*

Proof

The algorithm to elicit the optimal allocation for $f,g \in H$ is as follows. For all $(a,b) \in R_H^2$, perform an $(a,b,\geq\geq)$ threshold query, and let X be the set of all non-false responses (a,b,S') returned by these queries. Return the allocation $(S'',S-S'')$ from the triple $(a'',b'',S'') \in X$ that maximizes $a''+b''$.

By the previous observation, all of these threshold queries can be simulated using at most $(k+2)m^2$ value queries. To see that this algorithm returns an optimal allocation, let $(S^*,S-S^*)$ be a fixed optimal allocation, and define $f^* = f(S^*)$ and $g^* = g(S-S^*)$. Observe that $(f^*,g^*) \in R_H^2$, so the algorithm will make the threshold query $(f^*,g^*,\geq,\geq)$. Furthermore, this query will return some set S'' such that $f(S'') = f^*$ and $g(S-S'') = g^*$, since such an S'' exists and since by the optimality of S^* any response (a,b,S'') to this query must have $a \leq f^*$ (since $b \geq g^*$) and $b \leq g^*$ (since $a \geq f^*$). Thus X contains at least one optimal allocation. ■

5.2 Many Small Ranges

We again consider a set of real valued preference functions H. However, we will not assume a finite co-domain shared by all of the functions $f \in H$. Instead, we will assume that there exists an m such that for every $f \in H$, the size of the range[5] of f is bounded by m.

5. The size of the range of a function is the number of elements in the range. Technically, for a function $f : X \to Y$, the size of the range is $|\{y \in Y : \exists x \in X \text{ such that } f(x) = y\}|$.

Such problems are not as conducive to the easy analysis of the earlier section. For instance, given a function $f \in H$, we may not be able to discover its entire range without a number of queries exponential in the number of items. However:

Theorem 8 *Given an integer m and a set H such that each function $f \in H$ has a range of size less than m, if H' is the boolean projection of H, and the optimal allocation for $f', g' \in H'$ can be elicited using k value queries, then the optimal allocation for $f, g \in H$ can be elicited using $2 + 4(k+2)m^2$ value queries.*

Proof

Let $(S^*, S - S^*)$ be any fixed optimal allocation, $f^* = f(S^*)$, and $g^* = g(S - S^*)$. We give an algorithm that will iteratively construct subsets of the ranges of f and g such that, when the construction is complete, the final subsets will contain f^* and g^*, respectively. By the analysis of Observation 5, if the algorithm of that observation is run using the cross product of these two subsets in place of R_H^2, the algorithm will still successfully locate an optimal allocation.

The full algorithm is as follows:

1. Initialize $R_f = \{f(\emptyset)\}$ and $R_g = \{g(\emptyset)\}$.
2. For all $(a,b) \in R_f \times R_g$, if one has not already done so, perform three threshold queries $(a, b, >>)$, $(a, b, >\geq)$, and $(a, b, \geq>)$.
3. For every triple (a', b', S') returned in the previous step, add a' to R_f and b' to R_g.
4. If R_f and R_g increased in size on the previous step, return to step 2.
5. If R_f and R_g did not increase in size, run the algorithm of Observation 5 using $R_f \times R_g$ in place of R_H^2.

Let the final values of R_f and R_g be denoted by R_f^* and R_g^*, respectively. Observe that R_f^* is a subset of the range of f, because only values of f are inserted into it. Similarly, R_g^* is a subset of the range of g. Thus $|R_f^* \times R_g^*| \leq m^2$. Observe that 2 value queries are made in step 1, $3|R_f^* \times R_g^*|$ threshold queries are made in step 2 (which, by Observation 4, can be simulated by at most $3(k+2)m^2$ value queries), and, by Observation 5 and the earlier analysis, no more than $(k+2)m^2$ value queries are made in step 5. Thus, no more than $2 + 4(k+2)m^2$ value queries are made.

Now, consider the sets $R_f \subseteq R_f^*$ and $R_g \subseteq R_g^*$ just before the ith execution of step 2 of the algorithm. We will show below that if at least one of f^* and g^* is not contained in R_f and R_g, respectively, then at least one of these sets will increase in size when this step is executed. Thus, the algorithm will continue iterating this step until $f^* \in R_f$ and $g^* \in R_g$.

First, assume that both $f^* \notin R_f$ and $g^* \notin R_g$, and let $f^<$ ($g^<$) be the largest value in R_f (R_g) that is less than f^* (g^*). Note that the value $f^<$ exists[6] because $f(\emptyset) \in R_f$ and for all S', $f(\emptyset) \leq f(S')$ by the monotonicity of preference functions. Similarly, $g^<$ exists. Therefore, at some iteration of the algorithm the threshold query $(f^<, g^<, >>)$ will be made, and when it is made it will not return "false" because (f^*, g^*, S^*) is a valid response. Furthermore, any response (a, b, S') must either have $a \leq f^*$ or $b \leq g^*$, since $a + b \leq f^* + g^*$. But, by the definition of $f^<$ and $g^<$ as well as of the

6. If $f(\emptyset) = f^*$, then $f^* \in R_f$ at all stages.

threshold query, this means that at least one of a or b is a value that is not contained in R_f or R_g. But both a and b will be contained in their respective R^* sets. Therefore, at least one of the conditions $R_f \neq R_f^*$ and $R_g \neq R_g^*$ holds, so at least one of the sets R_f and R_g must grow during execution i of step 2.

Next, consider the case when $f^* \in R_f$ and $g^* \notin R_g$ (the remaining case $f^* \notin R_f$ and $g^* \in R_g$ is symmetric). Define $g^<$ as above and consider the threshold query $(f^*, g^<, \geq >)$. Reasoning as above shows that this query will produce a response (a, b, S') such that $b \notin R_g$. Thus, in all cases when at least one of f^* or g^* is not in its respective set, one of the sets grows at step 2.

■

5.3 When Real Values Make a Problem More Difficult

Theorem 9 *There exists a class of real-valued functions that requires* $2^n - 1$ *value queries to elicit while its boolean projection requires at most* $n+1$ *value queries to exactly learn.*

Proof

Imagine that the items are "more or less" unrelated. In particular, each item has a basic value in $\{1, 2, 4, 8, \ldots, 2^{n-1}\}$. For all $a \in S$, define $V(a)$ to be the basic value of a, and assume that this mapping is known. For all $S' \subseteq S$, define $V(S') = \sum_{a \in S'} V(a)$. Thus, the basic value of any set is the sum of the basic values in that set. Observe that for any $S' \subseteq S$, $V(S') + V(S - S') = 2^n - 1$.

Now, each agent has a special set that they value slightly more than other agents do. Thus, for all $S', S'' \subseteq S$, define $f_{S'}(S'') = V(S'')$ if $S' \neq S''$, and $f_{S'}(S') = V(S') + \frac{1}{2}$. The class of preferences of interest is therefore $\{f_{S'}\}_{S' \subseteq S}$.

In order to determine the optimal allocation when preferences are drawn from this class, the elicitor must find the special set for one agent. And in the worst case it requires $2^n - 1$ value queries in order to find a special set, since the only information obtained from a value query on a non-special set is that it is not special. Therefore, $2^n - 1$ value queries are required to elicit this preference class.

Now, consider the boolean projection of this class. For all $a \in R$, for all $S' \subseteq S$, define $g_a(S')$ to be true if and only if $V(S') \geq a$. Then the projection can be represented as $\{g_a\}_{a \in \{0, \ldots, 2^n\}}$. Now, using value queries, we can perform a binary search for the value of a defining a target member of this class. Thus, we can exactly learn a target g in the boolean projection with at most $n+1$ value queries. ■

One point to observe is that it is easy to approximate any function with an exponential number of values with a function with a polynomial number of values. However, one must be careful when one computes an allocation that is only approximately optimal, because the traditional techniques to motivate the agents to answer truthfully (which we describe in the next section) will no longer work.

6. Truthfulness and VCG

In combinatorial auctions and mechanism design, one key issue that arises is that bidders have their own interests: they each want to receive as valuable a bundle as possible, and therefore may lie in their responses if they perceive it to be to their advantage. For example, if the auctioneer is

not going to actually charge the bidders anything for the bundles they get, then bidders have an incentive to report overly high valuations, in order to make the auctioneer think that social welfare will be improved by giving more to them. On the other hand, if the bidders are charged exactly the valuations that they report, then they have an incentive to underbid, in the hope of making a profit (paying less for a bundle than it is actually worth to them).[7] In preference elicitation, the issue of motivating the bidders to answer queries truthfully is exacerbated by the fact that the elicitor's queries leak information to the bidder about the answers that other bidders have given.

Recently, a methodology was proposed by which elicitors can be made incentive compatible in the sense that every bidder answering the queries truthfully is an *ex post equilibrium* (Conen and Sandholm, 2001).[8] This is accomplished by organizing the mechanism so that if all the bidders answer truthfully, the final allocation and payments follow the *Vickrey-Clarke-Groves scheme (VCG)* (Vickrey, 1961; Clarke, 1971; Groves, 1973). In this scheme (the Clarke version), the amount bidder i has to pay is the sum of others' revealed valuations for the bundles they get had bidder i not participated, minus the sum of others' revealed valuations for the bundles they get in the actual optimal allocation. The elicitor can determine these payments by asking enough queries to be able to determine the welfare maximizing allocation overall, and *by asking extra queries to determine the welfare maximizing allocation for the auctions where each agent is ignored in turn*. The essence of the argument is that the auction in which agent i is removed serves only to determine agent i's payment, and therefore in this auction there is no motivation for any of the participating agents to lie. This then means that the payments given to the bidders can be assumed to be the correct VCG payments, which then implies by standard VCG arguments that the optimal strategy for the bidders in the first auction is to tell the truth as well. Conceptually, one could think of $k+1$ "elicitors", each working to solve one of these problems. Once all of the "elicitors" have found their welfare maximizing allocations respectively, the process can terminate. Note that the extra overhead of motivating the bidders to bid truthfully is just solving k additional elicitation problems beyond the original elicitation problem. Therefore, if elicitation can be done in a polynomial number of queries, then so can elicitation that motivates the bidders to answer the queries truthfully.

7. Subsequent Work

Since the conference (COLT-03) version of this paper appeared, a significant amount of closely related work has been done. In this section we summarize that work.

7. Throughout this paper we have let each bidder i have some valuation function f_i from bundles of items to reals. This notation implicitly makes the following common economic assumptions: (1) *private values*: bidder i knows f_i (in other words, the *function* f_i does not depend on the other bidders in any way); and (2) *no externalities*: bidder i does not care who gets the items that i does not get. For the truthfulness discussion we additionally make the common economic assumption (3) *quasilinear preferences*: the utility that bidder i tries to maximize is $u_i(S_i, p_i) = f_i(S_i) - p_i$, where S_i is the set of items that i gets and p_i is the total price that i has to pay.
8. This means that bidding truthfully is each bidder's best strategy (for any prior probability distribution that he may hold about the other bidders) given that the other bidders bid truthfully. In other words, truthful bidding strategies form a Nash equilibrium even in hindsight. This does not mean that bidding truthfully is a dominant strategy: if others bid insincerely, one may also do better by bidding insincerely. For example, in a 2-bidder setting, if bidder 1's strategy involves dropping out (bidding zero from then on) whenever it receives a particular query stream, then it can be bidder 2's best strategy to answer queries in a way that causes the elicitor to submit that query stream to bidder 1. In summary, implementation in *ex post* equilibrium is stronger than implementation in Nash equilibrium, but weaker than implementation in dominant strategies.

One of the oldest techniques for preference elicitation is an ascending auction. An ascending auction can be considered to be a sequence of increasing demand queries, where if one asks a query w' after a query w, then it must be the case that for all i, $w'_i \geq w_i$. In the conference (COLT-03) version of this paper we presented the following problem as an interesting open question:

Open Problem 2 *Does there exist a preference elicitation problem that is hard (or impossible) to elicit using an ascending auction but easy to elicit using demand queries?*

Since then, this question has been answered, and the answer is affirmative. Nisan (2003) presents a 2-item auction where no ascending item-price auction can determine the optimal allocation (using *any* number of queries), but the optimal allocation can easily be determined using (nonascending) item-price demand queries.

On the other hand, if *bundle-price* demand queries are allowed — i.e., prices are not assigned to items only, but potentially also to bundles — then ascending auctions exist that always determine the optimal allocation (using potentially an exponential number of queries), at least if each bidder is assumed to act truthfully (Parkes and Ungar, 2002; Ausubel and Milgrom, 2002). As pointed out by Nisan (2003), it remains an open question whether there exists an ascending bundle-price auction that always determines the optimal allocation if the auction is restricted to being *anonymous*, that is, at any time, the price of a bundle is the same for each agent. Bundle-price demand queries are quite powerful: as mentioned in footnote 3, one can use them to efficiently elicit monotone DNF formulas, and this fact as well as other results on these queries are given by Lahaie and Parkes (2004).

As to preference elicitation using value queries only, new valuation classes learnable in a polynomial number of queries have been introduced in Conitzer et al. (2003) and Santi et al. (2004). These include valuations where items have at most k-wise dependencies, and certain other valuations. Furthermore, if two classes of valuations are each learnable in a polynomial number of queries, then so is their union—even though the elicitor does not know in advance in which of the two classes (or both) the bidder's valuation belongs. Santi et al. (2004) also present severely restricted valuation classes where learning nevertheless requires an exponential number of value queries. First steps toward a characterization of polynomial learnability of valuation functions are also given.

8. Conclusions and Open Problems

In machine learning, one's objective is nearly always to learn or approximately learn some target function. In this paper, we relate this to the notion of preference elicitation, in which the goal instead is to find the optimal partitioning of some set of items among the bidders. In the case of two bidders, preference elicitation can be thought of as a learning problem with *two* target functions f and g, where the goal is rather than necessarily learning f and g to instead find the example x that maximizes $f(x)+g(\overline{x})$.

We now describe several open problems left by this work. We begin with a problem stated in Section 3.

Open Problem 2 *Can preferences expressible as polynomial-size DNF formulas be elicited in* $2^{O(\sqrt{n})}$ *value queries or demand queries?*

A somewhat fuzzier question related to our results on $\log(n)$-DNF is the following. Our algorithm in this case was non-adaptive: the questions asked did not depend on answers to previous questions. It seems natural that for some classes adaptivity should help. In fact, it is not hard to generate artificial examples in which this is the case. However, we know of no natural example having this property.

Open Problem 3 *Are there natural classes of functions for which exact learning is information-theoretically hard, preference elicitation via a non-adaptive algorithm is hard (i.e., an algorithm in which the questions can all be determined in advance) but elicitation by an adaptive algorithm is easy.*

Acknowledgments

This material is based upon work supported under NSF grants CCR-0105488, ITR CCR-0122581, CCR-0209064, ITR IIS-0081246, and ITR IIS-0121678. Any opinion, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. Tuomas Sandholm is also supported by a Sloan Fellowship. We would like to thank the referees for their helpful comments.

References

Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, pages 39–46, Boston, MA, 2000.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

Dana Angluin, Lisa Hellerstein, and Marek Karpinski. Learning read-once formulas with queries. In *Journal of the ACM*, volume 40, pages 185–210, 1993.

Lawrence M. Ausubel and Paul Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1, 2002. No. 1, Article 1.

Sushil Bikhchandani, Sven de Vries, James Schummer, and Rakesh V. Vohra. Linear programming and Vickrey auctions, 2001. Draft.

Sushil Bikhchandani and Joseph M. Ostroy. The package assignment model. UCLA Working Paper Series, mimeo, 2001.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions: Extended abstract. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 256–259, Tampa, FL, October 2001. A more detailed description of the algorithmic aspects appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.

Wolfram Conen and Tuomas Sandholm. Differential-revelation VCG mechanisms for combinatorial auctions. In *AAMAS-02 workshop on Agent-Mediated Electronic Commerce (AMEC)*, Bologna, Italy, 2002a. Springer Lecture Notes in Computer Science LNCS 2531.

Wolfram Conen and Tuomas Sandholm. Partial-revelation VCG mechanism for combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 367–372, Edmonton, Canada, 2002b.

Vincent Conitzer, Tuomas Sandholm, and Paolo Santi. Combinatorial auctions with *k*-wise dependent valuations, October 2003. Draft.

Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, August 1999.

Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

Holger Hoos and Craig Boutilier. Bidding languages for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217, Seattle, WA, 2001.

Benoit Hudson and Tuomas Sandholm. Effectiveness of query types and policies for preference elicitation in combinatorial auctions. In *International Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, USA, 2004.

Sebastién Lahaie and David Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, New York, NY, 2004.

Daniel Lehmann, Lidian Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, Baltimore, 1990.

Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 1–12, Minneapolis, MN, 2000.

Noam Nisan. The power and limitations of item price combinatorial auctions, 2003. Slides from the FCC Combinatorial Bidding Conference, Queenstown, MD, Nov. 21–23.

Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 242–252, Minneapolis, MN, 2000.

Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting Lindahl prices, 2003. Working Paper (version: March 2003).

David C. Parkes. iBundle: An efficient ascending price bundle auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 148–157, Denver, CO, November 1999a.

David C. Parkes. Optimal auction design for agents with hard valuation problems. In *Agent-Mediated Electronic Commerce Workshop at the International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999b.

David C. Parkes and Lyle Ungar. An ascending-price generalized Vickrey auction, 2002. Draft, Jun.

Michael H. Rothkopf, Aleksandar Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, January 2002a.

Tuomas Sandholm. eMediator: A next generation electronic commerce server. *Computational Intelligence*, 18(4):656–676, 2002b.

Tuomas Sandholm and Subhash Suri. Side constraints and non-price attributes in markets. In *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, pages 55–61, Seattle, WA, 2001.

Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1102–1108, Seattle, WA, 2001.

Paolo Santi, Vincent Conitzer, and Tuomas Sandholm. Towards a characterization of polynomial preference elicitation with value queries in combinatorial auctions. In *Conference on Learning Theory (COLT)*, Banff, Alberta, Canada, 2004.

Trey Smith, Tuomas Sandholm, and Reid Simmons. Constructing and clearing combinatorial exchanges using preference elicitation. In *AAAI-02 workshop on Preferences in AI and CP: Symbolic Approaches*, pages 87–93, 2002.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

Peter R. Wurman and Michael P. Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 21–29, Minneapolis, MN, October 2000.

Martin Zinkevich, Avrim Blum, and Tuomas Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 176–185, San Diego, CA, 2003.

Journal of Machine Learning Research 5 (2004) 669–695 Submitted 1/04; Published 6/04

Distance-Based Classification with Lipschitz Functions

Ulrike von Luxburg ULRIKE.LUXBURG@TUEBINGEN.MPG.DE
Olivier Bousquet OLIVIER.BOUSQUET@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstrasse 38
72076 Tübingen, Germany

Editors: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

The goal of this article is to develop a framework for large margin classification in metric spaces. We want to find a generalization of linear decision functions for metric spaces and define a corresponding notion of margin such that the decision function separates the training points with a large margin. It will turn out that using Lipschitz functions as decision functions, the inverse of the Lipschitz constant can be interpreted as the size of a margin. In order to construct a clean mathematical setup we isometrically embed the given metric space into a Banach space and the space of Lipschitz functions into its dual space. To analyze the resulting algorithm, we prove several representer theorems. They state that there always exist solutions of the Lipschitz classifier which can be expressed in terms of distance functions to training points. We provide generalization bounds for Lipschitz classifiers in terms of the Rademacher complexities of some Lipschitz function classes. The generality of our approach can be seen from the fact that several well-known algorithms are special cases of the Lipschitz classifier, among them the support vector machine, the linear programming machine, and the 1-nearest neighbor classifier.

1. Introduction

Support vector machines (SVMs) construct linear decision boundaries in Hilbert spaces such that the training points are separated with a large margin. The goal of this article is to extend this approach from Hilbert spaces to metric spaces: we want to find a generalization of linear decision functions for metric spaces and define a corresponding notion of margin such that the decision function separates the training points with a large margin. The reason why we are interested in metric spaces is that in many applications it is easier or more natural to construct distance functions between objects in the data space than positive definite kernel functions as they are used for support vector machines. Examples for this situation are the edit distance used to compare strings or graphs and the earth mover's distance on images.

SVMs can be seen from two different points of view. In the regularization interpretation, for a given positive definite kernel k, the SVM chooses a decision function of the form $f(x) = \sum_i \alpha_i k(x_i, x) + b$ which has a low empirical error R_{emp} and is as smooth as possible. According to the large margin point of view, SVMs construct a linear decision boundary in a Hilbert space $\mathcal{H}$ such that the training points are separated with a large margin and the sum of the margin errors is small. Both viewpoints can be connected by embedding the sample space $\mathcal{X}$ into the reproducing kernel Hilbert space $\mathcal{H}$ via the so called "feature map" and the function space $\mathcal{F}$ into the dual $\mathcal{H}'$. Then the regularizer (which is a functional on $\mathcal{F}$) corresponds to the inverse margin (which is a

norm of a linear operator), and the empirical error corresponds to the margin error (cf. Sections 4.3 and 7 of Schölkopf and Smola, 2002). The benefits of these two dual viewpoints are that the regularization framework gives some intuition about the geometrical meaning of the norm on $\mathcal{H}$, and the large margin framework leads to statistical learning theory bounds on the generalization error of the classifier.

Now consider the situation where the sample space is a metric space $(\mathcal{X}, d)$. From the regularization point of view, a convenient set of functions on a metric space is the set of Lipschitz functions, as functions with a small Lipschitz constant have low variation. Thus it seems desirable to separate the different classes by a decision function which has a small Lipschitz constant. In this article we want to construct the dual point of view to this approach. To this end, we embed the metric space $(\mathcal{X}, d)$ in a Banach space $\mathcal{B}$ and the space of Lipschitz functions into its dual space $\mathcal{B}'$. Remarkably, both embeddings can be realized as isometries simultaneously. By this construction, each $x \in \mathcal{X}$ will correspond to some $m_x \in \mathcal{B}$ and each Lipschitz function f on $\mathcal{X}$ to some functional $T_f \in \mathcal{B}'$ such that $f(x) = T_f m_x$ and the Lipschitz constant $L(f)$ is equal to the operator norm $\|T_f\|$. In the Banach space $\mathcal{B}$ we can then construct a large margin classifier such that the size of the margin will be given by the inverse of the operator norm of the decision functional. The basic algorithm implementing this approach is

$$\text{minimize } \mathrm{R}_{\mathrm{emp}}(f) + \lambda L(f)$$

in regularization language and

$$\text{minimize } L(f) + C\sum_i \xi_i \text{ subject to } y_i f(x_i) \geq 1 - \xi_i,\ \xi_i \geq 0$$

in large margin language. In both cases, $L(f)$ denotes the Lipschitz constant of the function f, and the minimum is taken over a subset of Lipschitz functions on $\mathcal{X}$. To apply this algorithm in practice, the choice of this subset will be important. We will see that by choosing different subsets we can recover the SVM (in cases where the metric on $\mathcal{X}$ is induced by a kernel), the linear programming machine (cf. Graepel et al., 1999), and even the 1-nearest neighbor classifier. In particular this shows that all these algorithms are large margin algorithms. So the Lipschitz framework can help to analyze a wide range of algorithms which do not seem to be connected at the first glance.

This paper is organized as follows: in Section 2 we provide the necessary functional analytic background for the Lipschitz algorithm, which is then derived in Section 3. We investigate representer theorems for this algorithm in Section 4. It will turn out that the algorithm always has a solution which can be expressed by distance functions to training points. In Section 5 we compute error bounds for the Lipschitz classifier in terms of Rademacher complexities. In particular, this gives valuable information about how fast the algorithm converges for different choices of subsets of Lipschitz functions. The geometrical interpretation for choosing different subsets of Lipschitz functions is further discussed in Section 6.

2. Lipschitz Function Spaces

In this section we introduce several Lipschitz function spaces and their properties. For a comprehensive overview we refer to Weaver (1999).

A metric space $(\mathcal{X}, d)$ is a set $\mathcal{X}$ together with a metric d, that is a non-negative, symmetric function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which fulfills $d(x,y) = 0 \Leftrightarrow x = y$ and the triangle inequality $d(x,y) +$

$d(y,z) \leq d(x,z)$. A function $f : \mathcal{X} \to \mathbb{R}$ on a metric space $(\mathcal{X},d)$ is called a Lipschitz function if there exists a constant L such that $|f(x) - f(y)| \leq Ld(x,y)$ for all $x,y \in \mathcal{X}$. The smallest constant L such that this inequality holds is called the Lipschitz constant of f, denoted by $L(f)$. For convenience, we recall some standard facts about Lipschitz functions:

Lemma 1 (Lipschitz functions) *Let $(\mathcal{X},d)$ be a metric space, $f,g : \mathcal{X} \to \mathbb{R}$ Lipschitz functions and $a \in \mathbb{R}$. Then $L(f+g) \leq L(f)+L(g)$, $L(af) \leq |a|L(f)$ and $L(\min(f,g)) \leq \max\{L(f),L(g)\}$, where $\min(f,g)$ denotes the pointwise minimum of the functions f and g. Moreover, let $f := \lim_{n\to\infty} f_n$ the pointwise limit of Lipschitz functions f_n with $L(f_n) \leq c$ for all $n \in \mathbb{N}$. Then f is a Lipschitz function with $L(f) \leq c$.*

For a metric space $(\mathcal{X},d)$ consider the set

$$\mathrm{Lip}(\mathcal{X}) := \{f : \mathcal{X} \to \mathbb{R};\ f \text{ is a bounded Lipschitz function}\}.$$

It forms a vector space, and the Lipschitz constant $L(f)$ is a seminorm on this space. To define a convenient norm on this space we restrict ourselves to *bounded* metric spaces. These are spaces which have a finite diameter $\mathrm{diam}(\mathcal{X}) := \sup_{x,y\in\mathcal{X}} d(x,y)$. For the learning framework this is not a big drawback as the training and test data can always be assumed to come from a bounded region of the underlying space. For a bounded metric space $\mathcal{X}$ we choose the norm

$$\|f\|_L := \max\left\{L(f), \frac{\|f\|_\infty}{\mathrm{diam}(\mathcal{X})}\right\}$$

as our default norm on the space $\mathrm{Lip}(\mathcal{X})$. It is easy to see that this indeed is a norm. Note that in the mathematical literature, $\mathrm{Lip}(\mathcal{X})$ is usually endowed with the slightly different norm $\|f\| := \max\{L(f),\|f\|_\infty\}$. But we will see that the norm $\|\cdot\|_L$ fits very naturally in our classification setting, as already can be seen by the following intuitive argument. Functions that are used as classifiers are supposed to take positive and negative values on the respective classes and satisfy

$$\|f\|_\infty = \sup_x |f(x)| \leq \sup_{x,y} |f(x) - f(y)| \leq \mathrm{diam}(\mathcal{X})L(f), \tag{1}$$

that is $\|f\|_L = L(f)$. Hence, the L-norm of a classification decision function is determined by the quantity $L(f)$ we use as regularizer later on. Some more technical reasons for the choice of $\|\cdot\|_L$ will become clear later.

Another important space of Lipschitz functions is constructed as follows. Let $(\mathcal{X}_0,d)$ be a metric space with a distinguished "base point" e which is fixed in advance. $(\mathcal{X}_0,d,e)$ is called a *pointed metric space*. We define

$$\mathrm{Lip}_0(\mathcal{X}_0) := \{f \in \mathrm{Lip}(\mathcal{X}_0);\ f(e) = 0\}.$$

On this space, the Lipschitz constant $L(\cdot)$ is a norm. However, its disadvantage in the learning framework is the condition $f(e) = 0$, which is an inconvenient a priori restriction on our classifier as e has to be chosen in advance. To overcome this restriction, for a given bounded metric space $(\mathcal{X},d)$ we define a corresponding extended pointed metric space $\mathcal{X}_0 := \mathcal{X} \cup \{e\}$ for a new base element e with the metric

$$d_{\mathcal{X}_0}(x,y) = \begin{cases} d(x,y) & \text{for } x,y \in \mathcal{X} \\ \mathrm{diam}(\mathcal{X}) & \text{for } x \in \mathcal{X}, y = e. \end{cases} \tag{2}$$

Note that $\mathrm{diam}(\mathcal{X}_0) = \mathrm{diam}(\mathcal{X})$. Then we define the map

$$\psi : \mathrm{Lip}(\mathcal{X}) \to \mathrm{Lip}_0(\mathcal{X}_0), \quad \psi(f)(x) = \begin{cases} f(x) & \text{if } x \in \mathcal{X} \\ 0 & \text{if } x = e. \end{cases} \tag{3}$$

Lemma 2 (Isometry between Lipschitz function spaces) *ψ is an isometric isomorphism between* $\mathrm{Lip}(\mathcal{X})$ *and* $\mathrm{Lip}_0(\mathcal{X}_0)$.

Proof Obviously, ψ is bijective and linear. Moreover, for $f_0 := \psi(f)$ we have

$$\begin{aligned} L(f_0) &= \sup_{x,y \in \mathcal{X}_0} \frac{|f_0(x) - f_0(y)|}{d_{\mathcal{X}_0}(x,y)} = \max\{ \sup_{x,y \in \mathcal{X}} \frac{|f(x) - f(y)|}{d(x,y)}, \sup_{x \in \mathcal{X}} \frac{|f(x) - f(e)|}{d_{\mathcal{X}_0}(x,e)} \} = \\ &= \max\{L(f), \frac{\|f\|_\infty}{\mathrm{diam}(\mathcal{X})}\} = \|f\|_L. \end{aligned}$$

Hence, ψ is an isometry. ■

In some respects, the space $(\mathrm{Lip}_0(\mathcal{X}_0), L(\cdot))$ is more convenient to work with than $(\mathrm{Lip}(\mathcal{X}), \|\cdot\|_L)$. In particular it has some very useful duality properties. Let $(\mathcal{X}_0, d, e)$ be a pointed metric space with some distinguished base element e. A *molecule* of $\mathcal{X}_0$ is a function $m : \mathcal{X}_0 \to \mathbb{R}$ such that its support (i.e., the set where m has non-zero values) is a finite set and $\sum_{x \in \mathcal{X}_0} m(x) = 0$. For $x, y \in \mathcal{X}_0$ we define the *basic molecules* $m_{xy} := \mathbb{1}_x - \mathbb{1}_y$. It is easy to see that every molecule m can be written as a (non unique) finite linear combination of basic molecules. Thus we can define

$$\|m\|_{AE} := \inf \left\{ \sum_i |a_i| d(x_i, y_i);\ m = \sum_i a_i m_{x_i y_i} \right\}$$

which is a norm on the space of molecules. The completion of the space of molecules with respect to $\|\cdot\|_{AE}$ is called the Arens-Eells space $AE(\mathcal{X}_0)$. Denoting its dual space (i.e., the space of all continuous linear forms on $AE(\mathcal{X}_0)$) by $AE(\mathcal{X}_0)'$ the following theorem holds true (cf. Arens and Eells, 1956; Weaver, 1999).

Theorem 3 (Isometry between $AE(\mathcal{X}_0)'$ and $\mathrm{Lip}_0(\mathcal{X}_0)$) *$AE(\mathcal{X}_0)'$ is isometrically isomorphic to* $\mathrm{Lip}_0(\mathcal{X}_0)$.

This means that we can regard a Lipschitz function f on $\mathcal{X}_0$ as a linear functional T_f on the space of molecules, and the Lipschitz constant $L(f)$ coincides with the operator norm of the corresponding functional T_f. For a molecule m and a Lipschitz function f this duality can be expressed as

$$\langle f, m \rangle = \sum_{x \in \mathcal{X}_0} m(x) f(x). \tag{4}$$

It can be proved that $\|m_{xy}\|_{AE} = d(x,y)$ holds for all basic molecules m_{xy}. Hence, it is possible to embed $\mathcal{X}_0$ isometrically in $AE(\mathcal{X}_0)$ via

$$\Gamma : \mathcal{X}_0 \to AE(\mathcal{X}_0),\ x \mapsto m_{xe}. \tag{5}$$

The norm $\|\cdot\|_{AE}$ has a nice geometrical interpretation in terms of the *mass transportation problem* (cf. Weaver, 1999): some product is manufactured in varying amounts at several factories and has to be distributed to several shops. The (discrete) transportation problem is to find an optimal way to transport the product from the factories to the shops. The costs of such a transport are defined as $\sum_{ij} a_{ij} d_{ij}$ where a_{ij} denotes the amount of the product transported from factory i to shop j and d_{ij} the distance between them. If f_i denotes the amount produced in factory i and s_i denotes the amount needed in shop i, the formal definition of the transportation problem is

$$\min_{i,j=1,\ldots,n} \sum a_{ij} d_{ij} \quad \text{subject to} \quad a_{ij} \geq 0, \sum_j a_{ij} = s_j, \sum_i a_{ij} = f_i. \tag{6}$$

To connect the Arens-Eells space to this problem we identify the locations of the factories and shops with a molecule m. The points x with $m(x) > 0$ represent the factories, the ones with $m(x) < 0$ the shops. It can be proved that $\|m\|_{AE}$ equals the minimal transportation costs for molecule m. A special case is when the given molecule has the form $m_0 = \sum m_{x_i y_j}$. In this case, the transportation problem reduces to the *bipartite minimal matching problem*: given $2m$ points $(x_1, \ldots, x_n, y_1, \ldots, y_n)$ in a metric space, we want to match each of the x-points to one of the y-points such that the sum of the distances between the matched pairs is minimal. The formal statement of this problem is

$$\min_{\pi} \sum_{i,j} d(x_i, y_{\pi(i)}) \tag{7}$$

where the minimum is taken over all permutations π of the set $\{1, ..., n\}$ (cf. Steele, 1997).

In Section 4 we will also need the notion of a vector lattice. A vector lattice is a vector space V with an ordering $\preceq$ which respects the vector space structure (i.e., for $x, y, z \in V, a > 0$: $x \preceq y \Longrightarrow x + z \preceq y + z$ and $ax \preceq ay$) and such that for any two elements $f, g \in V$ there exists a greatest lower bound $\inf(f, g)$. In particular, the space of Lipschitz functions with the ordering $f \preceq g \Leftrightarrow \forall x\ f(x) \leq g(x)$ forms a vector lattice.

3. The Lipschitz Classifier

Let $(\mathcal{X}, d)$ be a metric space and $(x_i, y_i)_{i=1,\ldots,n} \subset \mathcal{X} \times \{\pm 1\}$ some training data. In order to be able to define hyperplanes, we want to embed $(\mathcal{X}, d)$ into a vector space, but without loosing or changing the underlying metric structure.

3.1 Embedding and Large Margin in Banach Spaces

Our first step is to embed $\mathcal{X}$ by the identity mapping into the extended space $\mathcal{X}_0$ as described in (2), which in turn is embedded into $AE(\mathcal{X}_0)$ via (5). We denote the resulting composite embedding by

$$\Phi : \mathcal{X} \to AE(\mathcal{X}_0),\ x \mapsto m_x := m_{xe}.$$

Secondly, we identify $\mathrm{Lip}(\mathcal{X})$ with $\mathrm{Lip}_0(\mathcal{X}_0)$ according to (3) and then $\mathrm{Lip}_0(\mathcal{X}_0)$ with $AE(\mathcal{X}_0)'$ according to Theorem 3. Together this defines the map

$$\Psi : \mathrm{Lip}(\mathcal{X}) \to AE(\mathcal{X}_0)',\ f \mapsto T_f.$$

Lemma 4 (Properties of the embeddings) *The mappings Φ and Ψ have the following properties:*

1. *Φ is an isometric embedding of $\mathcal{X}$ into $AE(\mathcal{X}_0)$: to every point $x \in \mathcal{X}$ corresponds a molecule $m_x \in AE(\mathcal{X}_0)$ such that $d(x,y) = \|m_x - m_y\|_{AE}$ for all $x,y \in \mathcal{X}$.*

2. *$\mathrm{Lip}(\mathcal{X})$ is isometrically isomorphic to $AE(\mathcal{X}_0)'$: to every Lipschitz function f on $\mathcal{X}$ corresponds an operator T_f on $AE(\mathcal{X}_0)$ such that $\|f\|_L = \|T_f\|$ and vice versa.*

3. *It makes no difference whether we evaluate operators on the image of $\mathcal{X}$ in $AE(\mathcal{X}_0)$ or apply Lipschitz functions on $\mathcal{X}$ directly: $T_f m_x = f(x)$.*

4. *Scaling a linear operator is the same as scaling the corresponding Lipschitz function: for $a \in \mathbb{R}$ we have $aT_f = T_{af}$.*

Proof All these properties are direct consequences of the construction and Equation (4). ■

The message of this lemma is that it makes no difference whether we classify our training data on the space $\mathcal{X}$ with the decision function $\operatorname{sgn} f(x)$ or on $AE(\mathcal{X}_0)$ with the hyperplane $\operatorname{sgn}(T_f m_x)$. The advantage of the latter is that constructing a large margin classifier in a Banach space is a well studied problem. In Bennett and Bredensteiner (2000) and Zhou et al. (2002) it has been established that constructing a maximal margin hyperplane between the set X^+ of positive and X^- of negative training points in a Banach space V is equivalent to finding the distance between the convex hulls of X^+ and X^-. More precisely, let C^+ and C^- the convex hulls of the sets X^+ and X^-. In the separable case, we define the margin of a separating hyperplane H between C^+ and C^- as the minimal distance between the training points and the hyperplane:

$$\rho(H) := \inf_{i=1,\ldots,n} d(x_i, H).$$

The margin of the maximal margin hyperplane coincides with half the distance between the convex hulls of the positive and negative training points. Hence, determining the maximum margin hyperplane can be understood as solving the optimization problem

$$\inf_{p^+ \in C^+, p^- \in C^-} \|p^+ - p^-\|.$$

By duality arguments (cf. Bennett and Bredensteiner, 2000) it can be seen that its solution coincides with the solution of

$$\sup_{T \in V'} \inf_{p^+ \in C^+, p^- \in C^-} \langle T, p^+ - p^- \rangle / \|T\|.$$

This can be equivalently rewritten as the optimization problem

$$\inf_{T \in V', b \in} \|T\| \text{ subject to } y_i(\langle T, x_i \rangle + b) \geq 1 \ \ \forall i = 1, \ldots, n. \tag{8}$$

A solution of this problem is called a large margin classifier. The decision function has the form $f(x) = \langle T, x \rangle + b$, and its margin is given by $1/\|T\|$. For details we refer to Bennett and Bredensteiner (2000) and Zhou et al. (2002).

3.2 Derivation of the Algorithm

Now we can apply this construction to our situation. We embed $\mathcal{X}$ isometrically into the Banach space $AE(\mathcal{X}_0)$ and use the above reasoning to construct a large margin classifier. As the dual space of $AE(\mathcal{X}_0)$ is $\mathrm{Lip}_0(\mathcal{X}_0)$ and $\langle f, m_x \rangle = f(x)$, the optimization problem (8) in our case is

$$\inf_{f_0 \in \mathrm{Lip}_0(\mathcal{X}_0), b \in} L(f_0) \text{ subject to } y_i(f_0(x_i)+b) \geq 1 \ \ \forall i = 1, ..., n.$$

By the isometry stated in Theorem 3, this is equivalent to the problem

$$\inf_{f \in \mathrm{Lip}(\mathcal{X}), b \in} \|f\|_L \text{ subject to } y_i(f(x_i)+b) \geq 1 \ \ \forall i = 1, ..., n.$$

Next we want to show that the solution of this optimization problem does not depend on the variable b. To this end, we first set $g := f + b \in \mathrm{Lip}(\mathcal{X})$ to obtain

$$\inf_{g \in \mathrm{Lip}(\mathcal{X}), b \in} \|g - b\|_L \text{ subject to } y_i g(x_i) \geq 1 \ \ \forall i = 1, ..., n.$$

Then we observe that

$$\|g-b\|_L = \max\{L(g-b), \frac{\|g-b\|_\infty}{\mathrm{diam}(\mathcal{X})}\} = \max\{L(g), \frac{\|g-b\|_\infty}{\mathrm{diam}(\mathcal{X})}\} \geq L(g) = \max\{L(g), \frac{\|g\|_\infty}{\mathrm{diam}(\mathcal{X})}\}.$$

Here the last step is true because of the fact that g takes positive and negative values and thus $\|g\|_\infty / \mathrm{diam}(\mathcal{X}) \leq L(g)$ as we explained in Equation (1) of Section 2. Hence, under the constraints $y_i g(x_i) \geq 1$ we have $\inf_b \|g-b\|_L = L(g)$, and we can rewrite our optimization problem in the final form

$$\inf_{f \in \mathrm{Lip}(\mathcal{X})} L(f) \text{ subject to } y_i f(x_i) \geq 1, \ i = 1, \ldots, n. \qquad (*)$$

We call a solution of this problem a (hard margin) *Lipschitz classifier*. So we have proved:

Theorem 5 (Lipschitz classifier) *Let $(\mathcal{X}, d)$ be a bounded metric space, $(x_i, y_i)_{i=1,\ldots,n} \subset \mathcal{X} \times \{\pm 1\}$ some training data containing points of both classes. Then a solution f of $(*)$ is a large margin classifier, and its margin is given by $1/L(f)$.*

One nice aspect about the above construction is that the margin constructed in the space $AE(\mathcal{X}_0)$ also has a geometrical meaning in the original input space $\mathcal{X}$ itself: it is a lower bound on the minimal distance between the "separation surface" $S := \{s \in \mathcal{X};\ f(s) = 0\}$ and the training points. To see this, normalize the function f such that $\min_{i=1,\ldots,n} |f(x_i)| = 1$. This does not change the set S. Because of

$$1 \leq |f(x_i)| = |f(x_i) - f(s)| \leq L(f) d(x_i, s)$$

we thus get $d(x_i, s) \geq 1/L(f)$.

Analogously to SVMs we also define the soft margin version of the Lipschitz classifier by introducing slack variables ξ_i to allow some training points to lie inside the margin or even be misclassified:

$$\inf_{f \in \mathrm{Lip}(\mathcal{X})} L(f) + C \sum_{i=1}^{n} \xi_i \text{ subject to } y_i f(x_i) \geq 1 - \xi_i, \ \xi_i \geq 0. \qquad (**)$$

In regularization language, the soft margin Lipschitz classifier can be stated as

$$\inf_{f \in \mathrm{Lip}(\mathcal{X})} \ell(y_i f(x_i)) + \lambda L(f)$$

where the loss function ℓ is given by $\ell(y_i f(x_i)) = \max\{0, 1 - y_i f(x_i)\}$.

In Section 4, we will give an analytic expression for a solution of $(*)$ and show how $(**)$ can be written as a linear programming problem. However, it may be sensible to restrict the set over which the infimum is taken in order to avoid overfitting. We thus suggest to consider the above optimization problems over subspaces of $\mathrm{Lip}(\mathcal{X})$ rather than the whole space $\mathrm{Lip}(\mathcal{X})$. In Section 6 we derive a geometrical interpretation of the choice of different subspaces. Now we want to point out some special cases.

Assume that we are given training points in some reproducing kernel Hilbert space H. As it is always the case for linear functions, the Lipschitz constant of a linear function in H' coincides with its Hilbert space norm. This means that the support vector machine in H chooses the same linear function as the Lipschitz algorithm, if the latter takes the subspace of linear functions as hypothesis space.

In the case where we optimize over the subset of all linear combinations of distance functions of the form $f(x) = \sum_{i=1}^n a_i d(x_i, x) + b$, the Lipschitz algorithm can be approximated by the linear programming machine (cf. Graepel et al., 1999):

$$\inf_{a,b} \sum_{i=1}^n |a_i| \text{ subject to } y_i(\sum_{i=1}^n a_i d(x_i, x) + b) \geq 1.$$

The reason for this is that the Lipschitz constant of a function $f(x) = \sum_{i=1}^n a_i d(x_i, x) + b$ is upper bounded by $\sum_i |a_i|$. Furthermore, if we do not restrict the function space at all, then we will see in the next section that the 1-nearest neighbor classifier is a solution of the Lipschitz algorithm.

These examples show that the Lipschitz algorithm is a very general approach. By choosing different subsets of Lipschitz functions we recover several well known algorithms. As the Lipschitz algorithm is a large margin algorithm according to Theorem 5, the same holds for the recovered algorithms. For instance the linear programming machine, originally designed with little theoretical justification, can now be understood as a large margin algorithm.

4. Representer Theorems

A crucial theorem in the context of SVMs and other kernel algorithms is the representer theorem (cf. Schölkopf and Smola, 2002). It states that even though the space of possible solutions of these algorithms forms an infinite dimensional space, there always exists a solution in the finite dimensional subspace spanned by the training points. It is because of this theorem that SVMs overcome the curse of dimensionality and yield computationally tractable solutions. In this section we prove a similar theorem for the Lipschitz classifiers $(*)$ and $(**)$. To simplify the discussion, we denote $\mathcal{D} := \{d(x, \cdot);\ x \in \mathcal{X}\} \cup \{\mathbb{1}\}$ and $\mathcal{D}_{\text{train}} := \{d(x_i, \cdot);\ x_i \text{ training point }\} \cup \{\mathbb{1}\}$, where $\mathbb{1}$ is the constant-1 function.

4.1 Soft Margin Case

We first start by recalling a general result which implies the classical representer theorem in the case of SVMs.

Lemma 6 (Minimum norm interpolation) *Let V be a function of $n+1$ variables which is non-decreasing in its $n+1$-st argument. Given n points $x_1,\dots,x_n$ and a functional Ω, any function which is a solution of the problem*

$$\inf_f V(f(x_1),\dots,f(x_n),\Omega(f)) \tag{9}$$

is a solution of the minimum norm interpolation problem

$$\inf_{f:\forall i,\, f(x_i)=a_i} \Omega(f) \tag{10}$$

for some $a_1,\dots,a_n \in \mathbb{R}$.

Here, f being a solution of a problem of the form $\inf W(f)$ means $f = \operatorname{argmin} W(f)$. We learned this theorem from M. Pontil, but it seems to be due to C. Micchelli.

Proof Let f_0 be a solution of the first problem. Take $a_i = f_0(x_i)$. Then for any function f such that $f(x_i) = a_i$ for all i, we have

$$V(f(x_1),\dots,f(x_n),\Omega(f)) \geq V(f_0(x_1),\dots,f_0(x_n),\Omega(f_0)) = V(f(x_1),\dots,f(x_n),\Omega(f_0)).$$

Hence, by monotonicity of V we get $\Omega(f) \geq \Omega(f_0)$, which concludes the proof. ■

The meaning of the above result is that if the solutions of problem (10) have specific properties, then the solutions of problem (9) will also have these properties. So instead of studying the properties of solutions of $(**)$ directly, we will investigate the properties of (10) when the functional Ω is the Lipschitz norm. We first need to introduce the concept of Lipschitz extensions.

Lemma 7 (Lipschitz extension) *Given a function f defined on a finite subset $x_1,\dots,x_n$ of $\mathcal{X}$, there exists a function f' which coincides with f on $x_1,\dots,x_n$, is defined on the whole space $\mathcal{X}$, and has the same Lipschitz constant as f. Additionally, it is possible to explicitly construct f' in the form*

$$f'(x) = \alpha \min_{i=1,\dots,n} \left(f(x_i) + L(f)d(x,x_i)\right) + (1-\alpha) \max_{i=1,\dots,n} \left(f(x_i) - L(f)d(x,x_i)\right),$$

for any $\alpha \in [0,1]$, with $L(f) = \max_{i,j=1,\dots,n}(f(x_i)-f(x_j))/d(x_i,x_j)$.

Proof Consider the function $g(x) = \min_{i=1,\dots,n}(f(x_i) + L(f)d(x,x_i))$. We have

$$|g(x)-g(y)| \leq \max_{i=1,\dots,n} |f(x_i)+L(f)d(x,x_i) - f(x_i) - L(f)d(y,x_i)| \leq L(f)d(x,y),$$

so that $L(g) \leq L(f)$. Also, by definition $g(x_i) \leq f(x_i) + L(f)d(x_i,x_i) = f(x_i)$. Moreover, if i_0 denotes the index where the minimum is achieved in the definition of $g(x_i)$, i.e. $g(x_i) = f(x_{i_0}) + L(f)d(x_i,x_{i_0})$, then by definition of $L(f)$ we have $g(x_i) \geq f(x_{i_0}) + (f(x_i) - f(x_{i_0})) = f(x_i)$. As a result, for all $i = 1,\dots,n$ we have $g(x_i) = f(x_i)$, which also implies that $L(g) = L(f)$.
Now the same reasoning can be applied to $h(x) = \max_{i=1,\dots,n}(f(x_i) - L(f)d(x,x_i))$. Since $\alpha \in [0,1]$ we have $f'(x_i) = f(x_i)$ for all i. Moreover, $L(\alpha g + (1-\alpha)h) \leq \alpha L(g) + (1-\alpha)L(h) = L(f)$ and thus $L(f') = L(f)$, which concludes the proof. ■

From the above lemma, we obtain an easy way to construct solutions of minimum norm interpolation problems like (10) with Lipschitz norms, as is expressed in the next lemma.

Lemma 8 (Solution of the Lipschitz minimal norm interpolation problem) *Let $a_1,\ldots,a_n \in \mathbb{R}^n$, $\alpha \in [0,1]$, $L_0 = \max_{i,j=1,\ldots,n}(a_i - a_j)/d(x_i,x_j)$, and*

$$f_\alpha(x) := \alpha \min_{i=1,\ldots,n}(a_i + L_0 d(x,x_i)) + (1-\alpha)\max_{i=1,\ldots,n}(a_i - L_0 d(x,x_i))\,.$$

Then f_α is a solution of the minimal norm interpolation problem (10) *with $\Omega(f) = L(f)$. Moreover, when $\alpha = 1/2$ then f_α is a solution of the minimal norm interpolation problem* (10) *with $\Omega(f) = \|f\|_L$.*

Proof Given that a solution f of (10) has to satisfy $f(x_i) = a_i$, it cannot have $L(f) < L_0$. Moreover, by Lemma 7 f_α satisfies the constraints and has $L(f) = L_0$, hence it is a solution of (10) with $\Omega(f) = L(f)$.

When one takes $\Omega(f) = \|f\|_L$, any solution f of (10) has to have $L(f) \geq L_0$ and $\|f\|_\infty \geq \max_i |a_i|$. The proposed solution f_α with $\alpha = 1/2$ not only satisfies the constraints $f_\alpha(x_i) = a_i$ but also has $L(f) = L_0$ and $\|f\|_\infty = \max_i |a_i|$, which shows that it is a solution of the considered problem.
To prove that $\|f\|_\infty = \max_i |a_i|$, consider $x \in \mathcal{X}$ and denote by i_1 and i_2 the indices where the minimum and the maximum, respectively, are achieved in the definition of $f_\alpha(x)$. Then one has

$$f_{1/2}(x) \leq \frac{1}{2}(a_{i_2} + L_0 d(x,x_{i_2})) + \frac{1}{2}(a_{i_2} - L_0 d(x,x_{i_2})) = a_{i_2}\,,$$

and similarly $f_{1/2}(x) \geq a_{i_1}$. ■

Now we can formulate a general representer theorem for the soft margin Lipschitz classifier.

Theorem 9 (Soft margin representer theorem) *There exists a solution of the soft margin Lipschitz classifier* $(**)$ *in the vector lattice spanned by $\mathcal{D}_{\text{train}}$ which is of the form*

$$f(x) = \frac{1}{2}\min(a_i + L_0 d(x,x_i)) + \frac{1}{2}\max(a_i - L_0 d(x,x_i))$$

for some real numbers $a_1,\ldots,a_n$ with $L_0 := \max_{i,j}(a_i - a_j)/d(x_i,x_j)$. Moreover one has $\|f\|_L = L(f) = L_0$.

Proof The first claim follows from Lemmas 6 and 8. The second claim follows from the fact that a solution of $(**)$ satisfies $\|f\|_L = L(f)$. ■

Theorem 9 is remarkable as the space $\mathrm{Lip}(\mathcal{X})$ of possible solutions of $(**)$ contains the whole vector lattice spanned by $\mathcal{D}$. The theorem thus states that even though the Lipschitz algorithm searches for solutions in the whole lattice spanned by $\mathcal{D}$ it always manages to come up with a solution in the sublattice spanned by $\mathcal{D}_{\text{train}}$.

4.2 Algorithmic Consequences

As a consequence of the above theorem, we can obtain a tractable algorithm for solving problem $(**)$. First, we determine the coefficients a_i by solving

$$\min_{a_1,\ldots,a_n \in} \sum_{i=1}^n \ell(y_i a_i) + \lambda \max_{i,j} \frac{(a_i - a_j)}{d(x_i,x_j)}\,,$$

which can be rewritten as a linear programming problem

$$\min_{a_1,\ldots,a_n,\xi_1,\ldots,\xi_n,\rho\in} \sum_{i=1}^{n} \xi_i + \lambda\rho\,,$$

under the constraints $\xi_i \geq 0$, $y_i a_i \geq 1-\xi_i$, $\rho \geq (a_i - a_j)/d(x_i,x_j)$. Once a solution is found, one can simply take the function $f_{1/2}$ defined in Theorem 9 with the coefficients a_i determined by the linear program. Note, however, that in practical applications, the solution found by this procedure might overfit as it optimizes $(**)$ over the whole class $\mathrm{Lip}(\mathcal{X})$.

4.3 Hard Margin Case

The representer theorem for the soft margin case clearly also holds in the hard margin case, so that there will always be a solution of $(*)$ in the vector lattice spanned by $\mathcal{D}_{\text{train}}$. But in the hard margin case, also a different representer theorem is valid. We denote the set of all training points with positive label by X^+, the set of the training points with negative label by X^-, and for two subsets $A, B \subset \mathcal{X}$ we define $d(A,B) := \inf_{a\in A, b\in B} d(a,b)$.

Theorem 10 (Hard margin representer theorem) *Problem* $(*)$ *always has a solution which is a linear combination of distances to* sets *of training points.*

To prove this theorem we first need a simple lemma.

Lemma 11 (Optimal Lipschitz constant) *The Lipschitz constant* L^* *of a solution of* $(*)$ *satisfies* $L^* \geq 2/d(X^+,X^-)$.

Proof For a solution f of $(*)$ we have

$$\begin{aligned} L(f) &= \sup_{x,y\in\mathcal{X}} \frac{|f(x)-f(y)|}{d(x,y)} \geq \max_{i,j=1,\ldots,n} \frac{|f(x_i)-f(x_j)|}{d(x_i,x_j)} \\ &\geq \max_{i,j=1,\ldots,n} \frac{|y_i-y_j|}{d(x_i,x_j)} = \frac{2}{\min_{x_i\in X^+, x_j\in X^-} d(x_i,x_j)} = \frac{2}{d(X^+,X^-)}. \end{aligned}$$

■

Lemma 12 (Solutions of $(*)$) *Let* $L^* = 2/d(X^+,X^-)$. *For all* $\alpha \in [0,1]$, *the following functions solve* $(*)$*:*

$$f_\alpha(x) := \alpha \min_i (y_i + L^* d(x,x_i)) + (1-\alpha)\max_i (y_i - L^* d(x,x_i))$$

$$g(x) := \frac{d(x,X^-) - d(x,X^+)}{d(X^+,X^-)}$$

Proof By Lemma 7, f_α has Lipschitz constant L^* and satisfies $f_\alpha(x_i) = y_i$. Moreover, it is easy to see that $y_i g(x_i) \geq 1$. Using the properties of Lipschitz constants stated in Section 2 and the fact that the function $d(x,\cdot)$ has Lipschitz constant 1 we see that $L(g) \leq L^*$. Thus f_α and g are solutions of $(*)$ by Lemma 11. ■

The functions f_α and g lie in the vector lattice spanned by $\mathcal{D}_{\text{train}}$. As g is a linear combination of distances to sets of training points we have proved Theorem 10.

It is interesting to have a closer look at the functions of Lemma 12. The functions f_0 and f_1 are the smallest and the largest functions, respectively, that solve problem $(*)$ with equality in the constraints: any function f that satisfies $f(x_i) = y_i$ and has Lipschitz constant L^* satisfies $f_0(x) \leq f(x) \leq f_1(x)$. The functions g and $f_{1/2}$ are especially remarkable:

Lemma 13 (1-nearest neighbor classifier) *The functions g and $f_{1/2}$ defined above have the sign of the 1-nearest neighbor classifier.*

Proof It is obvious that $g(x) > 0 \iff d(x,X^+) < d(x,X^-)$ and $g(x) < 0 \iff d(x,X^+) > d(x,X^-)$. For the second function, we rewrite $f_{1/2}$ as follows:

$$f_{1/2}(x) = \frac{1}{2}(\min(L^* d(x,X^+)+1, L^* d(x,X^-)-1) - \min(L^* d(x,X^+)-1, L^* d(x,X^-)+1))\,.$$

Consider x such that $d(x,X^+) \geq d(x,X^-)$. Then $d(x,X^+)+1 \geq d(x,X^-)-1$ and thus

$$f_{1/2}(x) = \frac{1}{2}\left(L^* d(x,X^-) - 1 - \min(L^* d(x,X^+)-1, L^* d(x,X^-)+1)\right) \leq 0\,.$$

The same reasoning applies to the situation $d(x,X^+) \leq d(x,X^-)$ to yield $f_{1/2}(x) \geq 0$ in this case. ■

Note that g needs not reach equality in the constraints on all the data points, whereas the function $f_{1/2}$ always satisfies equality in the constraints. Lemma 13 has the surprising consequence that according to Section 3, the 1-nearest neighbor classifier actually is a large margin classifier.

4.4 Negative Results

So far we have proved that $(*)$ always has a solution which can be expressed as a linear combination of distances to sets of training points. But maybe we even get a theorem stating that we always find a solution which is a linear combination of distance functions to single training points? Unfortunately, in the metric space setting such a theorem is not true in general. This can be seen by the following counterexample:

Example 1 *Assume four training points x_1, x_2, x_3, x_4 with distance matrix*

$$D = \begin{pmatrix} 0 & 2 & 1 & 1 \\ 2 & 0 & 1 & 1 \\ 1 & 1 & 0 & 2 \\ 1 & 1 & 2 & 0 \end{pmatrix}$$

and label vector $y = (1,1,-1,-1)$. Then the set

$$\{f : \mathcal{X} \to \mathbb{R} |\, y_i f(x_i) \geq 1,\ f(x) = \sum_{i=1}^{4} a_i d(x_i,x) + b\}$$

is empty. The reason for this is that the distance matrix is singular and we have $d(x_1,\cdot)+d(x_2,\cdot) = d(x_3,\cdot) = d(x_4,\cdot)$. *Hence, in this example,* $(*)$ *has no solution which is a linear combination of distances to single training points. But it still has a solution as linear combination of distances to sets of training points according to Theorem 10.*

Another negative result is the following. Assume that instead of looking for solutions of $(*)$ in the space of all Lipschitz functions we only consider functions in the vector space spanned by $\mathcal{D}$. Is it in this case always possible to find solution in the linear span of $\mathcal{D}_{train}$? The answer is no again. An example for this is the following:

Example 2 *Let* $\mathcal{X} = \{x_1,...,x_5\}$ *consist of five points with distance matrix*

$$D = \begin{pmatrix} 0 & 2 & 1 & 1 & 1 \\ 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 2 & 1 \\ 1 & 1 & 2 & 0 & 2 \\ 1 & 1 & 1 & 2 & 0 \end{pmatrix}.$$

Let the first four points be training points with the label vector $y = (-1,-1,-1,1)$. *As above there exists no feasible function in the vector space spanned by* $\mathcal{D}_{train}$. *But as the distance matrix of all five points is invertible, there exist feasible functions in the vector space spanned by* $\mathcal{D}$.

In the above examples the problem was that the distance matrix on the training points was singular. But there are also other sources of problems that can occur. In particular it can be the case that the Lipschitz constant of a function restricted to the training set takes the minimal value L^*, but the Lipschitz constant on the whole space $\mathcal{X}$ is larger. Then it can happen that although we can find a linear combination of distance functions that satisfies $f(x_i) = y_i$, the function f has a Lipschitz constant larger than L^* and thus is no solution of $(*)$. An example for this situation is the following:

Example 3 *Let* $\mathcal{X} = \{x_1,...,x_5\}$ *consist of five points with distance matrix*

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 2 & 1 \\ 1 & 1 & 2 & 0 & 1 \\ 1 & 2 & 1 & 1 & 0 \end{pmatrix}.$$

Let the first four points be training points with the label vector $y = (1,1,-1,-1)$. *The optimal Lipschitz constant in this problem is* $L^* = 2/d(X^+,X^-) = 2$. *The function* $f(x) = -2d(x_1,x) - 2d(x_2,x) + 3$ *has this Lipschitz constant if we evaluate it on the training points only. But if we also consider* x_5, *the function has Lipschitz constant* 4.

These examples show that, in general, Theorem 10 cannot be improved to work in the vector space instead of the vector lattice spanned by $\mathcal{D}_{\text{train}}$. This also holds if we consider some subspaces of the set of Lipschitz functions. Thus we are in the interesting situation that it is not enough to consider distance functions to single training points – we have to deal with distances to sets of training points.

5. Error Bounds

In this section we compute error bounds for the Lipschitz classifier using Rademacher averages. This can be done following techniques introduced for example in Chapter 3 of Devroye and Lugosi (2001) or in Bartlett and Mendelson (2002). The measures of capacity we consider are the Rademacher average R_n and the related maximum discrepancy $\tilde{R}_n$. For an arbitrary class $\mathcal{F}$ of functions, they are defined as

$$R_n(\mathcal{F}) := E\left(\frac{1}{n}\sup_{f\in\mathcal{F}}|\sum_{i=1}^{n}\sigma_i f(X_i)|\right) \geq \frac{1}{2}E\left(\frac{1}{n}\sup_{f\in\mathcal{F}}|\sum_{i=1}^{n}(f(X_i)-f(X_i'))\right)| =: \frac{1}{2}\tilde{R}_n(\mathcal{F})$$

where σ_i are iid Rademacher random variables (i.e., $Prob(\sigma_i = +1) = Prob(\sigma_i = -1) = 1/2$), X_i and X_i' are iid sample points according to the (unknown) sample distribution, and the expectation is taken with respect to all occurring random variables. Sometimes we also consider the conditional Rademacher average $\hat{R}_n$, where the expectation is taken only conditionally on the sample points $X_1,...,X_n$. For decision function f, consider the loss function $\ell(f(x),y) = 1$ if $yf(x) \leq -1$, $1-yf(x)$ if $0 \leq yf(x) \leq 1$, and 0 if $yf(x) \geq 1$. Let $\mathcal{F}$ be a class of functions, denote by E the expectation with respect to the unknown sample distribution and by E_n the expectation with respect to the empirical distribution of the training points.

Lemma 14 (Error bounds) *With probability at least* $1-\delta$ *over the iid drawing of n sample points, every* $f \in \mathcal{F}$ *satisfies*

$$E(\ell(f(X),Y)) \leq E_n(\ell(f(X),Y)) + 2R_n(\mathcal{F}) + \sqrt{\frac{8\log(2/\delta)}{n}}.$$

Proof The proof is based on techniques of Devroye and Lugosi (chap. 3 of 2001) and Bartlett and Mendelson (2002): McDiarmid's concentration inequality, symmetrization and contraction property of Rademacher averages. ■

A similar bound can be obtained with the maximum discrepancy (see Bartlett and Mendelson, 2002).

We will describe two different ways to compute Rademacher averages for sets of Lipschitz functions. One way is a classical approach using entropy numbers and leads to an upper bound on R_n. For this approach we always assume that the metric space $(\mathcal{X},d)$ is precompact (i.e., it can be covered by finitely many balls of radius ε for every $\varepsilon > 0$).

The other way is more elegant: because of the definition of $\|\cdot\|_L$ and the resulting isometries, the maximum discrepancy of a $\|\cdot\|_L$-unit ball of $\mathrm{Lip}(\mathcal{X})$ is the same as of the corresponding unit ball in $AE(\mathcal{X}_0)'$. Hence it will be possible to express $\tilde{R}_n$ as the norm of an element of the Arens-Eells space. This norm can then be computed via bipartite minimal matching. In the following, B always denotes the unit ball of the considered function space.

5.1 The Duality Approach

The main insight to compute the maximum discrepancy by the duality approach is the following observation:

$$\sup_{\|f\|_L\leq 1} |\sum_{i=1}^{n} f(x_i) - f(x_i')| = \sup_{\|T_f\|\leq 1} |\sum_{i=1}^{n} T_f m_{x_i} - T_f m_{x_i'}| =$$

$$= \sup_{\|T_f\|\leq 1} |\langle T_f, \sum_{i=1}^{n} m_{x_i} - m_{x_i'}\rangle| = \|\sum_{i=1}^{n} m_{x_i x_i'}\|_{AE}$$

Applying this to the definition of the maximum discrepancy immediately yields

$$\tilde{R}_n(B) = \frac{1}{n} E \|\sum_{i=1}^{n} m_{X_i X_i'}\|_{AE}. \tag{11}$$

As we already explained in Section 2, the norm $\|\sum_{i=1}^{n} m_{X_i X_i'}\|_{AE}$ can be interpreted as the costs of a minimal bipartite matching between $\{X_1,\ldots,X_n\}$ and $\{X_1',\ldots,X_n'\}$. To compute the right hand side of (11) we need to know the expected value of random instances of the bipartite minimal matching problem, where we assume that the points X_i and X_i' are drawn iid from the sample distribution. In particular we want to know how this value scales with the number n of points as this indicates how fast we can learn. This question has been solved for some special cases of random bipartite matching. Let the random variable C_n describe the minimal bipartite matching costs for a matching between the points $X_1,\ldots,X_n$ and $X_1',\ldots,X_n'$ drawn iid according to some distribution P. In Dobric and Yukich (1995) it has been proved that for an arbitrary distribution on the unit square of $\mathbb{R}^d$ with $d \geq 3$ we have $\lim C_n/(n^{d-1/d}) = c > 0$ a.s. for some constant c. The upper bound $EC_n \leq c\sqrt{n\log n}$ for arbitrary distributions on the unit square in $\mathbb{R}^2$ was presented in Talagrand (1992). These results, together with Equation (11), lead to the following maximum discrepancies:

Theorem 15 (Maximum discrepancy of unit ball of $\mathrm{Lip}([0,1]^d)$) *Let $\mathcal{X} = [0,1]^d \subset \mathbb{R}^d$ with the Euclidean metric. Then the maximum discrepancy of the $\|\cdot\|_L$-unit ball B of $\mathrm{Lip}(\mathcal{X})$ satisfies*

$$\tilde{R}_n(B) \leq c_2\sqrt{\log n}/\sqrt{n} \quad \text{for all } n \in \mathbb{N} \qquad \text{if } d = 2$$
$$\lim_{n\to\infty} \tilde{R}_n(B)\sqrt[d]{n} = c_d > 0 \qquad \text{if } d \geq 3$$

where c_d ($d \geq 2$) are constants which are independent of n but depend on d.

Note that this procedure gives (asymptotically) exact results rather than upper bounds in cases where we have (asymptotically) exact results on the bipartite matching costs. This is for example the case for cubes in $\mathbb{R}^d, d \geq 3$ as Dobric and Yukich (1995) gives an exact limit result, or for $\mathbb{R}^2$ with the uniform distribution.

5.2 Covering Number Approach

To derive the Rademacher complexity in more general settings than Euclidean spaces we use an adapted version of the classical entropy bound of Dudley based on covering numbers. The covering number $N(\mathcal{X},\varepsilon,d)$ of a totally bounded metric space $(\mathcal{X},d)$ is the smallest number of balls of radius ε with centers in $\mathcal{X}$ which can cover $\mathcal{X}$ completely. The proof of the following theorem can be found in the appendix.

Theorem 16 (Generalized entropy bound) *Let $\mathcal{F}$ be a class of functions and $X_1,\ldots,X_n$ iid sample points with empirical distribution μ_n. Then, for every $\varepsilon > 0$,*

$$\hat{R}_n(\mathcal{F}) \leq 2\varepsilon + \frac{4\sqrt{2}}{\sqrt{n}} \int_{\varepsilon/4}^{\infty} \sqrt{\log N(\mathcal{F}, u, L_2(\mu_n))}\, du.$$

To apply this theorem we need to know covering numbers of spaces of Lipschitz functions. This can be found for example in Kolmogorov and Tihomirov (1961), pp.353–357.

Theorem 17 (Covering numbers for Lipschitz function balls) *For a totally bounded metric space $(\mathcal{X}, d)$ and the unit ball B of $(\mathrm{Lip}(\mathcal{X}), \|\cdot\|_L)$,*

$$2^{N(\mathcal{X}, 4\varepsilon, d)} \leq N(B, \varepsilon, \|\cdot\|_\infty) \leq \left(2\left\lceil \frac{2\,\mathrm{diam}(\mathcal{X})}{\varepsilon} \right\rceil + 1\right)^{N(\mathcal{X}, \frac{\varepsilon}{4}, d)}.$$

If, in addition, $\mathcal{X}$ is connected and centered (i.e., for all subsets $A \subset \mathcal{X}$ with $\mathrm{diam}(A) \leq 2r$ there exists a point $x \in \mathcal{X}$ such that $d(x,a) \leq r$ for all $a \in A$),

$$2^{N(\mathcal{X}, 2\varepsilon, d)} \leq N(B, \varepsilon, \|\cdot\|_\infty) \leq \left(2\left\lceil \frac{2\,\mathrm{diam}(\mathcal{X})}{\varepsilon} \right\rceil + 1\right) \cdot 2^{N(\mathcal{X}, \frac{\varepsilon}{2}, d)}.$$

Combining Theorems 16 and 17 and using $N(\mathcal{F}, u, L_2(\mu_n)) \leq N(\mathcal{F}, u, \|\cdot\|_\infty)$ now gives a bound on the Rademacher complexity of balls of $\mathrm{Lip}(\mathcal{X})$:

Theorem 18 (Rademacher complexity of unit ball of $\mathrm{Lip}(\mathcal{X})$) *Let $(\mathcal{X}, d)$ be a totally bounded metric space with diameter $\mathrm{diam}(\mathcal{X})$ and B the ball of Lipschitz functions with $\|f\|_L \leq 1$. Then, for every $\varepsilon > 0$,*

$$R_n(B) \leq 2\varepsilon + \frac{4\sqrt{2}}{\sqrt{n}} \int_{\varepsilon/4}^{4\,\mathrm{diam}(\mathcal{X})} \sqrt{N(\mathcal{X}, \frac{u}{4}, d) \log\left(2\left\lceil \frac{2\,\mathrm{diam}(\mathcal{X})}{u} \right\rceil + 1\right)}\, du.$$

If, in addition, $\mathcal{X}$ is connected and centered, we have

$$R_n(B) \leq 2\varepsilon + \frac{4\sqrt{2}}{\sqrt{n}} \int_{\varepsilon/4}^{2\,\mathrm{diam}(\mathcal{X})} \sqrt{N(\mathcal{X}, \frac{u}{2}, d) \log 2 + \log(2\left\lceil \frac{2\,\mathrm{diam}(\mathcal{X})}{u} \right\rceil + 1)}\, du.$$

In our framework this is a nice result as the bound on the complexity of balls of $\mathrm{Lip}(\mathcal{X})$ only uses the metric properties of the underlying space $\mathcal{X}$. Now we want to compare the results of Theorems 15 and 18 for two simple examples.

Example 4 (d-dimensional unit square, $d \geq 3$) *Let $\mathcal{X} = [0,1]^d \subset \mathbb{R}^d, d \geq 3$, with the Euclidean metric $\|\cdot\|_2$. This is a connected and centered space. In Theorem 15 we showed that $\tilde{R}_n(B)$ asymptotically scales as $1/\sqrt[d]{n}$, and this result cannot be improved. Now we want to check whether Theorem 18 achieves a similar scaling rate. To this end we choose $\varepsilon = 1/\sqrt[d]{n}$ (as we know that we cannot obtain a rate smaller than this) and use that the covering numbers of $\mathcal{X}$ have the form $N(\mathcal{X}, \varepsilon, \|\cdot\|_2) = c/\varepsilon^d$ (e.g., page 1 of Mendelson and Vershynin, 2003). After evaluating the second integral of Theorem 18 we find that $R_n(B)$ indeed scales as $1/\sqrt[d]{n}$.*

Example 5 (2-dimensional unit square) *Let $\mathcal{X} = [0,1]^2 \subset \mathbb{R}^2$ with the Euclidean metric. Applying Theorem 18 similar to Example 4 yields a bound on $R_n(B)$ that scales as $\log n/\sqrt{n}$.*

In case of Example 4 the scaling behavior of the upper bound on $R_n(B)$ obtained by the covering number approach coincides with the exact result for $\tilde{R}_n(B)$ derived in Theorem 15. In case of Example 5 the covering number result $\log n/\sqrt{n}$ is slightly worse than the result $\sqrt{\log(n)}/\sqrt{n}$ obtained in Theorem 15.

5.3 Complexity of Lipschitz RBF Classifiers

In this section we want to derive a bound for the Rademacher complexity of radial basis function classifiers of the form

$$\mathcal{F}_{rbf} := \{ f : \mathcal{X} \to \mathbb{R} |\ f(x) = \sum_{k=1}^{l} a_k g_k(d(p_k,x)),\ g_k \in \mathcal{G},\ l < \infty \}, \tag{12}$$

where $p_k \in \mathcal{X}$, $a_k \in \mathbb{R}$, and $\mathcal{G} \subset \mathrm{Lip}(\mathcal{X})$ is a (small) set of $\|\cdot\|_\infty$-bounded Lipschitz functions on $\mathbb{R}$ whose Lipschitz constants are bounded from below by a constant $c > 0$. As an example, consider $\mathcal{G} = \{g : \mathbb{R} \to \mathbb{R} |\ g(x) = \exp(-x^2/\sigma^2), \sigma \geq 1\}$. The special case $\mathcal{G} = \{id\}$ corresponds to the function class which is used by the linear programming machine. It can easily be seen that the Lipschitz constant of an RBF function satisfies $L(\sum_k a_k g_k(d(p_k,\cdot))) \leq \sum_k |a_k| L(g_k)$. We define a norm on $\mathcal{F}_{rbf}$ by

$$\|f\|_{rbf} := \inf \left\{ \sum_k |a_k| L(g_k);\ f = \sum_k a_k g_k(d(p_k,\cdot)) \right\}$$

and derive the Rademacher complexity of a unit ball B of $(\mathcal{F}_{rbf}, \|\cdot\|_{rbf})$. Substituting a_k by $c_k/L(g_k)$ in the expansion of f we get

$$\begin{aligned}
\sup_{f \in B} |\sum_{i=1}^{n} \sigma_i f(x_i)| &= \sup_{\sum |a_k| L(g_k) \leq 1, p_k \in \mathcal{X}, g_k \in \mathcal{G}} |\sum_{i=1}^{n} \sigma_i \sum_{k=1}^{l} a_k g_k(d(p_k,x_i))| \\
&= \sup_{\sum |c_k| \leq 1, p_k \in \mathcal{X}, g_k \in \mathcal{G}} |\sum_{i=1}^{n} \sigma_i \sum_{k=1}^{l} \frac{c_k}{L(g_k)} g_k(d(p_k,x_i))| \\
&= \sup_{\sum |c_k| \leq 1, p_k \in \mathcal{X}, g_k \in \mathcal{G}} |\sum_{k=1}^{l} c_k \sum_{i=1}^{n} \sigma_i \frac{1}{L(g_k)} g_k(d(p_k,x_i))| \\
&= \sup_{p \in \mathcal{X}, g \in \mathcal{G}} |\sum_{i=1}^{n} \sigma_i \frac{1}{L(g)} g(d(p,x_i))|. \qquad (13)
\end{aligned}$$

For the last step observe that the supremum in the linear expansion in the second last line is obtained when one of the c_k is 1 and all the others are 0. To proceed we introduce the notations $h_{p,g}(x) := g(d(p,x_i))/L(g)$, $\mathcal{H} := \{h_{p,g};\ p \in \mathcal{X}, g \in \mathcal{G}\}$, and $\mathcal{G}_1 := \{g/L(g);\ g \in \mathcal{G}\}$. We rewrite the right hand side of Equation (13) as

$$\sup_{p \in \mathcal{X}, g \in \mathcal{G}} |\sum_{i=1}^{n} \sigma_i \frac{1}{L(g)} g(d(p,x_i))| = \sup_{h_{p,g} \in \mathcal{H}} |\sum_{i=1}^{n} \sigma_i h_{p,g}(x_i)|$$

and thus obtain $R_n(B) = R_n(\mathcal{H})$. To calculate the latter we need the following:

Lemma 19 $N(\mathcal{H}, 2\varepsilon, \|\cdot\|_\infty) \leq N(\mathcal{X}, \varepsilon, d) N(\mathcal{G}_1, \varepsilon, \|\cdot\|_\infty)$.

Proof First we observe that for $h_{p_1,g_1}, h_{p_2,g_2} \in \mathcal{H}$

$$\begin{aligned}
\|h_{p_1,g_1} - h_{p_2,g_2}\|_\infty &= \sup_{x\in\mathcal{X}} \left| \frac{g_1(d(p_1,x))}{L(g_1)} - \frac{g_2(d(p_2,x))}{L(g_2)} \right| \\
&\leq \sup_{x\in\mathcal{X}} \left(\left| \frac{g_1(d(p_1,x))}{L(g_1)} - \frac{g_1(d(p_2,x))}{L(g_1)} \right| + \left| \frac{|g_1(d(p_2,x))}{L(g_1)} - \frac{g_2(d(p_2,x))}{L(g_2)} \right| \right) \\
&\leq \sup_{x\in\mathcal{X}} |d(p_1,x) - d(p_2,x)| + \left\| \frac{g_1}{L(g_1)} - \frac{g_2}{L(g_2)} \right\|_\infty \\
&\leq d(p_1,p_2) + \left\| \frac{g_1}{L(g_1)} - \frac{g_2}{L(g_2)} \right\|_\infty =: d_{\mathcal{H}}(h_{p_1,g_1}, h_{p_2,g_2}) \qquad (14)
\end{aligned}$$

For the step from the second to the third line we used the Lipschitz property of g_1. Finally, it is easy to see that $N(\mathcal{H}, 2\varepsilon, d_{\mathcal{H}}) \leq N(\mathcal{X}, \varepsilon, d) N(\mathcal{G}_1, \varepsilon, \|\cdot\|_\infty)$. ■

Plugging lemma 19 in Theorem 16 yields the following Rademacher complexity:

Theorem 20 (Rademacher complexity of unit ball of $\mathcal{F}_{rbf}$) *Let B be the unit ball of $(\mathcal{F}_{rbf}, \|\cdot\|_{rbf})$, $\mathcal{G}_1$ the rescaled functions of $\mathcal{G}$ as defined above, and $w := \max\{\mathrm{diam}(\mathcal{X}, d), \mathrm{diam}(\mathcal{G}_1, \|\cdot\|_\infty)\}$. Then, for every $\varepsilon > 0$,*

$$R_n(B) \leq 2\varepsilon + \frac{4\sqrt{2}}{\sqrt{n}} \int_{\varepsilon/4}^{w} \sqrt{\log N(\mathcal{X}, \frac{u}{2}, d) + \log N(\mathcal{G}_1, \frac{u}{2}, \|\cdot\|_\infty)}\ du.$$

This theorem is a huge improvement compared to Theorem 18 as instead of the covering numbers we now have log-covering numbers in the integral. As an example consider the linear programming machine on $\mathcal{X} = [0,1]^d$. Because of $\mathcal{G} = \{id\}$, the second term in the square root vanishes, and the integral over the log-covering numbers of $\mathcal{X}$ can be bounded by a constant independent of ε. As result we obtain that in this case $R_n(B)$ scales as $1/\sqrt{n}$.

6. Choosing Subspaces of $\mathrm{Lip}(\mathcal{X})$

So far we always considered the isometric embedding of the given metric space into the Arens-Eells space and discovered many interesting properties of this embedding. But there exist many different isometric embeddings which could be used instead. Hence, the construction of embedding the metric space isometrically into some Banach space and then using a large margin classifier in this Banach space is also possible with different Banach spaces than the Arens-Eells space. For example, Hein and Bousquet (2003) used the Kuratowski embedding, which maps a metric space $\mathcal{X}$ isometrically in the space of continuous functions $(\mathcal{C}(\mathcal{X}), \|\cdot\|_\infty)$ (see Example 6 below). Now it is a natural question whether there are interesting relationships between large margin classifiers constructed by the different isometric embeddings, especially with respect to the Lipschitz classifier.

A second question concerns the choice of subspaces of $\mathrm{Lip}(\mathcal{X})$. At the end of Section 3 we already explained that we have to work on some "reasonable" subspace of Lipschitz functions to apply the Lipschitz classifier in practice. This is justified by complexity arguments, but does the large margin interpretation still hold if we do this? Is there some geometric intuition which could

help choosing a subspace?

It will turn out that both questions are inherently related to each other. We will show that there is a correspondence between embedding $\mathcal{X}$ into a Banach space V and constructing the large margin classifier on V on the one hand, and choosing a subspace F of $\mathrm{Lip}(\mathcal{X})$ and constructing the Lipschitz classifier from F on the other hand. Ideally, we would like to have a one-to-one correspondence between V and F. In one direction this would mean that we could realize any large margin classifier on any Banach space V with the Lipschitz classifier on an appropriate subspace F of Lipschitz functions. In the other direction this would mean that choosing a subspace F of Lipschitz functions corresponds to a large margin classifier on some Banach space V. We could then study the geometrical implications of a certain subspace F via the geometric properties of V.

Unfortunately, such a nice one-to-one correspondence between V and F is not always true, but in many cases it is. We will show that given an embedding into some vector space V, the hypothesis class of the large margin classifier on V always corresponds to a subspace F of Lipschitz functions (Lemma 24). In general, this correspondence will be an isomorphism, but not an isometry. The other way round, given a subspace F of Lipschitz functions, under some conditions we can construct a vector space V such that $\mathcal{X}$ can be isometrically embedded into V and the large margin classifiers on V and F coincide (Lemma 25).

The key ingredient in this section is the fact that $AE(\mathcal{X}_0)$ is a free Banach space. The following definition can be found for example in Pestov (1986).

Definition 21 (Free Banach space) *Let $(\mathcal{X}_0, d, e)$ be a pointed metric space. A Banach space $(E, \|\cdot\|_E)$ is a free Banach space over $(\mathcal{X}_0, d, e)$ if the following properties hold:*

1. *There exists an isometric embedding $\Phi : \mathcal{X}_0 \to E$ with $\Phi(e) = 0$, and E is the closed linear span of $\Phi(\mathcal{X}_0)$.*
2. *For every Banach space $(V, \|\cdot\|_V)$ and every Lipschitz map $\Psi : \mathcal{X}_0 \to V$ with $L(\Psi) = 1$ and $\Psi(e) = 0$ there exists a linear operator $T : E \to V$ with $\|T\| = 1$ such that $T \circ \Phi = \Psi$.*

It can be shown that the free Banach space over $(\mathcal{X}, d, e)$ always exists and is unique up to isomorphism (cf. Pestov, 1986).

Lemma 22 (*AE* is a free Banach space) *For any pointed metric space $(\mathcal{X}_0, d, e)$, $AE(\mathcal{X}_0)$ is a free Banach space.*

Proof Property (1) of Definition 21 is clear by construction. For a proof of property (2), see for example Theorem 2.2.4 of Weaver (1999). ■

We are particularly interested in the case where the mapping $\Psi : \mathcal{X}_0 \to V$ of Definition 21 is an isometric embedding of $\mathcal{X}_0$ into some vector space V. Firstly we want to find out under which conditions its dual V' is isometric isomorphic to some subspace F of $\mathrm{Lip}(\mathcal{X})$. Secondly, given a subspace F of $\mathrm{Lip}(\mathcal{X})$ the question is whether there exists a Banach space V such that $\mathcal{X}_0$ can be

embedded isometrically into V and simultaneously V' is isometric isomorphic to F. Both questions will be answered by considering the mapping T of Definition 21 and its adjoint T'. The following treatment will be rather technical, and it might be helpful to have Figure 1 in mind, which shows which relations we want to prove.

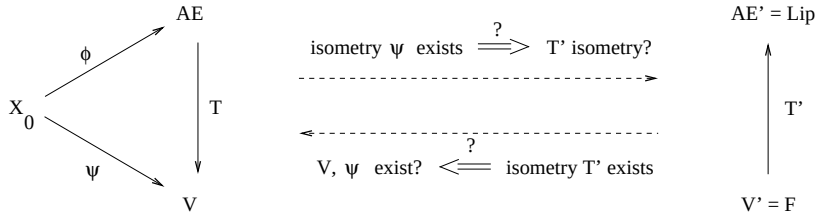


Figure 1: *Relations between Banach spaces and subspaces of Lipschitz functions. The left part shows the commutative diagram corresponding to the free Banach space property of $AE(\mathcal{X}_0)$. The right part shows the adjoint mapping T' of T. The dotted arrows in the middle show the relationships we want to investigate.*

Now we want to go into detail and start with the first question. For simplicity, we make the following definition.

Definition 23 (Dense isometric embedding) *Let $(\mathcal{X}_0, d)$ a metric space and V a normed space. A mapping $\Psi : \mathcal{X}_0 \to V$ is called a* dense isometric embedding *if Ψ is an isometry and if V is the norm-closure of* $\operatorname{span}\{\Psi(x); x \in \mathcal{X}_0\}$.

Lemma 24 (Construction of F for given V) *Let $(\mathcal{X}_0, d)$ be a pointed metric space, $(V, \|\cdot\|_V)$ a normed space and $\Psi : \mathcal{X}_0 \to V$ a dense isometric embedding. Then V' is isomorphic to a closed subspace $F \subset \operatorname{Lip}_0(\mathcal{X}_0)$, and the canonical injection $i : F \to \operatorname{Lip}_0(\mathcal{X}_0)$ satisfies $\|i\| \leq 1$.*

Proof Recall the notation $m_x := \Phi(x)$ from Section 3 and analogously denote $v_x := \Psi(x)$. Let $T : AE(\mathcal{X}_0) \to V$ the linear mapping with $T \circ \Phi = \Psi$ as in Definition 21. As Ψ is an isometry, T satisfies $\|T\| = 1$, and maps $AE(\mathcal{X}_0)$ on some dense subspace of V. Consider the adjoint $T' : V' \to AE(\mathcal{X}_0)'$. It is well known (e.g., Chapter 4 of Rudin, 1991) that $\|T\| = \|T'\|$ and that T' is injective iff the range of T is dense. Thus, in our case T' is injective. As by construction also $\langle T m_x, v' \rangle = \langle T' v', m_x \rangle$, we have a unique correspondence between the linear functions in V' and some subspace $F := T'V' \subset AE(\mathcal{X}_0)'$: for $g \in V'$ and $f = T'g \in \operatorname{Lip}_0(\mathcal{X}_0)$ we have $g(v_x) = f(m_x)$ for every $x \in \mathcal{X}_0$. The canonical inclusion i corresponds to the adjoint T'. ■

Lemma 24 shows that the hypothesis space V' constructed by embedding $\mathcal{X}$ into V is isomorphic to a subset $F \subset \operatorname{Lip}_0(\mathcal{X}_0)$. But it is important to note that this isomorphism is not isometric in general. Let $g \in V'$ and $f \in \operatorname{Lip}_0(\mathcal{X}_0)$ be corresponding functions, that is $f = T'g$. Because of $\|T'\| = 1$ we know that $\|f\|_{AE'} \leq \|g\|_{V'}$, but in general we do not have equality. This means that the margins $\|g\|_{V'}$ and $\|f\|_{AE'}$ of corresponding functions are measured with respect to different norms and might have

different sizes. As a consequence, the solutions of the two large margin problems

$$\min_{g\in V'} \|g\|_{V'} \text{ subject to } y_i g(v_{x_i}) \geq 1$$

and

$$\min_{f\in F} \|f\|_L \text{ subject to } y_i f(x_i) \geq 1$$

might be different, even though the sets of feasible functions are the same in both cases.

To illustrate this we will consider two examples. The first one shows how the large margin classifier in V can give different results than the one constructed by using the corresponding subspace for the Lipschitz classifier. In the second example we show a situation where both classifiers coincide.

Example 6 (Kuratowski embedding) *Let $(\mathcal{X},d)$ be an arbitrary compact metric space and $(\mathcal{C}(\mathcal{X}),\|\cdot\|_\infty)$ the space of continuous functions on $\mathcal{X}$. Define $\Psi : \mathcal{X} \to \mathcal{C}(\mathcal{X})$, $x \mapsto d(x,\cdot)$. This mapping is an isometric embedding called Kuratowski embedding, and it has been used in Hein and Bousquet (2003) to construct a large margin classifier. We want to compare the large margin classifiers resulting from the Kuratowski embedding and the embedding in the Arens-Eells space. As an example consider the finite metric space $\mathcal{X} = \{x_1,...,x_4\}$ with distance matrix*

$$D = \begin{pmatrix} 0 & 5 & 3 & 6 \\ 5 & 0 & 4 & 1 \\ 3 & 4 & 0 & 5 \\ 6 & 1 & 5 & 0 \end{pmatrix}.$$

Let $V = \text{span}\{d(x,\cdot);\, x \in \mathcal{X}\} \subset \mathcal{C}(\mathcal{X})$, endowed with the norm $\|\cdot\|_\infty$. V is a 4-dimensional vector space. Let V' its dual space. Via the mapping T', each linear operator $g \in V'$ corresponds to the linear operator $f \in \text{Lip}_0(\mathcal{X}_0)$ with $f(x_i) = \langle g, d(x_i,\cdot)\rangle =: c_i$. Now we want to compare the norms of g in V' and f in $\text{Lip}(\mathcal{X})$.The norm of g in V' can be computed as follows:

$$\begin{aligned}
\|g\|_{V'} &= \sup\{\langle g, v\rangle :\ v \in V, \|v\|_V \leq 1\} \\
&= \sup\{\langle g, \sum_{i=1}^4 a_i d(x_i,\cdot)\rangle :\ a_i \in \mathbb{R}, \|\sum_{i=1}^4 a_i d(x_i,\cdot)\|_\infty \leq 1\} \\
&= \sup\{\sum_{i=1}^4 a_i c_i :\ a_i \in \mathbb{R}, -1 \leq \sum_{i=1}^4 a_i d(x_i,x_j) \leq 1 \text{ for all } j = 1,...,4\}.
\end{aligned}$$

For given function $g \in V'$ (that is, for given values c_i) this norm can be computed by a linear program. Consider the two functions $g_1, g_2 \in V'$ with values on x_1,x_2,x_3,x_4 given as $(-1,-1,-1,-1)$ and $(1,0,1,0)$, respectively, and let $f_1, f_2 \in \text{Lip}_0(\mathcal{X}_0)$ be the corresponding Lipschitz functions. Then we have $\|f_1\|_L = 0.166 < 0.25 = \|f_2\|_L$ and $\|g_1\|_{V'} = 0.366 > 0.28 = \|g_2\|_{V'}$. So the norms $\|\cdot\|_{V'}$ and $\|\cdot\|_L$ do not coincide, and moreover there is no monotonic relationship between them. If the maximal margin algorithm had to choose between functions f_1 and f_2, it would come to different solutions, depending whether the underlying norm is $\|\cdot\|_{V'}$ as for the large margin classifier in V' or $\|\cdot\|_L$ as for the Lipschitz classifier in $T'V'$.

Example 7 (Normed space) *Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ be a normed vector space with dual $(\mathcal{X}', \|\cdot\|_{\mathcal{X}'})$. As the norm of linear functions coincides with their Lipschitz constant, $\mathcal{X}'$ is* isometrically *isomorphic to a subspace of* $\mathrm{Lip}_0(\mathcal{X}_0)$. *This means that it makes no difference whether we construct a large margin classifier on the normed space $\mathcal{X}$ directly or ignore the fact that $\mathcal{X}$ is a normed space, embed $\mathcal{X}$ into $AE(\mathcal{X}_0)$ and then construct the Lipschitz classifier on $AE(\mathcal{X}_0)$ with the subspace $T'\mathcal{X}'$. We already mentioned this fact in Section 3 when we stated that the SVM solution is the same one as the Lipschitz classifier on $\mathcal{X}'$.*

Now we want to investigate our second question: given some subspace $F \subset \mathrm{Lip}_0(\mathcal{X}_0)$, is F the dual space of some Banach space V such that $\mathcal{X}_0$ can be embedded isometrically into V and $V' \simeq F$? To answer this question we have to deal with some technical problems. First of all, F has to possess a *pre-dual*, that is a vector space V whose dual V' coincides with F. In general, not every Banach space possesses a pre-dual, and if it exists, it needs not be unique. Secondly, it turns out that the canonical injection $T' : F \to \mathrm{Lip}_0(\mathcal{X}_0)$ has to have a *pre-adjoint*, that is a mapping $T : AE(\mathcal{X}_0) \to V$ whose adjoint coincides with T'. Pre-adjoints also not always exist. In general, neither the existence of a pre-dual nor the existence of pre-adjoints are easy to prove. One situation where both can be handled is the case where F is closed under pointwise convergence:

Lemma 25 (Construction of V for given F) *Let $\mathcal{X}_0$ be a bounded metric space, and F a subspace of $(\mathrm{Lip}_0(\mathcal{X}_0), L(\cdot))$ which is closed under pointwise convergence and satisfies the condition*

$$\sup_{f \in F, L(f) \leq 1} |f(x) - f(y)| = d(x,y) \tag{15}$$

for all $x, y \in \mathcal{X}_0$. Then there exists a normed space V such that $\mathcal{X}_0$ can be isometrically embedded into V and its dual V' is isometrically isomorphic to F.

Before we can start with the proof we need two more definitions: Let M be a subspace of some Banach space V and N a subspace of the dual space V'. Then the annihilator $M^{\perp}$ and the pre-annihilator ${}^{\perp}N$ are defined as $M^{\perp} = \{T \in V';\ Tm = 0 \text{ for all } m \in M\}$ and ${}^{\perp}N = \{e \in V;\ Te = 0 \text{ for all } T \in N\}$. As the proof is a bit technical, we refer to Megginson (1998) for background reading.

Proof For a bounded metric space $\mathcal{X}_0$, the topology of pointwise convergence on $\mathrm{Lip}_0(\mathcal{X}_0)$ coincides with its weak* topology. Thus by assumption, F is weak*-closed, which implies that ${}^{\perp}F$ is a closed subspace of $AE(\mathcal{X}_0)$. Hence, the quotient space $V := AE(\mathcal{X}_0)/{}^{\perp}F$ exists, and there exists an isometric isomorphism between V' and $({}^{\perp}F)^{\perp}$. As F is weak*-closed, $({}^{\perp}F)^{\perp} = F$. So V is a pre-dual of F. Let $T' : F \to \mathrm{Lip}_0(\mathcal{X}_0)$ be the canonical inclusion. It has a pre-adjoint, namely the quotient mapping $\pi : AE(\mathcal{X}_0) \to V$. Define the mapping $\Psi : \mathcal{X}_0 \to V$, $x \mapsto \pi m_x =: v_x$. We have

$$\langle f, v_x \rangle = \langle f, \pi m_x \rangle = \langle T' f, m_x \rangle = \langle f, m_x \rangle = f(x).$$

Hence, by assumption (15), Ψ is an isometry:

$$\|\Psi(x) - \Psi(y)\|_V = \sup_{f \in F, L(f) \leq 1} \{|\langle f, v_x - v_y \rangle|\} = \sup_{f \in F, L(f) \leq 1} \{|f(x) - f(y)|\} = d(x,y).$$

■

Lemma 25 gives a nice interpretation of what it means geometrically to choose a subspace F of Lipschitz functions: the Lipschitz classifier with hypothesis space F corresponds to embedding $\mathcal{X}$ isometrically into the pre-dual V of F and constructing the large margin classifier on V directly. Condition (15), which F has to satisfy to allow this interpretation, intuitively means that F has to be a "reasonably large" subspace.

Example 8 (Linear combination of distance functions) *Let F be the subspace of $\mathrm{Lip}(\mathcal{X})$ consisting of functions of the form $f(x) = \sum_i a_i d(x_i, x) + b$, and $\bar{F} \subset \mathrm{Lip}(\mathcal{X})$ its closure under pointwise convergence. As norm on $\bar{F}$ we take the Lipschitz constant. On $\bar{F}$, condition* (15) *is satisfied: trivially, we always have $\leq$ in* (15)*, and for given $x, y \in \mathcal{X}$, equality is reached for the function $f = d(x, \cdot)$. So we can conclude by Lemma 25 that the Lipschitz classifier on $\bar{F}$ has the geometrical interpretation explained above.*

7. Discussion

We derived a general approach to large margin classification on metric spaces which uses Lipschitz functions as decision functions. Although the Lipschitz algorithm, which implements this approach, has been derived in a rather abstract mathematical framework, it boils down to an intuitively plausible mechanism: it looks for a decision function which has a small Lipschitz constant. This agrees with the regularization principle that tries to avoid choosing functions with a high variation. The solution of the Lipschitz algorithm is well behaved as, by the representer theorems of Section 4, it can always be expressed by distance functions to training points. For some special cases, the solution corresponds to solutions of other well known algorithms, such as the support vector machine, the linear programming machine, or the 1-nearest neighbor classifier. We provide Rademacher complexity bounds for some of the involved function classes which can be used to bound the generalization error of the classifier.

In spite of all those nice properties there are several important questions which remain unanswered. To apply the Lipschitz algorithm in practice it is important to choose a suitable subspace of Lipschitz functions as hypothesis space. In Section 6 we found a geometrical explanation of what the choice of certain subspaces F means: it is equivalent to using a different isometric embedding of the metric space into some Banach space. But this explanation does not solve the question of which subspace we should choose in the end. Moreover, there exist isometric embeddings in certain Banach spaces which have no such interpretation in terms of subspaces of Lipschitz functions. For example, Hein and Bousquet (2003) studied the Kuratowski embedding of a metric space into its space of continuous functions to construct a large margin algorithm. As we explained in Example 6, the large margin classifier resulting from this embedding can be different from the Lipschitz classifier. It is an interesting question how different embeddings into different Banach spaces should be compared. One way to do this could be comparing the capacities of the induced function spaces. An interesting question in this context is to find the "smallest space" (for instance, in terms of the Rademacher complexities) in which a given data space can be embedded isometrically.

There is also a more practical problem connected to the choice of the subspace of Lipschitz functions. To implement the Lipschitz algorithm for a given subspace of Lipschitz functions, we

need to know some way to efficiently compute the Lipschitz constants of the functions in the chosen subspace. For example, in case of the linear programming machine it was possible to bound the Lipschitz constants of the functions in the parameterized subspace of functions $\sum_i a_i d(x_i, \cdot) + b$ in terms of their parameters by $\sum_i |a_i|$. But in many cases, there is no obvious parametric representation of the Lipschitz constant of a class of functions. Then it is not clear how the task of minimizing the Lipschitz constant can be efficiently implemented.

An even more heretic question is whether isometric embeddings should be used at all. In our approach we adopted the point of view that a meaningful distance function between the training points is given by some external knowledge, and that we are not allowed to question it. But in practical applications it is often the case that distances are estimated by some heuristic procedure which might not give a sensible result for all the training points. In those cases the paradigm of isometric embedding might be too strong. Instead we could look for bi-Lipschitz embeddings or low distortion embeddings of the metric space into some Banach space, or even into some Hilbert space. We would then loose some (hopefully unimportant) information on the distances in the metric space, but the gain might consist in a simpler structure of the classification problem in the target space.

Finally, many people argue that for classification only "local properties" should be considered. One example is the assumption that the data lies on some low dimensional manifold in a higher dimensional space. In this case, the meaningful information consists of the intrinsic distances between points along the manifold. In small neighborhoods, those distances are close to the distances measured in the enclosing space, but for points which are far away from each other this is not true any more. In this setting it is not very useful to perform an isometric embedding of the metric space into a Banach space as the additional linear structure the Banach space imposes on the training data might be more misleading than helpful. Here a different approach has to be taken, but it is not clear how a large margin algorithm in this setting can be constructed, or even whether in this case the large margin paradigm should be applied at all.

Acknowledgments

We would like to thank Matthias Hein and Bernhard Schölkopf for helpful discussions.

Appendix A. Proof of Theorem 16

The idea of the proof of Theorem 16 is the following. Instead of bounding the Rademacher complexity on the whole set of functions $\mathcal{F}$, we first consider a maximal ε-separating subset $\mathcal{F}_\varepsilon$ of $\mathcal{F}$. This is a maximal subset such that all its points have distance at least ε to each other. To this special set we will apply the classical entropy bound of Dudley (1987):

Theorem 26 (Classical entropy bound) *For every class $\mathcal{F}$ of functions there exists a constant C such that*

$$\hat{R}_n(\mathcal{F}) \leq \frac{C}{\sqrt{n}} \int_0^\infty \sqrt{\log N(\mathcal{F}, u, L_2(\mu_n))}\, du$$

where μ_n is the empirical distribution of the sample.

As a second step we then bound the error we make by computing the Rademacher complexity of $\mathcal{F}_\varepsilon$ instead of $\mathcal{F}$. This will lead to the additional offset of 2ε in Theorem 16. The following lemma can be found as Lemma 3.10 in Bousquet (2002) (for the definition of a separable process see also van der Vaart and Wellner 1996).

Lemma 27 (ε-separations of an empirical process) *Let $\{Z_t; t \in T\}$ be a separable stochastic process satisfying for $\lambda > 0$ the increment condition*

$$\forall s,t \in T : E\left(e^{\lambda(Z_t - Z_s)}\right) \leq e^{\lambda^2 c^2 d^2(s,t)/2}.$$

Let $\varepsilon \geq 0$ and $\delta > 0$. If $\varepsilon > 0$, let T_ε denote a maximal ε-separated subset of T and let $T_\varepsilon = T$ otherwise. Then for all t_0,

$$E\left(\sup_{t \in T_\varepsilon, d(t,t_0) \leq \delta} Z_t - Z_{t_0}\right) \leq 4\sqrt{2}c \int_{\varepsilon/4}^{\delta/2} \sqrt{\log N(T,u,d)}du.$$

To apply this lemma to the Rademacher complexity of a function class $\mathcal{F}$, we choose the index set $T = \mathcal{F}$, the fixed index $t_0 = f_0$ for some $f_0 \in \mathcal{F}$, the empirical process $Z_f = \frac{1}{n}\sum \sigma_i f(X_i)$, and $\delta \to \infty$. Note that the Rademacher complexity satisfies the increment condition of Lemma 27 with respect to the $L_2(\mu_n)$–distance with constant $c = \sqrt{n}$. Moreover, observe that $E(\sup_t Z_t - Z_{t_0}) = E(\sup_t Z_t) - E(Z_{t_0})$ and $E(Z_{t_0}) = E(\frac{1}{n}\sum \sigma_i f_0(X_i)) = 0$. Together with the symmetry of the distribution of Z_f we thus get the next lemma:

Lemma 28 (Entropy bound for ε-separations) *Let $(X_i)_{i=1,\ldots,n}$ be iid training points with empirical distribution μ_n, $\mathcal{F}$ an arbitrary class of functions, and $\mathcal{F}_\varepsilon$ a maximal ε-separating subset of $\mathcal{F}$ with respect to $L_2(\mu_n)$- norm. Then*

$$E\left(\sup_{f \in \mathcal{F}_\varepsilon} \frac{1}{n}|\sum_i \sigma_i f(X_i)| \middle| X_1, \ldots, X_n\right) \leq \frac{4\sqrt{2}}{\sqrt{n}} \int_{\varepsilon/4}^{\infty} \sqrt{\log N(\mathcal{F}, u, L_2(\mu_n))}\ du.$$

With this lemma we achieved that the integral over the covering numbers starts at $\varepsilon/4$ instead of 0 as it is the case in Theorem 26. The price we pay is that the supremum on the left hand side is taken over the smaller set $\mathcal{F}_\varepsilon$ instead of the whole class $\mathcal{F}$. Our next step is to bound the mistake we make by this procedure.

Lemma 29 *Let $\mathcal{F}$ be a class of functions and $\mathcal{F}_\varepsilon$ a maximal ε-separating subset of $\mathcal{F}$ with respect to $\|\cdot\|_{L_2(\mu_n)}$. Then $|R_n(\mathcal{F}) - R_n(\mathcal{F}_\varepsilon)| \leq 2\varepsilon$.*

Proof We want to bound the expression

$$|R_n(\mathcal{F}) - R_n(\mathcal{F}_\varepsilon)| = E\frac{1}{n}\left|\sup_{f \in \mathcal{F}} |\sum \sigma_i f(X_i)| - \sup_{f \in \mathcal{F}_\varepsilon} |\sum \sigma_i f(X_i)|\right|.$$

First look at the expression inside the expectation, assume that the σ_i and X_i are fixed and that $\sup_{f \in \mathcal{F}} |\sum \sigma_i f(x_i)| = |\sum \sigma_i f^*(x_i)|$ for some function f^* (if f^* does not exist we additionally have to use a limit argument). Let $f_\varepsilon \in \mathcal{F}_\varepsilon$ such that $\|f^* - f_\varepsilon\|_{L_2(\mu_n)} \leq 2\varepsilon$. Then,

$$\begin{aligned}\frac{1}{n}\left|\sup_{f \in \mathcal{F}} |\sum \sigma_i f(x_i)| - \sup_{f \in \mathcal{F}_\varepsilon} |\sum \sigma_i f(x_i)|\right| &\leq \frac{1}{n}\left||\sum \sigma_i f^*(x_i)| - |\sum \sigma_i f_\varepsilon(x_i)|\right| \\ &\leq \frac{1}{n}\left|\sum \sigma_i (f^*(x_i) - f_\varepsilon(x_i))\right| \leq \|f^* - f_\varepsilon\|_{L_1(\mu_n)} \leq \|f^* - f_\varepsilon\|_{L_2(\mu_n)} \leq 2\varepsilon.\end{aligned}$$

As this holds conditioned on all fixed values of σ_i and X_i we get the same for the expectation. This proves the lemma. ■

To prove Theorem 16 we now combine lemmas 28 and 29.

References

R. Arens and J. Eells. On embedding uniform and topological spaces. *Pacific Journal of Mathematics*, 6:397–403, 1956.

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

K. Bennett and E. Bredensteiner. Duality and geometry in SVM classifiers. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 57–64. Morgan Kaufmann, San Francisco, 2000.

O. Bousquet. *Concentration Inequalities and Empirical Processes Theory Applied to the Analysis of Learning Algorithms*. PhD thesis, Ecole Polytechnique, 2002.

L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer, New York, 2001.

V. Dobric and J. Yukich. Asymptotics for transportation costs in high dimensions. *Journal of Theoretical Probability*, 8(1):97–118, 1995.

R. M. Dudley. Universal Donsker classes and metric entropy. *Annals of Probability*, 15(4):1306–1326, 1987.

T. Graepel, R. Herbrich, B. Schölkopf, A. Smola, P. Bartlett, K. Müller, K. Obermayer, and R. Williamson. Classification of proximity data with LP machines. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, pages 304–309, 1999.

M. Hein and O. Bousquet. Maximal margin classification for metric spaces. In M. Warmuth B. Schölkopf, editor, *Proceedings of the 16th Annual Conference on Computational Learning Theory*, pages 72–86. Springer Verlag, Heidelberg, 2003.

A. N. Kolmogorov and V. M. Tihomirov. ε-entropy and ε-capacity of sets in functional space. *American Mathematical Society Translations (2)*, 17:277–364, 1961.

R. Megginson. *An Introduction to Banach Space Theory*. Springer, New York, 1998.

S. Mendelson and R. Vershynin. Entropy and the combinatorial dimension. *Inventiones Mathematicae*, 152(1):37–55, 2003.

V. Pestov. Free Banach spaces and representations of topological groups. *Functional Analysis and Its Applications*, 20:70–72, 1986.

W. Rudin. *Functional Analysis*. McGraw-Hill Inc., Singapore, 2nd edition, 1991.

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

J. M. Steele. *Probability theory and combinatorial optimization*, volume 69 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

M. Talagrand. The Ajtai-Komlós-Tusnády matching theorem for general measures. In *Probability in Banach spaces, 8 (Brunswick, ME, 1991)*, volume 30 of *Progress in Probability*, pages 39–54. Birkhäuser Boston, MA, 1992.

A. W. van der Vaart and J. A. Wellner. *Weak Convergence and Empirical Processes*. Springer, New York, 1996.

N. Weaver. *Lipschitz algebras*. World Scientific, Singapore, 1999.

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 5 (2004) 697-723 Submitted 4/02; Revised 1/03; Published 6/04

Hierarchical Latent Class Models for Cluster Analysis

Nevin L. Zhang LZHANG@CS.UST.HK
Department of Computer Science
Hong Kong University of Science and Technology
Hong Kong, China

Editor: Craig Boutilier

Abstract

Latent class models are used for cluster analysis of categorical data. Underlying such a model is the assumption that the observed variables are mutually independent given the class variable. A serious problem with the use of latent class models, known as local dependence, is that this assumption is often untrue. In this paper we propose hierarchical latent class models as a framework where the local dependence problem can be addressed in a principled manner. We develop a search-based algorithm for learning hierarchical latent class models from data. The algorithm is evaluated using both synthetic and real-world data.

Keywords: Model-based clustering, latent class models, local dependence, Bayesian networks, latent structure discovery

1. Introduction

Cluster analysis is the partitioning of similar objects into meaningful classes, when both the number of classes and the composition of the classes are to be determined (Kaufman and Rousseeuw 1990; Everitt 1993). In model-based clustering, it is assumed that the objects under study are generated by a mixture of probability distributions, with one component corresponding to each class. When the attributes of objects are continuous, cluster analysis is sometimes called *latent profile analysis* (Gibson, 1959; Lazarsfeld and Henry, 1968; Bartholomew and Knott, 1999; Vermunt and Magidson, 2002). When the attributes are categorical, cluster analysis is sometimes called *latent class analysis (LCA)* (Lazarsfeld and Henry, 1968; Goodman, 1974b; Bartholomew and Knott, 1999; Uebersax, 2001). There is also cluster analysis of *mixed-mode data* (Everitt, 1993) where some attributes are continuous while others are categorical.

This paper is concerned with LCA, where data are assumed to be generated by a *latent class (LC)* model. An LC model consists of a class variable that represents the clusters to be identified and a number of other variables that represent attributes of objects.[1] The class variable is not observed and hence said to be *latent*. On the other hand, the attributes are observed and are called *manifest variables*.

LC models assume *local independence*, i.e., manifest variables are mutually independent in each latent class, or equivalently, given the latent variable. A serious problem with the use of LCA, known as *local dependence*, is that this assumption is often violated. If one does not deal with local

1. Latent class models are sometimes also referred to as naïve Bayes models. We suggest that the term "naïve Bayes models" be used only in the context of classification and the term "latent class models" be used in the context of clustering.

dependence explicitly, one implicitly attributes it to the latent variable. This can lead to spurious latent classes and poor model fit. It can also degenerate the accuracy of classification because locally dependent manifest variables contain overlapping information (Vermunt and Magidson, 2002).

The local dependence problem has attracted some attention in the LCA literature (Espeland and Handelman, 1989; Garrett and Zeger, 2000; Hagenaars, 1988; Vermunt and Magidson, 2000). Methods for detecting and modeling local dependence have been proposed. To detect local dependence, one typically compares observed and expected cross-classification frequencies for pairs of manifest variables. To model local dependence, one can join manifest variables, introduce multiple latent variables, or reformulate LC models as loglinear models and then impose constraints on them. All existing methods are preliminary proposals and suffer from a number of deficiencies (Section 2).

1.1 Our Work

This paper describes the first systematic approach to the problem of local dependence. We address the problem in the framework of *hierarchical latent class (HLC) models*. HLC models are Bayesian networks whose structures are rooted trees and where the leaf nodes are observed while all other nodes are latent. This class of models is chosen for two reasons. First it is significantly larger than the class of LC models and can accommodate local dependence. Second inference in an HLC model takes time linear in model size, which makes it computationally feasible to run EM.

We develop a search-based algorithm for learning HLC models from data. The algorithm systematically searches for the optimal model by hill-climbing in a space of HLC models with the guidance of a model selection criterion. When there is no local dependence, the algorithm returns an LC model. When local dependence is present, it returns an HLC model where local dependence is appropriately modeled. It should be noted, however, that the algorithm might not work well on data generated by models that neither are HLC models nor can be closely approximated by HLC models.

The motivation for this work originates from an application in traditional Chinese medicine. In that application, there are approximately seventy manifest variables and local dependence is an important issue. The aim is to learn a statistical model from data and hence provide doctors with an objective picture about the structure of the application domain.[2] As such, model quality is of utmost importance, while it is reasonable to assume abundant data and computing resources. So we take a principled (as opposed to heuristic) approach when designing our algorithm and we empirically show that the algorithm yields models of good quality. In subsequent work, we will explore ways to scale up the algorithm.

1.2 Related Literature

This paper is an addition to the growing literature on hidden variable discovery in Bayesian networks (BN). Here is a brief discussion of some of this literature. Elidan *et al.* (2001) discuss how to introduce latent variables to BNs constructed for observed variables by BN structure learning algorithms. The idea is to look for structural signatures of latent variables. Elidan and Friedman (2001) give a fast algorithm for determining the cardinalities — the numbers of possible states — of latent variables introduced this way. Meila-Predoviciu (1999) studies how mixtures of trees can

2. Currently, diagnosis in Chinese medicine is based on theories that have not been scientifically validated.

be induced from data. This work is based on the method of approximating joint probability distributions with dependence trees by Chow and Liu (1968). The new component is a latent variable that specifies how several trees over observed nodes fit into one model.

The algorithms described in Connolly (1993) and Martin and VanLehn (1994) are closely related the algorithm presented in this paper. They all aim at inducing from data a latent structure that explains correlations among observed variables. The algorithm by Martin and VanLehn (1994) builds a two-level Bayesian network where the lower level consists of observed variables while the upper level consists of latent variables. The algorithm is based on tests of association between pairs of observed variables. The algorithm by Connolly (1993) constructs exactly what we call HLC models. Mutual information is used to group variables, a latent variable is introduced for each group, and the cardinality of the latent variable is determined using a technique called conceptual clustering. In comparison with Connolly's method, our method is more principled in the sense that it determines model structure and cardinalities of latent variables using one criterion, namely (some approximation) of the marginal likelihood.

The task of learning HLC models is similar to the reconstruction of phylogenetic trees, which is a major topic in biological sequence analysis (Durbin *et al*., 1998). As a matter of fact, phylogenetic trees are special HLC models where the model structures are binary (bifurcating) trees and all the variables share the same set of possible states. However, phylogenetic trees cannot be directly used for general cluster analysis because the constraints imposed on them. And techniques for phylogenetic tree reconstruction do not necessarily carry over to HLC models. For example, the structural EM algorithm for phylogenetic tree reconstruction by Friedman *et al*., (2002) does not work for HLC models because we do not know, a priori, the number of latent variables and their cardinalities.

HLC models should not be confused with model-based hierarchical clustering (e.g., Hanson *et al*., 1991; Fraley, 1998). In an LC model (or similar models with continuous manifest variables), there is only one latent variable and each state of the variable corresponds to a cluster in data. HLC models generalize LC models by allowing multiple latent variables and hence open up the possibility of multiple clusterings in one model. An HLC model contains a hierarchy of latent variables, with each corresponding to one way to cluster data. In model-based hierarchical clustering, on the other hand, one has a hierarchy of classes. Conceptually there is only one latent variable. Classes at different levels of the hierarchy correspond to states of the variable at different levels of granularity.

1.3 Organization of Paper

The rest of this paper is organized as follows. In the next section we give a brief review of latent class models and survey previous work on local dependence. In Section 3 we formally define HLC models and study a number of theoretical issues related to the task of learning HLC models. A hill-climbing algorithm for inducing regular HLC models from data is described in Section 4. Section 5 reports empirical results on synthetic data and Section 6 discusses experiments with real-world data. Conclusions and remarks about future directions are provided in the final section.

2. Latent Class Models and Local Dependence

A *latent class (LC)* model involves a latent variable X and a number of manifest variables $Y_1, Y_2, \ldots, Y_n$. All the variables are categorical and the relationships among them are described by the simple Bayesian network shown in Figure 1. In applications, the latent variable X represents concepts

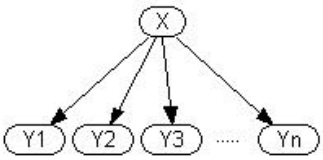


Figure 1: Structure of LC models.

such as "depression" that cannot be directly measured (Eaton *et al.*, 1989). States of the latent variable correspond to classes of individuals in a population. The manifest variables Y_i represent manifestations of the latent concept such as "loss of appetite", "trouble falling asleep", "thoughts of death", and so on.

The latent variable influences all the manifest variables at the same time and hence renders them correlated. The essence of *latent class analysis* (LCA) is to characterize the latent concept by analyzing those correlations. This is possible due to the assumption that the manifest variables are mutually independent given the latent variable, which can be intuitively interpreted as saying that the latent variable is the only reason for the correlations. Since each state of the latent variable corresponds to a class of individuals in a population, the conditional independence assumption can be restated as that the manifest variables are independent within each latent class. Because of this, it is sometimes called the *local independence* assumption.

Learning an LC model from data means to (1) determine the cardinality for variable X, i.e., the number of latent classes; and (2) estimate the model parameters $P(X)$ and $P(Y_i|X)$. Parameters are usually estimated using the EM algorithm (Dempster *et al.*, 1977; Lauritzen, 1995). The cardinality of X is determined by comparing alternatives using goodness-of-fit indices or scoring metrics. The most commonly used scoring metric is BIC (Schwarz, 1978). Equivalent to the MDL score (Lanterman, 2001), the BIC score is an approximation of the marginal likelihood that is derived in a setting when all variables are observed. Geiger *et al.*, (1998) have recently cautioned against its use in LC models. Extensive experiments by Chickering and Heckerman (1997) show that BIC is less accurate than other efficient approximations of marginal likelihood such as the Cheeseman-Stutz (CS) score (Cheeseman and Stutz, 1995).

A serious problem with the use of LCA is that the local independence assumption is often violated. The term *local dependence* is used to refer to this problem. Previous methods for dealing with local independence are surveyed by Uebersax (2000). In this survey, Uebersax distinguishes between two subtasks, namely the diagnosis and modeling of local dependence.

Diagnostic methods compare observed and expected frequencies for pairs of manifest variables. For concreteness, consider two manifest variables A and B in an LC model. Denote the observed and expected frequencies on A and B by $O(A,B)$ and $E(A,B)$ respectively. For any state a of A and b of B, $O(a,b)$ is the number of records where A is in state a and B is in state b.[3] On the other hand, $E(a,b) = P(a,b) * N$, where $P(A,B)$ is the joint probability of A and B in the LC model and N is the total number of records. Hagenaars (1988) suggests that one examine the standardized residuals

$$R(a,b) = \frac{O(a,b) - E(a,b)}{\sqrt{E(a,b)}}$$

3. We use upper case letters for variable names and the corresponding lower case letters for their states.

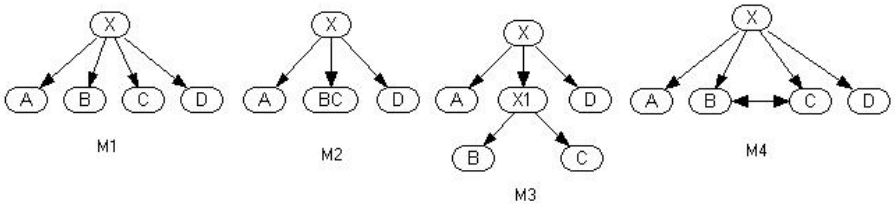


Figure 2: Modeling of local dependence.

for each combination (a,b) of states of A and B. If the residuals deviate from zero significantly, one concludes that A and B are locally dependent. Espeland and Handelman (1988) propose computing the likelihood ratio statistic

$$L(A,B) = \sum_{a,b} 2O(a,b)log\frac{O(a,b)}{E(a,b)}.$$

The larger the statistic, the stronger the evidence for local dependence between A and B. When A and B are binary variables, we denote the possible states for the two variables by a, $\neg a$, b, and $\neg b$. Garret and Zeger (2000) recommend to compare the observed and expected log odds ratio

$$log\frac{O(\neg a,b)/O(a,b)}{O(\neg a,\neg b)/O(a,\neg b)}, log\frac{E(\neg a,b)/E(a,b)}{E(\neg a,\neg b)/E(a,\neg b)}.$$

Again larger differences indicate stronger evidence for local dependence.

An obvious way to model local dependence is to introduce joint variables. Consider the LC model M1 in Figure 2. If variables B and C are locally dependent, we can combine those two variables and introduce a joint variable BC. This lead to the model M2. A second method is to introduce new latent variables (Goodman, 1974a). Uebersax calls it the *multiple indicator method*. To account for the local dependence between B and C in M1, for instance, we can introduce a new latent variable X_1 and thereby get model M3. By doing this, we are assuming that the reason for B and C being locally dependent is that they are jointly influenced by a latent variable X_1 that is not completely determined by the latent variable X. In a third approach (Hagenaars, 1988), one views LC models as special loglinear models. When two manifest variables are locally dependent, one simply adds a direct effect between them. In model M1, adding a direct effect between B and C yields the model M4. Note that M4 is no longer a Bayesian network. It is the path-diagram (see Bohrnstedt and Knoke, 1994, Chapter 11) for a loglinear model.

Previous work in the LCA community for dealing with local dependence is not sufficient for a number of reasons. First, the criteria for detecting local dependence is heuristic in nature. Judgments are required as to how the various thresholds should be set. Second, there are no criteria for making the trade-off between increasing the cardinalities of existing latent variables versus increasing the complexity of model structure. In Hagenaars (1988) and Uebersax (2000), cardinalities of all latent variables are fixed at 2 while model structures are allowed to change. In most other work the standard one-latent-variable structure is assumed and fixed, while the cardinality of the latent variable is allowed to change. Third, the search for the best model is carried out manually. Typically only a few simple models are considered (Goodman, 1974a; Hagenaars, 1988). The search space

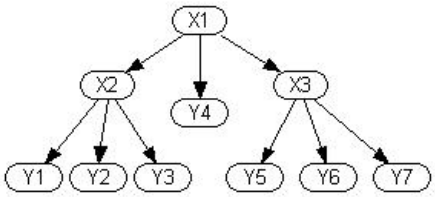


Figure 3: An example HLC model. The X_i's are latent variables and the Y_j's are manifest variables.

for the multiple indicator method is not even clearly defined. Finally, when there are multiple pairs of locally dependent manifest variables, it is not clear which pair should be tackled first, or if all pairs should be handled simultaneously.

The purpose of this paper is to develop a principled and systematic method for dealing with local dependence. In the next section, we describe the models that serve as the framework for our work.

3. Hierarchical Latent Class Models

A *hierarchical latent class (HLC) model* is a Bayesian network where

1. The network structure is a rooted tree; and
2. The variables at the leaf nodes are observed and all the other variables are not.[4]

Figure 3 shows an example of an HLC model. Following the LCA literature, we refer to the observed variables as *manifest variables* and all the other variables as *latent variables*. In this paper we do not distinguish between variables and nodes. So we sometimes speak also of *manifest nodes* and *latent nodes*. For technical convenience, we assume that there are at least two manifest variables.

We use θ to refer to the collection of parameters in an HLC model M and use m to refer to what is left when the parameters are removed from M. So we usually write an HLC model as a pair $M = (m, \theta)$. We sometimes refer to the first component m of the pair also as an HLC model. When it is necessary to distinguish between m and the pair (m, θ), we call m an *uninstantiated HLC model* and the pair an *instantiated HLC model*. The term *HLC model structure* is reserved for what is left if information about cardinalities of latent variables are removed from an uninstantiated model m. Model structures will be denoted by the letter S, possibly with subscripts.

3.1 Parsimonious HLC Models

In this paper we study the learning of HLC models. We assume that there is a collection of identical and independently distributed (i.i.d.) samples generated by some HLC model. Each sample consists of states for all or some of the manifest variables. The task is to reconstruct the HLC model from data. As will be seen later, not all HLC models can be reconstructed from data. It is hence natural

4. The concept of a variable being observed is always w.r.t some given data set. A variable is *observed* in a data set if there is at least one record that contains the state for that variable.

to ask what models can be reconstructed. In this subsection we provide a partial answer to this question.

Consider two instantiated HLC models $M = (m, \theta)$ and $M' = (m', \theta')$ that share the same manifest variables Y_1, Y_2, ..., Y_n. We say that M and M' are *marginally equivalent* if the probability distribution over the manifest variables is the same in both models, i.e.,

$$P(Y_1, \ldots, Y_n | m, \theta) = P(Y_1, \ldots, Y_n | m', \theta'). \quad (1)$$

Two marginally equivalent instantiated models are *equivalent* if they also have the same number of independent parameters. Two uninstantiated HLC models m and m' are *equivalent* if for any parameterization θ of m there exists a parameterization θ' of m' such that (m, θ) and (m, θ') are equivalent and vice versa. Two HLC model structures S_1 and S_2 are *equivalent* if there are equivalent uninstantiated models m_1 and m_2 whose underlying structures are S_1 and S_2 respectively.

An instantiated HLC model M is *parsimonious* if there does not exist another model M' that is marginally equivalent to M and that has fewer independent parameters than M. An uninstantiated HLC model m is *parsimonious* if there exists a parameterization θ of m such that (m, θ) is parsimonious.

Let M be an instantiated HLC model and D be a set of i.i.d. samples generated by M. If M is not parsimonious, then there must exist another HLC model whose penalized loglikelihood score given D (Green, 1998; Lanternman, 2001) is greater than that of M. This means that, if one uses penalized loglikelihood for model selection, one would prefer this other parsimonious models over the non-parsimonious model M. The following theorem states that, to some extent, the opposite is also true, i.e., one would prefer M to other models if M is parsimonious.

Theorem 1 *Let M and M' be two instantiated HLC models with the same manifest variables. Let D be a set of i.i.d. samples generated from M.*

1. *If M and M' are not marginally equivalent, then the loglikelihood $l(M|D)$ of M is strictly greater than the loglikelihood $l(M'|D)$ of M' when the sample size is large enough.*

2. *If M is parsimonious and is not equivalent to M', then the penalized loglikelihood of M is strictly larger than that of M' when the sample size is large enough.*

Proof: Use P and P' to denote the marginal probability distributions over the manifest variables in M and M' respectively. Let N be the sample size. It follows from the law of large numbers that, as N goes to infinity, $[l(M|D) - l(M'|D)]/N$ approaches the Kullback-Leibler (KL) distance $I(P:P')$. The first part hence follows from the well-known property of the KL distance that $I(P:P') \geq 0$ and the equality is true only when P and P' are identical (e.g., Cover and Thomas, 1991).

The second part can be divided into two cases. The first case is when M and M' are marginally equivalent and M' has more parameters than M. Here the statement is trivially true for all sample sizes. In the second case, M and M' are not marginally equivalent. According to the first part of the theorem, $l(M|D) - l(M'|D)$ is positive when N is large enough. Moreover the quantity increases linearly with N. On the other hand, the penalty on model complexity increases logarithmically with N. Hence the statement is true when N is large enough. Q.E.D.

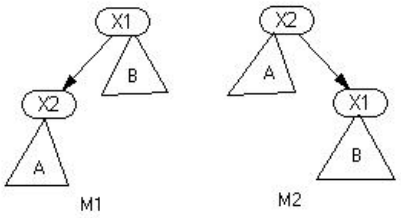


Figure 4: The operation of root walking.

3.2 Model Equivalence

In this subsection we give an operational characterization of model equivalence. Let X_1 be the root of an instantiated HLC model M_1. Suppose X_2 is a child of X_1 and it is a latent node (see Figure 4). Define another HLC model M_2 by reversing the arrow $X_1 \rightarrow X_2$ and, while leaving the values for all other parameters unchanged, defining $P_{M_2}(X_2)$ and $P_{M_2}(X_1|X_2)$ as follows:

$$\begin{aligned} P_{M_2}(X_2) &= \sum_{X_1} P_{M_1}(X_1) P_{M_1}(X_2|X_1) \\ P_{M_2}(X_1|X_2) &= \begin{cases} \frac{P_{M_1}(X_1)P_{M_1}(X_2|X_1)}{P_{M_2}(X_2)} & \text{if } P_{M_2}(X_2) > 0. \\ \frac{1}{|X_1|} & \text{otherwise.} \end{cases} \end{aligned}$$

We use the term *root walking* to refer to the process of obtaining M_2 from M_1. In the process, the root has walked from X_1 to X_2.

Theorem 2 *Let M_1 and M_2 be two instantiated HLC models. If M_2 is obtained from M_1 by one or more steps of root walking, then M_1 and M_2 are equivalent.*[5]

Proof: We will prove this theorem for the case when M_2 is obtained from M_1 by one step of root walking. The general case follows from this special case by induction.

Assume the models are as shown in Figure 4. Model M_2 is obtained by letting the root of M_1 walk from X_1 to X_2. Let A be the set of variables in the subtrees rooted at X_2 except those in the subtree rooted at X_1 and let B be the set of variables in the subtrees rooted at X_1 except those in the subtree rooted at X_2. We have,

$$\begin{aligned} P_{M_1}(X_1, X_2, A, B) &= P_{M_1}(X_1) P_{M_1}(X_2|X_1) P_{M_1}(A|X_2) P_{M_1}(B|X_1) \\ &= P_{M_2}(X_2) P_{M_2}(X_1|X_2) P_{M_2}(A|X_2) P_{M_2}(B|X_1) \\ &= P_{M_2}(X_1, X_2, A, B). \end{aligned}$$

Consequently, M_1 and M_2 are marginally equivalent.

It is easy to see that $P_{M_1}(X_1)$ and $P_{M_1}(X_2|X_1)$ encapsulate $|X_1||X_2|-1$ parameters. The same is true for $P_{M_2}(X_2)$ and $P_{M_2}(X_1|X_2)$. Hence M_1 and M_2 have the same number of parameters. The theorem is therefore proved. Q.E.D.

5. A similar but different theorem was proved by Chickering (1996) for Bayesian networks with no latent variables. In Chickering (1996), model equivalence implies equal number of parameters. Here equal number of parameters is part of the definition of model equivalence.

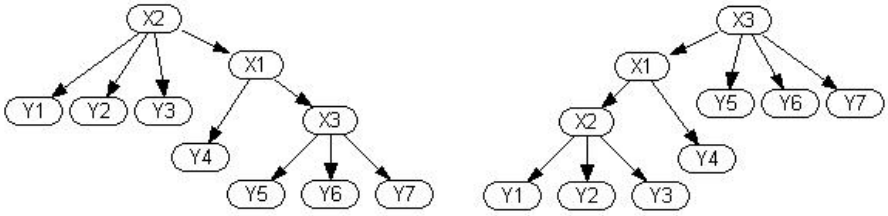


Figure 5: HLC models that are equivalent to the one in Figure 3.

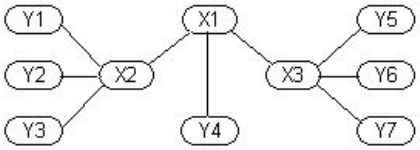


Figure 6: The unrooted HLC model that corresponds to the HLC model in Figure 3.

The two HLC models shown in Figure 5 are equivalent to the model in Figure 3. The model on the left is obtained by letting the root of the original model walk from X_1 to X_2, while the model on the right is obtained by letting the root walk from X_1 to X_3.

In general, the root of an HLC model can walk to any latent node. This implies the root node cannot be determined from data.[6] A question about the suitability of HLC models for cluster analysis naturally arises. We take the position that the root node can be determined from the objective in clustering and domain knowledge. Moreover we view the presence of multiple latent variables as an advantage because it enables one to cluster data in multiple ways. Each latent variable represents one possible way to cluster data. Note that multiple clusterings due to multiple latent variables are very different from multiple clusterings in hierarchical clustering. In the latter case, a clustering at a lower level of the hierarchy is a refinement of a clustering at a higher level. The same relationship does not exist in the former case.

The inability of determining the root node from data also has some technical consequences. We can never induce HLC models from data. Instead we obtain what might be called unrooted HLC models. An *unrooted HLC model* is an HLC model with all directions on the edges dropped. Figure 6 shows the unrooted HLC model that corresponds to the HLC model in Figure 3. An unrooted HLC model represents a class of HLC models; members of the class are obtained by rooting the model at various latent nodes and by directing the edges away from the root. Semantically it is a Markov random field on an undirected tree. The leaf nodes are observed while the interior nodes are latent. The concepts of marginal equivalence, equivalence, and parsimony can be defined for unrooted HLC models in the same way as for rooted models.

From now on when we speak of HLC models we always mean unrooted HLC models unless it is explicitly stated otherwise.

6. In the case of phylogenetic trees, this is a well-known fact (Durbin *et al*., 1998).

3.3 Regular HLC Models

In this subsection we first introduce the concept of regular HLC models and show that all parsimonious models must regular. We then show that the set of uninstantiated regular HLC models for a given set of manifest variables is finite. This provides a search space for the learning algorithm to be developed in the next section.

For any variable X, use Ω_X and $|X|$ to denote its domain and cardinality respectively. For a latent variable Z in an HLC model, enumerate its neighbors as X_1, X_2, ..., X_k. An HLC model is *regular* if

1. It consists of at least two manifest variables; and
2. For any latent variable Z,
   (a) If Z has only two neighbors, then one of the two neighbors must be a latent node and
   $$|Z| < \frac{|X_1||X_2|}{\max\{|X_1|,|X_2|\}} \tag{2}$$
   (b) If Z has more than two neighbors, then
   $$|Z| \leq \frac{\prod_{i=1}^{k}|X_i|}{\max_{i=1}^{k}|X_i|}. \tag{3}$$

Note that this definition applies to instantiated as well as uninstantiated models.

Theorem 3 *Let M be an instantiated HLC model. If M is irregular, then there exists another model M' that is marginally equivalent to and has fewer parameters than M.*

Proof: Parameters of a rooted HLC model include the prior probability distribution of the root and the conditional probability distribution of each of the non-root nodes given its parent. On the other hand, parameters of an unrooted HLC model include a potential for each edge in the model structure. The potential is a function of the two variables connected by the edge. Referring to the definition of regularity, let $f(Z,X_1,\ldots,X_k)$ be the multiplication of all potentials for edges between Z and its neighbors and let $g(X_1,\ldots,X_k)=\sum_Z f(Z,X_1,\ldots,X_k)$.

Consider the case when inequality (3) is violated, i.e., $|Z| > \prod_{i=1}^{k}|X_i|/\max_{i=1}^{k}|X_i|$. Without loss of generality, suppose $|X_k|=\max_{i=1}^{k}|X_i|$. Let M' be the same as M except that the domain of Z is redefined to be $\prod_{i=1}^{k-1}\Omega_{X_i}$. An state of Z can be written as $<z_1,\ldots,z_{k-1}>$, where z_i is a state of X_i. For each $i=1,\ldots,k-1$, set the potential $f_i(Z,X_i)$ for the edge between X_i and Z as follows:

$$f_i(<z_1,\ldots,z_{k-1}>,x_i) = \begin{cases} 1 & \text{if } z_i = x_i \\ 0 & \text{if } z_i \neq x_i. \end{cases}$$

Set the potential $f_k(Z,X_k)$ for the edge between X_k and Z as follows:

$$f_i(<z_1,\ldots,z_{k-1}>,x_k) = g(z_1,\ldots,z_{k-1},x_k).$$

Then

$$\sum_Z \prod_{i=1}^{k} f_i(Z,X_i) = g(X_1,\ldots,X_k).$$

Hence M' is marginally equivalent to M. Because Z has fewer states in M' than in M, M' has fewer parameters than M. Therefore M is not parsimonious.

Now consider the case when inequality (2) is violated. In this case, the latent variable Z has two neighbors X_1 and X_2, one of which being a latent node, such that

$$|Z| \geq \frac{|X_1||X_2|}{\max\{|X_1|,|X_2|\}} \tag{4}$$

We assume that X_1 is a latent node. Let M' be the model obtained by eliminating Z from M.[7] Then M' is marginally equivalent to M. To calculate the difference in the number of independent parameters between M' and M, imagine rooting both models at X_1. Then it is easy to see that the difference is

$$|X_1|(|X_2|-1)-[|X_1|(|Z|-1)+|Z|(|X_2|-1)] = |X_1||X_2|-|Z|(|X_1|+|X_2|-1).$$

This quantity is negative because of inequality (4) and of the fact that both X_1 and X_2 have more than one state. Hence M' has fewer parameters than M. Therefore M is not parsimonious. The theorem is proved. Q.E.D

Corollary 1 *Parsimonious HLC models must be regular.*

Theorem 4 *The set of all regular uninstantiated HLC models for a given set of manifest variables is finite.*

Before proving this theorem, we need to introduce several lemmas, which are interesting in their own right. A latent node in an HLC model has at least two neighbors. A *singly connected* latent node is one that has exactly two neighbors.

Lemma 1 *In a regular HLC model, no two singly connected latent nodes can be neighbors.*

Proof: We know from 2 that the cardinality of a singly connected node is strictly smaller than those of its two neighbors. If two singly connected latent nodes Z_1 and Z_2 were neighbors, then we would have both $|Z_1|>|Z_2|$ and $|Z_1|<|Z_2|$. Therefore two singly connected latent nodes cannot be neighbors. Q.E.D.

This lemma inspires the following two definitions. We say that an HLC model structure is *regular* if no two singly connected latent nodes are neighbors. If there are no singly connected latent nodes at all, we say that the model structure is *strictly regular*.

Lemma 2 *Let S be an HLC model structure with n manifest variables. If S is regular, then there are fewer than* $3n$ *latent nodes. If S is strictly regular, then there are fewer than n latent nodes.*

Proof: We prove the second part first.[8] Let h be the number of latent nodes. Then the total number of nodes is $n+h$. Hence the number of edges is $n+h-1$.

7. This means to (1) remove Z and connect X_1 and X_2; and (2) set the potential for the new edge between X_1 and X_2 to be $\sum_Z f_1(X_1,Z)f_2(X_2,Z)$, where f_1 and f_2 are the potentials for the edge between X_1 and Z and the edge between X_2 and Z respectively.
8. This proof is contributed by Tomáš Kočka.

On the other hand, each manifest node appears in exactly one edge and, because of strict regularity, each latent node appears in at least three edges. Because each edge involves exactly two variables, there are at least $(n+3h)/2$ edges. Hence $n+h-1 \geq (n+3h)/2$. Solving this inequality yields $h \leq n-2 < n$.

To prove the first part, let m be the total number of nodes in a regular structure. Imagine that we root the structure at an arbitrary latent node. Then the child of a singly connected latent node is either a manifest node or another latent node that is not singly connected. Moreover, the children for different singly connected nodes are different. So if we eliminate all the singly connected latent nodes, the resulting structure will have at least $m/2$ nodes. The resulting structure is strictly regular. Hence $m/2 < 2n$. This implies that $m < 4n$. Since there are n manifest nodes, the number of latent must be smaller than $3n$. Q.E.D

Lemma 3 *There are fewer than 2^{3n^2} different regular HLC model structures for a given set of n manifest nodes.*

Proof: Let $\mathcal{P}$ be the power set of the set of manifest nodes and let $\mathcal{V}$ be the collection of vectors that consist $3n$ elements of $\mathcal{P}$. Duplicates are allowed in any given vector. Since the cardinality of $\mathcal{P}$ is 2^n, the cardinality of $\mathcal{V}$ is $(2^n)^{3n} = 2^{3n^2}$.

Let $\mathcal{S}$ be the set of all regular HLC model structures for the given manifest nodes. Define a mapping from $\mathcal{S}$ to $\mathcal{V}$ as follows: For any given model structure in $\mathcal{S}$, first root the structure at the parent of the first manifest node. Second, arrange all the latent nodes into a vector according to the depth-first traversal order. According to Lemma 2, the length of the vector cannot exceed $3n$. Third, replace each latent node with the subset of manifest nodes in its subtrees. Finally, add copies of the empty set to the end so that the length of the vector is $3n$. It is not difficult to see that the mapping is bijective. Therefore the cardinality of $\mathcal{S}$ cannot exceed that of $\mathcal{V}$, which is 2^{3n^2}. The theorem is proved. Q.E.D.

Proof of Theorem 4: According to Lemma 3, the number of regular model structures is finite. It is clear from (3), the number of uninstantiated model for a given model structure must also be finite. The theorem is therefore proved. Q.E.D

4. Searching for Optimal Models

In this section we present a hill-climbing algorithm for learning HLC models. Hill-climbing requires a scoring metric for comparing candidate models. In this work we experiment with four existing scoring metrics, namely AIC (Akaike, 1974), BIC (Schwarz, 1978), the Cheeseman-Stutz (CS) score (Cheeseman and Stutz, 1995), and the holdout logarithmic score (LS) (Cowell *et al*., 1999).

Hill-climbing also requires the specification of a search space and search operators. According to Corollary 1, a natural search space for our task is the set of all regular (uninstantiated) HLC models for the set of manifest variables that appear in data. By Theorem 4, we know that this space is finite.

Instead of searching this space directly, we structure the space into two levels according to the following two subtasks and we search those two levels separately:

1. Given a model structure, find optimal cardinalities for the latent variables.

2. Find an optimal model structure.

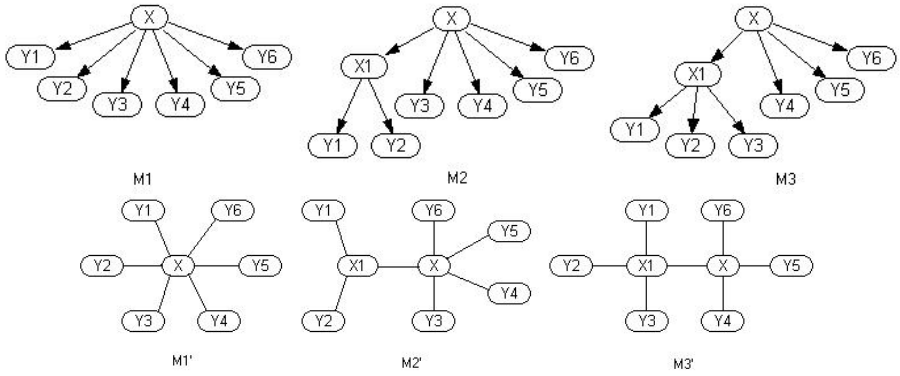


Figure 7: Illustration of structural search operators.

This search space restructuring is motivated by the fact that natural search operators exist for each of the two levels, while operators for the flat space are less obvious.

4.1 Estimating Cardinalities of Latent Variables

The search space for the first subtask consists of all the regular models with the given model structure. To hill-climb in this space we start with the model where the cardinalities of all the latent variables are the minimum. In most cases, the minimum cardinality for a latent variable is 2. For a latent variable next to a singly connected latent node, however, the minimum possible cardinality is 3 because of the inequality (2). At each step, we modify the current model to get a number of new models. The operator for modifying a model is to increase the cardinality of a latent variable by one. Irregular new models are discarded. We then evaluate each of the new models and picks the best one to seed the next search step. To evaluate a model, one needs to estimate its parameters. We use the EM algorithm (Dempster *et al.*, 1977; Lauritzen, 1995) for this task.

4.2 Search for Optimal Model Structures

The search space for the subtask of finding an optimal model structure consists of all the regular HLC model structures for the given manifest variables. To search this space, we start with the simplest HLC model structure, namely the LC model structure (viewed as an unrooted HLC model structure). At each step, we modify the current structure to construct a number of new structures. The new structures are then evaluated and the best structure is selected as the starting point for the next step. To evaluate a model structure, one needs to estimate the cardinalities of its latent variables. This issue is addressed in subtask 1.

We use three search operators to modify model structures, namely node introduction, node elimination, and neighbor relocation.

4.2.1 NODE INTRODUCTION

To motivate the node introduction operator, we need to go back to rooted models. Consider the rooted HLC model M_1 shown in Figure 7. Suppose variables Y_1 and Y_2 are locally dependent. A natural way to model this local dependence is to introduce a new parent for Y_1 and Y_2, as shown in

M_2. This is precisely the idea behind the multiple indicator approach to local dependence that we mentioned in Section 2.

When translated to unrooted model structures, the new parent introduction operator becomes the *node introduction* operator. Let X be a latent node in an unrooted model structure. Suppose X has more than two neighbors. Then for any two neighbors of X, say Z_1 and Z_2, we can introduce a new latent node Z to separate X from Z_1 and Z_2. Afterwards, X is no longer connected to Z_1 and Z_2. Instead X is connected to Z and Z is connected to Z_1 and Z_2. To see an example, consider the model structure M_1' in Figure 7. Introducing a new latent node X_1 to separate X from Y_1 and Y_2 results in the model structure M_2'.

In the case of rooted model structures, we do not consider introducing new parents for groups of three or more nodes for the sake of computational efficiency. This constraint implies that the model M_3 in Figure 7 cannot be reached from M_1 in one step. In the case of unrooted model structures, we do not allow the introduction of a new node to separate a latent node from three or more of its neighbors. This implies that we cannot reach M_3' from M_1' in one step.

Node introduction is not allowed when it results in irregular model structures. This means that we cannot introduce a new node to separate a latent node X from two of its neighbors if it has only one other neighbor and that neighbor is a singly connected latent node. Moreover, we cannot introduce a new node to separate a singly connected latent node from its two neighbors.[9]

4.2.2 NODE ELIMINATION

The opposite of node introduction is *node elimination*. We notice that a newly introduced node has exactly three neighbors. Consequently we allow a latent node be eliminated only when it has three neighbors. Of course, node elimination cannot be applied if there is only one latent node.

4.2.3 NEIGHBOR RELOCATION

The third search operator is called *neighbor relocation*. Suppose a latent node X has a neighbor Z that is also a latent node. Then we can relocate any of the other neighbors Z' of X to Z, which means to disconnect Z' from X and reconnect it to Z. To see an example, consider the model structure M_2' in Figure 7. If we relocate the neighbor Y_3 of X to X_1, we reach structure M_3'.

For the sake of computational efficiency, we do not allow neighbor relocation between two non-neighboring latent nodes. In Figure 6, for example, we cannot relocate neighbors of X_2 to X_3 and vice versa. Moreover neighbor relocation is not allowed when it results in irregular model structures. To be more specific, suppose X is a latent node that has a latent node neighbor Z. We cannot relocate another neighbor Z' of X to Z if X has only three neighbors and the third neighbor is a singly connected latent node. The relocation is not allowed, of course, if X has only two neighbors. Finally note that the effects of any particular neighbor relocation can always be undone by another application of the operator.[10]

9. Node introduction is similar to an operator that PROMTL, a system for inferring phylogenetic trees, uses to search for optimal tree topologies via star decomposition (Kishino *et al.*, 1990). The former is slightly less constrained than the latter in that it is allowed to create singly connected nodes as by-products.
10. Neighbor relocation is related to but significantly different than an operator called branch swapping that PAUP, a system for inferring phylogenetic trees, uses to search for optimal tree topologies (Swofford, 1998). The latter includes what are called nearest neighbor interchange; subtree pruning and regrafting; and tree bisection/reconnection.

4.2.4 A PROPERTY

Theorem 5 *Consider the collection of regular HLC model structures for a given set of manifest variables. One can go between any two structures in the collection without visiting irregular structures using node introduction, node elimination, and neighbor relocation.*

Proof: It suffices to show that any regular structure S in the collection can be reached from the (unrooted) LC model structure without visiting irregular structures. We do this by induction on the number of latent nodes in S. If there is only one latent node, the proposition is trivially true. Suppose the proposition is true for the case of $n-1$ latent nodes where $n>1$. Consider the case of n. Because S is regular and there are more than one latent node, there must be a latent node X that has 3 or more neighbors. We modify S by first relocating some of the neighbors of X to other (neighboring) latent nodes such that it has only 3 neighbors afterwards. We then eliminate X. Denote the resulting structure by S'. It is evident that S' and all the intermediate structures are regular. By the induction hypothesis, we can reach S' from the LC model structure without visiting irregular structures using node introduction, node elimination, and neighbor relocation. By the construction of S', we know that we can reach S from S' without visiting irregular structures using those three operators. The theorem is therefore proved. Q.E.D

4.3 Complexity Analysis

The description of our learning algorithm is now complete. In this subsection we analyze the worst case complexity of the algorithm.

Let n be the number of manifest variables. According to Lemma 2, a regular HLC model with n manifest variables has fewer than $3n$ latent nodes. The total number of nodes in the model is hence bounded by $4n-1$.

In each step of structural search, our algorithm applies node introduction, node elimination, and neighbor relocation to the current model structure and produces a set of new structures. Each application of node introduction involves a pair of neighbors of a latent nodes. Such pairs for different applications of the operator cannot be the same. Hence the number of new structures produced by node introduction is bounded by $(4n-1)4n/2<8n^2$. The node-elimination operator produces no more than $3n$ new structures. An application of neighbor relocation also involves a pair of nodes, a latent node and one of its neighbors. Such pairs for different applications of the operator are different. Hence the total number of new structures produced by neighbor relocation is bounded by the number of edges, which is no more than $4n$. The total number of new structures produced at each search step is hence bounded by $8n^2+3n+4n=8n^2+7n$. If the entire search process takes N steps, then the total number of model structures that we need to examine is no more than $N(8n^2+7n)$.

For each model structure, we need to determine the cardinalities of all its latent variables. Our algorithm does so by hill-climbing. The search starts from the model where all latent variables have the minimum numbers of states possible and at each step a new model is generated for each latent variable by increasing its cardinality by one. Since the number of latent variables is no larger than $3n$, no more than $3n$ models can be generated at each step. Let k be the maximum number of states a variable can have in all models that we encounter. Then the search takes no more than $3nk$ steps. Consequently, the total number of models examined for an given model structure does not exceed $n*3nk=3n^2k$.

For each model, we need to estimate its parameters using the EM algorithm. The complexity of inference in an HLC model is linear in the number of nodes. Since there are no more than $4n$ nodes, inference takes $O(4n)$ time. Suppose there are d distinct data records ($d \leq k^n$). Then each iteration of EM takes $O(4nd)$ time. Let M be the maximum number of EM iterations allowed on a model. Then parameter estimation for a given model takes $(4ndM)$ time.

Totaling up all the parts, we conclude that the time complexity of our algorithm is

$$O(N(8n^2+7n)*3n^2k*4ndM) = O(96MNkdn^5). \quad (5)$$

5. Empirical Results on Synthetic Data

We have empirically evaluated the algorithm described in the previous section using both synthetic and real-world data. This section discusses experiments with synthetic data. Two experiments were conducted. We report their results separately.

5.1 Experiment 1

In this experiment, synthetic data were generated using the HLC model structure in Figure 3. The cardinalities of all variables were set at 3. The model was randomly instantiated. Four training sets with 5,000, 10,000, 50,000, and 100,000 records were sampled. A test set of 5,000 records was also sampled. Each sample record consists of states for all the manifest variables.

We ran our learning algorithm on each of the four training sets, once for each of the four scoring metrics BIC, AIC, CS, and LS. There are 16 settings in total. For the LS scoring metric, 25% of the training data was set aside and used as validation data. Candidate models were compared using their logarithmic scores on the validation data. During model selection, EM was terminated when the increase in real (not expected) loglikelihood fell below 0.01. When estimating parameters for the final model, the threshold was set at 0.0001. Irrespective of the threshold, EM was allowed to run no more than 200 iterations on any given model. For local maxima avoidance, we used the Chickering and Heckerman (1997) variant of the multiple-restart approach.

The experiments were conducted on a PC with a 1 GHz Pentium III processor. On the training set with 10,000 records, the algorithm took 97 hours to terminate. The running times for other cases are in the same scale.

The logarithmic scores of the learned models on the testing data are shown in Figure 8. The scores are grouped into four curves according to the four scoring metrics. The score of the original model is also shown for comparison. We see that, in the relative sense, the scores of the learned models are quite close to that of the original model. This indicates that those models are as good as the original model when it comes to predicting the testing set. We also see that scores do not vary significantly across the scoring metrics.

The structures of the learned models do depend on the scoring metrics. There are 7 different model structures. The first one is the original structure and will be denoted by $M0$. The other six are shown in Figure 9. Model structures produced by our algorithm are unrooted. In this and the next section, we root them in certain ways for readability. Table 1 gives information about which structure was obtained in what setting and how far away the structures are from the original structure in terms the number of structural search operations.

We see that, when combined with either BIC or CS, our algorithm obtained, from the 50k and 100k training sets, the correct structure. In the other two training sets, the model structures found

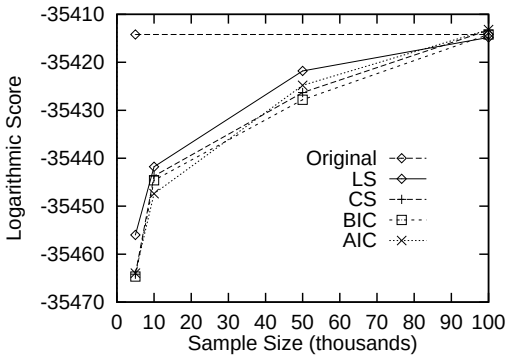


Figure 8: Logarithmic scores of learned models on testing data.

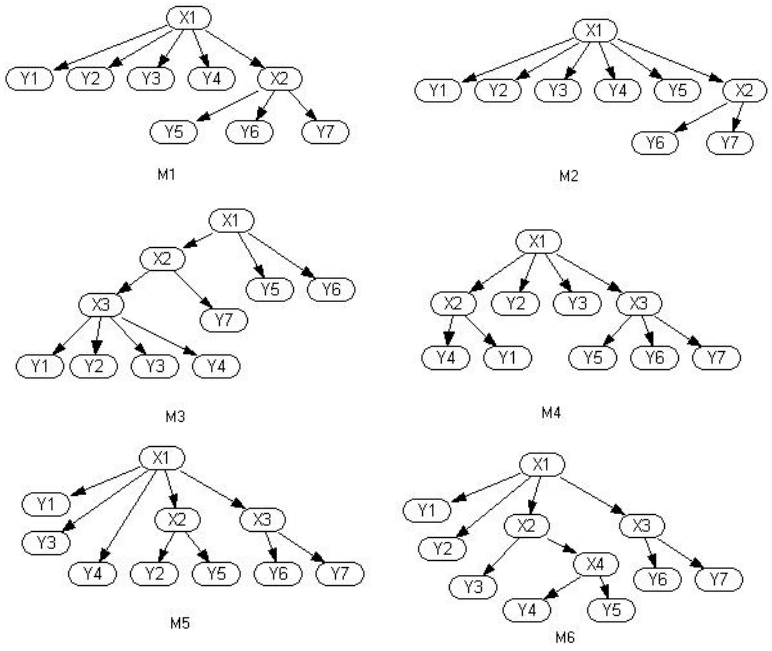


Figure 9: Structures of learned models.

are quite close to the original structure. For example, the BIC scoring metric gave us M1. This model structure is very similar to the original structure. The only thing that our algorithm failed to recognize in M1 is that Y_4 and X_2 are not independent given X_1. The fact that M1 is only one step away from the generative model implies that M1 was compared with the generative model during search. It was choosen over the generative model because it is better than the latter according to data. This happened due to insufficient data. As a matter of fact, in the two cases where there were 50k and 100k records, the generative model was selected over M1.

	5k	10k	50k	100k
BIC	1 (M1)	1 (M1)	0 (M0)	0 (M0)
CS	1 (M1)	2 (M2)	0 (M0)	0 (M0)
LS	2 (M3)	4 (M6)	0 (M0)	0 (M0)
AIC	3 (M4)	4 (M5)	0 (M0)	0 (M0)

Table 1: Model structures found in the 16 settings and the numbers of operations it would take to reach the original structure.

	50k				100k			
	BIC	CS	LS	AIC	BIC	CS	LS	AIC
X_1	2	2	2	2	2	2	2	2
X_2	3	3	3	3	3	3	3	4
X_3	2	2	4	4	3	3	4	4

Table 2: Cardinalities of latent variables in the learned models of Experiment 1 that have the correct structure. In the original model, all variables have 3 states.

When combined with AIC and LS, our algorithm also recovered the correct structure from the 50k and 100k training sets. In the other two training sets, however, it obtained structures that are significantly different from the original structure.

Although the correct model structure was recovered from the 50k and 100k training sets no matter which the scoring metric was used, different scoring metrics gave different estimates for the cardinalities of the latent variables. As can be seen from Table 2, BIC and CS tend to produce underestimates and with more data, they tend to give better estimates. On the other hand, AIC and LS tend to bring about overestimates and the estimates do not seem to improve with more data.

5.2 Experiment 2

The setup of this experiment is the same as that of Experiment 1 except the way model parameters were generated. Here we generated the parameters also in random fashion but ensured that each conditional probability distribution has one component with mass no smaller than 0.6. The objective is to see how our algorithm would perform in cases where the parameters are more extreme compared with those in Experiment 1.

In terms of logarithmic scores of learned models on testing data, results of this experiment are more or less the same as in Experiment 1. When it comes to structures and cardinalities of latent nodes, there are significant differences. With BIC and CS, our algorithm was able to recover the correct model structure from all four training sets. Moreover, the cardinalities of X_2 and X_3 were always estimated correctly. The cardinality of X_1 was estimated correctly in the 100k training sets, while it was underestimated by 1 in all other training sets. These results are significantly better than those in Experiment 1.

When AIC and LS was used, on the other hand, the performance is worse than in Experiment 1. The algorithm was unable to recover the correct model structure even from the 50k and 100k training sets.

6. Empirical Results on Real-World Data

This section reports empirical results on four data sets taken from the LCA literature, namely the Hannover rheumatoid arthritis data (Wasmus *et al.*, 1989), the Coleman data (Coleman, 1964), the HIV data (Alvord *et al.*, 1988), and the housing building data (Hagenaars, 1988). For the convenience of the reader, the data sets are reproduced in Tables 3 and 4 near the end of the paper. These experiments were also conducted on a PC with a 1 GHz Pentium III processor. The running times range from a few seconds to a few minutes.

6.1 The Hannover Rheumatoid Arthritis Data

The Hannover rheumatoid arthritis data was taken from a study by Wasmus *et al.* (1989) on the prevalence of rheumatoid arthritis in the adult population. A random sample of 25 to 74 year old German residents of Hannover, Germany was surveyed by means of a mailed questionnaire. Among others, this questionnaire contained five questions about the presence of five symptoms "today": back pain, neck pain, pain in one or several joints, joint swelling, and morning stiffness. Each item was to be answered in a simple yes/no response format. The data set consists of 7,162 records.

This data set has been analyzed by Kohlmann and Formann (1997). They conclude that the best model for this data set is a four class LC model. This model fits the data well ($L = 8.2$, $df = 8$, $p = 0.414$) and is meaningful to epidemiologists.

Using scoring metrics BIC, CS, and AIC, our algorithm discovered exactly the same model as the one obtained by Kohlmann and Formann (1997). When LS was used, however, it computed a very different model that does not fit data well.

6.2 The Coleman Data

The Coleman data summarize responses of 3,398 schoolboys, each was asked to respond to the following question and statement at two different points in time: (1) "Are you a member of the leading crowd?" (2) "If a fellow wants to be a part of the leading crowd around here, he sometimes has to go against his principles." There are four binary manifest variables A, B, C, and D. The variable A stands for the answer to the question in October 1957 and C that in May 1958. The variable B stands for the response to the statement in October 1957 and D that in May 1958. The value of 0 means "yes" and 1 means "no".

This data set has been previously analyzed by Goodman (1974a) and Hagenaars (1988). Goodman started with a 2-class LC model and found that it does not fit the data well ($L = 249.50$, $df = 6$, $p < 0.001$). He went on to consider the loglinear model that is represented by the path diagram M1 in Figure 10. In the model, both X_1 and X_2 are binary variables. This model fits data well ($L = 1.27$, $df = 4$, $p = 0.87$). Hagenaars examined several possible models and reached the conclusion that the loglinear model M2, where X is a binary variable, best explains the data. This model also fits the data very well ($L = 1.43$, $df = 5$, $p = 0.92$).

Using scoring metrics AIC, BIC, and CS, our algorithm found the model M3, where X_1 and X_2 are both binary variables. It's obvious that M3 is equivalent to M1 and hence fit data equally well.

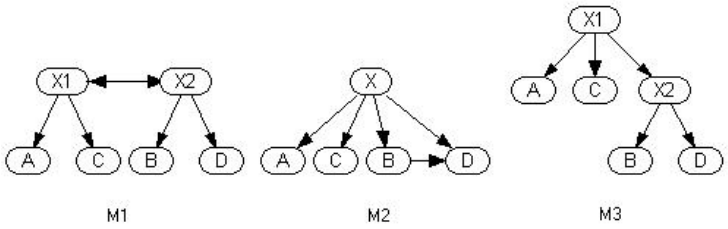


Figure 10: Models for the Coleman data.

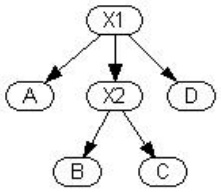


Figure 11: Model for the HIV data.

Our algorithm does not examine model M2 because it is not an HLC model. This is, however, no problem because M2 and M3 are almost identical as generative models for the manifest variables.

Using LS, our algorithm found a model that is the same as M3 except the cardinality of X_1 is 3. This model does not fit data well ($L = 1.27$, $df = 0$, $p = 0.0$).

6.3 The HIV Test Data

This data set consists of results on 428 subjects of four diagnostic tests for human HIV virus: "radioimmunoassay of antigen ag121" (*A*); "radioimmunoassay of HIV p24" (*B*); "radioimmunoassay of HIV gp120" (*C*); and "enzyme-linked immunosorbent assay" (*D*). A negative result is represented by 0 and a positive result by 1.

Alvord *et al.* (1988) reasoned that there should be two latent classes, corresponding to the presence and absence of the HIV virus. However, the two-class LC model does not fit data well ($L = 16.23$, $df = 6$, $p = 0.01$). This indicates the presence of local dependence.

The performance of our algorithm on this data set is similar to that on the Coleman data set. Using AIC, BIC, and CS, it found the model in Figure 11, where both latent variables are binary variables. The model is identical to one of the equivalent models Uebersax (2000) reached using some heuristic techniques. The model fit data well ($L = 3.056$, $df = 4$, $p = 0.548$).

With LS score, our algorithm produced the same model structure. However, the cardinalities of both latent variables are overestimated by 2. The model fits data poorly.

6.4 The House Building Data

The house building data are taken from a study by Hagenaars on people's view about what a new government should do. Again there are four binary manifest variables *A*, *B*, *C*, and *D*. Roughly speaking, *A* and *C* represent answers by respondents, in November and December 1970 respectively,

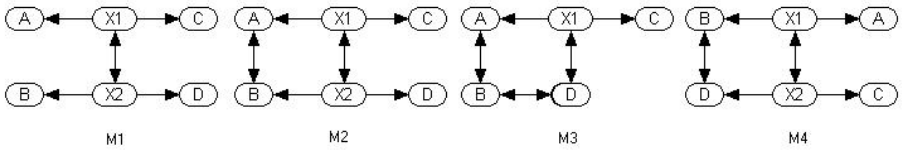


Figure 12: Models for the house building data.

to the question whether house building was an important problem. *B* and *D* represent the views of respondents, in November and December 1970 respectively, on how important house building was in relation to several other issues. We refer the reader to Hagenaars (1988) for the details.

When analyzing the data, Hagenaars started with the model M1 shown in Figure 12. In M1 and all the other three models, all latent variables are binary. The idea behind M1 is to capture the test-retest effects between the two interviews. This model does not fit data well ($L = 45.37$, $df = 4$, $p = 0.000$). Hagenaars went on to examine the standardized residuals and concluded that an direct effect should be added between *A* and *B*. This led to the model M2. This model fit data well ($L = 1.94$, $df = 3$, $p = 0.59$). Starting from some other initial models and adding direct effects properly among manifest variables, Hagenaars also derived the models M3 and M4, which fit data as well as M2.

It is clear that models M2, M3, and M4 are not close to any HLC models. Consequently, we cannot expect our algorithm to find satisfactory models for this data set. This turned out to be the case. Regardless of the scoring metric, our algorithm always found LC models. With BIC and CS it found 3-class LC models. With AIC and LS, it found 4-class LC models. None of the models fit data well.

To summarize the experiments with real-world data, we note that the performance of our algorithm on the Hannover rheumatoid arthritis data supports the claim made in the introduction that the algorithm would return LC models when there is no local dependence. The performances on the Coleman and HIV data sets support that claim that when local dependence is present, the algorithm would return HLC models with local dependence properly encoded. Finally, the performance on the housing building data shows the limitation of HLC models.

7. Conclusions and Future Directions

We have introduced a new class of models for cluster analysis, namely HLC models. HLC models are significantly more general than LC models and can accommodate local dependence. Yet they remain computationally attractive because of simplicity of their structures.

A search-based algorithm has been developed for learning HLC models from data. Both synthetic and real-world data have been used to evaluate the algorithm with four different scoring metrics, namely AIC, BIC, CS, and LS. The results indicate that the algorithm works well with BIC and CS.

The models that our algorithm, with BIC or CS, reconstructed from synthetic data predicate test data as well as the original model. Their structures are the same or equivalent to that of the original model, except in a couple of cases where minor differences exist (probably due to insufficient data). The cardinalities of latent variables were, however, often underestimated. On three of the four real-

world data sets, our algorithm found models that are considered optimal or close to optimal in the literature. However, it failed on the fourth data set, owing to the limitations of HLC models.

We end the paper with a list of issues that should be addressed. On top of the list is the issue of complexity. The focus of this paper has been on developing a principled search-based method for learning HLC models. Not much consideration was given to computational complexity. It is clear from Section 4.3 that our algorithm is computationally expensive because it, at each step of search, examines a large number of models and runs the EM algorithm on each of the models. To improve scability, we need to reduce the number of candidate models and to reduce the number of times the EM algorithm is called. Although not straightforward, both tasks are possible. For example, the number of calls to the EM algorithm can be reduced by applying the idea of structural EM (Friedman, 1997). We have recently developed a new algorithm based on this and other ideas. The algorithm was tested on, among others, a data set derived from the CoIL Challenge 2000 benchmark data (van der Putten and van Someren, 2000). There are 42 mostly binary attributes and 5822 records. The new algortihm finished analyzing the data in 121 hours on a PC with a 2.4 GHz Pentium 4 processor and it obtained a very interesting model. The details will be reported in upcoming papers.

The second issue concerns scoring metric. It has been shown that the BIC score is a consistent model selection criterion for Bayesian networks with no latent variables in the sense that, given sufficient data, the BIC score of the generative model, i.e., the model from which data were sampled, is larger than those of any other models that are not equivalent to the generative model (Geiger *et al.*, 2001). Although our empirical studies suggest that the BIC score is well-behaved in practice for the task of learning HLC models, BIC has not been proved to be consistent for latent variable models. The use of effective dimensions makes BIC a better approximation of the marginal likelihood (Geiger *et al.*, 1996); and a method for effectively computing effective dimensions of HLC models has been found (Kočka and Zhang, 2002). However, the marginal likelihood itself has not been shown to be consistent for latent variable models. Finding a consistent model selection criterion for HLC models in particular and for latent variable models in general is an important research topic.

Finally, we choose to study HLC models because they significantly generalize LC models and are computationally attractive. As we have seen in Section 6, HLC models are well suited for some applications while inadequate for others. Sometimes more complex models are needed. The challenge is to keep computation feasible while considering more and more complex models.

Acknowledgments

This work was partially supported by Hong Kong Research Grants Council under grants HKUST6093/99E and HKUST6088/01E. The bulk of the work was done while the author is on leave at Department of Computer Science, Aalborg University, Denmark. I thank Tomáš Kočka for insightful discussions on parsimonious and regular HLC models. I am also grateful to Craig Boutilier, Tao Chen, Finn V. Jensen, Thomas Nielsen, Kristian G. Olesen, Olav Bangso, Jose Pena, Jiri Vomlel, Marta Vomlelova and the anonymous reviewers for valuable feedback on earlier versions of this paper.

Back Pain	Neck Pain	Joint Pain	Swelling	Stiffness	Frequency
no	no	no	no	no	3,634
no	no	no	no	yes	73
no	no	no	yes	no	87
no	no	no	yes	yes	10
no	no	yes	no	no	440
no	no	yes	no	yes	89
no	no	yes	yes	no	106
no	no	yes	yes	yes	75
no	yes	no	no	no	295
no	yes	no	no	yes	25
no	yes	no	yes	no	15
no	yes	no	yes	yes	5
no	yes	yes	no	no	137
no	yes	yes	no	yes	42
no	yes	yes	yes	no	35
no	yes	yes	yes	yes	39
yes	no	no	no	no	489
yes	no	no	no	yes	37
yes	no	no	yes	no	23
yes	no	no	yes	yes	7
yes	no	yes	no	no	255
yes	no	yes	no	yes	116
yes	no	yes	yes	no	71
yes	no	yes	yes	yes	65
yes	yes	no	no	no	306
yes	yes	no	no	yes	48
yes	yes	no	yes	no	16
yes	yes	no	yes	yes	11
yes	yes	yes	no	no	229
yes	yes	yes	no	yes	162
yes	yes	yes	yes	no	44
yes	yes	yes	Yes	yes	176

Table 3: The Hannover rheumatoid arthritis data.

References

[1] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 716-723.

[2] Alvord, W. G., Drummond, J. E., Arthur. L.O., Biggar, R. J., Goedert, J. J., Levine, P. H., Murphy, E. L. Jr, Weiss, S. H., Blattner, W. A. (1988). A method for predicting individual HIV infection status in the absence of clinical information. *AIDS Res Hum Retroviruses*, 4(4):295-304.

A	B	C	D	Coleman	HIV	House Building
0	0	0	0	458	170	193
0	0	0	1	140	15	44
0	0	1	0	110	0	26
0	0	1	1	49	0	34
0	1	0	0	171	6	103
0	1	0	1	182	0	77
0	1	1	0	56	0	15
0	1	1	1	87	0	58
1	0	0	0	184	4	58
1	0	0	1	75	17	16
1	0	1	0	531	0	32
1	0	1	1	281	83	48
1	1	0	0	85	1	84
1	1	0	1	97	4	54
1	1	1	0	338	0	60
1	1	1	1	554	128	283

Table 4: The Coleman, HIV, and housing building data sets.

[3] Bartholomew, D. J. and Knott, M. (1999). *Latent variable models and factor analysis*, 2nd edition. Kendall's Library of Statistics 7. London: Arnold.

[4] Bohrnstedt, G. W. and Knoke D. (1994). *Statistics for social data analysis (3rd Edition)*. F. E. Peacock Publishers Inc., Itasca, Illinois.

[5] Cheeseman, P. and Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. In Fayyad, U., Piatesky-Shaoiro, G., Smyth, P., and Uthurusamy, R. (eds.), *Advancesin Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA.

[6] Chickering, D. M. and Heckerman, D. (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning* 29(2-3): 181-212.

[7] Chow, C. K. and Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3): 462-467.

[8] Coleman, J. S. (1964). *Introduction to Mathematical Sociology*. London: Free Press.

[9] Connolly, D. (1993). Constructing hidden variables in Bayesian networks via conceptual learning. *Proceedings of 10th International Conference on Machine Learning (ICML-93), Amherst, MA, USA*, 65-72.

[10] Cover, T. M., Thomas, J. A. (1991). *Elements of Information Theory*, John Wiley and Sons.

[11] Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic networks and expert systems*, Springer.

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

[13] Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press.

[14] Eaton, W. W., Dryman, A., Sorenson, A., and McCutcheon, A. (1989). DSM-III Major depressive disorder in the community: A latent class analysis of data from the NIMH epidemiologic catchment area programme. *British Journal of Psychiatry*, 155, 48-54.

[15] Elidan, G., Lotner, N., Friedman, N. and Koller, D. (2000). Discovering hidden variables: A structure-based approach. *Advances in Neural Information Processing Systems 13 (NIPS-00), Denver, CO, USA*, 479-485.

[16] Elidan, G. and N. Friedman (2001). Learning the dimensionality of hidden variables. *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI-01), Seattle, Washington, USA*, 144-151.

[17] Espeland, M. A. and Handelman, S. L. (1989). Using latent class models to characterize and assess relative error in discrete measurements. *Biometrics*, 45, 587-599.

[18] Everitt, B. S. (1993). *Cluster Analysis*. London: Edward Arnold.

[19] Fraley, C. (1998). Algorithms for model-based Gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 20 (1), 270-281.

[20] Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. *Proceedings of the 14th International Conference on Machine Learning (ICML), Nashville, TN, USA ICML-97*, 125-133.

[21] Friedman, N., Ninio, M., Pe'er, I., and Pupko, T. (2002). A structural EM algorithm for phylogenetic inference. *Journal of Computational Biology*, 9:331-353.

[22] Garrett, E. S. and Zeger, S. L. (2000). Latent class model diagnosis. *Biometrics*, 56, 1055-1067.

[23] Geiger, D., Heckerman, D., and C. Meek, C. (1996). Asymptotic Model Selection for Directed Networks with Hidden Variables. *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence, Portland, Oregon, USA (UAI-96)*, 158-168.

[24] Geiger, D., Heckerman, D., King, H., and Meek, C. (2001). Stratified exponential families: Graphical models and model selection. *The Annals of Statistics, 29 (1), 505-529.*

[25] Gibson, W. A. (1959). Three multivariate models: Factor analysis, latent structure analysis, and latent profile analysis. *Psychometrika*, 24: 229-252.

[26] Goodman, L. A. (1974a). The analysis of systems of qualitative variables when some of the variables are unobservable. Part I-A Modified latent structure approach. *American Journal of Sociology*, 7(5), 1179-1259.

[27] Green, P. (1998). Penalized likelihood. In *Encyclopedia of Statistical Sciences, Update Volume 3, S. Kotz, C. Read, D. L. Banks (eds.)*, 578-586, John Wiley and Sons.

[28] Goodman, L. A. (1974b). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61, 215-231.

[29] Hagenaars, J. A. (1988). Latent structure models with direct effects between indicators: local dependence models. *Sociological Methods and Research*, 16, 379-405.

[30] Hanson, R., Stutz, J., and Cheeseman, P. (1991). Bayesian classification with correlation and inheritance. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, New South Wales, Australia*, 2, 692-698.

[31] Kaufman, L. and Rousseeuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis.* New York: John Wiley and Sons, Inc.

[32] Kishino, H., Miyata, T., and Hasegawa, M. (1990). Maximum likelihood inference of protein phylogeny and the origin of the chloroplasts. J. Mol. Evol. 31, 151-160.

[33] Kohlmann, T., and Formann, A. K. (1997). Using latent class models to analyze response patterns in epidemiologic mail surveys. Rost, J. and Langeheine, R. (eds.). *Applications of latent trait and latent class models in the social sciences*. Muenster: Waxman Verlag.

[34] Lanterman, A. D. (2001). Schwarz, Wallace, and Rissanen: Intertwining themes in theories of model order estimation. *International Statistical Review*, 69(2), 185-212.

[35] Lauritzen, S. L. (1995). The EM-algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 1, 191-201.

[36] Lazarsfeld, P. F., and Henry, N.W. (1968). *Latent structure analysis*. Boston: Houghton Mifflin.

[37] Martin, J. and VanLehn, K. (1994). Discrete factor analysis: learning hidden variables in Bayesian networks. Technical Report LRGC_ONR-94-1, Department of Computer Science, University of Pittsburgh.

[38] Meila-Predoviciu, M. (1999). *Learning with mixtures of trees*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[39] Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6(2), 461-464.

[40] Swofford, D. L. (1998). *PAUP* 4.0 - Phylogenetic Analysis Using Parsimony (*and Other Methods)*. Sinauer Assoc., Sunderland, MA.

[41] Uebersax, J. (2000). A practical guide to local dependence in latent class models. http://ourworld.compuserve.com/homepages/jsuebersax/condep.htm.

[42] van der Putten, P. and van Someren, M. (eds) (2000). *CoIL Challenge 2000: The Insurance Company Case.* Sentient Machine Research, Amsterdam. See also: http://www.liacs.nl/%7Eputten/library/cc2000/.

[43] Vermunt, J.K. and Magidson, J. (2000). *Latent GOLD User's Guide*. Belmont, Mass.: Statistical Innovations, Inc.

[44] Vermunt, J.K. and Magidson, J. (2002). Latent class cluster analysis. in Hagenaars, J. A. and McCutcheon A. L. (eds.), *Advances in latent class analysis*. Cambridge University Press.

[45] Wasmus, A., Kindel, P., Mattussek, S. and Raspe, H. H. (1989). Activity and severity of rheumatoid arthritis in Hannover/FRG and in one regional referral center. *Scandinavian Journal of Rheumatology*, Suppl. 79, 33-44.

Journal of Machine Learning Research 5 (2004) 725–775 Submitted 3/03; Revised 11/03; Published 7/04

Bias-Variance Analysis of Support Vector Machines for the Development of SVM-Based Ensemble Methods

Giorgio Valentini VALENTINI@DSI.UNIMI.IT
DSI - Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Comelico 39, Milano, Italy

Thomas G. Dietterich TGD@CS.ORST.EDU
Department of Computer Science
Oregon State University
Corvallis, OR 97331, USA

Editor: Nello Cristianini

Abstract

Bias-variance analysis provides a tool to study learning algorithms and can be used to properly design ensemble methods well tuned to the properties of a specific base learner. Indeed the effectiveness of ensemble methods critically depends on accuracy, diversity and learning characteristics of base learners. We present an extended experimental analysis of bias-variance decomposition of the error in Support Vector Machines (SVMs), considering Gaussian, polynomial and dot product kernels. A characterization of the error decomposition is provided, by means of the analysis of the relationships between bias, variance, kernel type and its parameters, offering insights into the way SVMs learn. The results show that the expected trade-off between bias and variance is sometimes observed, but more complex relationships can be detected, especially in Gaussian and polynomial kernels. We show that the bias-variance decomposition offers a rationale to develop ensemble methods using SVMs as base learners, and we outline two directions for developing SVM ensembles, exploiting the SVM bias characteristics and the bias-variance dependence on the kernel parameters.

Keywords: Bias-variance analysis, support vector machines, ensemble methods, multi-classifier systems.

1. Introduction

Ensembles of classifiers represent one of the main research directions in machine learning (Dietterich, 2000a). Empirical studies showed that both in classification and regression problems ensembles are often much more accurate than the individual base learner that make them up (Bauer and Kohavi, 1999; Dietterich, 2000b; Freund and Schapire, 1996), and recently different theoretical explanations have been proposed to justify the effectiveness of some commonly used ensemble methods (Kittler et al., 1998; Schapire, 1999; Kleinberg, 2000; Allwein et al., 2000).

Two main theories are invoked to explain the success of ensemble methods. The first one considers the ensembles in the framework of large margin classifiers (Mason et al., 2000), showing that ensembles enlarge the margins, enhancing the generalization capabilities of learning algorithms (Schapire et al., 1998; Allwein et al., 2000). The second is based on the classical bias-

variance decomposition of the error (Geman et al., 1992), and it shows that ensembles can reduce variance (Breiman, 1996b) and also bias (Kong and Dietterich, 1995).

Recently Domingos proved that Schapire's notion of margins (Schapire et al., 1998) can be expressed in terms of bias and variance and vice versa (Domingos, 2000c), and hence Schapire's bounds of ensemble's generalization error can be equivalently expressed in terms of the distribution of the margins or in terms of the bias-variance decomposition of the error, showing the equivalence of margin-based and bias-variance-based approaches.

The effectiveness of ensemble methods depends on the specific characteristics of the base learners; in particular on the relationship between diversity and accuracy of the base learners (Dietterich, 2000a; Kuncheva et al., 2001b; Kuncheva and Whitaker, 2003), on their stability (Breiman, 1996b; Bousquet and Elisseeff, 2002), and on their general geometrical properties (Cohen and Intrator, 2001).

From this standpoint the analysis of the features and properties of the base learners used in ensemble methods is crucial in order to design ensemble methods well tuned to the characteristics of a specific base learner. For instance, considering that the agglomeration of many classifiers into one classification rule reduces variance (Breiman, 1996a), we could apply low-bias base learners to reduce both bias and variance using ensemble methods. To this purpose in this paper we study Support Vector Machines (SVMs), that are "strong" dichotomic classifiers, well founded on Vapnik's statistical learning theory (Vapnik, 1998), in order to establish if and how we can exploit their specific features in the context of ensemble methods. We analyze the learning properties of SVMs using the bias-variance decomposition of the error as a tool to understand the relationships between kernels, kernel parameters, and learning processes in SVM.

Historically, the bias-variance insight was borrowed from the field of regression, using squared-loss as the loss function (Geman et al., 1992). For classification problems, where the $0/1$ loss is the main criterion, several authors proposed bias-variance decompositions related to $0/1$ loss. Kong and Dietterich (1995) proposed a bias-variance decomposition in the context of ECOC ensembles (Dietterich and Bakiri, 1995), but their analysis is extensible to arbitrary classifiers, even if they defined variance simply as a difference between loss and bias.

In Breiman's decomposition (Breiman, 1996b) bias and variance are always non-negative (while Dietterich definition allows a negative variance), but at any input the reducible error (i.e. the total error rate less noise) is assigned entirely to variance if the classification is unbiased, and to bias if biased. Moreover he forced the decomposition to be purely additive, while for the 0/1 loss this is not the case. Kohavi and Wolpert (1996) approach leads to a biased estimation of bias and variance, assigning a non-zero bias to a Bayes classifier, while Tibshirani (1996) did not use directly the notion of variance, decomposing the $0/1$ loss in bias and an unrelated quantity he called "aggregation effect", which is similar to the James' notion of *variance effect* (James, 2003).

Friedman (1997) showed that bias and variance are not purely additive: in some cases increasing variance increases the error, but in other cases can also reduce the error, especially when the prediction is biased.

Heskes (1998) proposed a bias-variance decomposition using as loss function the Kullback-Leibler divergence. By this approach the error between the target and the predicted classifier densities is measured; anyway when he tried to extend this approach to the zero-one function interpreted as the limit case of log-likelihood type error, the resulting decomposition produces a definition of bias that loses his natural interpretation as systematic error committed by the classifier.

As briefly outlined, these decompositions suffer of significant shortcomings: in particular they lose the relationship to the original squared loss decomposition, forcing in most cases bias and variance to be purely additive.

We consider classification problems and the $0/1$ loss function in the Domingos' unified framework of bias-variance decomposition of the error (Domingos, 2000c,b). In this approach bias and variance are defined for an arbitrary loss function, showing that the resulting decomposition specializes to the standard one for squared loss, but it holds also for the $0/1$ loss (Domingos, 2000c).

A similar approach has been proposed by James (2003): he extended the notion of variance and bias for general loss functions, distinguishing also between bias and variance, interpreted respectively as the systematic error and the variability of an estimator, and the actual effect of bias and variance on the error.

Using Domingos' theoretical framework, we tried to answer two main questions:

1. Can we characterize bias and variance in SVMs with respect to the kernel and its parameters?

2. Can the bias-variance decomposition offer guidance for developing ensemble methods using SVMs as base learners?

In order to answer these two questions, we planned and performed an extensive series of experiments on synthetic and real data sets to evaluate bias variance-decomposition of the error with different kernels and different kernel parameters.

The paper is organized as follows. In Section 2, we summarize the main results of Domingos' unified bias-variance decomposition of error. Section 3 outlines how to measure in practice bias and variance decomposition of the error with artificial or large benchmark data sets, or when only a small "real" data set is available. Section 4 outlines the main characteristics of the data sets employed in our experiments and the main experimental tasks performed. Then we present the main results of our experiments about bias-variance decomposition of the error in SVMs, considering separately Gaussian, polynomial and and dot product SVMs, and comparing also the results between different kernels. Section 6 provides a characterization of bias-variance decomposition of the error for Gaussian, polynomial and and dot product SVMs, highlighting the common patterns for each different kernel. Section 7 exploits the knowledge achieved by the bias-variance decomposition of the error to formulate hypotheses about the effectiveness of SVMs as base learners in ensembles of learning machines, and two directions for developing new ensemble models of SVM are proposed. An outline of ongoing and future developments of this work concludes the paper.

2. Bias-Variance Decomposition for the 0/1 Loss Function

The analysis of bias-variance decomposition of the error has been originally developed in the standard regression setting, where the squared error is usually used as loss function. Considering a prediction $y = f(x)$ of an unknown target t, provided by a learner f on input x, with $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$, the classical decomposition of the error in bias and variance for the squared error loss is (Geman et al., 1992)

$$\begin{aligned} E_{y,t}[(y-t)^2] &= E_t[(t-E[t])^2] + E_y[(y-E[y])^2] + (E[y]-E[t])^2 \\ &= Noise(t) + Var(y) + Bias^2(y). \end{aligned}$$

In words, the expected loss of using y to predict t is the sum of the variances of t (noise) and y plus the squared bias. $E_y[\cdot]$ indicates the expected value with respect to the distribution of the random variable y.

This decomposition cannot be automatically extended to the standard classification setting, as in this context the 0/1 loss function is usually applied, and bias and variance are not purely additive. As we are mainly interested in analyzing bias-variance for classification problems, we introduce the bias-variance decomposition for the 0/1 loss function, according to the Domingos unified bias-variance decomposition of the error (Domingos, 2000b).

2.1 Expected Loss Depends on the Randomness of the Training Set and the Target

Consider a (potentially infinite) population U of labeled training data points, where each point is a pair $(\mathbf{x}_j, t_j)$, $t_j \in \mathcal{C}$, $\mathbf{x}_j \in \mathbb{R}^d$, $d \in \mathbb{N}$, where $\mathcal{C}$ is the set of the class labels. Let $P(\mathbf{x},t)$ be the joint distribution of the data points in U. Let D be a set of m points drawn identically and independently from U according to P. We think of D as being the training sample that we are given for training a classifier. We can view D as a random variable, and we will let $E_D[\cdot]$ indicate the expected value with respect to the distribution of D.

Let $\mathcal{L}$ be a learning algorithm, and define $f_D = \mathcal{L}(D)$ as the classifier produced by $\mathcal{L}$ applied to a training set D. The model produces a prediction $f_D(\mathbf{x}) = y$. Let $L(t,y)$ be the $0/1$ loss function, that is $L(t,y) = 0$ if $y = t$, and $L(t,y) = 1$ otherwise.

Suppose we consider a fixed point $\mathbf{x} \in \mathbb{R}^d$. This point may appear in many labeled training points in the population. We can view the corresponding labels as being distributed according to the conditional distribution $P(t|\mathbf{x})$. Recall that it is always possible to factor the joint distribution as $P(\mathbf{x},t) = P(\mathbf{x})P(t|\mathbf{x})$. Let $E_t[\cdot]$ indicate the expectation with respect to t drawn according to $P(t|\mathbf{x})$.

Suppose we consider a *fixed* predicted class y for a given $\mathbf{x}$. This prediction will have an expected loss of $E_t[L(t,y)]$. In general, however, the prediction y is not fixed. Instead, it is computed from a model f_D which is in turn computed from a training sample D.

Hence, the expected loss EL of learning algorithm $\mathcal{L}$ at point $\mathbf{x}$ can be written by considering both the randomness due to the choice of the training set D and the randomness in t due to the choice of a particular test point $(\mathbf{x},t)$:

$$EL(\mathcal{L},\mathbf{x}) = E_D[E_t[L(t, f_D(\mathbf{x}))]],$$

where $f_D = \mathcal{L}(D)$ is the classifier learned by $\mathcal{L}$ on training data D. The purpose of the bias-variance analysis is to decompose this expected loss into terms that separate the bias and the variance.

2.2 Optimal and Main Prediction

To derive this decomposition, we must define two things: the *optimal prediction* and the *main prediction*: according to Domingos, bias and variance can be defined in terms of these quantities.

The *optimal prediction* y_* for point $\mathbf{x}$ minimizes $E_t[L(t,y)]$:

$$y_*(\mathbf{x}) = \arg\min_y E_t[L(t,y)]. \tag{1}$$

It is equal to the label t that is observed more often in the universe U of the data points, and corresponds to the prediction provided by the Bayes classifier. The *optimal model* $\hat{f}(\mathbf{x}) = y_*$, $\forall \mathbf{x}$

makes the optimal prediction at each point $\mathbf{x}$, and corresponds to the Bayes classifier; its error rate corresponds to the Bayes error rate.

The noise $N(\mathbf{x})$, is defined in terms of the optimal prediction, and represents the remaining loss that cannot be eliminated, even by the optimal prediction:

$$N(\mathbf{x}) = E_t[L(t, y_*)].$$

Note that in the deterministic case $y_* = t$ and $N(\mathbf{x}) = 0$.

The *main prediction* y_m at point $\mathbf{x}$ is defined as

$$y_m = \arg\min_{y'} E_D[L(f_D(\mathbf{x}), y')]. \tag{2}$$

This is a value that would give the lowest expected loss if it were the "true label" of $\mathbf{x}$. It expresses the "central tendency" of a learner, that is its systematic prediction, or, in other words, it is the label for $\mathbf{x}$ that the learning algorithm "wishes" were correct. For 0/1 loss, the main prediction is the class predicted most often by the learning algorithm $\mathcal{L}$ when applied to training sets D.

2.3 Bias, Unbiased and Biased Variance.

Given these definitions, the *bias* $B(\mathbf{x})$ (of a learning algorithm $\mathcal{L}$ on training sets of size m) is the loss of the main prediction relative to the optimal prediction:

$$B(\mathbf{x}) = L(y_*, y_m).$$

For 0/1 loss, the bias is always 0 or 1. We will say that $\mathcal{L}$ is *biased at point* $\mathbf{x}$, if $B(\mathbf{x}) = 1$.

The *variance* $V(\mathbf{x})$ is the average loss of the predictions relative to the main prediction:

$$V(\mathbf{x}) = E_D[L(y_m, f_D(\mathbf{x}))]. \tag{3}$$

It captures the extent to which the various predictions $f_D(\mathbf{x})$ vary depending on D.

In the case of the 0/1 loss we can also distinguish two opposite effects of variance (and noise) on the error: in the unbiased case variance and noise increase the error, while in the biased case variance and noise decrease the error.

There are three components that determine whether $t = y$:

1. Noise: is $t = y_*$?
2. Bias: is $y_* = y_m$?
3. Variance: is $y_m = y$?

Note that bias is either 0 or 1 because neither y_* nor y_m are random variables. From this standpoint we can consider two different cases: the unbiased and the biased case.

In the unbiased case, $B(\mathbf{x}) = 0$ and hence $y_* = y_m$. In this case we suffer a loss if the prediction y differs from the main prediction y_m (variance) and the optimal prediction y_* is equal to the target t, or y is equal to y_m, but y_* is different from t (noise).

In the biased case, $B(\mathbf{x}) = 1$ and hence $y_* \neq y_m$. In this case we suffer a loss if the prediction y is equal to the main prediction y_m and the optimal prediction y_* is equal to the target t, or if both y is different from y_m (variance), and y_* is different from t (noise). Figure 1 summarizes the different

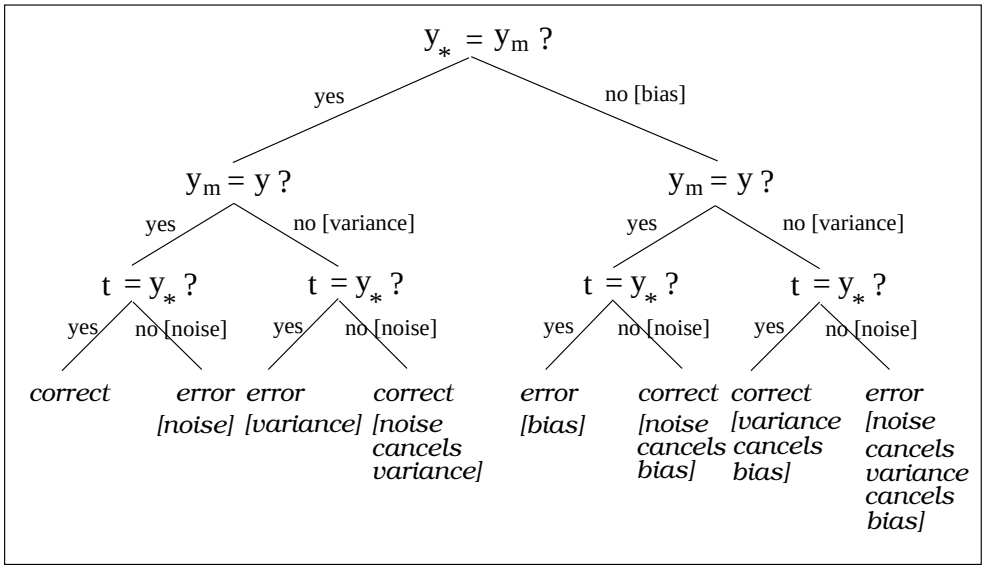


Figure 1: Case analysis of error.

conditions under which an error can arise, considering the combined effect of bias, variance and noise on the learner prediction.

Considering the above case analysis of the error, if we let $P(t \neq y_*) = N(\mathbf{x}) = \tau$ and $P(y_m \neq y) = V(\mathbf{x}) = \sigma$, in the unbiased case we have

$$\begin{aligned} L(t,y) &= \tau(1-\sigma)+\sigma(1-\tau) \\ &= \tau+\sigma-2\tau\sigma \\ &= N(\mathbf{x})+V(\mathbf{x})-2N(\mathbf{x})V(\mathbf{x}), \end{aligned} \tag{4}$$

while in the biased case

$$\begin{aligned} L(t,y) &= \tau\sigma+(1-\tau)(1-\sigma) \\ &= 1-(\tau+\sigma-2\tau\sigma) \\ &= B(\mathbf{x})-(N(\mathbf{x})+V(\mathbf{x})-2N(\mathbf{x})V(\mathbf{x})). \end{aligned} \tag{5}$$

Note that in the unbiased case (Equation 4) the variance is an additive term of the loss function, while in the biased case (Equation 5) the variance is a subtractive term of the loss function. Moreover the interaction terms will usually be small, because, for instance, if both noise and variance term will be both lower than 0.1, the interaction term $2N(\mathbf{x})V(\mathbf{x})$ will be reduced to less than 0.02.

In order to distinguish between these two different effects of the variance on the loss function, Domingos defines the *unbiased variance*, $V_u(\mathbf{x})$, to be the variance when $B(\mathbf{x}) = 0$ and the *biased variance*, $V_b(\mathbf{x})$, to be the variance when $B(\mathbf{x}) = 1$. We can also define the *net variance* $V_n(\mathbf{x})$ to take into account the combined effect of the unbiased and biased variance:

$$V_n(\mathbf{x}) = V_u(\mathbf{x}) - V_b(\mathbf{x}).$$

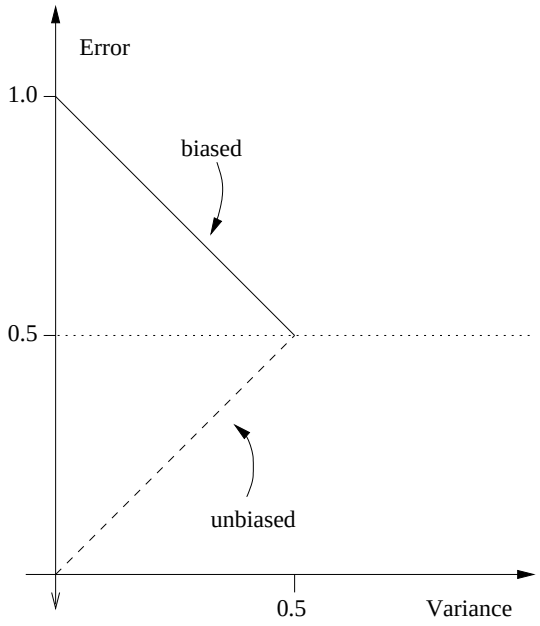


Figure 2: Effects of biased and unbiased variance on the error. The unbiased variance increments, while the biased variance decrements the error.

Figure 2 summarizes in graphic form the opposite effects of biased and unbiased variance on the error.

If we can disregard the noise, the unbiased variance captures the extents to which the learner deviates from the correct prediction y_m (in the unbiased case $y_m = y_*$), while the biased variance captures the extents to which the learner deviates from the incorrect prediction y_m (in the biased case $y_m \neq y_*$).

2.4 Domingos' Bias-Variance Decomposition

Domingos (2000a) showed that for a quite general loss function the expected loss is

$$EL(\mathcal{L}, \mathbf{x}) = c_1 N(\mathbf{x}) + B(\mathbf{x}) + c_2 V(\mathbf{x}). \tag{6}$$

For the $0/1$ loss function c_1 is $2P_D(f_D(\mathbf{x}) = y_*) - 1$ and c_2 is +1 if $B(\mathbf{x}) = 0$ and -1 if $B(\mathbf{x}) = 1$. Note that $c_2 V(\mathbf{x}) = V_u(\mathbf{x}) - V_b(\mathbf{x}) = V_n(\mathbf{x})$ (Equation 3), and if we disregard the noise, Equation 6 can be simplified to

$$EL(\mathcal{L}, \mathbf{x}) = B(\mathbf{x}) + V_n(\mathbf{x}). \tag{7}$$

Summarizing, one of the most interesting aspects of Domingos' decomposition is that variance hurts on unbiased points $\mathbf{x}$, but it helps on biased points. Nonetheless, to obtain low overall expected loss, we want the bias to be small, and hence, we see to reduce both the bias and the unbiased variance. A good classifier will have low bias, in which case the expected loss will approximately equal the variance.

This decomposition for a single point $\mathbf{x}$ can be generalized to the entire population by defining $E_{\mathbf{x}}[\cdot]$ to be the expectation with respect to $P(\mathbf{x})$. Then we can define the *average bias* $E_{\mathbf{x}}[B(\mathbf{x})]$, the *average unbiased variance* $E_{\mathbf{x}}[V_u(\mathbf{x})]$, and the *average biased variance* $E_{\mathbf{x}}[V_b(\mathbf{x})]$. In the noise-free case, the expected loss over the entire population is

$$E_{\mathbf{x}}[EL(\mathcal{L},\mathbf{x})] = E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[V_u(\mathbf{x})] - E_{\mathbf{x}}[V_b(\mathbf{x})].$$

3. Measuring Bias and Variance

The procedures to measure bias and variance depend on the characteristics and on the cardinality of the data sets used.

For synthetic data sets we can generate different sets of training data for each learner to be trained. Then a large synthetic test set can be generated in order to estimate the bias-variance decomposition of the error for a specific learner model.

Similarly, if a large data set is available, we can split it in a large learning set and in a large testing set. Then we can randomly draw subsets of data from the large training set in order to train the learners; bias-variance decomposition of the error is measured on the large independent test set.

However, in practice, for real data we dispose of only one and often small data set. In this case, we can use cross-validation techniques for estimating bias-variance decomposition, but we propose to use out-of-bag (Breiman, 2001) estimation procedures, as they are computationally less expensive.

3.1 Measuring with Artificial or Large Benchmark Data Sets

Consider a set $\mathcal{D} = \{D_i\}_{i=1}^{n}$ of learning sets $D_i = \{\mathbf{x}_k, t_k\}_{k=1}^{m}$. The data sets D_i can be generated according to some known probability distribution or can be drawn with replacement from a large data set $\mathcal{D}$ according to an uniform probability distribution. Here we consider only a two-class case, i.e. $t_k \in \mathcal{C} = \{-1,1\}, \quad \mathbf{x}_k \in \mathbf{X}$, for instance $\mathbf{X} = \mathbb{R}^d, \quad d \in \mathbb{N}$, but the extension to the multiclass case is straightforward.

The estimates of the error, bias, unbiased and biased variance are performed on a test set $\mathcal{T}$ separated from the training set $\mathcal{D}$. In particular these estimates with respect to a single example $(\mathbf{x},t) \in \mathcal{T}$ are performed using the classifiers $f_{D_i} = \mathcal{L}(D_i)$ produced by a learner $\mathcal{L}$ using training sets D_i drawn from $\mathcal{D}$. These classifiers produce a prediction $y \in \mathcal{C}$, that is $f_{D_i}(\mathbf{x}) = y$. The estimates are performed for all the $(\mathbf{x},t) \in \mathcal{T}$, and the overall loss, bias and variance can be evaluated averaging over the entire test set $\mathcal{T}$.

In presence of noise and with the 0/1 loss, the optimal prediction y_* is equal to the label t that is observed more often in the universe U of data points:

$$y_*(\mathbf{x}) = \arg\max_{t \in \mathcal{C}} P(t|\mathbf{x}).$$

The noise $N(\mathbf{x})$ for the 0/1 loss can be estimated if we can evaluate the probability of the targets for a given example $\mathbf{x}$:

$$N(\mathbf{x}) = \sum_{t \in \mathcal{C}} L(t,y_*)P(t|\mathbf{x}) = \sum_{t \in \mathcal{C}} ||t \neq y_*||P(t|\mathbf{x}),$$

where $||z|| = 1$ if z is true, 0 otherwise. In practice it is difficult to estimate the noise for "real world" data sets, and to simplify the computation we consider the noise free case. In this situation we have $y_* = t$.

The main prediction is a function of the $y = f_{D_i}(\mathbf{x})$. Considering a $0/1$ loss, we have

$$y_m = \arg\max(p_1, p_{-1}),$$

where $p_1 = P_D(y = 1|\mathbf{x})$ and $p_{-1} = P_D(y = -1|\mathbf{x})$, i.e. the main prediction is the mode. To calculate p_1, having a test set $\mathcal{T} = \{\mathbf{x}_j, t_j\}_{j=1}^{r}$, it is sufficient to count the number of learners that predict class 1 on a given input $\mathbf{x}$:

$$p_1(\mathbf{x}_j) = \frac{\sum_{i=1}^{n} \| f_{D_i}(\mathbf{x}_j) = 1 \|}{n},$$

where $\|z\| = 1$ if z is true and $\|z\| = 0$ if z is false

The bias can be easily calculated after the evaluation of the main prediction:

$$B(\mathbf{x}) = \begin{cases} 1 & \text{if } y_m \neq t \\ 0 & \text{if } y_m = t \end{cases} = \left| \frac{y_m - t}{2} \right|, \tag{8}$$

or equivalently:

$$B(\mathbf{x}) = \begin{cases} 1 & \text{if } p_{corr}(\mathbf{x}) \leq 0.5 \\ 0 & \text{otherwise,} \end{cases}$$

where p_{corr} is the probability that a prediction is correct, i.e. $p_{corr}(\mathbf{x}) = P(y = t|\mathbf{x}) = P_D(f_D(\mathbf{x}) = t)$.

In order to measure the variance $V(\mathbf{x})$, if we define $y_{D_i} = f_{D_i}(\mathbf{x})$, we have

$$V(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} L(y_m, y_{D_i}) = \frac{1}{n} \sum_{i=1}^{n} ||(y_m \neq y_{D_i})||$$

The *unbiased variance* $V_u(\mathbf{x})$ and the *biased variance* $V_b(\mathbf{x})$ can be calculated evaluating if the prediction of each learner differs from the main prediction respectively in the unbiased and in the biased case:

$$V_u(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} ||(y_m = t) \text{ and } (y_m \neq y_{D_i})||,$$

$$V_b(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} ||(y_m \neq t) \text{ and } (y_m \neq y_{D_i})||.$$

In the noise-free case, the *average loss on the example* $\mathbf{x}$ $E_D(\mathbf{x})$ is calculated by a simple algebraic sum of bias, unbiased and biased variance:

$$E_D(\mathbf{x}) = B(\mathbf{x}) + V_u(\mathbf{x}) - V_b(\mathbf{x}) = B(\mathbf{x}) + (1 - 2B(\mathbf{x}))V(\mathbf{x}).$$

We can easily calculate the *average bias, variance, unbiased, biased and net variance*, averaging over the entire set of the examples of the test set $\mathcal{T} = \{(\mathbf{x}_j, t_j)\}_{j=1}^{r}$. In the remaining part of this section the indices j refer to the examples that belong to the test set $\mathcal{T}$, while the indices i refer to the training sets D_i, drawn with replacement from the separated training set $\mathcal{D}$, and used to train the classifiers f_{D_i}.

The average quantities are

Average bias:

$$E_{\mathbf{x}}[B(\mathbf{x})] = \frac{1}{r} \sum_{j=1}^{r} B(\mathbf{x}_j) = \frac{1}{r} \sum_{j=1}^{r} \left| \frac{y_m(\mathbf{x}_j) - t_j}{2} \right|,$$

Average variance:

$$\begin{aligned} E_{\mathbf{x}}[V(\mathbf{x})] &= \frac{1}{r}\sum_{j=1}^{r} V(\mathbf{x}_j) \\ &= \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} L(y_m(\mathbf{x}_j), f_{D_i}(\mathbf{x}_j)) \\ &= \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} ||y_m(\mathbf{x}_j) \neq f_{D_i}(\mathbf{x}_j)||, \end{aligned}$$

Average unbiased variance:

$$E_{\mathbf{x}}[V_u(\mathbf{x})] = \frac{1}{r}\sum_{j=1}^{r} V_u(\mathbf{x}_j) = \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} ||(y_m(\mathbf{x}_j) = t_j) \text{ and } (y_m(\mathbf{x}_j) \neq f_{D_i}(\mathbf{x}_j))||,$$

Average biased variance:

$$E_{\mathbf{x}}[V_b(\mathbf{x})] = \frac{1}{r}\sum_{j=1}^{r} V_b(\mathbf{x}_j) = \frac{1}{nr}\sum_{j=1}^{r}\sum_{i=1}^{n} ||(y_m(\mathbf{x}_j) \neq t_j) \text{ and } (y_m(\mathbf{x}_j) \neq f_{D_i}(\mathbf{x}_j))||,$$

and the *Average net variance*:

$$E_{\mathbf{x}}[V_n(\mathbf{x})] = \frac{1}{r}\sum_{j=1}^{r} V_n(\mathbf{x}_j) = \frac{1}{r}\sum_{j=1}^{r} (V_u(\mathbf{x}_j) - V_b(\mathbf{x}_j)).$$

Finally, the average loss on all the examples (with no noise) is the algebraic sum of the average bias, unbiased and biased variance.

$$E_{\mathbf{x}}[L(t,y)] = E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[V_u(\mathbf{x})] - E_{\mathbf{x}}[V_b(\mathbf{x})]$$

3.2 Measuring with Small Data Sets

In practice (unlike in theory), we have only one and often small data set $\mathcal{S}$. We can simulate multiple training sets by bootstrap replicates $S_b = \{\mathbf{x}|\mathbf{x} \text{ is drawn at random with replacement from } \mathcal{S}\}$. In order to measure bias and variance we can use out-of-bag points, providing in such a way an unbiased estimate of the error. At first we need to construct B bootstrap replicates of $\mathcal{S}$ (e. g., $B = 200$): $S_1, \ldots, S_B$. Then we apply a learning algorithm $\mathcal{L}$ to each replicate S_b to obtain hypotheses $f_b = \mathcal{L}(S_b)$.

Let $T_b = \mathcal{S} \backslash S_b$ be the data points that do not appear in S_b (out of bag points). We can use these data sets T_b to evaluate the bias-variance decomposition of the error; that is we compute the predicted values $f_b(\mathbf{x})$, $\forall \mathbf{x}$ s.t. $\mathbf{x} \in T_b$. For each data point $\mathbf{x}$, we have now the observed corresponding value t and several predictions $y_1, \ldots, y_K$, where $K = |\{T_b|\mathbf{x} \in T_b, 1 \leq b \leq B\}|$, $K \leq B$, and on the average $K \simeq B/3$, because about $1/3$ of the predictors is not trained on a specific input $\mathbf{x}$. Note that the value of K depends on the specific example $\mathbf{x}$ considered. Moreover if $\mathbf{x} \in T_b$ then $\mathbf{x} \notin S_b$, hence $f_b(\mathbf{x})$ makes a prediction on an unknown example $\mathbf{x}$.

In order to compute the main prediction, for a two-class classification problem, we can define

$$p_1(\mathbf{x}) = \frac{1}{K}\sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (f_b(\mathbf{x}) = 1)||,$$

$$p_{-1}(\mathbf{x}) = \frac{1}{K}\sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (f_b(\mathbf{x}) = -1)||.$$

The main prediction $y_m(\mathbf{x})$ corresponds to the mode:

$$y_m = \arg\max(p_1, p_{-1}).$$

The bias can be calculated as in Equation 8, and the variance $V(\mathbf{x})$ is

$$V(\mathbf{x}) = \frac{1}{K}\sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (y_m \neq f_b(\mathbf{x}))||.$$

Similarly easily computed are the unbiased, biased and net-variance:

$$V_u(\mathbf{x}) = \frac{1}{K}\sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (B(\mathbf{x}) = 0) \text{ and } (y_m \neq f_b(\mathbf{x}))||,$$

$$V_b(\mathbf{x}) = \frac{1}{K}\sum_{b=1}^{B} ||(\mathbf{x} \in T_b) \text{ and } (B(\mathbf{x}) = 1) \text{ and } (y_m \neq f_b(\mathbf{x}))||,$$

$$V_n(\mathbf{x}) = V_u(\mathbf{x}) - V_b(\mathbf{x}).$$

Average bias, variance, unbiased, biased and net variance, can be easily calculated averaging over all the examples.

4. Bias-Variance Analysis in SVMs

The bias-variance decomposition of the error represents a powerful tool to analyze learning processes in learning machines. According to the procedures described in the previous section, we measured bias and variance in SVMs, in order to study the relationships with different kernel types and their parameters. To accomplish this task we computed bias-variance decomposition of the error on different synthetic and "real" data sets.

4.1 Experimental Setup

In the experiments we employed seven different data sets, both synthetic and "real". *P2* is a synthetic bidimensional two-class data set;[1] each region is delimited by one or more of four simple polynomial and trigonometric functions (Figure 3). The synthetic data set *Waveform* is generated from a combination of two of three "base" waves; we reduced the original three classes of *Waveform* to two, deleting all samples pertaining to class 0. The other data sets are all from the UCI repository (Merz and Murphy, 1998). Table 4.1 summarizes the main features of the data sets used in the experiments.

1. The application `gensimple`, that we developed to generate the data, is freely available on line at `ftp://ftp.disi.unige.it/person/ValentiniG/BV/gensimple`.

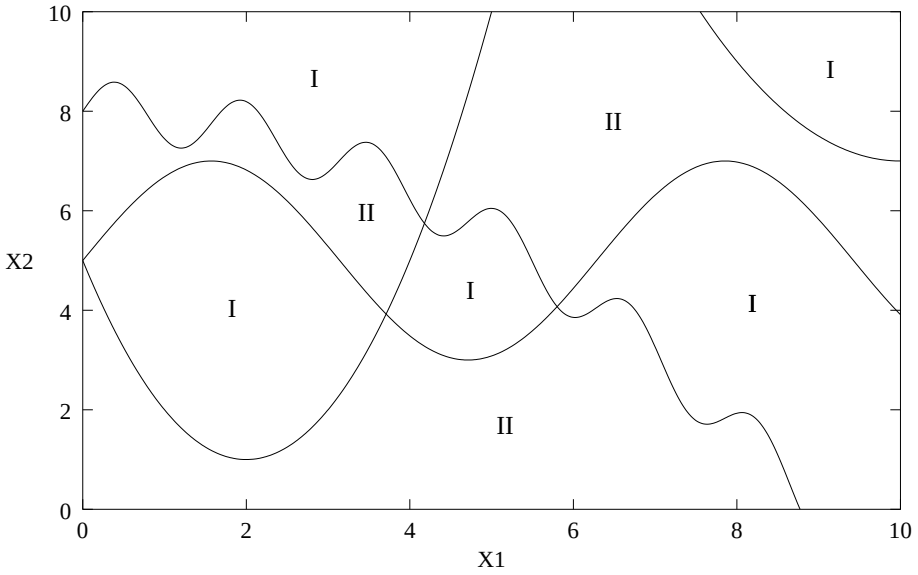


Figure 3: P2 data set, a bidimensional two class synthetic data set. Roman numbers label the regions belonging to the two classes.

Data set	# of attr.	# of tr. samples	# of tr. sets	# base tr. set	# of test samples
P2	2	100	400	synthetic	10000
Waveform	21	100	200	synthetic	10000
Grey-Landsat	36	100	200	4425	2000
Letter	16	100	200	614	613
Letter w. noise	16	100	200	614	613
Spam	57	100	200	2301	2300
Musk	166	100	200	3299	3299

Table 1: Data sets used in the experiments.

In order to perform a reliable evaluation of bias and variance we used small training set and large test sets. For synthetic data we generated the desired number of samples. For real data sets we used bootstrapping to replicate the data. In both cases we computed the main prediction, bias, unbiased and biased variance, net-variance according to the procedures explained in Section 3.1. In our experiments, the computation of James' variance and systematic effect (James, 2003) is reduced to the measurements of the net-variance and bias, and hence we did not explicitly compute these quantities (see Appendix *A* for details).

With synthetic data sets, we generated small training sets of about 100 examples and reasonably large test sets using computer programs. In fact small samples show bias and variance more clearly than having larger samples. We produced 400 different training sets for *P2* and 200 training sets for

Waveform. The test sets were chosen reasonably large (10000 examples) to obtain reliable estimates of bias and variance.

For real data sets we first divided the data into a training $\mathcal{D}$ and a test $\mathcal{T}$ sets. If the data sets had a predefined training and test sets reasonably large, we used them (as in *Grey-Landsat* and *Spam*), otherwise we split them in a training and test set of equal size. Then we drew from $\mathcal{D}$ bootstrap samples. We chose bootstrap samples much smaller than $|\mathcal{T}|$ (100 examples). More precisely we drew 200 data sets from $\mathcal{D}$, each one consisting of 100 examples uniformly drawn with replacement.

Summarizing, both with synthetic and real data sets we generated small training sets for each data set and a much larger test set. Then all the data were normalized in such a way that for each attribute the mean was 0 and the standard deviation 1. In all our experiments we used *NEURObjects* (Valentini and Masulli, 2002),[2] a C++ library for the development of neural networks and machine learning applications, and *SVM-light* (Joachims, 1999), a set of C applications for training and testing SVMs.

We developed and used the C++ application `analyze_BV`, to perform bias-variance decomposition of the error.[3] This application analyzes the output of a generic learning machine model and computes the main prediction, error, bias, net-variance, unbiased and biased variance using the $0/1$ loss function. Other C++ applications have been developed to process and analyze the results, using also Cshell scripts to train, test and analyze bias-variance decomposition of all the SVM models for each specific data set.

4.2 Experimental Tasks

To evaluate bias and variance in SVMs we conducted experiments with different kernels (Gaussian, polynomial and dot product) and different kernel parameters. For each kernel we considered the same set of values for the parameter C that controls the trade-off between training error and margin, ranging from $C = 0.01$ to $C = 1000$.

1. **Gaussian kernels**. We evaluated bias-variance decomposition varying the parameters σ of the kernel and the C parameter. In particular we analyzed:

   (a) The relationships between average error, bias, net-variance, unbiased and biased variance, the σ parameter of the kernel and the C parameter.

   (b) The relationships between generalization error, training error, number of support vectors and capacity with respect to σ.

   We trained RBF-SVM with all the combinations of the parameters σ and C, using the a set of values for σ ranging from $\sigma = 0.01$ to $\sigma = 1000$. We evaluated about 200 different RBF-SVM models for each data set.

2. **Polynomial kernels**. We evaluated bias-variance decomposition varying the degree of the kernel and the C parameter. In particular we analyzed the relationships between average error, bias, net-variance, unbiased and biased variance, the degree of the kernel and the C parameter.

2. This library may be downloaded from the web at `http://www.disi.unige.it/person/ValentiniG/NEURObjects`.
3. The source code is available at `ftp://ftp.disi.unige.it/person/ValentiniG/BV`. Moreover C++ classes for bias-variance analysis have been developed as part of the *NEURObjects* library.

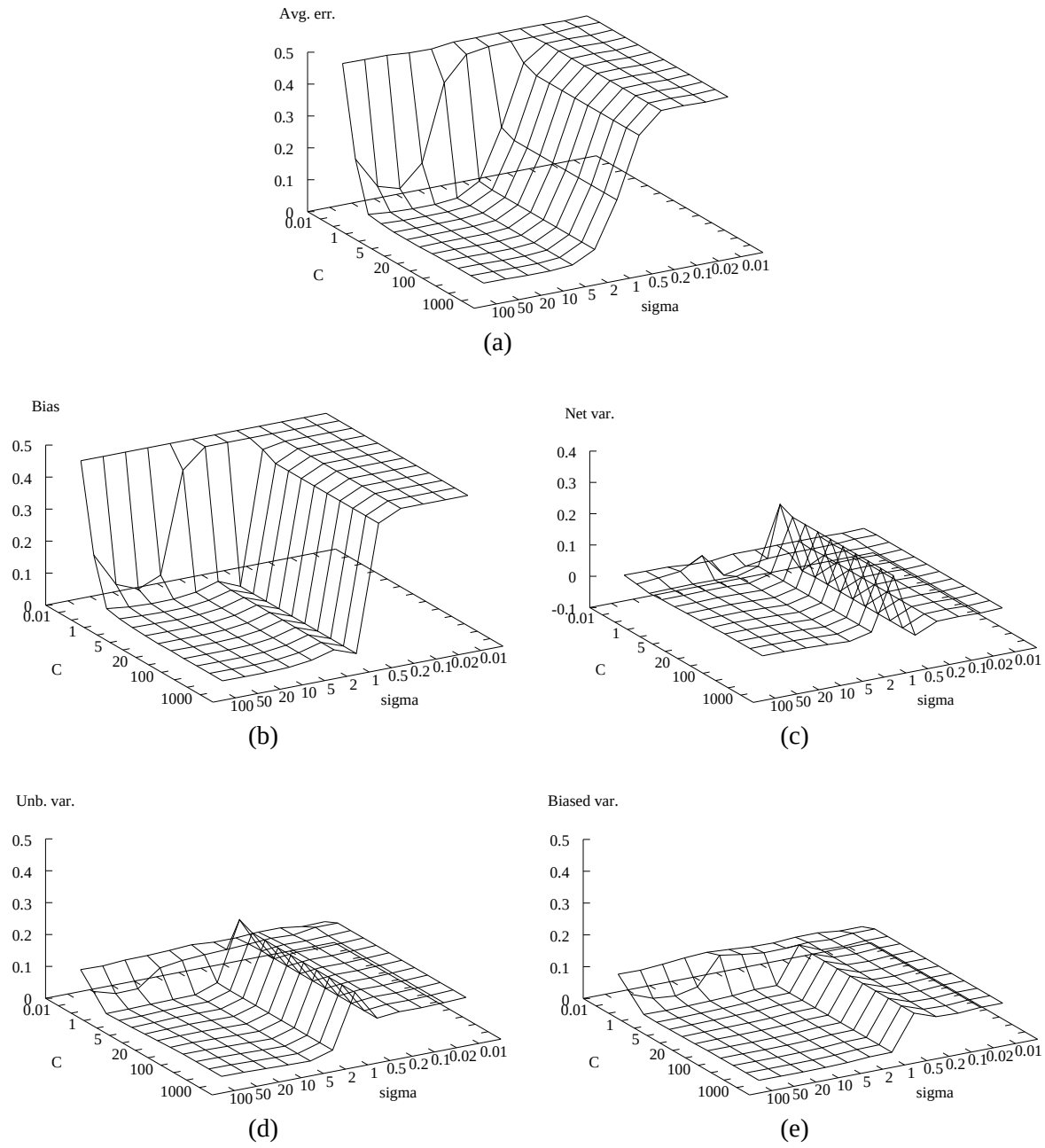


Figure 4: Grey-Landsat data set. Error (a) and its decomposition in bias (b), net variance (c), unbiased variance (d), and biased variance (e) in SVM RBF, varying both C and σ.

We trained polynomial-SVM with several combinations of the degree parameter of the kernel and C values, using all the polynomial degrees between 1 and 10, evaluating in such a way

about 120 different polynomial-SVM models for each data set. Following the heuristic of Jakkola, the dot product of polynomial kernel was divided by the dimension of the input data, to "normalize" the dot product before to raise to the degree of the polynomial.

3. **Dot product kernels**. We evaluated bias-variance decomposition varying the C parameter. We analyzed the relationships between average error, bias, net-variance, unbiased and biased variance and the parameter C (the regularization factor) of the kernel. We trained dot-product-SVM considering different values for the C parameter, evaluating in such a way 12 different dot-product-SVM models for each data set.

Each SVM model required the training of 200 different SVMs, one for each synthesized or bootstrapped data set, for a total of $(204+120+12)\times 200 = 67200$ trained SVM for each data set (134400 for the data set *P2*, as for this data set we used 400 data sets for each model). The experiments required the training of more than half million of SVMs, considering all the data sets and of course the testing of all the SVM previously trained in order to evaluate the bias-variance decomposition of the error of the different SVM models. For each SVM model we computed the main prediction, bias, net-variance, biased and unbiased variance and the error on each example of the test set, and the corresponding average quantities on the overall test set.

5. Results

In this section we present the results of the experiments. We analyzed bias-variance decomposition with respect to the kernel parameters considering separately Gaussian, polynomial and dot product SVMs, comparing also the results among different kernels. Here we present the main results. Full results, data and graphics are available by anonymous ftp at `ftp://ftp.disi.unige.it/person/ValentiniG/papers`

5.1 Gaussian Kernels

Figure 4 depicts the average loss, bias net-variance, unbiased and biased variance varying the values of σ and the regularization parameter C in *RBF-SVM* on the *Grey-Landsat* data set. We note that σ is the most important parameter: although for very low values of C the SVM cannot learn, independently of the values of σ, (Figure 4 a), the error, the bias, and the net-variance depend mostly on the σ parameter. In particular for low values of σ, bias is very high (Figure 4 b) and net-variance is 0, as biased and unbiased variance are about equal (Figure 4d and 4e). Then the bias suddenly drops (Figure 4b), lowering the average loss (Figure 4a), and then stabilizes for higher values of σ. Interestingly enough, in this data set (but also in others, data not shown), we note an increment followed by a decrement of the net-variance, resulting in a sort of "wave shape" of the net variance graph (Figure 4c).

Figure 5 shows the bias-variance decomposition on different data sets, varying σ, and for a fixed value of C, that is a sort of "slice" along the σ axis of the Figure 4. The plots show that average loss, bias, and variance depend significantly on σ for all the considered data sets, confirming the existence of a "high biased region" for low values of σ. In this region, biased and unbiased variance are about equal (net-variance $V_n = V_u - V_b$ is low). Then unbiased variance increases while biased variance decreases (Figure 5 a,b,c and d), and finally both stabilize for relatively high values of σ. Interestingly, the average loss and the bias do not increase for high values of σ, especially if C is high.

Bias and average loss increases with σ only for very small C values. Note that net-variance and bias show opposite trends only for small values of C (Figure 5 c). For larger C values the symmetric trend is limited only to $\sigma \leq 1$ (Figure 5 d), otherwise bias stabilizes and net-variance slowly decreases. Figure 6 shows more in detail the effect of the C parameter on bias-variance decomposition. For $C \geq 1$ there are no variations of the average error, bias and variance for a fixed value of σ. Note that for very low values of σ (Figure 6a and b) there is no learning. In the Letter-Two data set, as in other data sets (figures not shown), only for small C values we have variations in bias and variance values (Figure 6).

5.1.1 DISCRIMINANT FUNCTION COMPUTED BY THE SVM-RBF CLASSIFIER

In order to get insights into the behaviour of the SVM learning algorithm with Gaussian kernels we plotted the real-valued functions computed without considering the discretization step performed through the sign function. The real valued function computed by a Gaussian SVM is

$$f(\mathbf{x}, \alpha, b) = \sum_{i \in SV} y_i \alpha_i \exp(-\|\mathbf{x_i} - \mathbf{x}\|^2 / \sigma^2) + b,$$

where the α_i are the Lagrange multipliers found by the solution of the dual optimization problem, the $\mathbf{x_i} \in SV$ are the support vectors, that is the points for which $\alpha_i > 0$.

We plotted the surface computed by the Gaussian SVM with the synthetic data set *P2*. Indeed it is the only surface that can be easily visualized, as the data are bidimensional and the resulting real valued function can be easily represented through a wireframe three-dimensional surface. The SVMs are trained with exactly the same training set composed by 100 examples. The outputs are referred to a test set of 10000 examples, selected in an uniform way through all the data domain. In particular we considered a grid of equi-spaced data at 0.1 interval in a two dimensional 10×10 input space. If $f(\mathbf{x}, \alpha, b) > 0$ then the SVM matches up the example $\mathbf{x}$ with class 1, otherwise with class 2.

With small values of σ we have "spiky" functions: the response is high around the support vectors, and is close to 0 in all the other regions of the input domain (Figure 7). In this case we have overfitting: a large error on the test set (about 46 % with $\sigma = 0.01$ and 42.5 % with $\sigma = 0.02$), and a training error near to 0.

If we enlarge the values of σ we obtain a wider response on the input domain and the error decreases (with $\sigma = 0.1$ the error is about 37 %). With $\sigma = 1$ we have a smooth function that fits quite well the data (Figure 8). In this case the error drops down to about 13 %.

Enlarging too much σ we have a too smooth function (Figure 9 (a)), and the error increases to about 37 %: in this case the high bias is due to an excessive smoothing of the function. Increasing the values of the regularization parameter C (in order to better fit the data), we can diminish the error to about 15 %: the shape of the function now is less smooth (Figure 9 (b)).

As noted in Scholkopf and Smola (2002), using very large values of sigma, we have a very smooth discriminant function (in practice a plane), and increasing it even further does not change anything. Indeed, enlarging σ to 500 we obtain a plane (Fig 9 (c)), and a very biased function (error about 45 %), and even if we increment C, we can obtain better results, but always with a large error (about 35 %, Fig 9 (d)).

5.1.2 BEHAVIOR OF SVMS WITH LARGE σ VALUES

Fig 4 and 5 show that σ parameter plays a sort of smoothing effect, as the value of σ increases. In particular with large values of σ we did not observe any increment of bias nor decrement of variance. In order to get insights into this counter-intuitive behaviour we tried to answer these two questions:

1. Does the bias increase while variance decrease with large values of σ, and what is the combined effect of bias-variance on the error?

2. In this situation (large values for σ), what is the effect of the C parameter?

In Figure 5 we do not observe an increment of bias with large values of σ, but we limited our experiments to values of $\sigma \leq 100$. Here we investigate the effect for larger values of σ (from 100 to 1000).

In most cases, also increasing the values of σ right to 1000 we do not observe an increment of the bias and a substantial decrement of the variance. Only for low values of C, that is $C < 1$, the bias and the error increase with large values of σ (Figure 10). With the P2 data set the situation is different: in this case we observe an increment of the bias and the error with large values of σ, even if with large values of C the increment rate is lower (Figure 11 a and b).

Also with the musk data set we note an increment of the error with very large values of σ, but surprisingly this is due to an increment of the unbiased variance, while the bias is quite stable, at least for values of $C > 1$, (Figure 11 c and d).

Larger values of C counter-balance the bias introduced by large values of σ. But with some distributions of the data too large values of σ produce too smooth functions, and also incrementing C it is very difficult to fit the data. Indeed, the discriminant function computed by the RBF-SVM with the P2 data set (that is the function computed without considering the sign function) is too smooth for large values of σ: for $\sigma = 20$, the error is about 37%, due almost entirely to the large bias, (Figure 9 a), and for $\sigma = 500$ the error is about 45 % and also incrementing the C value to 1000, we obtain a surface that fits the data better, but with an error that remains large (about 35%). Indeed with very large values of σ the Gaussian kernel becomes nearly linear (Scholkopf and Smola, 2002) and if the data set is very far from being linearly separable, as with the *P2* data set (Figure 3), the error increases, especially in the bias component (Figure 11 (a) and (b)). Summarizing with large σ values bias can increment, while net-variance tends to stabilize, but this effect can be counter-balanced by larger C values.

5.1.3 RELATIONSHIPS BETWEEN GENERALIZATION ERROR, TRAINING ERROR, NUMBER OF SUPPORT VECTORS AND CAPACITY

Looking at Figure 4 and 5, we see that SVMs do not learn for small values of σ. Moreover the low error region is relatively large with respect to σ and C.

In this section we evaluate the relationships between the estimated generalization error, the bias, the training error, the number of support vectors and the estimated Vapnik Chervonenkis dimension, in order to answer the following questions:

1. Why SVMs do not learn for small values of σ?

2. Why we have a so large bias for small values of σ?

3. Can we use the variation of the number of support vectors to predict the "low error" region?

4. Is there any relationship between the bias, variance and VC dimension, and can we use this last one to individuate the "low error" region?

The generalization error, bias, training error, number of support vectors and the Vapnik Chervonenkis dimension are estimated averaging with respect to 400 SVMs (*P2* data set) or 200 SVMs (other data sets) trained with different bootstrapped training sets composed by 100 examples each one. The test error and the bias are estimated with respect to an independent and sufficiently large data set.

The *VC dimension* is estimated using the Vapnik's bound based on the radius R of the sphere that contains all the data (in the feature space), approximated through the sphere centered in the origin, and on the norm of the weights in the feature space (Vapnik, 1998). In this way the VC dimension is overestimated but it is easy to compute and we are interested mainly in the comparison of the VC dimension of different SVM models:

$$VC \leq R^2 \cdot \|\mathbf{w}\|^2 + 1,$$

where

$$\|\mathbf{w}\|^2 = \sum_{i \in SV} \sum_{j \in SV} \alpha_i \alpha_j K(\mathbf{x_i}, \mathbf{x_j}) y_i y_j$$

and

$$R^2 = \max_i K(\mathbf{x_i}, \mathbf{x_i}).$$

The number of support vectors is expressed as the halved ratio of the number (% *SV*) of support vectors with respect to the total number of training data:

$$\%SV = \frac{\#SV}{\#trainingdata \cdot 2}.$$

In the graphs shown in Figure 12 and Figure 13, on the left y axis is represented the error, training error and bias, and the halved ratio of support vectors. On the right y axis is reported the estimated Vapnik Chervonenkis dimension.

For very small values of σ the training error is very small (about 0), while the number of support vectors is very high, and high is also the error and the bias (Figure12 and 13). These facts support the hypothesis of overfitting problems with small values of σ. Indeed the real-valued function computed by the SVM (that is the function computed without considering the sign function, see Section 5.1.1) is very spiky with small values of σ (Figure 7). The response of the SVM is high only in small areas around the support vectors, while in all the other areas "not covered" by the support vectors the response is very low (about 0), that is the SVM is not able to get a decision, with a consequently very high bias. In the same region (small values for σ) the net variance is usually very small, for either one of these reasons: 1) biased and unbiased variance are almost equal because the SVM performs a sort of random guessing for the most part of the unknown data; 2) both biased and unbiased variance are about 0, showing that all the SVMs tend to answer in the same way independently of a particular instance of the training set (Figure 5 a, b and f). Enlarging σ we obtain a wider response on the input domain: the real-valued function computed by the SVM becomes smoother (Figure 8), as the "bumps" around the support vectors become wider and the

SVM can decide also on unknown examples. At the same time the number of support vectors decreases (Figure 12 and 13).

Considering the variation of the ratio of the support vectors with σ, in all data sets the trend of the rate of support vectors follows the error, with a sigmoid shape that sometimes becomes an U shape for small values of C (Figure12 and 13). This is not surprising because it is known that the support vector ratio offers an approximation of the generalization error of the SVMs (Vapnik, 1998). Moreover, on all the data sets the %SV decreases in the "stabilized" region, while in the transition region remains high. As a consequence the decrement in the number of support vectors shows that we are entering the "low error" region, and in principle we can use this information to detect this region.

In our experiments, an inspection of the support vectors relative to the *Grey-Landsat* and *Waveform* data sets found that most of the support vectors are shared in polynomial and Gaussian kernels with respectively the best degree and σ parameters. Even if these results confirmed the ones found by other authors (see e.g. Vapnik (1998)), it is worth noting that we did not perform a systematic study on this topic: we considered only two data sets and we compared only few hundreds of different SVMs.

In order to analyze the role of the VC dimension on the generalization ability of learning machines, we know from statistical learning theory that the form of the bounds of the generalization error E of SVMs is

$$E(f(\sigma,C)_n^k)) \leq E_{emp}(f(\sigma,C)_n^k)) + \Phi(\frac{h_k}{n}), \tag{9}$$

where $f(\sigma,C)_n^k$ represents the set of functions computed by an RBF-SVM trained with n examples and with parameters (σ_k,C_k) taken from a set of parameters $S = \{(\sigma_i,C_i), i \in \mathbb{N}\}$, E_{emp} represents the empirical error and Φ the confidence interval that depends on the cardinality n of the data set and on the VC dimension h_k of the set of functions identified by the actual selection of the parameters (σ_k,C_k). In order to obtain good generalization capabilities we need to minimize both the empirical risk and the confidence interval. According to Vapnik's bounds (Equation 9), in Figure 12 and 13 the lowest generalization error is obtained for a small empirical risk and a small estimated VC dimension.

But sometimes with relatively small values of *VC* we may have a very large error, as the training error and the number of support vectors increase with very large values of σ (Figure 12 a and 13 a). Moreover with a very large estimate of the VC dimension and low empirical error (Figure 12 and 13) we may have a relatively low generalization error. In conclusion it seems very difficult to use in practice these estimate of the VC dimension to infer the generalization abilities of the SVM. In particular it seems unreliable to use the VC dimension to infer the "low error" region of the RBF-SVM.

5.2 Polynomial and Dot Product Kernels

In this section we analyze the characteristics of bias-variance decomposition of the error in polynomial SVMs, varying the degree of the kernel and the regularization parameter C.

Error shows a U shape with respect to the degree. This shape depends on unbiased variance (Figure 14 a and b), or both by bias and unbiased variance (Figure 14 c and d). The U shape of the error with respect to the degree tends to be more flat for increasing values of C, and net-variance and bias show often opposite trends (Figure 15).

Average error and bias tends to be higher for low C and degree values, but, incrementing the degree, the error is less sensitive to *C* values (Figure 16).

Bias is flat (Figure 17 a) or decreasing with respect to the degree (Figure 15 b), or it can be constant or decreasing, depending on C (Figure 17 b). Unbiased variance shows an U shape (Figure 14 a and b) or it increases (Figure 14 c) with respect to the degree, and the net-variance follows the shape of the unbiased variance. Note that in the P2 data set (Figure 15) bias and net-variance follow the classical opposite trends with respect to the degree. This is not the case with other data sets (see, e.g. Figure 14).

For large values of C bias and net-variance tend to converge, as a result of the bias reduction and net-variance increment (Figure 18), or because both stabilize at similar values (Figure 16).

In dot product SVMs bias and net-variance show opposite trends: bias decreases, while net-variance and unbiased variance tend to increase with C (Figure 19). On the data set P2 this trend is not observed, as in this task the bias is very high and the SVM does not perform better than random guessing (Figure 19a). The minimum of the average loss for relatively low values of C is the result of the decrement of the bias and the increment of the net-variance: it is achieved usually before the crossover of bias and net-variance curves and before the stabilization of the bias and the net-variance for large values of C. The biased variance remains small independently of C.

5.3 Comparing Kernels

In this section we compare the bias-variance decomposition of the error with respect to the C parameter, considering Gaussian, polynomial and dot product kernels. For each kernel and for each data set the best results are selected. Table 5.3 shows the best results achieved by the SVM, considering each kernel and each data set used in the experiments. Interestingly enough in 3 data sets (Waveform, Letter-Two with added noise and Spam) there are not significant differences in the error between the kernels, but there are differences in bias, net-variance, unbiased or biased variance. In the other data sets Gaussian kernels outperform polynomial and dot product kernels, lowering bias or net-variance or both. Considering bias and net-variance, in some cases they are lower for polynomial or dot product kernel, showing that different kernels learn in different ways with different data.

Considering the data set *P2* (Figure 20 a, c, e), RBF-SVMs achieve the best results, as bias is lower. Unbiased variance is comparable between polynomial and Gaussian kernel, while net-variance is lower, as biased variance is higher for polynomial-SVM. In this task the bias of dot product SVM is very high. Also in the data set *Musk* (Figure 20 b, d, f) RBF-SVM obtains the best results, but in this case unbiased variance is responsible for this fact, while bias is similar. With the other data sets the bias is similar between RBF-SVM and polynomial-SVM, but for dot product SVM often the bias is higher (Figure 21 b, d, f). Interestingly enough RBF-SVM seems to be more sensible to the *C* value with respect to both polynomial and dot product SVM: for $C < 0.1$ in some data sets the bias is much higher (Figure 21 a, c, e). With respect to *C* bias and unbiased variance show sometimes opposite trends, independently of the kernel: bias decreases, while unbiased variance increases, but this does not occur in some data sets. We outline also that the shape of the error, bias and variance curves is similar between different kernels in all the considered data sets: that is, well tuned SVMs having different kernels tend to show similar trends of the bias and variance curves with respect to the *C* parameter.

	Parameters	**Avg. Error**	**Bias**	**Var. unb.**	**Var. bias.**	**Net Var.**
Data set *P2*						
RBF-SVM	$C = 20, \quad \sigma = 2$	0.1516	0.0500	0.1221	0.0205	0.1016
Poly-SVM	$C = 10, \quad degree = 5$	0.2108	0.1309	0.1261	0.0461	0.0799
D-prod SVM	$C = 500$	0.4711	0.4504	0.1317	0.1109	0.0207
Data set *Waveform*						
RBF-SVM	$C = 1, \quad \sigma = 50$	0.0706	0.0508	0.0356	0.0157	0.0198
Poly-SVM	$C = 1, \quad degree = 1$	0.0760	0.0509	0.0417	0.0165	0.0251
D-prod SVM	$C = 0.1$	0.0746	0.0512	0.0397	0.0163	0.0234
Data set *Grey-Landsat*						
RBF-SVM	$C = 2, \quad \sigma = 20$	0.0382	0.0315	0.0137	0.0069	0.0068
Poly-SVM	$C = 0.1, \; degree = 5$	0.0402	0.0355	0.0116	0.0069	0.0047
D-prod SVM	$C = 0.1$	0.0450	0.0415	0.0113	0.0078	0.0035
Data set *Letter-Two*						
RBF-SVM	$C = 5, \quad \sigma = 20$	0.0743	0.0359	0.0483	0.0098	0.0384
Poly-SVM	$C = 2, \quad degree = 2$	0.0745	0.0391	0.0465	0.0111	0.0353
D-prod SVM	$C = 0.1$	0.0908	0.0767	0.0347	0.0205	0.0142
Data set *Letter-Two with added noise*						
RBF-SVM	$C = 10, \quad \sigma = 100$	0.3362	0.2799	0.0988	0.0425	0.0563
Poly-SVM	$C = 1, \quad degree = 2$	0.3432	0.2799	0.1094	0.0461	0.0633
D-prod SVM	$C = 0.1$	0.3410	0.3109	0.0828	0.0527	0.0301
Data set *Spam*						
RBF-SVM	$C = 5, \quad \sigma = 100$	0.1263	0.0987	0.0488	0.0213	0.0275
Poly-SVM	$C = 2, \quad degree = 2$	0.1292	0.0969	0.0510	0.0188	0.0323
D-prod SVM	$C = 0.1$	0.1306	0.0965	0.0547	0.0205	0.0341
Data set *Musk*						
RBF-SVM	$C = 2, \quad \sigma = 100$	0.0884	0.0800	0.0217	0.0133	0.0084
Poly-SVM	$C = 2, \quad degree = 2$	0.1163	0.0785	0.0553	0.0175	0.0378
D-prod SVM	$C = 0.01$	0.1229	0.1118	0.0264	0.0154	0.0110

Table 2: Compared best results with different kernels and data sets. RBF-SVM stands for SVM with Gaussian kernel; Poly-SVM for SVM with polynomial kernel and D-prod SVM for SVM with dot product kernel. Var unb. and Var. bias. stand for unbiased and biased variance.

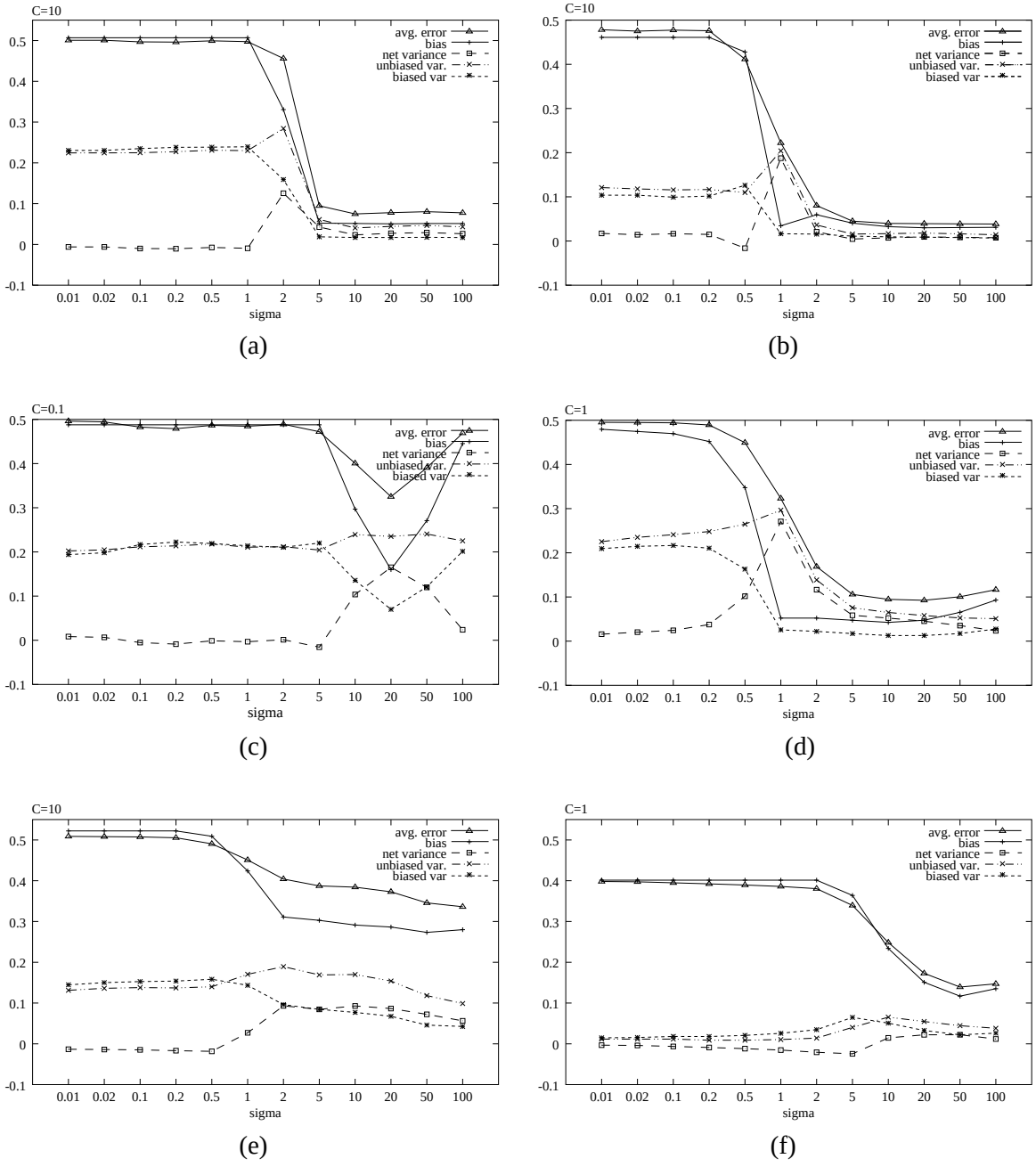


Figure 5: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in SVM RBF, varying σ and for fixed C values: (a) Waveform, (b) Grey-Landsat, (c) Letter-Two with $C = 0.1$, (c) Letter-Two with $C = 1$, (e) Letter-Two with added noise and (f) Spam.

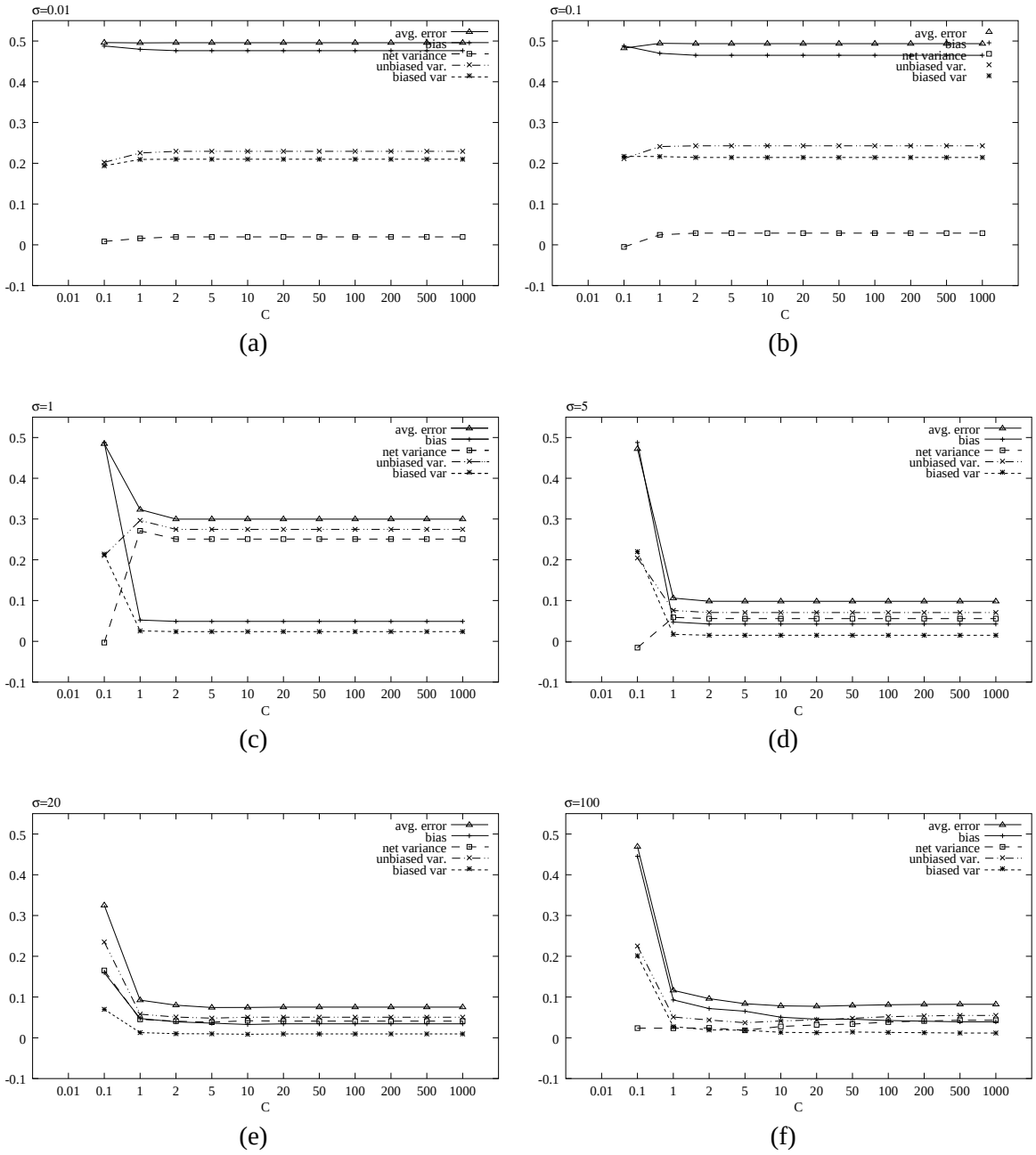


Figure 6: Letter-Two data set. Bias-variance decomposition of the error in bias, net variance, unbiased and biased variance in SVM RBF, while varying C and for some fixed values of σ: (a) $\sigma = 0.01$, (b) $\sigma = 0.1$, (c) $\sigma = 1$, (d) $\sigma = 5$, (e) $\sigma = 20$, (f) $\sigma = 100$.

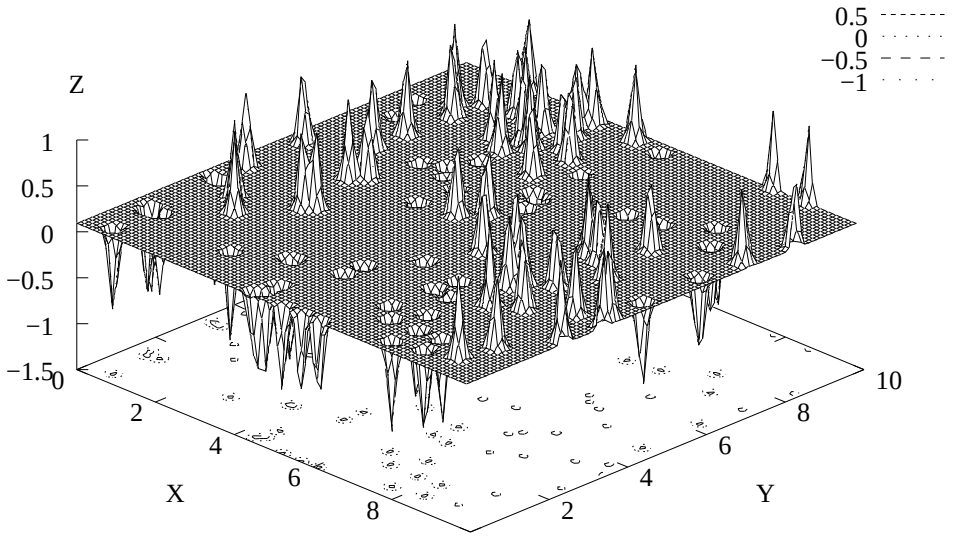


Figure 7: The real valued function computed by the SVM on the P2 data set with $\sigma = 0.01$, $C = 1$.

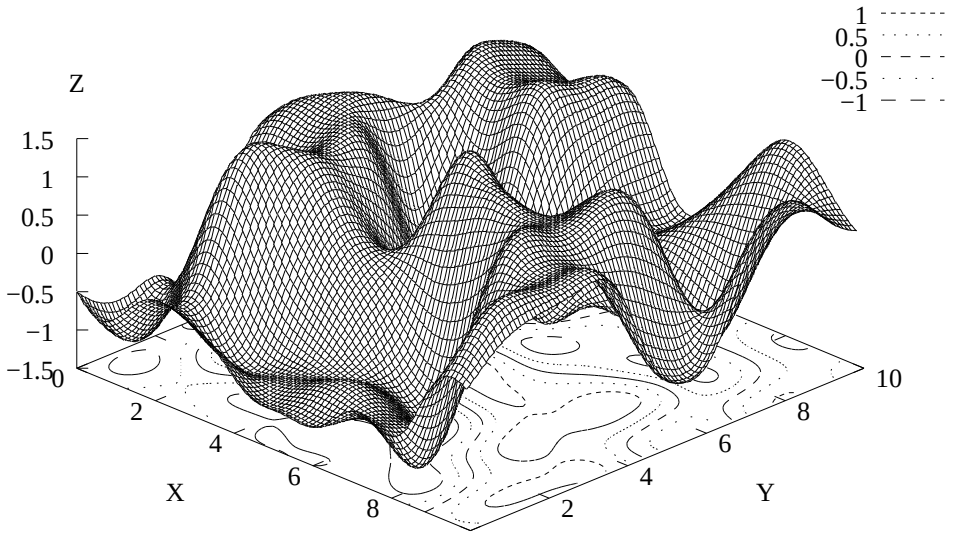


Figure 8: The real valued function computed by the SVM on the P2 data set, with $\sigma = 1$, $C = 1$.

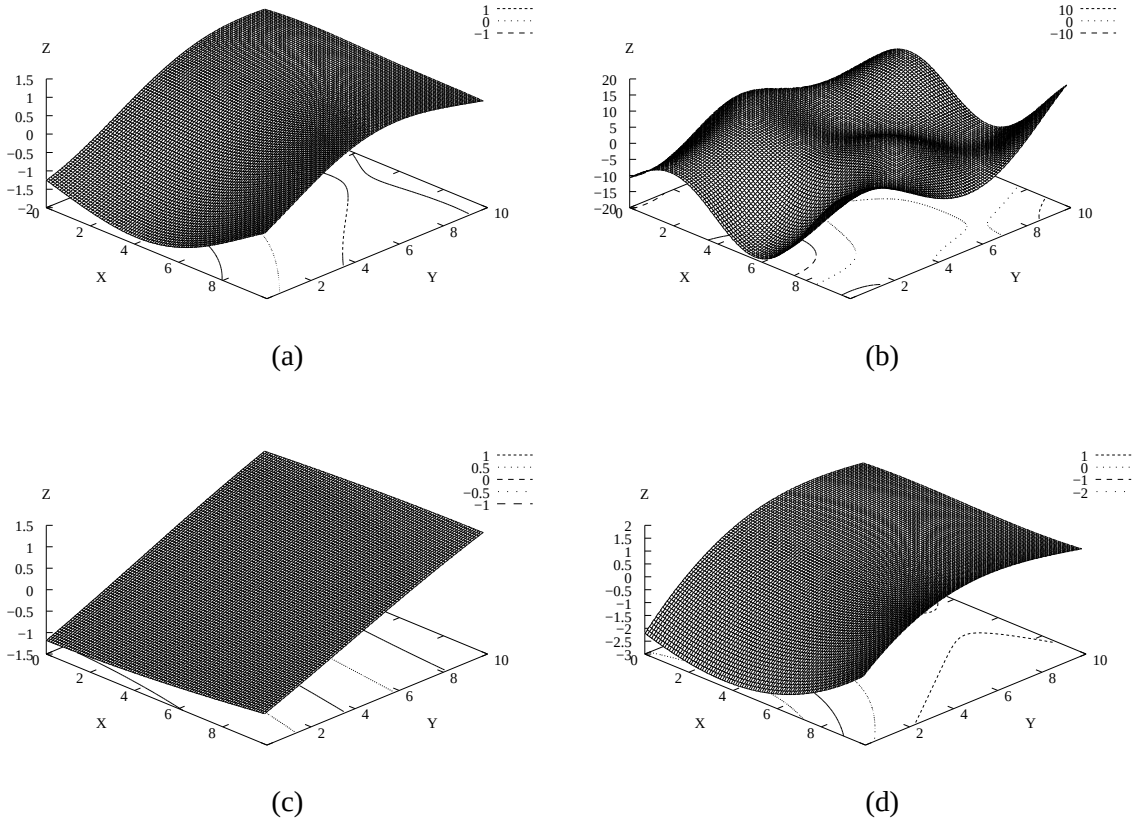


Figure 9: The real valued function computed by the SVM on the P2 data set. (a) $\sigma = 20$ $C = 1$, (b) $\sigma = 20$ $C = 1000$, (c) $\sigma = 500$ $C = 1$, (d) $\sigma = 500$ $C = 1000$.

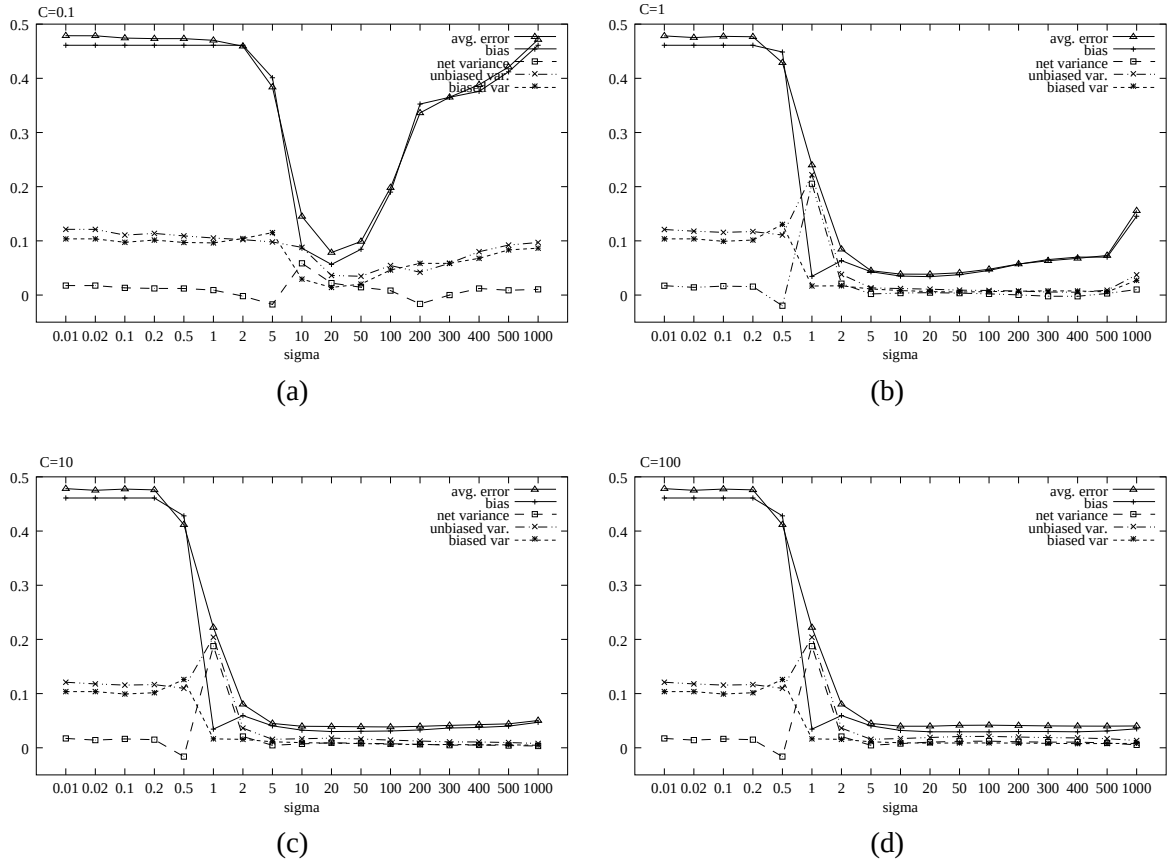


Figure 10: Grey-Landsat data set. Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in SVM RBF, while varying σ and for some fixed values of C: (a) $C = 0.1$, (b) $C = 1$, (c) $C = 10$, (d) $C = 100$.

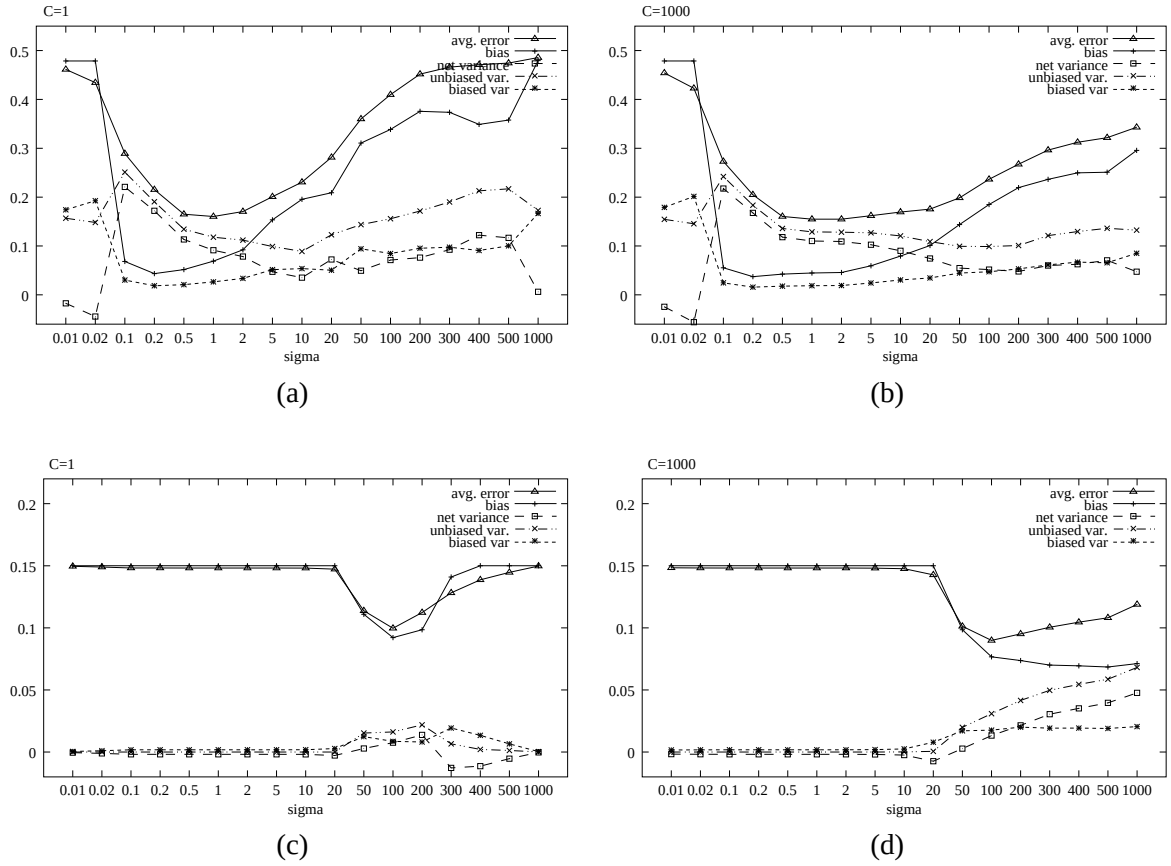


Figure 11: Bias-variance decomposition of the error in bias, net variance, unbiased and biased variance in SVM RBF, while varying σ and for some fixed values of C: (a) P2, with $C = 1$, (b) P2, with $C = 1000$, (c) Musk, with $C = 1$, (d) Musk, with $C = 1000$.

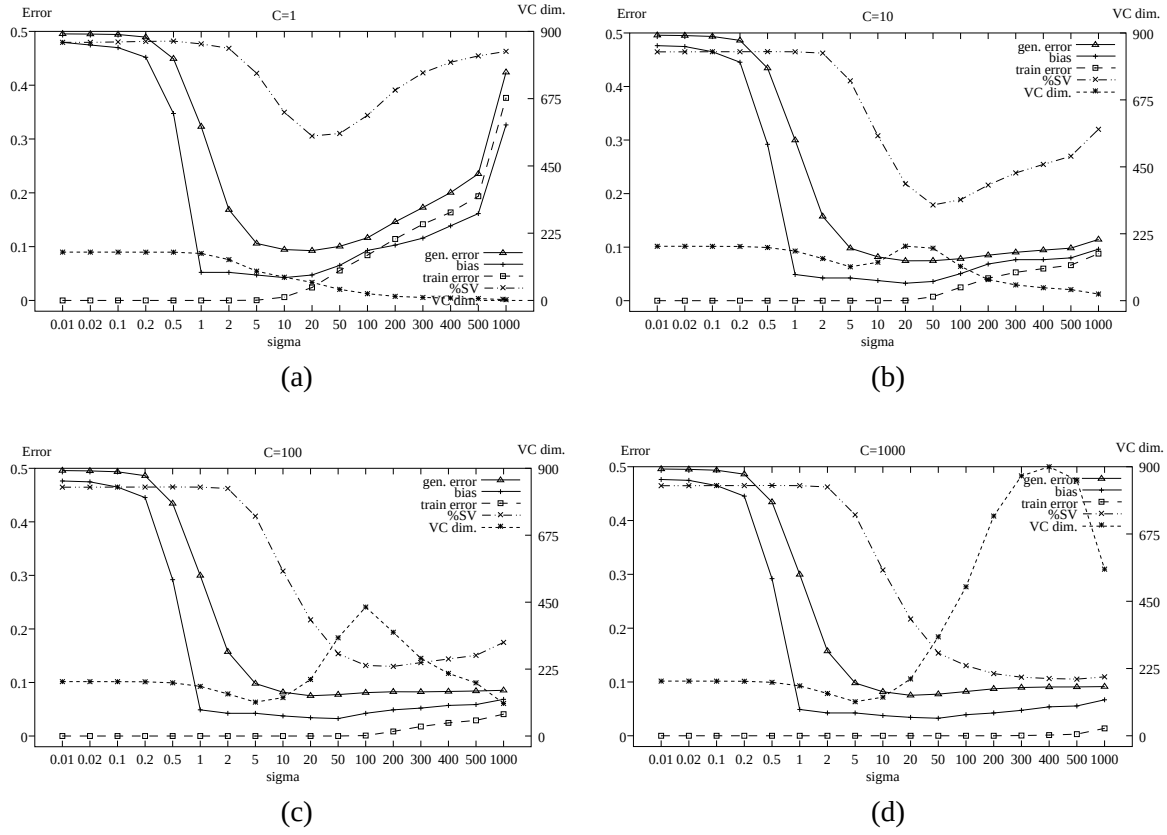


Figure 12: Letter-Two data set. Error, bias, training error, support vector rate, and estimated VC dimension in SVM RBF, while varying the σ parameter and for some fixed values of C: (a) $C = 1$, (b) $C = 10$, (c) $C = 100$, and $C = 1000$.

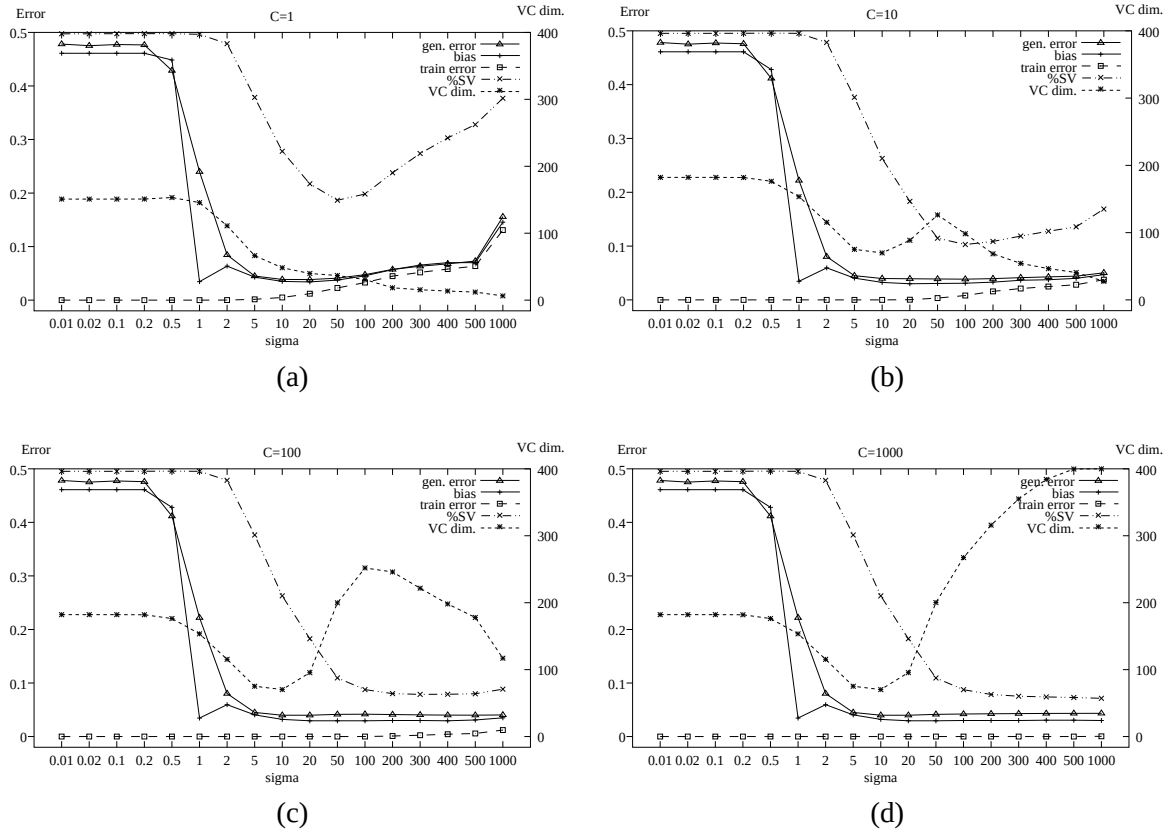


Figure 13: Grey-Landsat data set. Error, bias, training error, support vector rate, and estimated VC dimension in SVM RBF, while varying the σ parameter and for some fixed values of C: (a) $C = 1$, (b) $C = 10$, (c) $C = 100$, and $C = 1000$.

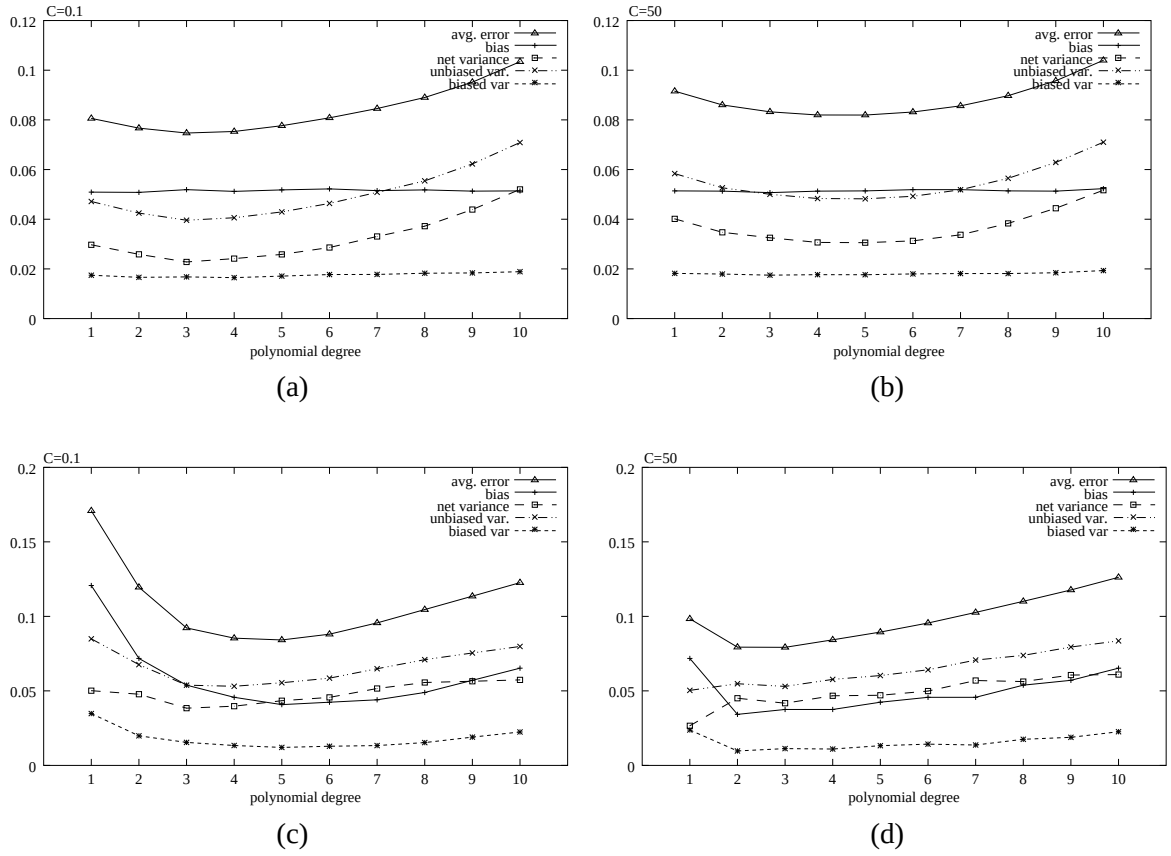


Figure 14: Bias-variance decomposition of the error in bias, net variance, unbiased and biased variance in polynomial SVM, while varying the degree and for some fixed values of C: (a) Waveform, $C = 0.1$, (b) Waveform, $C = 50$, (c) Letter-Two, $C = 0.1$, (d) Letter-Two, $C = 50$.

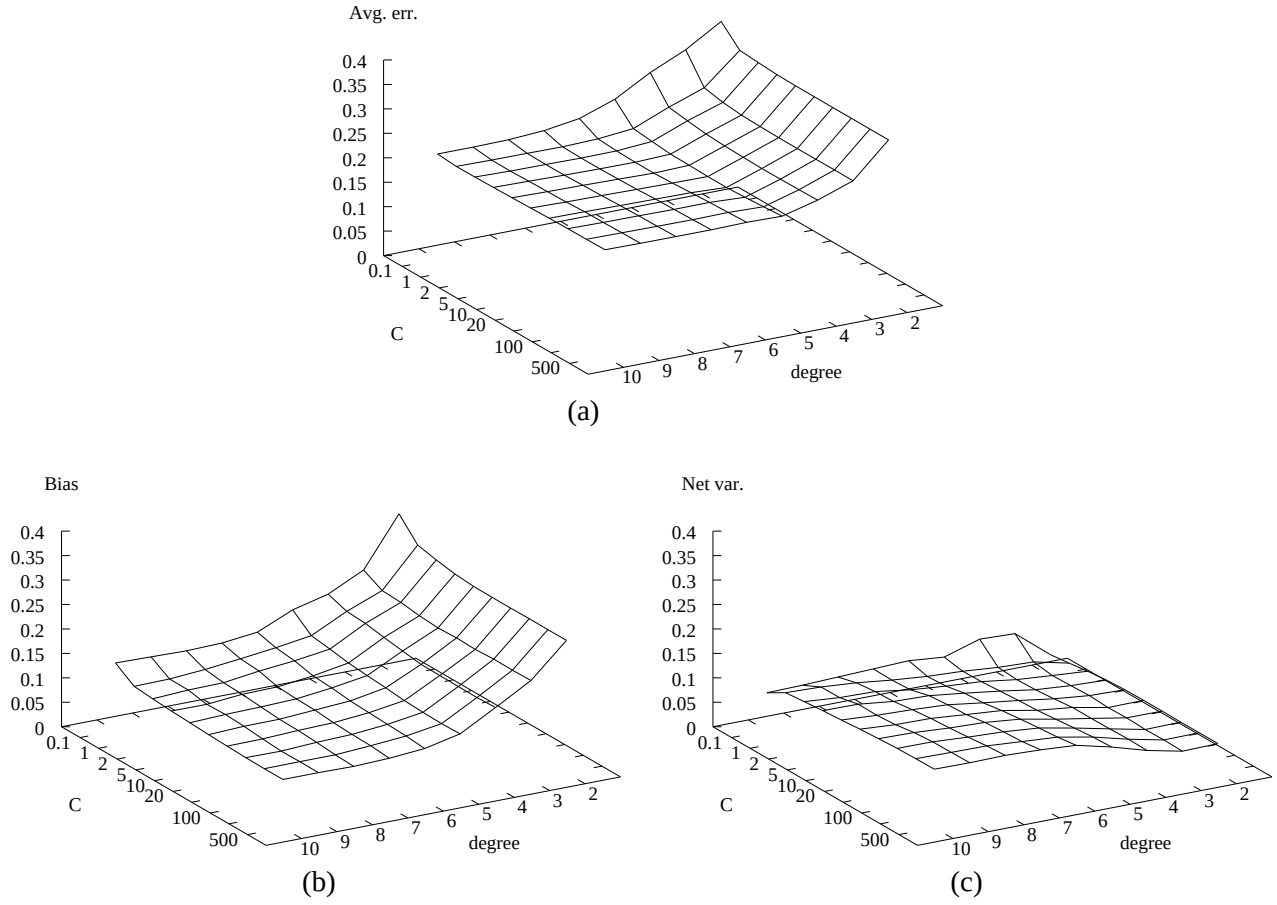


Figure 15: P2 data set. Error (a) and its decomposition in bias (b) and net variance (c), varying both *C* and the polynomial degree.

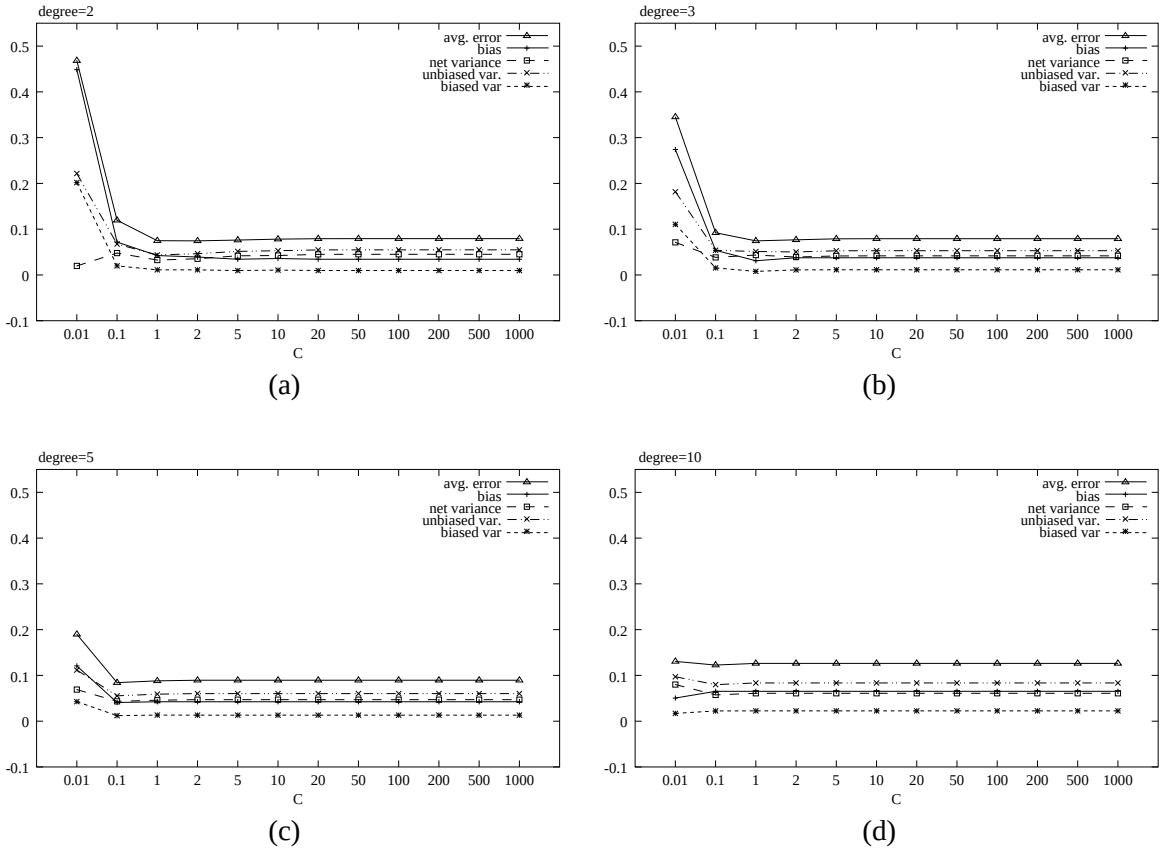


Figure 16: Letter-Two data set. Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in polynomial SVM, while varying C and for some polynomial degrees: (a) $degree = 2$, (b) $degree = 3$, (c) $degree = 5$, (d) $degree = 10$

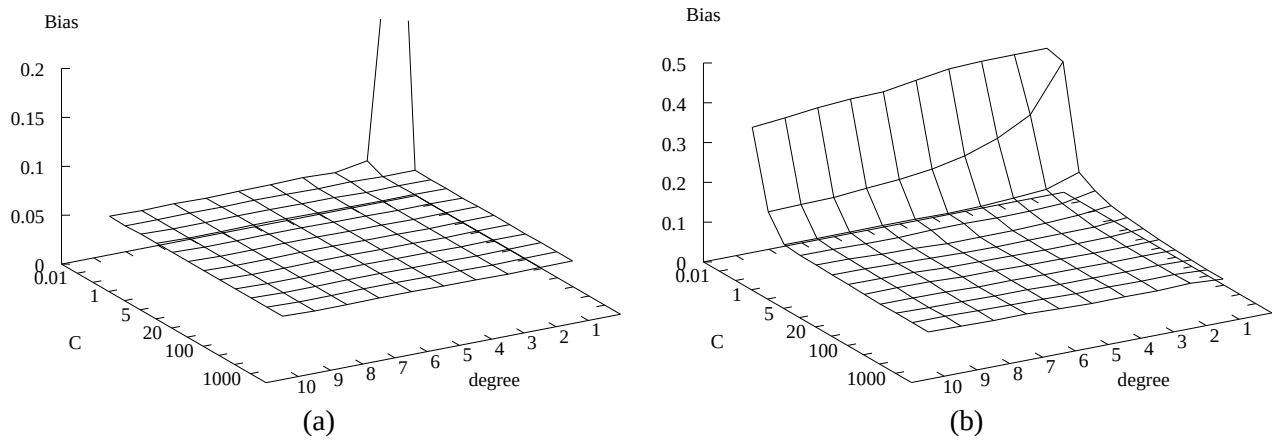


Figure 17: Bias in polynomial SVMs with (a) Waveform and (b) Spam data sets, varying both C and polynomial degree.

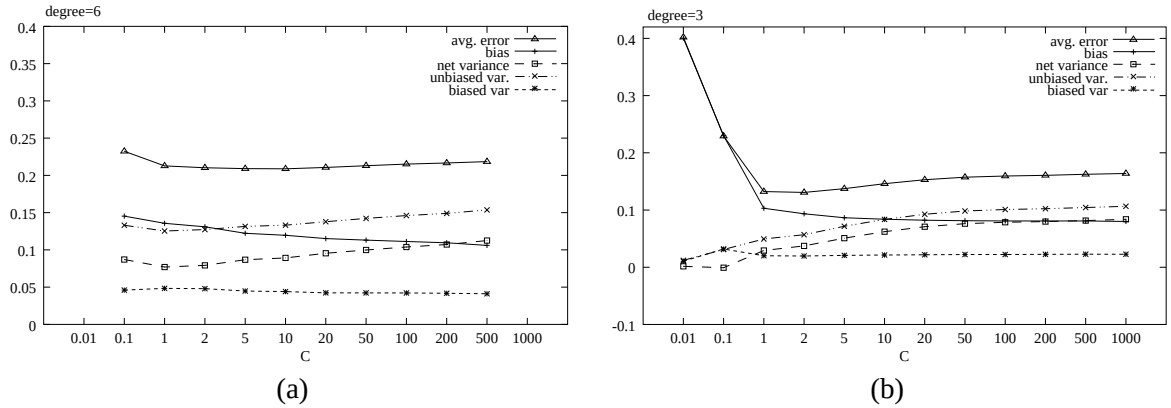


Figure 18: Bias-variance decomposition of the error in bias, net variance, unbiased and biased variance in polynomial SVM, varying C: (a) P2 data set with $degree = 6$, (b) Spam data set with $degree = 3$.

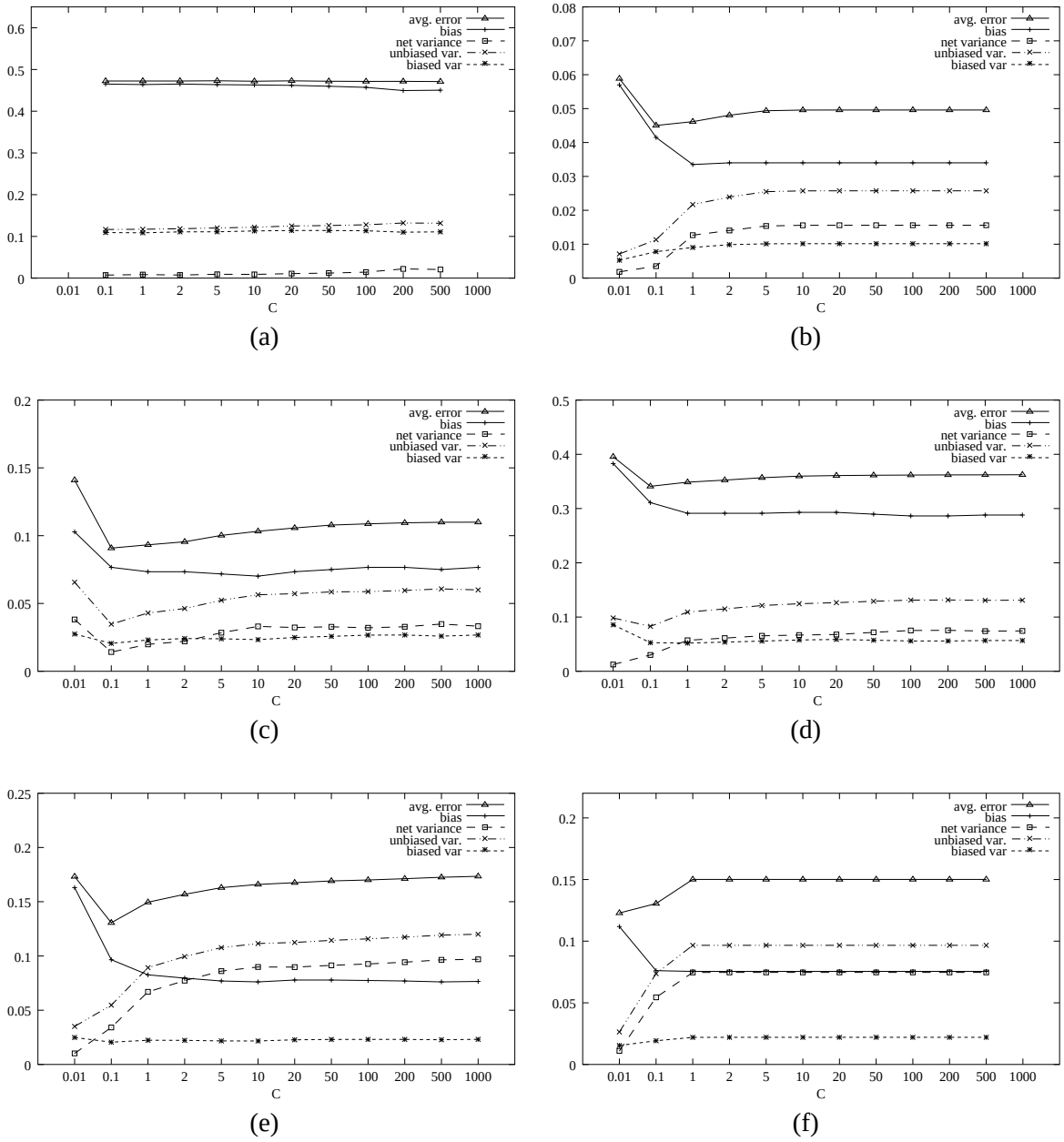


Figure 19: Bias-variance decomposition of error in bias, net variance, unbiased and biased variance in dot product SVM, varying C: (a) P2, (b) Grey-Landsat, (c) Letter-Two, (d) Letter-Two with added noise, (e) Spam, (f) Musk.

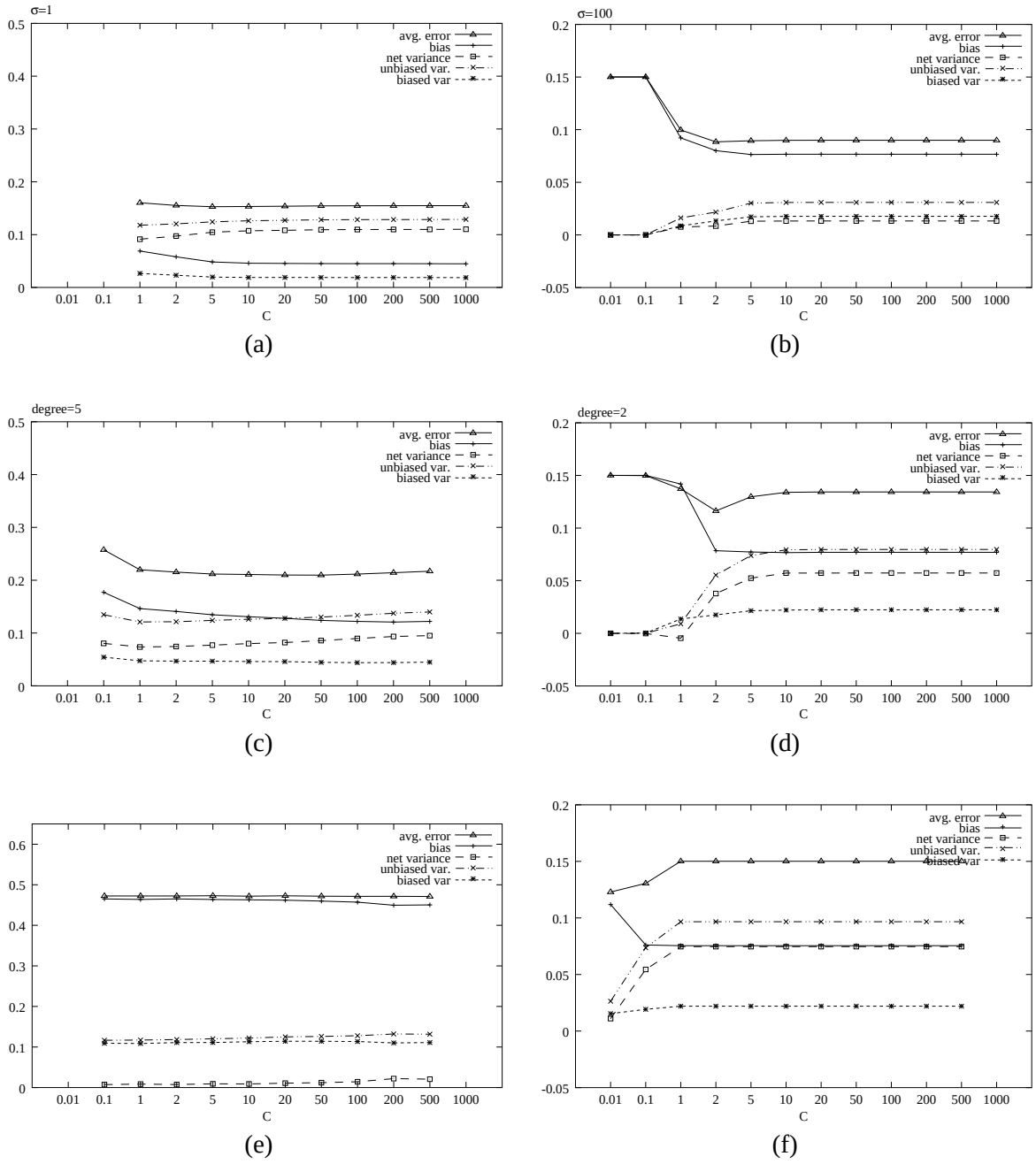


Figure 20: Bias-variance decomposition of the error in bias, net variance, unbiased and biased variance with respect to *C*, considering different kernels. (a) P2, Gaussian; (b) Musk, Gaussian (c) P2, polynomial; (d) Musk, polynomial; (e) P2, dot product; (f) Musk, dot product.

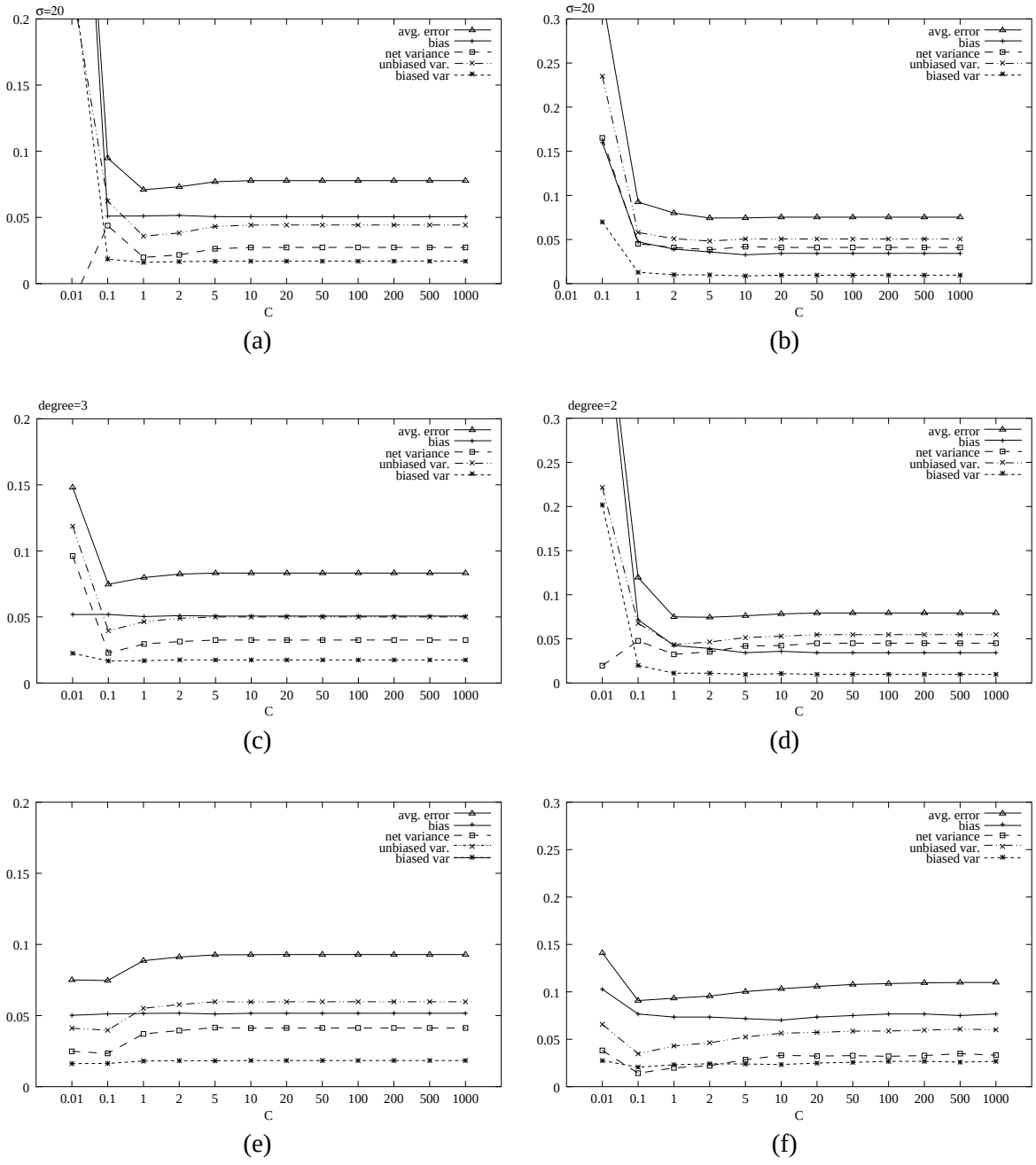


Figure 21: Bias-variance decomposition of the error in bias, net variance, unbiased and biased variance with respect to C, considering different kernels. (a) Waveform, Gaussian; (b) Letter-Two, Gaussian (c) Waveform, polynomial; (d) Letter-Two, polynomial; (e) Waveform, dot product; (f) Letter-Two, dot product.

Kernel type	**Avg. Error**	**Bias**	**Var. unb.**	**Var. bias.**	**Net Var.**
RBF	0.0901 ± 0.0087	0.0805 ± 0.0126	0.0237 ± 0.0039	0.0141 ± 0.0025	0.0096 ± 0.0019
Poly	0.1158 ± 0.0069	0.0782 ± 0.0083	0.0585 ± 0.0071	0.0109 ± 0.0018	0.0376 ± 0.0047
D-prod	0.1305 ± 0.0133	0.1179 ± 0.0140	0.0285 ± 0.0084	0.0159 ± 0.0045	0.0126 ± 0.0035

Table 3: Evaluation of the variation of the estimated values of bias variance decomposition with the *Musk* data set. RBF-SVM stands for SVM with Gaussian kernel; Poly-SVM for SVM with polynomial kernel and D-prod SVM for SVM with dot product kernel. Net Var. Var unb. and Var. bias. stand for net, unbiased and biased variance. For each value is represented the mean value over 100 replicated experiments and the corresponding value of the standard deviation.

In our experiments we used relatively small training sets (100 examples), while the number of input variables ranged from 2 (*P2* data set) to 166 (*Musk* data set). Hence, even if for each SVM model (that is for each combination of SVM parameters) we used 200 training sets $D_i, 1 \leq i \leq 200$ in order to train 200 different classifiers f_{D_i}, you could wonder whether the estimated quantities (average error, bias, net-variance, unbiased and biased variance) could be noisy. An extensive evaluation of the sensitivity of the estimated quantities to the sampling procedure would be very expensive. Indeed if we replicate only 10 times our experiments on all the data sets, we should train and test more than 5 millions of different SVMs. Anyway, in order to get insights about this problem, we performed 100 replicates of our experiments limited only to the *Musk* data set (that is the data set with the largest dimensionality in our experiments), for a subset of the parameters near the optimal ones. We found that the standard deviation of the estimated values is not too large. For instance, considering the best model for Gaussian, polynomial and dot product kernels we obtained the values shown in Table 5.3. It seems that the computed quantities are not too noisy, even if we need more experiments to confirm this result.

5.4 Bias-Variance Decomposition with Noisy Data

While the estimation of the noise is quite straightforward with synthetic data, it is a difficult task with "real" data James (2003). For these reasons, and in order to simplify the computation and the overall analysis, in our experiments we did not explicitly consider noise.

Anyway, noise can play a significant role in the bias-variance analysis. Indeed, according to Domingos, with the $0/1$ loss the noise is linearly added to the error with a coefficient equal to $2P_D(f_D(\mathbf{x}) = y_*) - 1$ (Equation 6). Hence, if the classifier is accurate, that is if $P_D(f_D(\mathbf{x}) = y_*) \gg 0.5$, then the noise $N(\mathbf{x})$, if present, influences the expected loss. In the opposite situation also, with very bad classifiers, that is when $P_D(f_D(\mathbf{x}) = y_*) \ll 0.5$, the noise influences the overall error in the opposite sense: it reduces the expected loss. If $P_D(f_D(\mathbf{x}) = y_*) \approx 0.5$, that is if the classifier performs a sort of random guessing, then $2P_D(f_D(\mathbf{x}) = y_*) - 1 \approx 0$ and the noise has no substantial impact on the error.

Hence if we know that the noise is small we can disregard it, but what about the effect of noise when it is present but not explicitly considered in the bias-variance decomposition of the error? The analysis of the results in the data set Letter-Two without and with 20 % added noise shows that the

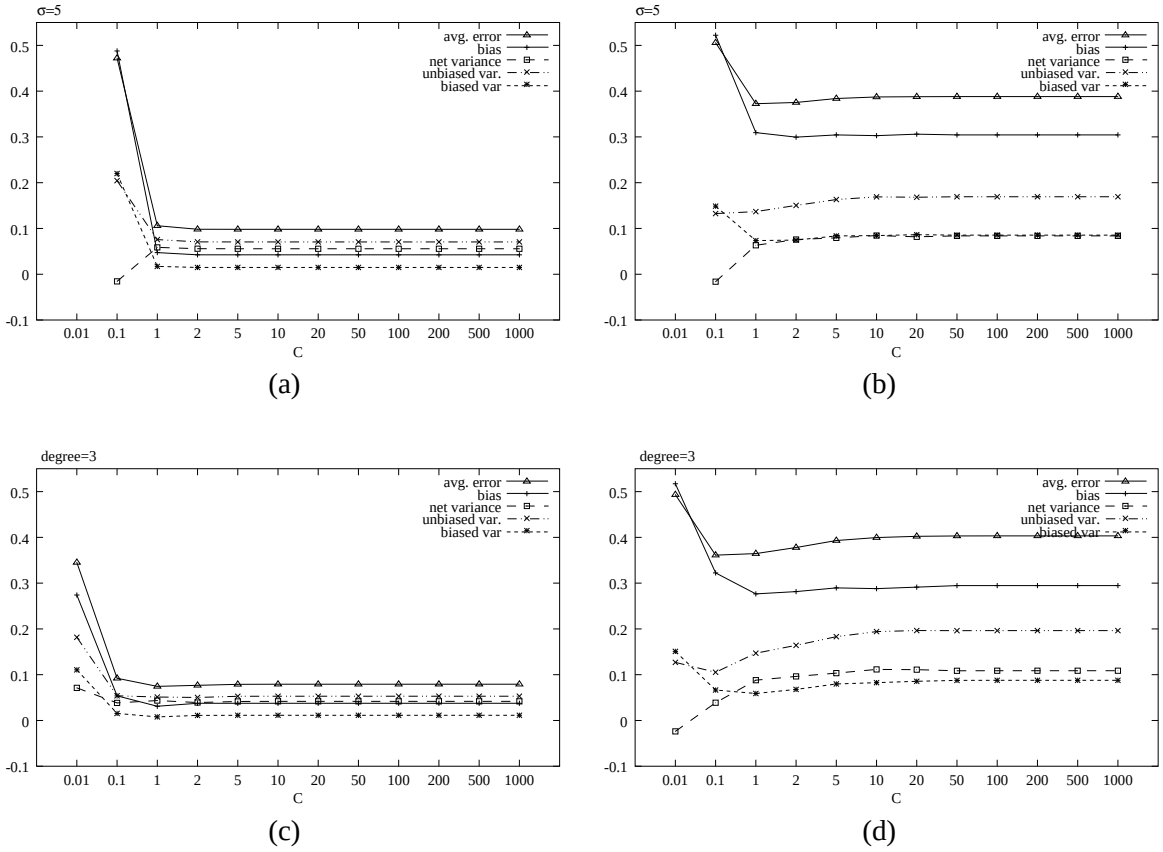


Figure 22: Effect of noise on bias and variance. The bias-variance decomposition of the error is shown while varying the C regularization parameter with polynomial and Gaussian kernels. (a) Letter-Two: Gaussian kernel, $\sigma = 5$, (b) Letter-Two with added noise: Gaussian kernel, $\sigma = 5$, (c) Letter-Two: polynomial kernel, degree = 3, (d) Letter-Two with added noise: polynomial kernel, degree = 3.

main effect of noise in this specific situation consists in incrementing the bias and consequently the average error. Indeed, with Gaussian kernels (Figure 22 (a) and (b)) the bias is raised to about 0.3, with an increment of about 0.25 with respect to the data set without noise, while the net-variance is incremented only by about 0.02, as the increment of the unbiased variance is counter-balanced by the increment of the biased variance. A similar behavior is registered also with polynomial (Figure 22 (c) and (d)) and dot product kernels (Figure 19 (c) and (d)).

6. Characterization of Bias-Variance Decomposition of the Error

Despite the differences observed in different data sets, common patterns of bias and variance can be detected for each of the kernels considered in this study. Each kernel presents a specific charac-

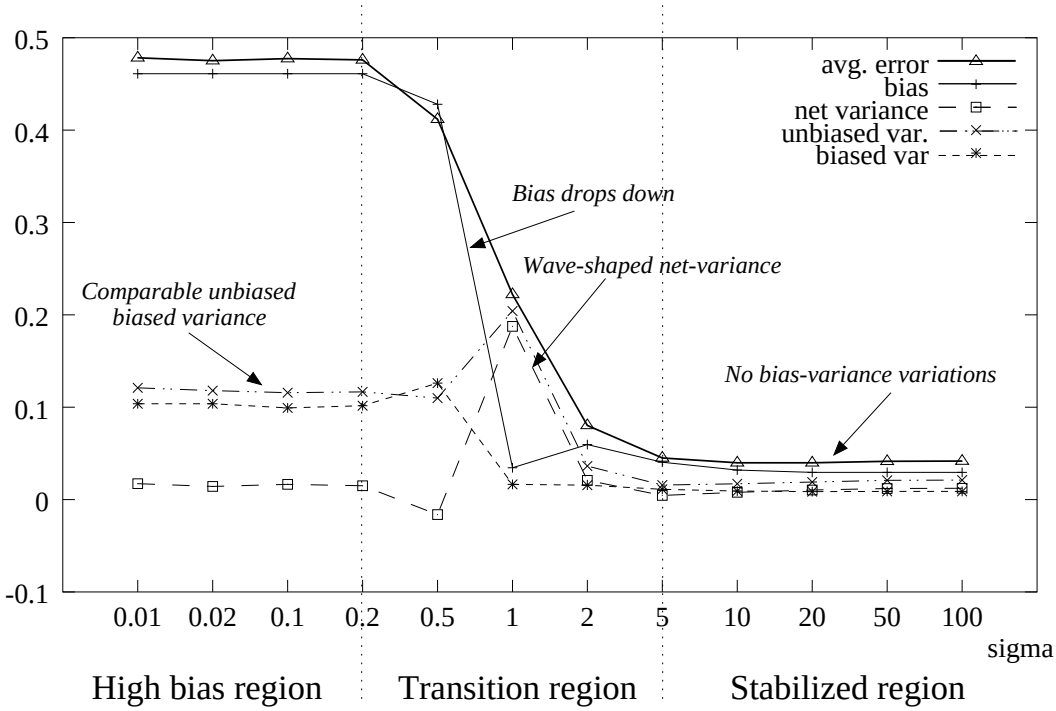


Figure 23: The 3 regions of the error in RBF-SVM with respect to σ.

terization of bias and variance with respect to its specific parameters, as explained in the following sections.

6.1 Gaussian Kernels

Error, bias, net-variance, unbiased and biased variance show a common trend in the 7 data sets we used in the experiments. Some differences, of course, arise in the different data sets, but we can distinguish three different regions in the error analysis of *RBF-SVM*, with respect to increasing values of σ (Figure 23):

1. **High bias region**. For low values of σ, error is high: it depends on high bias. Net-variance is about 0 as biased and unbiased variance are equivalent. In this region there are no remarkable fluctuations of bias and variance: both remain constant, with high values of bias and comparable values of unbiased and biased variance, leading to net-variance values near to 0. In some cases biased and unbiased variance are about equal, but different from 0, in other cases they are equal, but near to 0.

2. **Transition region**. Suddenly, for a critical value of σ, the bias decreases rapidly. This critical value depends also on C: for very low values of C, we have no learning, then for higher values the bias drops. Higher values of C cause the critical value of σ to decrease (Figure 4 (b) and 5). In this region the increase in net-variance is lower than the decrease in bias: so the average error decreases. The boundary of this region can be determined at the point where the error

stops decrementing. This region is characterized also by a particular trend of the net-variance. We can distinguish two main behaviors:

(a) **Wave-shaped net-variance**. Net-variance first increases and then decreases, producing a wave-shaped curve with respect to σ. The initial increment of the net-variance is due to the simultaneous increment of the unbiased variance and decrement of the biased variance. In the second part of the transition region, biased variance stabilizes and unbiased variance decreases, producing a parallel decrement of the net-variance. The rapid decrement of the error with σ is due to the rapid decrement of the bias, after which the bias stabilizes and the further decrement of the error with σ is determined by the net-variance reduction (Figure 4c, 5).

(b) **Semi-wave-shaped net-variance**. In other cases the net-variance curve with σ is not so clearly wave-shaped: the descending part is very reduced (Figure 5 e, f). In particular in the musk data set we have a continuous increment of the net-variance (due to the continuous growing of the unbiased variance with σ), and no wave-shaped curve is observed (at least for $C > 10$, Figure 11 d).

In both cases the increment of net-variance is slower than the increment in bias: as a result, the average error decreases.

3. **Stabilized region**. This region is characterized by small or no variations in bias and net-variance. For high values of σ both bias and net-variance stabilize and the average error is constant (Figure 4, 5). In other data sets the error increases with σ, because of the increment of the bias (Figure 11 a,b) or the unbiased variance (Figure 11 c,d).

In the first region, bias rules SVM behavior: in most cases the bias is constant and close to 0.5, showing that we have a sort of random guessing, without effective learning. It appears that the area of influence of each support vector is too small (Figure 7), and the learning machine overfits the data. This is confirmed by the fact that in this region the training error is about 0 and almost all the training points are support vectors.

In the transition region, the SVM starts to learn, adapting itself to the data characteristics. Bias rapidly goes down (at the expenses of a growing net-variance), but for higher values of σ (in the second part of the transition region), sometimes net-variance also goes down, working to lower the error (Figure 5).

Even if the third region is characterized by no variations in bias and variance, sometimes for low values of C, the error increases with σ (Figure 10 a, 12 a), as a result of the bias increment; on the whole RBF-SVMs are sensitive to low values of C: if C is too low, then bias can grow quickly. High values of C lower the bias(Figure 12 c, d).

6.2 Polynomial and Dot Product Kernels

For polynomial and dot product SVMs, we have also characterized the behavior of SVMs in terms of average error, bias, net-variance, unbiased and biased variance, even if we are not able to distinguish between different regions clearly defined.

However, common patterns of the error curves with respect to the polynomial degree, considering bias, net-variance and unbiased and biased variance can be noticed.

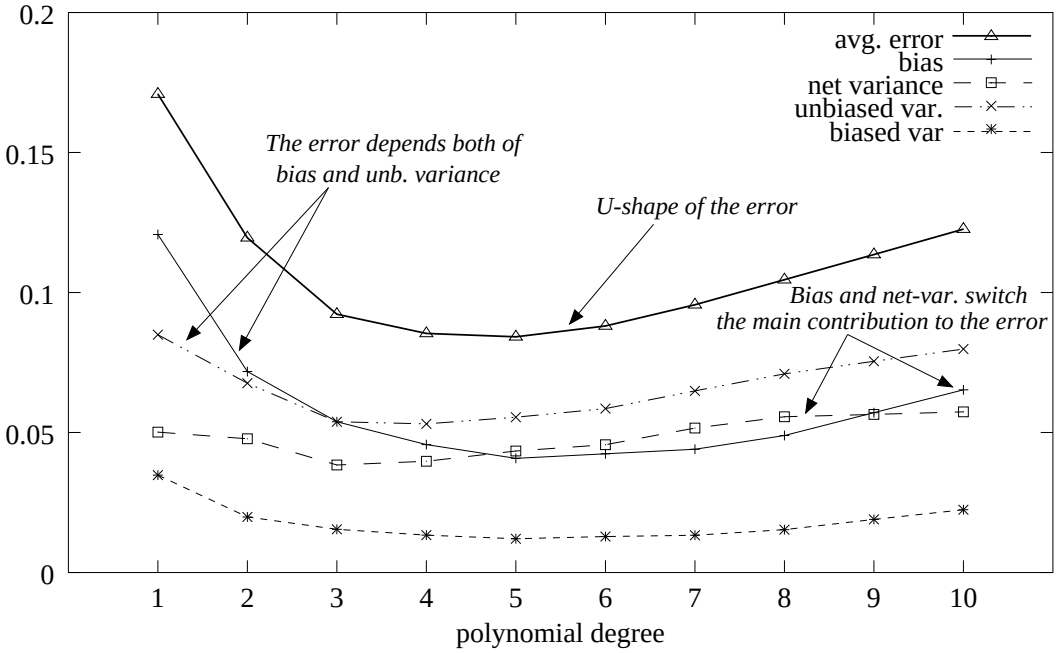


Figure 24: Behaviour of polynomial SVM with respect of the bias-variance decomposition of the error.

The average loss curve shows in general a U shape with respect to the polynomial degree, and this shape may depend on both bias and unbiased variance or in some cases mostly on the unbiased variance according to the characteristics of the data set. From these general observations we can schematically distinguish two main global pictures of the behaviour of polynomial SVM with respect to the bias-variance decomposition of the error:

1. **Error curve shape bias-variance dependent**.
   In this case the shape of the error curve is dependent both on the unbiased variance and the bias. The trend of bias and net-variance can be symmetric or they can also have non coincident paraboloid shape, depending on C parameter values (Figure 14 c, d and 15). Note that bias and net variance show often opposite trends (Figure 15).

2. **Error curve shape unbiased variance dependent**.
   In this case the shape of the error curve is mainly dependent on the unbiased variance. The bias (and the biased variance) tend to be degree independent, especially for high values of C (Figure 14 a, b) .

Figure 24 schematically summarizes the main characteristics of the bias-variance decomposition of error in polynomial SVM. Note however that the error curve depends for the most part on both variance and bias: the prevalence of the unbiased variance (Figure 14 a, b) or the bias seems to depend mostly on the distribution of the data.

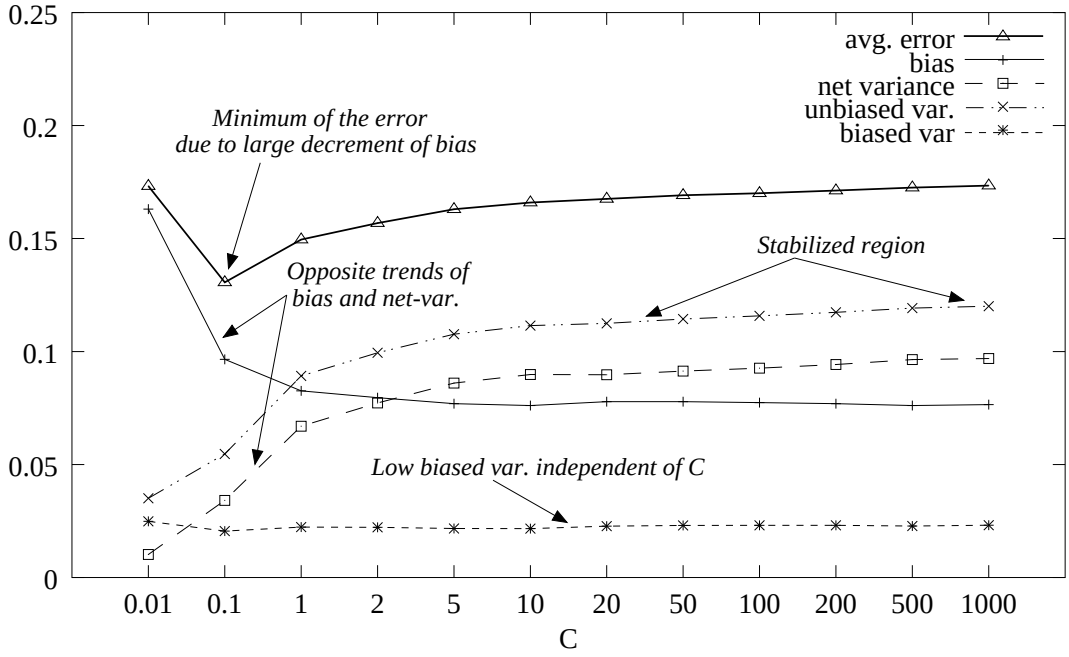


Figure 25: Behaviour of the dot product SVM with respect of the bias-variance decomposition of the error.

The increment of the values of C tends to flatten the U shape of the error curve: in particular for large C values bias becomes independent with respect to the degree (Figure 17). Moreover the C parameter plays also a regularization role (Figure 18)

Dot product SVM are characterized by opposite trends of bias and net-variance: bias decrements, while net-variance grows with respect to C; then, for higher values of C both stabilize. The combined effect of these symmetric curves produces a minimum of the error for low values of C, as the initial decrement of bias with C is larger than the initial increment of net-variance. Then the error slightly increases and stabilizes with C (Figure 19). The shape of the net-variance curve is determined mainly by the unbiased variance: it increases and then stabilizes with respect to C. On the other hand the biased variance curve is flat, remaining small for all values of C. A schematic picture of this behaviour is given in Figure 25.

7. Two Directions for Developing Ensembles of SVMs

In addition to providing insights into the behavior of SVMs, the analysis of the bias-variance decomposition of the error can identify the situations in which ensemble methods might improve SVM performance.

On several real-world problems, SVM ensembles are reported to give improvements over single SVMs (Kim et al., 2002; Valentini et al., 2003), but few works showed also negative experimental results about ensembles of SVMs (Buciu et al., 2001; Evgeniou et al., 2000). In particular Evgeniou et al. (2000) experimentally found that leave-one-out error bounds for kernel machines ensembles

are tighter that the equivalent ones for single machines, but they showed that with accurate parameters tuning single SVMs and ensembles of SVMs perform similarly.

In this section we propose to exploit bias-variance analysis in order to develop ensemble methods well tuned to the bias-variance characteristics of the base learners. In particular we present two possible ways of applying bias-variance analysis to develop SVM-based ensemble methods.

7.1 Bagged Ensemble of Selected Low-Bias SVMs

From a general standpoint, considering different kernels and different parameters of the kernel, we can observe that the minimum of the error, bias and net-variance (and in particular unbiased variance) do not match. For instance, considering RBF-SVM we see that we achieve the minimum of the error, bias and net-variance for different values of σ (see, for instance, Figure 5). Similar considerations can also be applied to polynomial and dot product SVM. Often, modifying parameters of the kernel, if we gain in bias we lose in variance and vice versa, even if this is not a rule.

Under the bootstrap assumption, bagging reduces only variance. From bias-variance decomposition we know that unbiased variance reduces the error, while biased variance increases the error. Hence bagging should be applied to low-biased classifiers, because the biased variance will be small.

Summarizing, we can schematically consider the following observations:

- We know that bagging lowers net-variance (in particular unbiased variance) but not bias (Breiman, 1996b).
- SVMs are strong, low-biased learners, but this property depends on the proper selection of the kernel and its parameters.
- If we can identify low-biased base learners with a relatively high unbiased variance, bagging can lower the error.
- Bias-variance analysis can identify SVMs with low bias.

Hence a basic high-level algorithm for a general *Bagged ensemble of selected low-bias SVMs* is the following:

1. Estimate bias-variance decomposition of the error for different SVM models
2. Select the SVM model with the lowest bias
3. Perform bagging using as base learner the SVM with the estimated lowest bias.

This approach combines the low bias properties of SVMs with the low unbiased variance properties of bagging and should produce ensembles with low overall error. We named this approach *Lobag*, that stands for *Lo*w bias *bag*ging. Using SVMs as base learners, depending on the type of kernel and parameters considered, and on the way the bias is estimated for the different SVM models, different algorithmic variants can be provided: For instance, depending on the type of kernel and parameters considered, different implementations can be given:

1. Selecting the RBF-SVM with the lowest bias with respect to the C and σ parameters.

2. Selecting the polynomial-SVM with the lowest bias with respect to the C and degree parameters.

3. Selecting the dot-prod-SVM with the lowest bias with respect to the C parameter.

4. Selecting the SVM with the lowest bias with respect to the kernel.

Another issue is how to implement the estimation of the bias-variance decomposition of the error for different SVM models. We could use cross-validation in conjunction with bootstrap replicates, or out-of-bag estimates (especially if we have small training sets), or hold-out techniques in conjunction with bootstrap replicates if we have sufficiently large training sets.

A first implementation of this approach, using an out-of-bag estimate of the bias-variance decomposition of the error, has been proposed, and quite encouraging results have been achieved (Valentini and Dietterich, 2003).

Another problem is the estimate of the noise in real data sets. A straightforward approach simply consists in disregarding it, but in this way we could overestimate the bias (see Section 5.4). Some heuristics are proposed in James (2003), but the problem remains substantially unresolved.

It is worth noting that this approach can be viewed as an alternative way for tuning SVM parameters, using an ensemble instead of a single SVM. From this standpoint recent works proposed to automatically choose multiple kernel parameters (Chapelle et al., 2002; Grandvalet and Canu, 2003), setting, for instance different σ values for each input dimension in Gaussian kernels, by applying a minimax procedure to iteratively maximize the margin of the SVM and to minimize an estimate of the generalization error over the set of kernel parameters (Chapelle et al., 2002). This promising approach could be in principle extended to minimize the bias, instead of the overall error. To this purpose we need to solve non trivial problems such as providing an upper bound for the bias and the variance, or at least an easy to compute their estimator having, if possible, an analytical expression. This approach could represent a new interesting research line that could improve the performances and/or reduce the computational burden of the Lobag method.

7.2 Heterogeneous Ensembles of SVM

The analysis of bias-variance decomposition of error in SVM shows that the minimum of the overall error, bias, net-variance, unbiased and biased variance occur often in different SVM models. These different behaviors of different SVM models could be in principle exploited to produce diversity in ensembles of SVMs. Although the diversity of base learner itself does not assure the error of the ensemble will be reduced (Kuncheva et al., 2001b), the combination of accuracy and diversity in most cases does (Dietterich, 2000a). As a consequence, we could select different SVM models as base learners by evaluating their accuracy and diversity through the bias-variance decomposition of the error.

Our results show that the "optimal region" (low average loss region) is quite large in RBF-SVMs (Figure 4). This means that C and σ do not need to be tuned extremely carefully. From this point of view, we can avoid time-consuming model selection by combining RBF-SVMs trained with different σ values all chosen from within the "optimal region." For instance, if we know that the error curve looks like the one depicted in Figure 23, we could try to fit a sigmoid-like curve using only few values to estimate where the stabilized region is located. Then we could train an heterogeneous ensemble of SVMs with different σ parameters (located in the low bias region) and average them according to their estimated accuracy.

A high-level algorithm for *Heterogeneous Ensembles of SVMs* could include the following steps:

1. Individuate the "optimal region" through bias-variance analysis of the error
2. Select the SVMs with parameters chosen from within the optimal region defined by bias-variance analysis.
3. Combine the selected SVMs by majority or weighted voting according to their estimated accuracy.

We could use different methods or heuristics to find the "optimal region" (see Section 5.1.3) and we have to define also the criterion used to select the SVM models inside the "optimal region" (for instance, improvement of the diversity). The combination could be performed using also other approaches, such as minimum, maximum, average and OWA aggregating operators (Kittler et al., 1998) or Behavior-Knowledge space method (Huang and C. Y., 1995), Fuzzy aggregation rules (Wang et al., 1998), Decision templates (Kuncheva et al., 2001a) or Meta-learning techniques (Prodromidis et al., 1999). Bagging and boosting (Freund and Schapire, 1996) methods can also be combined with this approach to further improve diversity and accuracy of the base learners.

7.3 Numerical Experiments with Low Bias Bagged SVMs

In order to show that these research directions could be fruitful to follow further, we performed numerical experiments on different data sets to test the Lobag ensemble method using SVMs as base learners. We compared the results with single SVMs and classical bagged SVM ensembles. We report here some preliminary results.More detailed results are reported in Valentini and Dietterich (2003).

We employed the 7 different two-class data sets described in Section 4.1, using small $\mathcal{D}$ training sets and large test $\mathcal{T}$ sets in order to obtain a reliable estimate of the generalization error: the number of examples for $\mathcal{D}$ was set to 100, while the size of $\mathcal{T}$ ranged from a few thousands for the "real" data sets to ten thousands for synthetic data sets. Then we applied the Lobag algorithm setting the number of samples bootstrapped from $\mathcal{D}$ to 100, and performing an out-of-bag estimate of the bias-variance decomposition of the error. The selected lobag, bagged and single SVMs were finally tested on the separated test set $\mathcal{T}$.

Table 7.3 shows the results of the experiments. We measured 20 outcomes for each method: 7 data sets, and 3 kernels (Gaussian, polynomial, and dot product) applied to each data set except P2 for which we did not apply the dot product kernel (because it was obviously inappropriate). For each pair of methods, we applied the McNemar test (Dietterich, 1998) to determine whether there was a significant difference in predictive accuracy on the test set.

On nearly all the data sets, both bagging and Lobag outperform the single SVMs independently of the kernel used. The null hypothesis that Lobag has the same error rate as a single SVM is rejected at or below the 0.1 significance level in 17 of the 20 cases, while the null hypothesis that bagging has the same error rate as a single SVM is rejected at or below the 0.1 level in 13 of the 20 cases. Most importantly, Lobag generally outperforms standard bagging. Lobag is statistically significantly better than bagging in 9 of the 20 cases, and significantly inferior only once.

These preliminary results show the feasibility of our approach, as shown also by similar experiments presented in Valentini and Dietterich (2003), but we need more experimental studies and

Kernel type	E_{lobag}	E_{bag}	E_{single}	Confidence level		
				L/B	**L/S**	**B/S**
	Data set *P2*					
Polyn.	0.1735	0.2008	0.2097	0.001	0.001	0.001
Gauss.	0.1375	0.1530	0.1703	0.001	0.001	0.001
	Data set *Waveform*					
Linear	0.0740	0.0726	0.0939	1	0.001	0.001
Polyn.	0.0693	0.0707	0.0724	1	0.1	0.1
Gauss.	0.0601	0.0652	0.0692	0.001	0.001	0.001
	Data set *Grey-Landsat*					
Linear	0.0540	0.0540	0.0650	1	0.001	0.001
Polyn.	0.0400	0.0440	0.0480	1	0.1	1
Gauss.	0.0435	0.0470	0.0475	0.1	0.1	1
	Data set *Letter-Two*					
Linear	0.0881	0.0929	0.1011	1	0.025	0.05
Polyn.	0.0701	0.0717	0.0831	1	0.05	0.1
Gauss.	0.0668	0.0717	0.0799	1	1	1
	Data set *Letter-Two with added noise*					
Linear	0.3535	0.3518	0.3747	1	1	0.1
Polyn.	0.3404	0.3715	0.3993	1	0.05	0.1
Gauss.	0.3338	0.3764	0.3829	0.05	0.025	1
	Data set *Spam*					
Linear	0.1408	0.1352	0.1760	0.05	0.001	0.001
Polyn.	0.0960	0.1034	0.1069	0.1	0.025	1
Gauss.	0.1130	0.1256	0.1282	0.005	0.001	1
	Data set *Musk*					
Linear	0.1291	0.1291	0.1458	1	0.001	0.001
Polyn.	0.1018	0.1157	0.1154	0.001	0.001	1
Gauss.	0.0985	0.1036	0.0936	0.05	1	0.05

Table 4: Results of the experiments using pairs of train $\mathcal{D}$ and test $\mathcal{T}$ sets. E_{lobag}, E_{bag} and E_{SVM} stand respectively for estimated error of lobag, bagged and single SVMs on the test set $\mathcal{T}$. The three last columns show the confidence level according to the Mc Nemar test. **L/B**, **L/S** and **B/S** stand respectively for the comparison Lobag/Bagging, Lobag/Single SVM and Bagging/Single SVM. If the confidence level is equal to 1, no significant difference is registered.

applications to real problems in order to better understand when and in which conditions this approach could be fruitful.

8. Conclusion and Future Works

We applied bias-variance decomposition of the error as a tool to gain insights into SVM learning algorithm. In particular we performed an analysis of bias and variance of SVMs, considering Gaussian, polynomial, and dot product kernels. The relationships between parameters of the kernel and

bias, net-variance, unbiased and biased variance have been studied through an extensive experimentation involving training, testing, and bias-variance analysis of more than half million of SVMs.

We discovered regular patterns in the behavior of the bias and variance, and we related those patterns to the parameters and kernel functions of the SVMs. The characterization of bias-variance decomposition of the error showed that in Gaussian kernels we can individuate at least three different regions with respect to the σ parameter, while in polynomial kernels the U shape of the error can be determined by the combined effects of bias and unbiased variance. The analysis revealed also that the expected trade-off between bias and variance holds systematically for dot product kernels, while other kernels showed more complex relationships.

The information supplied by bias-variance analysis suggests two promising approaches for designing ensembles of SVMs. One approach is to employ low-bias SVMs as base learners in a bagged ensemble. The other approach is to apply bias-variance analysis to construct a heterogeneous, diverse set of accurate and low-bias classifiers. We are designing and experimenting with both of these approaches.

An outgoing development of this work extends this analysis to bagged and boosted ensemble of SVMs, in order to achieve more insights about the behavior of SVM ensembles based on resampling methods.

In our experiments we did not explicitly consider the noise: analyzing the role of the noise in the decomposition of the error (Section 5.4) could help to develop ensemble methods specifically designed for noisy data.

Moreover in our experiments we did not explicitly consider the characteristics of the data. Nonetheless, such as we could expect and as our experiments suggested, different data characteristics influence bias-variance patterns in learning machines. To this purpose we plan to explicitly analyze the relationships between bias-variance decomposition of the error and data characteristics, using data complexity measures based on geometrical and topological characteristics of the data (Li and Vitanyi, 1993; Ho and Basu, 2002).

Acknowledgments

We thanks the anonymous reviewers for their comments and suggestions.

Appendix A.

In this appendix we discuss the notions of systematic and variance effect introduced by James (2003), showing that these quantities are reduced respectively to the bias and the net-variance when the $0/1$ loss is used and the noise is disregarded.

James (2003) provides definitions of bias and variance that are similar to those provided by Domingos (2000c). Indeed bias and variance definitions are based on quantities that he named the systematic part *sy* of the prediction *y* and the systematic part *st* of the target *t*. These correspond respectively to the Domingos main prediction (Equation2) and optimal prediction (Equation1). Moreover James distinguishes between bias and variance and *systematic* and *variance effects*. Bias and variance satisfy respectively the notion of the difference between the systematic parts of *y* and *t*, and the variability of the estimate *y*. Systematic effect *SE* represents the change in error of predicting *t* when using *sy* instead of *st* and the variance effect *VE* the change in prediction error when using *y*

instead of *sy* in order to predict t. Using Domingos' notation (y_m for *sy*, and y_* for *st*) the variance effect is

$$VE(y,t) = E_{y,t}[L(y,t)] - E_t[L(t,y_m)],$$

while the systematic effect corresponds to

$$SE(y,t) = E_t[L(t,y_m)] - E_t[L(t,y_*)].$$

In other words the systematic effect represents the change in prediction error caused by bias, while the variance effect the change in prediction error caused by variance.

While for the squared loss the two sets of bias-variance definitions match, for general loss functions the identity does not hold. In particular for the 0/1 loss James proposes the following definitions for noise, variance and bias with 0/1 loss:

$$\begin{aligned} N(\mathbf{x}) &= P(t \neq y_*), \\ V(\mathbf{x}) &= P(y \neq y_m), \\ B(\mathbf{x}) &= I(y_* \neq y_m), \end{aligned} \tag{10}$$

where $I(z)$ is 1 if z is true and 0 otherwise.

The variance effect for the 0/1 loss can be expressed as

$$\begin{aligned} VE(y,t) = E_{y,t}[L(y,t) - L(t,y_m)] &= P_{y,t}(y \neq t) - P_t(t \neq y_m) = \\ = 1 - P_{y,t}(y = t) - (1 - P_t(t = y_m)) &= P_t(t = y_m) - P_{y,t}(y = t), \end{aligned} \tag{11}$$

while the systematic effect is

$$\begin{aligned} SE(y,t) = E_t[L(t,y_m)] - E_t[L(t,y_*)] &= P_t(t \neq y_m) - P_t(t \neq y_*) = \\ = 1 - P_t(t = y_m) - (1 - P_t(t = y_*)) &= P_t(t = y_*) - P_t(t = y_m). \end{aligned} \tag{12}$$

If we let $N(\mathbf{x}) = 0$, considering Equation 7, 10 and Equation 11 the variance effect becomes

$$\begin{aligned} VE(y,t) = P_t(t = y_m) - P_{y,t}(y = t) &= P(y_* = y_m) - P_y(y = y_*) = \\ = 1 - P(y_* \neq y_m) - (1 - P_y(y \neq y_*)) &= 1 - B(\mathbf{x}) - (1 - EL(\mathcal{L}, \mathbf{x})) = \\ EL(\mathcal{L}, \mathbf{x}) - B(\mathbf{x}) = V_n(\mathbf{x}), & \end{aligned} \tag{13}$$

while from Equation 10 and Equation 12 the systematic effect becomes

$$\begin{aligned} SE(y,t) = P_t(t = y_*) - P_t(t = y_m) &= 1 - P_t(t \neq y_*) - (1 - P_t(t \neq y_m)) = \\ P(y* \neq y_m) = I(y* \neq y_m) = B(\mathbf{x}). & \end{aligned} \tag{14}$$

Hence if $N(\mathbf{x}) = 0$, it follows that the variance effect is equal to the net-variance (Equation 13), and the systematic effect is equal to the bias (Equation 14).

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting and variants. *Machine Learning*, 36(1/2):525–536, 1999.

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

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996a.

L. Breiman. Bias, variance and arcing classifiers. Technical Report TR 460, Statistics Department, University of California, Berkeley, CA, 1996b.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

I. Buciu, C. Kotropoulos, and I. Pitas. Combining Support Vector Machines for Accurate Face Detection. In *Proc. of ICIP'01*, volume 1, pages 1054–1057, 2001.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1):131–159, 2002.

S. Cohen and N. Intrator. Automatic Model Selection in a Hybrid Perceptron/Radial Network. In *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 349–358. Springer-Verlag, 2001.

T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, (7):1895–1924, 1998.

T. G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2000a.

T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2):139–158, 2000b.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, (2):263–286, 1995.

P. Domingos. A unified bias-variance decomposition. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2000a.

P. Domingos. A Unified Bias-Variance Decomposition and its Applications. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 231–238, Stanford, CA, 2000b. Morgan Kaufmann.

P. Domingos. A Unified Bias-Variance Decomposition for Zero-One and Squared Loss. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 564–569, Austin, TX, 2000c. AAAI Press.

T. Evgeniou, L. Perez-Breva, M. Pontil, and T. Poggio. Bounds on the Generalization Performance of Kernel Machine Ensembles. In P. Langley, editor, *Proc. of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 271–278. Morgan Kaufmann, 2000.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kauffman, 1996.

J. H. Friedman. On bias, variance, 0/1 loss and the curse of dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias-variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

Y. Grandvalet and S. Canu. Adaptive Scaling for Feature Selection in SVMs. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS 2002 Conference Proceedings, Advances in Neural Information Processing Systems*, volume 15, Cambridge, MA, 2003. MIT Press.

T. Heskes. Bias/Variance Decompostion for Likelihood-Based Estimators. *Neural Computation*, 10:1425–1433, 1998.

T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

Y. S. Huang and Suen. C. Y. Combination of multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17: 90–94, 1995.

G. James. Variance and bias for general loss function. *Machine Learning*, (2):115–135, 2003.

T. Joachims. Making large scale SVM learning practical. In Smola A. Scholkopf B., Burges C., editor, *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, Cambridge, MA, 1999.

H. C. Kim, S. Pang, H. M. Je, D. Kim, and S. Y. Bang. Pattern Classification Using Support Vector Machine Ensemble. In *Proceedings of the International Conference on Pattern Recognition, 2002*, volume 2, pages 20160–20163. IEEE, 2002.

J. Kittler, M. Hatef, R. P. W. Duin, and Matas J. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

E. M. Kleinberg. A Mathematically Rigorous Foundation for Supervised Learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. First International Workshop, MCS 2000, Cagliari, Italy*, volume 1857 of *Lecture Notes in Computer Science*, pages 67–76. Springer-Verlag, 2000.

R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proc. of the Thirteenth International Conference on Machine Learning, The Seventeenth International Conference on Machine Learning*, pages 275–283, Bari, Italy, 1996. Morgan Kaufmann.

E. Kong and T. G. Dietterich. Error-correcting output coding correct bias and variance. In *The XII International Conference on Machine Learning*, pages 313–321, San Francisco, CA, 1995. Morgan Kauffman.

L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001a.

L. I. Kuncheva, F. Roli, G. L. Marcialis, and C. A. Shipp. Complexity of Data Subsets Generated by the Random Subspace Method: An Experimental Investigation. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems. Second International Workshop, MCS 2001, Cambridge, UK*, volume 2096 of *Lecture Notes in Computer Science*, pages 349–358. Springer-Verlag, 2001b.

L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles. *Machine Learning*, 51:181–207, 2003.

M. Li and P Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, 1993.

L. Mason, P. Bartlett, and J. Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 2000.

C. J. Merz and P. M. Murphy. UCI repository of machine learning databases, 1998. www.ics.uci.edu/mlearn/MLRepository.html.

A. Prodromidis, P. Chan, and S. Stolfo. Meta-Learning in Distributed Data Mining Systems: Issues and Approaches. In H. Kargupta and P. Chan, editors, *Advances in Distributed Data Mining*, pages 81–113. AAAI Press, 1999.

R. E. Schapire. A brief introduction to boosting. In Thomas Dean, editor, *16th International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kauffman, 1999.

R. E. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

R. Tibshirani. Bias, variance and prediction error for classification rules. Technical report, Department of Preventive Medicine and Biostatistics and Department od Statistics, University of Toronto, Toronto, Canada, 1996.

G. Valentini and T. G. Dietterich. Low Bias Bagged Support Vector Machines. In T. Fawcett and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003)*, pages 752–759, Washington D. C., USA, 2003. AAAI Press.

G. Valentini and F. Masulli. NEURObjects: an object-oriented library for neural network development. *Neurocomputing*, 48(1–4):623–646, 2002.

G. Valentini, M. Muselli, and F. Ruffino. Bagged Ensembles of SVMs for Gene Expression Data Analysis. In *IJCNN2003, The IEEE-INNS-ENNS International Joint Conference on Neural Networks*, pages 1844–49, Portland, USA, 2003. IEEE.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

D. Wang, J. M. Keller, C. A. Carson, K. K. McAdoo-Edwards, and C. W. Bailey. Use of fuzzy logic inspired features to improve bacterial recognition through classifier fusion. *IEEE Transactions on Systems, Man and Cybernetics*, 28B(4):583–591, 1998.

Journal of Machine Learning Research 5 (2004) 777–800 Submitted 12/03; Revised 5/04; Published 7/04

A Fast Algorithm for Joint Diagonalization with Non-orthogonal Transformations and its Application to Blind Source Separation

Andreas Ziehe ZIEHE@FIRST.FHG.DE
Pavel Laskov LASKOV@FIRST.FHG.DE
Fraunhofer FIRST.IDA
Kekuléstrasse 7
12489 Berlin, Germany

Guido Nolte NOLTEG@NINDS.NIH.GOV
National Institutes of Health
10 Center Drive MSC 1428
Bethesda, MD 20892, USA

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

Editor: Michael Jordan

Abstract

A new efficient algorithm is presented for joint diagonalization of several matrices. The algorithm is based on the Frobenius-norm formulation of the joint diagonalization problem, and addresses diagonalization with a general, non-orthogonal transformation. The iterative scheme of the algorithm is based on a multiplicative update which ensures the invertibility of the diagonalizer. The algorithm's efficiency stems from the special approximation of the cost function resulting in a sparse, block-diagonal Hessian to be used in the computation of the quasi-Newton update step. Extensive numerical simulations illustrate the performance of the algorithm and provide a comparison to other leading diagonalization methods. The results of such comparison demonstrate that the proposed algorithm is a viable alternative to existing state-of-the-art joint diagonalization algorithms. The practical use of our algorithm is shown for blind source separation problems.

Keywords: joint diagonalization, common principle component analysis, independent component analysis, blind source separation, nonlinear least squares, Newton method, Levenberg-Marquardt algorithm

1. Introduction

Joint diagonalization of square matrices is an important problem of numeric computation. Many applications make use of joint diagonalization techniques as their main algorithmic tool, for example

independent component analysis (ICA) and blind source separation (BSS) (Comon, 1994; Molgedey and Schuster, 1994; Belouchrani et al., 1997; Wu and Principe, 1999; Cardoso, 1999; Ziehe and Müller, 1998; Ziehe et al., 2003; Pham and Cardoso, 2000; Ziehe et al., 2000a; Yeredor, 2002; Haykin, 2000; Hyvärinen et al., 2001), common principal component analysis (CPC) (Flury, 1988; Airoldi and Flury, 1988; Fengler et al., 2001), various signal processing applications (van der Veen et al., 1992, 1998) and, more recently, kernel-based nonlinear BSS (Harmeling et al., 2003).

This paper pursues two goals. First, we propose a new efficient algorithm for joint approximate matrix diagonalization. Our algorithm is based on the second-order approximation of a cost function for the simultaneous diagonalization problem. Second, we demonstrate an application of our algorithm to BSS, which allows to perform BSS without pre-whitening the data.

Let us begin by defining the notion of joint diagonalization. It is well known that exact joint diagonalization is in general only possible for two matrices and amounts to the generalized eigenvalue problem. Extensive literature exists on this topic (e.g. Noble and Daniel, 1977; Golub and van Loan, 1989; Bunse-Gerstner et al., 1993; Van der Vorst and Golub, 1997, and references therein). When more than two matrices are to be diagonalized, exact diagonalization may also be possible if the matrices possess a certain common structure. Otherwise one can only speak of approximate joint diagonalization. Our paper focuses on the investigation of algorithms for exact—whenever this is possible—or otherwise approximate diagonalization of more than two matrices. In the remainder of the paper we will refer to such problems as "joint diagonalization" problems.

A number of algorithms for joint diagonalization have been previously proposed in the literature (Flury and Gautschi, 1986; Cardoso and Souloumiac, 1993, 1996; Hori, 1999; Pham, 2001; van der Veen, 2001; Yeredor, 2002; Joho and Rahbar, 2002). To understand the challenges of the joint diagonalization problem, as well as the need for further improvement of currently known algorithms and possible directions of such improvement some insight into the main issues of joint diagonalization is now provided.

Let us consider a set $\{C^1,\ldots,C^K\}$ of real-valued symmetric matrices of size $N \times N$.[1] The goal of a joint diagonalization algorithm is to find a transformation V that in some sense "diagonalizes" all the given matrices. The notion of diagonality and the corresponding formal statement of the joint diagonalization problem can be defined in at least three different ways:

1. *Frobenius norm formulation.* This formulation is used in Cardoso and Souloumiac (1993, 1996); Joho and Rahbar (2002) and, in a generalized form, in Hori (1999). Let

$$F^k = VC^kV^T \tag{1}$$

denote the result of applying transformation V to matrix C^k. Joint diagonalization is defined as the following optimization problem:

$$\min_{V\in\mathbb{R}^{N\times N}} \sum_{k=1}^{K} \mathfrak{M}_D(F^k), \tag{2}$$

where the diagonality measure $\mathfrak{M}_D$ is the Frobenius norm of the off-diagonal elements in F^k:

$$\mathfrak{M}_D(F^k) = \mathrm{off}(F^k) = \sum_{i\neq j}(F^k_{ij})^2. \tag{3}$$

1. The formulations and the proposed algorithm will be presented for symmetric matrices. Extensions to the unsymmetric or complex-valued case can be obtained in a similar manner.

A more careful look at the cost function in Equation (2) reveals a serious problem with the Frobenius norm formulation: this cost function is obviously minimized by the trivial solution $V = 0$. The problem can be avoided by additionally requiring orthogonality of V. In fact, this assumption is very natural if the joint diagonalization problem is seen as an extension of the eigenvalue problem to several matrices. However, restricting V to the group of orthogonal matrices may limit the applicability and unduly degrade the performance of the method.[2]

2. *Positive definite formulation.* Another reasonable assumption on the initial problem is the positive-definiteness of the matrices C^k. This assumption is motivated by the fact that in many applications matrices C^k are covariance matrices of some random variables. In this case, as proposed in Matsuoka et al. (1995); Pham (2001) the criterion[3]

$$\mathfrak{M}_D(F^k) = \log\det(\mathrm{ddiag}(F^k)) - \log\det(F^k) \tag{4}$$

can be used in the cost function (2) instead of the criterion (3). The additional advantage of this criterion is that it allows for super-efficient estimation (Pham and Cardoso, 2001). However in certain applications, such as blind source separation based on time-delayed decorrelation (Belouchrani et al., 1997; Ziehe and Müller, 1998), correlation matrices are no longer guaranteed to be positive-definite, and diagonalization based on this criterion may fail.

3. *Subspace fitting formulation*. The fact that exact joint diagonalization may not be possible can be explicitly accounted for in the problem formulation. This is to say that, instead of applying the transformation directly to the matrices C^k, another set of diagonal matrices Λ^k is sought for, along with the transformation so as to best approximate the target matrices. The optimization problem resulting from this approach

$$\min_{A,\Lambda^1,\ldots,\Lambda^K} \sum_{k=1}^{K} ||C^k - A\Lambda^k A^T||_F^2 \tag{5}$$

constitutes an instance of a subspace fitting problem (van der Veen, 2001; Yeredor, 2002).

Compared to the previous approaches, the algorithms based on subspace fitting have two advantages: they do not require orthogonality, positive-definiteness or any other normalizing assumptions on the matrices A and C^k, and they are able to handle non-square mixture matrices. These advantages, however, come at a high computational cost: the algorithm of van der Veen (2001) has quadratic convergence in the vicinity of the minimum but its running time per iteration is $O(KN^6)$, whereas the AC-DC algorithm of Yeredor (2002) converges linearly with a running time per iteration of order $O(KN^3)$.

As a short resume of the above mentioned approaches we notice the following. The algorithms using the Frobenius norm formulation are efficient but rely on the orthogonality assumption to prevent convergence to the trivial solution. The algorithms using the positive-definiteness assumption are also quite efficient but they may fail if this assumption is not satisfied. Subspace fitting algorithms, which do not require such strong prior assumptions, are computationally much more demanding. A

2. In principle, a pre-sphering step could be applied to alleviate this problem, nevertheless a performance degradation is to be expected in this case, especially in the context of blind source separation (Cardoso, 1994a; Yeredor, 2002).
3. Here the operator $\mathrm{ddiag}(F)$ returns a diagonal matrix containing only the diagonal entries of F.

natural question arises: could a single algorithm combine the positive and avoid the negative features of the previous joint diagonalization algorithms? In this contribution we present an algorithm using the Frobenius norm formulation that strives towards this goal. In particular, the algorithm, to be called FFDIAG (Fast Frobenius Diagonalization), possesses the following features:

- computational efficiency: quadratic convergence (in the neighborhood of the solution) and $O(KN^2)$ running time per iteration,
- guaranteed avoidance of the trivial solution,
- no orthogonality and no positive-definiteness assumptions; nonetheless, orthogonality can be used to constrain the solution, which further reduces the computational complexity by a factor of two.

On top of that, the algorithm is simple and easy to implement.

The remainder of the paper is organized as follows. In Section 2 the main idea of the FFDIAG algorithm is proposed. The computational details regarding the algorithm's update rule are derived in Section 3. Section 4, in a slight digression from the main topic of the article, presents a connection of our algorithm to the classical Levenberg-Marquardt algorithm, and points out the main differences between the two. The application of the FFDIAG algorithm to blind source separation is developed in Section 5. Extensive numerical simulations are presented in Section 6. Finally, Section 7 is devoted to the discussion and conclusions.

2. General Structure of the Algorithm

The FFDIAG algorithm is an iterative scheme to approximate the solution of the following optimization problem:

$$\min_{V \in \mathbb{R}^{N \times N}} \sum_{k=1}^{K} \sum_{i \neq j} ((VC^k V^T)_{ij})^2. \tag{6}$$

The basic idea is to use the invertibility of the matrix V as a constraint preventing convergence of the minimizer of the cost function in Equation (6) to the zero solution. Invertibility is tacitly assumed in many applications of diagonalization algorithms, e.g. in blind source separation, therefore making use of such constraint is very natural and does not limit the generality from the practical point of view.

Invertibility can be enforced by carrying out the update of V in multiplicative form as

$$V_{(n+1)} \leftarrow (I + W_{(n)}) V_{(n)}, \tag{7}$$

where I denotes the identity matrix, the update matrix $W_{(n)}$ is constrained to have zeros on the main diagonal, and n is the iteration number. Such update is rarely used in classical unconstrained optimization algorithms; however, it is common for many successful BSS algorithms, such as relative-gradient (Laheld and Cardoso, 1996; Amari et al., 2000), relative Newton (Akuzawa and Murata, 2001; Zibulevsky, 2003), as well as for joint diagonalization (Pham, 2001). The off-diagonal component $W_{(n)}$ of the update multiplier is to be determined so as to minimize the cost function (6). In order to maintain invertibility of V it clearly suffices to enforce invertibility of $I + W_{(n)}$. The latter can be carried out using the following well-known results of matrix analysis (Horn and Johnson, 1985).

Definition 1 *An $n \times n$ matrix A is said to be strictly diagonally dominant if*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|, \quad \text{for all } i = 1, \ldots, n.$$

Theorem 2 (Levi-Desplanques) *If an $n \times n$ matrix A is strictly diagonally-dominant, then it is invertible.*

The Levi-Desplanques theorem can be used to control invertibility of $I + W_{(n)}$ in a straightforward way. Observe that the diagonal entries in $I + W_{(n)}$ are all equal to 1; therefore, it suffices to ensure that

$$\max_i \sum_{j \neq i} |W_{ij}| = ||W_{(n)}||_\infty < 1.$$

This can be done by dividing $W_{(n)}$ by its infinity norm whenever the latter exceeds some fixed $\theta < 1$. An even stricter condition can be imposed by using a Frobenius norm in the same way:

$$W_{(n)} \leftarrow \frac{\theta}{||W_{(n)}||_F} W_{(n)}. \tag{8}$$

To determine the optimal updates $W_{(n)}$ at each iteration, first-order optimality constraints for the objective (6) are used. A special approximation of the objective function will enable us to efficiently compute $W_{(n)}$. For this reason, we consider the expression for updating the matrices to be diagonalized

$$C^k_{(n+1)} \leftarrow F^k = (I + W_{(n)})\, C^k_{(n)}\, (I + W_{(n)})^T. \tag{9}$$

Let $D^k_{(n)}$ and $E^k_{(n)}$ denote the diagonal and off-diagonal parts of $C^k_{(n)}$, respectively. In order to simplify the optimization problem we assume that the norms of $W_{(n)}$ and $E^k_{(n)}$ are small, i.e. quadratic terms in the expression for the new set of matrices can be ignored

$$\begin{aligned} C^k_{(n+1)} &= (I + W_{(n)})(D^k_{(n)} + E^k_{(n)})(I + W_{(n)})^T \\ &\approx D^k_{(n)} + W_{(n)} D^k_{(n)} + D^k_{(n)} W^T_{(n)} + E^k_{(n)}. \end{aligned} \tag{10}$$

With these simplifications, and ignoring already diagonal terms D^k, the diagonality measure (3) can be computed using expressions linear in W.[4]

$$F^k \approx \tilde{F}^k = W D^k + D^k W^T + E^k. \tag{11}$$

The linearity of terms (11) allows us to explicitly compute the optimal update matrix $W_{(n)}$ minimizing the approximated diagonality criterion

$$\min_W \sum_{k=1}^{K} \sum_{i \neq j} ((W D^k + D^k W^T + E^k)_{ij})^2. \tag{12}$$

Details of the efficient solution of problem (12) are presented in Section 3.

The simplifying assumptions used in (10) require some further discussion. The motivation behind them is that in the neighborhood of the optimal solution, the optimal steps W that take us to

4. The iteration indices will be dropped in the following if all quantities refer to the same iteration.

the optimum are small and the matrices C^k are almost diagonal. Hence, in the neighborhood of the optimal solution the algorithm is expected to behave similarly to Newton's algorithm and converge quadratically. The assumption of small E^k is potentially problematic, especially in the case where exact diagonalization is impossible. A similar derivation can be carried out with E^k fully accounted for, which leads to additional WE^k and E^kW^T terms in the expression (12). However, the resulting algorithm, will not give rise to a computationally efficient solution of problem (12). As for the assumption of small W, it is crucial for the convergence of the algorithm and needs to be carefully controlled. The latter is done by the normalization (8).

Pseudo-code describing the FFDIAG method is outlined in Algorithm 1.[5]

Algorithm 1 FFDIAG

INPUT: C^k { Matrices to be diagonalized}
$W_{(1)} \leftarrow 0$, $V_{(1)} \leftarrow I$, $n \leftarrow 1$ { $V_{(1)}$ could also be initialized by a more clever guess.}
$C^k_{(1)} \leftarrow V_{(1)}\, C^k V^T_{(1)}$
repeat
  compute $W_{(n)}$ from $C^k_{(n)}$ according to Equation (17) or (18)
  if $||W_{(n)}||_F > \theta$ **then**
    $W_{(n)} \leftarrow \frac{\theta}{||W_{(n)}||_F} W_{(n)}$
  end if
  $V_{(n+1)} \leftarrow (I + W_{(n)})\, V_{(n)}$
  $C^k_{(n+1)} \leftarrow (I + W_{(n)})\, C^k_{(n)}\, (I + W_{(n)})^T$
  $n \leftarrow n + 1$
until convergence
OUTPUT: V, C^k

Some remarks on convergence properties of the proposed algorithmic scheme are due at this point. In general, Newton-like algorithms are known to converge only in the neighborhood of the optimal solution; however, when they converge, the rate of convergence is quadratic (e.g. Kantorovich, 1949). Since the essential components of our algorithm—the second-order approximation of the objective function and the computation of optimal steps by solving the linear system arising from first-order optimality conditions—are inherited from Newton's method, the same convergence behavior can be expected. In practice, however, the known theoretical estimates of convergence regions of Newton's method, such as the ones provided, e.g., in Theorems 1 and 2 in Kantorovich (1949), are of little utility since they provide no guidance how to reach the convergence region from an arbitrary starting point.

3. Computation of the Update Matrix

The key to computational efficiency of the FFDIAG algorithm lies in exploiting the sparseness introduced by the approximation (11). The special structure of the problem can be best seen in the matrix-vector notation presented next.

5. MATLAB code for FFDIAG can be obtained at `http://www.first.fhg.de/~ziehe/research/FFDiag`.

The $N(N-1)$ off-diagonal entries of the update matrix W are arranged as

$$w = (W_{12}, W_{21}, \ldots, W_{ij}, W_{ji}, \ldots)^T. \tag{13}$$

Notice that this is *not* the usual vectorization operation vec$W$, as the order of elements in w reflects the pairwise relationship of the elements in W. In a similar way the $KN(N-1)$ off-diagonal entries of the matrices E^k are arranged as

$$e = (E^1_{12}, E^1_{21}, \ldots, E^1_{ij}, E^1_{ji}, \ldots, E^k_{ij}, E^k_{ji}, \ldots)^T. \tag{14}$$

Finally, a large but very sparse, $KN(N-1) \times N(N-1)$ matrix J is built, in the form:

$$J = \begin{pmatrix} J_1 \\ \vdots \\ J_K \end{pmatrix} \text{ with } J_k = \begin{pmatrix} \mathcal{D}^k_{12} & & & \\ & \ddots & & \\ & & \mathcal{D}^k_{ij} & \\ & & & \ddots \end{pmatrix},$$

where each J_k is block-diagonal, containing $N(N-1)/2$ matrices of dimension 2×2

$$\mathcal{D}^k_{ij} = \begin{pmatrix} D^k_j & D^k_i \\ D^k_j & D^k_i \end{pmatrix}, \quad i, j = 1, \ldots, N,\ i \neq j,$$

where D^k_i is a short-hand notation for the ii-th entry of a diagonal matrix D^k. Now the approximate cost function can be re-written as the linear least-squares problem

$$\mathcal{L}(w) = \sum_k \sum_{i \neq j} (\tilde{F}^k_{ij})^2 = (Jw + e)^T (Jw + e).$$

The well-known solution of this problem (Press et al., 1992) reads

$$w = -(J^T J)^{-1} J^T e. \tag{16}$$

We can now make use of the sparseness of J and e to enable the direct computation of the elements of w in (16). Writing out the matrix product $J^T J$ yields a block-diagonal matrix

$$J^T J = \begin{pmatrix} \sum_k (\mathcal{D}^k_{12})^T \mathcal{D}^k_{12} & & & \\ & \ddots & & \\ & & \sum_k (\mathcal{D}^k_{ij})^T \mathcal{D}^k_{ij} & \\ & & & \ddots \end{pmatrix}$$

whose blocks are 2×2 matrices. Thus the system (16) actually consists of decoupled equations

$$\begin{pmatrix} W_{ij} \\ W_{ji} \end{pmatrix} = -\begin{pmatrix} z_{jj} & z_{ij} \\ z_{ij} & z_{ii} \end{pmatrix}^{-1} \begin{pmatrix} y_{ij} \\ y_{ji} \end{pmatrix}, \quad i, j = 1, \ldots, N,\ i \neq j,$$

where

$$z_{ij} = \sum_k D^k_i D^k_j$$

$$y_{ij} = \sum_k D^k_j \frac{E^k_{ij} + E^k_{ji}}{2} = \sum_k D^k_j E^k_{ij}.$$

The matrix inverse can be computed in closed form, leading to the following expressions for the update of the entries of W:

$$\begin{aligned} W_{ij} &= \frac{z_{ij}y_{ji} - z_{ii}y_{ij}}{z_{jj}z_{ii} - z_{ij}^2} \\ W_{ji} &= \frac{z_{ij}y_{ij} - z_{jj}y_{ji}}{z_{jj}z_{ii} - z_{ij}^2}. \end{aligned} \tag{17}$$

(Here, only the off-diagonal elements ($i \neq j$) need to be computed and the diagonal terms of W are set to zero.) Thus, instead of performing inversion and multiplication of large matrices, which would have brought us to the same $O(KN^6)$ complexity as in van der Veen (2001), computation of the optimal $W_{(n)}$ leads to a simple formula (17) which has to be evaluated for each of $N(N-1)$ components of $W_{(n)}$. Since the computation of z_{ij} and y_{ij} also involves a loop over K, the overall complexity of the update step is $O(KN^2)$.

An even simpler solution can be obtained if the diagonalization matrix V is assumed to be orthogonal from the very beginning. Orthogonality of V can be preserved to the first order by requiring W to be skew-symmetric, i.e., $W = -W^T$. Hence only one of each pair of its entries needs to be computed. In this case the structure of the problem is already apparent in the scalar notation, and one can easily obtain the partial derivatives of the cost function. Equating the latter to zero yields the following expression for the update of W:

$$W_{ij} = \frac{\sum_k E_{ij}^k (D_i^k - D_j^k)}{\sum_k (D_i^k - D_j^k)^2}, \quad i,j = 1,\ldots,N,\ i \neq j, \tag{18}$$

which agrees with the result of Cardoso (1994b). To ensure orthogonality of V beyond the first order the update (7) should be replaced by the matrix exponential update

$$V_{(n+1)} \leftarrow \exp(W_{(n)})V_{(n)},$$

where $W_{(n)}$ is skew-symmetric (cf. Akuzawa and Murata, 2001).

4. Comparison with the Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm (Levenberg, 1944; Marquardt, 1963) is one of the powerful and popular algorithms for solving nonlinear least-squares problems. Interestingly, the motivation in the original article of Marquardt (1963) was somewhat similar to ours: he knew that quadratic convergence of Newton's method was attainable only in the neighborhood of the solution, and therefore he looked for an efficient means of steering the algorithm to the area of quadratic convergence. The particular mechanism used in the LM algorithm consists of a controllable trade-off between Newton and gradient steps.

Although the problem of simultaneous diagonalization is essentially a nonlinear (quadratic) least-squares problem, the LM algorithm cannot be directly applied to it. An implicit assumption of the simultaneous diagonalization problem is the invertibility of the diagonalizing matrix, and the classical LM algorithm does not provide for incorporation of additional constraints. In what follows we present a modification which allows one to incorporate the additional structure of our problem into the LM algorithm.

A general problem of the nonlinear regression to be solved by the LM algorithm is usually formulated as

$$\min_p \sum_k ||y_k - f_p(x_k)||^2.$$

The goal of the optimization is to find the parameters p of the regression function f so as to minimize the squared deviations between $f_p(x_k)$ and y_k for all data points $1,\ldots,k$. The signature of the function f can be arbitrary, with an appropriate norm to be chosen. The cost function (6) can be seen as a nonlinear regression problem over the vector-valued function $f_V(C)$, parameterized by V, of matrix argument C, with zero target values:

$$\min_{V\in\mathbb{R}^{n\times n}} \sum_k ||0 - f_V(C^k)||^2.$$

The construction and dimensionality of the function f is explained below, and the zero vector is set to have the appropriate dimension.

To enforce invertibility, the diagonal entries of V are set to 1 and only off-diagonal entries are considered as free parameters. As in Section 3 such a representation of V is constructed by means of symmetric vectorization $\operatorname{vecs} V \stackrel{\text{def}}{=} [V_{21}, V_{12}, V_{31}, V_{13}, \ldots]^T$.[6]

The same vectorization is applied to construct the regression function

$$f_V(C) : \mathbb{R}^{N\times N} \to \mathbb{R}^{N(N-1)\times 1} \stackrel{\text{def}}{=} \operatorname{vecs} VCV^T.$$

As a result of such vectorization, the diagonal entries of VCV^T are discarded, and the Euclidean norm of this vector is equivalent to the "off" function.

The LM algorithm requires computation of the Jacobian matrix of the regression function (w.r.t. parameters $\operatorname{vecs} V$) at all data points:

$$J_{LM} \stackrel{\text{def}}{=} [\mathsf{D}_{\operatorname{vecs} V} f_V(C^1), \ldots, \mathsf{D}_{\operatorname{vecs} V} f_V(C^K)]^T.$$

The Jacobian matrices (of dimension $N(N-1)\times N(N-1)$) at individual data points can be computed as:

$$\mathsf{D}_{\operatorname{vecs} V} f_V(C^k) = S_{NN}(I_{N^2} + K_{NN})(VC^k \otimes I_N)S_{NN}^T,$$

where K_{NN} is the commutation matrix (Lütkepohl, 1996), I is the identity matrix of the appropriate size, and $\otimes$ is the Kronecker matrix product.

Denoting $f = [f^T(C^1), \ldots, f^T(C^K)]^T$, the main step of the LM algorithm consists of solving the following linear system:

$$((J_{LM})^T J_{LM} + \lambda I) \operatorname{vecs} V = -(J_{LM})^T f. \tag{20}$$

The parameter λ controls the trade-off between Newton-like and gradient-based strategies: $\lambda = 0$ results in the pure Newton-direction, whereas with large λ, the steps approach the gradient direction. We use the original heuristic of Marquardt to choose λ: if the value of the cost function provided by the current step $\operatorname{vecs} V$ decreases, λ can be decreased by a factor of 10 while descent is maintained; if the value of the cost function increases, increase λ by a factor of 10 until descent is achieved. This heuristic is very intuitive and easy to implement; however, since it doesn't involve any line search,

6. The symmetric vectorization vecs is related to column vectorization vec by the special permutation matrix S_{NN} such that $\operatorname{vecs} X = S_{NN} \operatorname{vec} X$.

the algorithm may fail to converge. More sophisticated strategies with convergence guarantees of Osborne (1976) and Moré (1978) can also be deployed.

From the theoretical point of view, one can draw the following parallels between the FFDIAG and LM algorithms:

- Both algorithms pursue a Newton direction (cf. equations (16) and (20)) to compute the update matrix V. Whereas the LM algorithms computes the update step directly on V, the update of the FFDIAG is performed in a multiplicative way by computing W to be used in the update rule (7).

- Unlike the LM algorithm using the Hessian of the original cost function, the Newton direction in FFDIAG is computed based on the Hessian of the second-order approximation of the cost function.[7] Taking advantage of the resulting special structure, this computation can be carried out very efficiently in FFDIAG.

- Regularization in the LM algorithm results in a gradual shift from the gradient to the Newton directions (and back when necessary). Regularization in the FFDIAG algorithm is of quite different flavor. Since the computed direction is only approximately Newton, one cannot fully trust it, and therefore the update heuristic (8) limits the impact of inaccurate computation of W. On the other hand, when FFDIAG converges to a close neighborhood of the optimal solution, the heuristic is turned off, and Newton-like convergence is no longer impeded.

It is interesting to compare performance of the LM and FFDIAG algorithms experimentally. We use two criteria: the cost function and the convergence ratio

$$\text{convergence ratio} = \frac{||f_{(n+1)} - f^*||}{||f_{(n)} - f^*||}.$$

The zero value of the convergence ratio indicates super-linear convergence. The evolution of our criteria in two runs of the algorithms are shown in Figure 1. In the cost function plot one can see that convergence of both algorithms is linear for the most part of their operation, with a gradual shift to quadratic convergence as the optimal solution is approached. The same conclusion can be drawn from the convergence ratio plot, in which one can see that this criterion approaches zero in the neighborhood of the optimal solution. Thus one can conclude that, similarly to the LM algorithm, the heuristic (8) steers the algorithm to the area where Newton-like convergence is achieved. Furthermore, we note that due to the use of the special structure, the per-iteration complexity of the FFDIAG algorithm is significantly lower than that of the LM algorithm.

5. Application to Blind Source Separation

First, we recall the definition of the blind source separation (BSS) problem (Jutten and Herault, 1991). We are given the instantaneous linear mixtures $x_i(t)$ of a number of source signals $s_j(t)$, obeying the model

$$x_i(t) = \sum_{j=1}^{m} a_{ij} s_j(t), \ (i = 1,\ldots,n,\ j = 1,\ldots,m), \tag{21}$$

7. In fact, in both algorithms, the Hessians are approximated by the product of the Jacobian matrices.

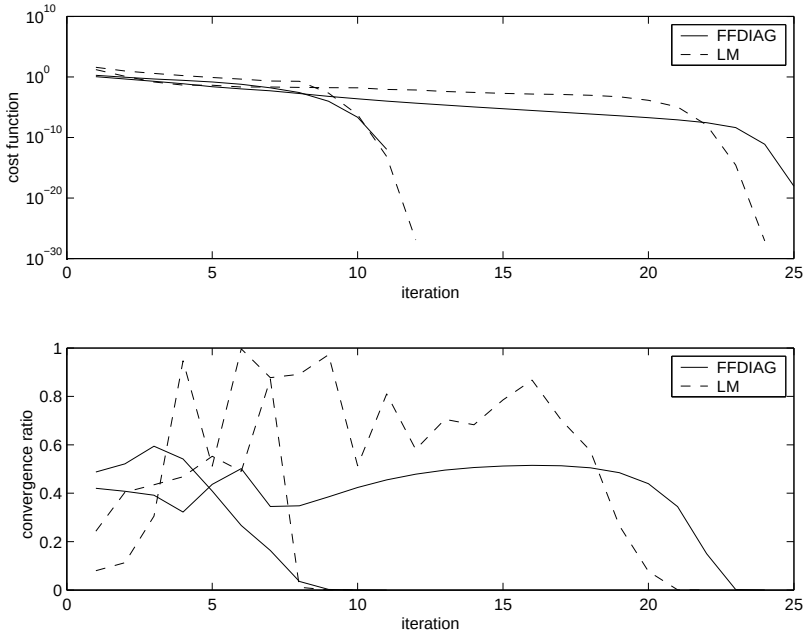


Figure 1: Comparison of the LM and the FFDIAG algorithms. The data matrices are generated as described in Section 6.1 with $K = 10$, $N = 10$. Two illustrative runs are shown.

with A being non-singular and $s_j(t)$ statistically independent. The goal is to estimate both A and $s(t)$ from $x(t)$.

Linear BSS methods have been successfully applied to a variety of problems. An example of such an application is the reduction of artifacts in electroencephalographic (EEG) and magnetoencephalographic (MEG) measurements. Due to the fact that the electromagnetic waves superimpose linearly and virtually instantaneously (because of the relatively small distance from sources to sensors) the model (21) is valid (Makeig et al., 1996; Vigário et al., 1998; Wübbeler et al., 2000; Ziehe et al., 2000a). Note, however, that in other applications, such as the so called cocktail-party problem in auditory perception (von der Malsburg and Schneider, 1986), the model from Equation (21) may be too simplistic, since time-delays in the signal propagation are no longer negligible. Extended models to deal with such convolutive mixtures have been considered (e.g. Parra and Spence, 1998; Lee et al., 1998; Murata et al., 2001). We will in the following only discuss how to solve the linear, instantaneous BSS problem stated in Equation (21). The usual approach is to define an appropriate cost function that can subsequently be optimized. Here our goal is to use the general joint diagonalization cost function (6) and to construct certain matrices in such a way that their approximate joint diagonalizer is an estimate for the demixing matrix V (up to an arbitrary permutation and scaling of its rows).

Let us consider for example the spatial covariance matrix of the mixed signals $x(t)$,

$$C_{(x)} \stackrel{\text{def}}{=} E\{x(t)x(t)^T\} = E\{(As(t))(As(t))^T\} = AE\{s(t)s(t)^T\}A^T,$$

where the expectation is taken over t. We see that theoretically $C_{(x)} = AC_{(s)}A^T$ is similar to a diagonal matrix, because the cross-correlation terms that form the off-diagonal part of $C_{(s)}$ are zero

for independent signals. There are many more possibilities to define matrices that have the same property as the covariance matrix, namely that they are diagonal for the source signals and 'similar to diagonal' for the observed mixtures and, most important, that the inverse $V = A^{-1}$ of the mixing matrix A diagonalizes them all simultaneously. Examples are time-lagged covariances (Molgedey and Schuster, 1994; Belouchrani et al., 1997; Ziehe and Müller, 1998), covariance matrices of different segments of the data (Pham and Cardoso, 2000; Choi et al., 2001), matrices obtained from spatial time-frequency distributions (Pham, 2001), slices of the cumulant tensor (Cardoso, 1999) or Hessians of the characteristic function (Yeredor, 2000). Generally, for stationary and temporally correlated signals we can define a set of matrices C^k with entries

$$(C_{(x)})^k_{ij} = \frac{1}{2}\sum_{t=1}^{T} x_i(t)(\Phi^k \star x_j)(t) + x_j(t)(\Phi^k \star x_i)(t), \tag{23}$$

where $\star$ denotes convolution and $\Phi^k(t)$, $k = 1,\ldots,K$ are linear filters (Ziehe et al., 2000b).

We note that in the popular special case where the Φ^k are simple time-shift operators $\Phi^\tau(t) = \delta_{t\tau}$ (cf. Tong et al., 1991; Molgedey and Schuster, 1994; Belouchrani et al., 1997) the matrices defined by Equation (23) may become indefinite for certain choices of τ. Furthermore, in practice, the above target matrices have always to be estimated from the available data which inevitably gives rise to estimation errors. Hence the best we can do is to find the matrix which diagonalizes the estimated target set "as good as possible". Since we are able to perform the approximate joint diagonalization with a non-orthogonal transformation, we avoid the problematic pre-whitening step and obtain an estimate of the mixing matrix $A = V^{-1}$ by applying our FFDIAG algorithm directly to the empirical matrices (23). Algorithm 2 summarizes the typical steps in an application to BSS.

Algorithm 2 The FFSEP algorithm

INPUT: $x(t)$, Φ^k
$C^k = \ldots$ {Estimate a number of matrices C^k according to Equation (23)}
$V = \text{FFDIAG}(C^k)$ {Apply joint diagonalization method}
$u(t) = Vx(t)$ {unmix signals}
OUTPUT: u(t), V

6. Numerical Simulations

The experiments in this Section are intended to compare the FFDIAG algorithm with state-of-the-art algorithms for simultaneous diagonalization and to illustrate the performance of our algorithm in BSS applications. As we mentioned in the introduction, there exist at least three alternative formulations of the simultaneous diagonalization problem. The most successful algorithms representing the respective approaches were chosen for comparison.

We present the results of five progressively more complex experiments. First, we perform a "sanity check" experiment on a relatively easy set of perfectly diagonalizable matrices. This experiment is intended to emphasize that for small-size diagonalizable matrices the algorithm's performance matches the expected quadratic convergence. In the second experiment we compare the FFDIAG algorithm with the extended Jacobi method as used in the JADE algorithm of Cardoso and Souloumiac (1993) (orthogonal Frobenius norm formulation), Pham's algorithm for positive-definite matrices (Pham, 2001) and Yeredor's AC-DC algorithm (Yeredor, 2002) (non-orthogonal,

subspace fitting formulation). In the third experiment we investigate the scaling behavior of our algorithm as compared to AC-DC. Furthermore, the performance of the FFDIAG algorithm is tested and compared with the AC-DC algorithm on noisy, non-diagonalizable matrices. Finally, the application of our algorithm to BSS is illustrated.

6.1 "Sanity Check" Experiment

The test data in this experiment is generated as follows. We use $K = 15$ diagonal matrices D^k of size 5×5 where the elements on the diagonal are drawn from a uniform distribution in the range $[-1, \ldots, 1]$ (cf. Joho and Rahbar, 2002). These matrices are 'mixed' by an orthogonal matrix A according to AD^kA^T to generate the set of target matrices $\{C^k\}$ to be diagonalized.[8] The FFDIAG algorithm is initialized with the identity matrix $V_{(0)} = I$, and the skew-symmetric update rule (18) is used.

The convergence behavior of the algorithm in 10 runs is shown in Figure 2. The diagonalization error is measured by the off$(\cdot)$ function. One can see that the algorithm has converged to the correct solution after less than 10 iterations in all trials. The quadratic convergence is observed from early iterations.

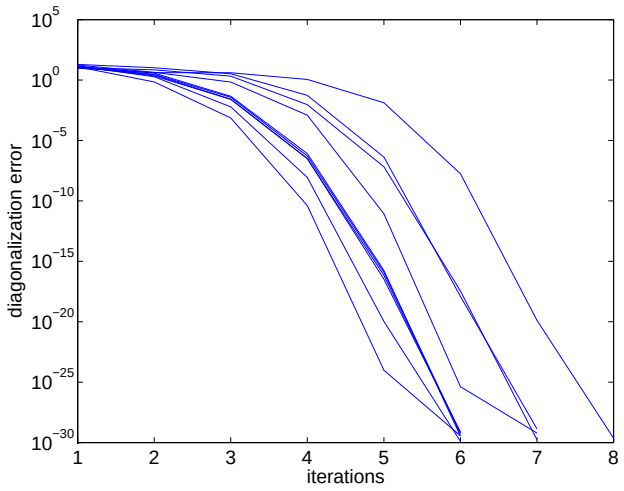


Figure 2: Diagonalization errors of the FFDIAG algorithm for a diagonalizable problem.

6.2 Comparison with the State-of-the-Art Algorithms

Two scenarios are considered for a comparison of the four selected algorithms: FFDIAG, the extended Jacobi method, Pham's algorithm and AC-DC. First, we test these algorithms on diagonalizable matrices under the conditions satisfying the assumptions of all of them. Such conditions are: positive-definiteness of the target matrices C^k and orthogonality of the true transformation A used to generate those matrices. These conditions are met by generating the target matrices $C^k = AD^kA^T$ where D^k are diagonal matrices with positive entries on the main diagonal. The data set consists of 100 random matrices of size 10×10 satisfying the conditions above.

8. The orthogonal matrix was obtained from a singular value decomposition of a random 5×5 matrix, where the entries are drawn from a standard normal distribution.

A comparison of the four algorithms on orthogonal positive-definite matrices is shown in Figure 3. Two runs of the algorithms are presented, for the AC-DC algorithm 5 AC steps were interlaced with 1 DC step at each iteration. Although the algorithms optimize different objective functions, the off$(\cdot)$ function is still an adequate evaluation criterion provided that the arbitrary scale is properly normalized.

To achieve this, we evaluate $\sum_k \text{off}(\hat{A}^{-1}C^k\hat{A}^{-T})$ where $\hat{A}$ is the normalized estimated mixing matrix. At the true solution the criterion must attain zero. One can see that the convergence of Pham's algorithm, the extended Jacobi method and FFDIAG is quadratic, whereas the AC-DC algorithm converges linearly. The average iteration complexity of the four algorithms is shown in Table 1. It

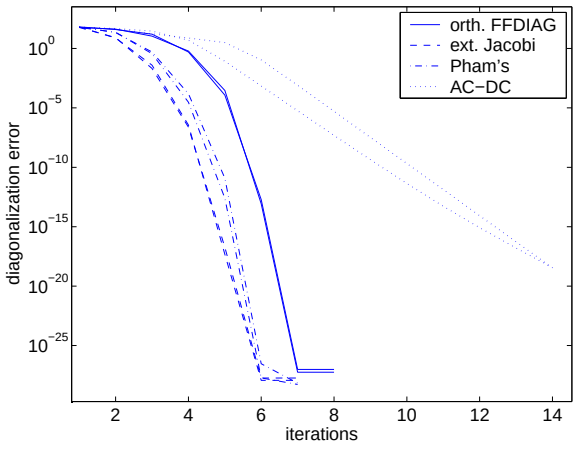


Figure 3: Comparison of the FFDIAG, the extended Jacobi method, Pham's algorithm and AC-DC in the orthogonal, positive-definite case: diagonalization error per iteration measured by the off$(\cdot)$ criterion.

follows from this table that the FFDIAG algorithm indeed lives up to its name: its running time per iteration is superior to both Pham's algorithm and AC-DC, and is comparable to the extended Jacobi method algorithm.[9]

FFDIAG	ext. Jacobi	Pham's	AC-DC
0.025	0.030	0.168	2.430

Table 1: Comparison of the FFDIAG, ext. Jacobi, Pham's and AC-DC algorithms in the orthogonal, positive-definite case: average running time per iteration in seconds.

In the second scenario, the comparison of the FFDIAG and the AC-DC algorithms is repeated for non-positive-definite matrices obtained from a non-orthogonal mixing matrix. This case cannot be handled by the other two algorithms, therefore they are omitted from the comparison. The convergence plots are shown in Figure 4; average running time per iteration is reported in Table 2.

9. In all experiments, MATLAB implementations of the algorithms were run on a standard PC with a 750MHz clock.

Convergence behavior of the two algorithms is the same as in the orthogonal, positive-definite case; the running time per iteration of FFDIAG increases due to the use of non-skew-symmetric updates.

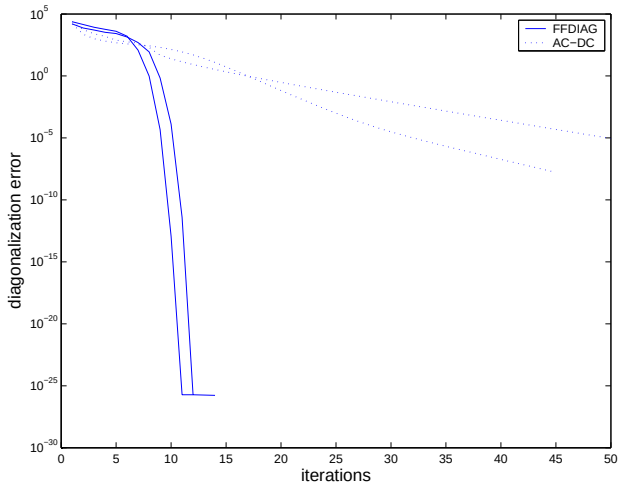


Figure 4: Comparison of the FFDIAG and AC-DC algorithms in the non-orthogonal, non-positive-definite case: diagonalization error per iteration measured by the off$(\cdot)$ criterion.

FFDIAG	AC-DC
0.034	2.64

Table 2: Comparison of the FFDIAG and AC-DC algorithms in the non-orthogonal, non-positive-definite case: average running time per iteration in seconds.

6.3 Scaling Behavior of FFDIAG

Scalability is essential for application of an algorithm to real-life problems. The most important parameter of the simultaneous diagonalization problem affecting the scalability of an algorithm is the size of the matrices. Figure 5 shows the running time per iteration of the FFDIAG and the AC-DC algorithms for problems with increasing matrix sizes, plotted at logarithmic scale. One can see that both algorithm exhibit running times of $O(N^2)$; however, in absolute terms the FFDIAG algorithm is almost two orders of magnitude faster.[10]

6.4 Non-diagonalizable Matrices

We now investigate the impact of non-diagonalizability of the set of matrices on the performance of the FFDIAG algorithm. Again, two scenarios are considered: the one of the "sanity check" ex-

10. This seemingly controversial result—theoretically expected scaling factor of AC-DC is $O(N^3)$—is due to high constants hidden in the setup phase of AC-DC. The setup phase has $O(N^2)$ complexity, but because of the constants it outweighs the main part of the algorithm in our experiment.

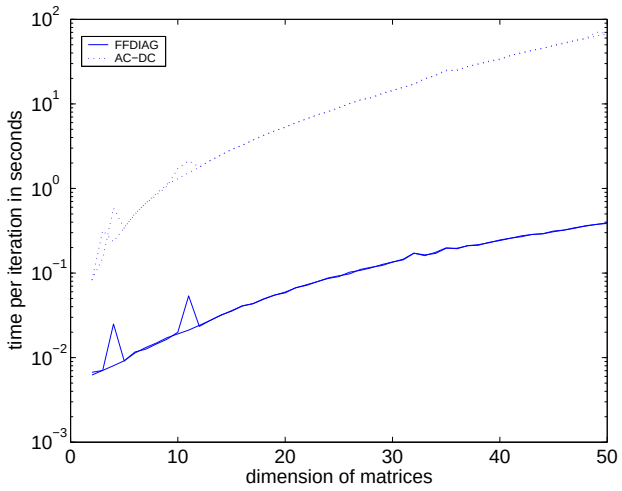


Figure 5: Scaling of the FFDIAG and AC-DC algorithms with respect to the matrix size. Two repetitions of the experiment have been performed.

periment and the comparative analysis against the established algorithms. Non-diagonalizability is modeled by adding a random non-diagonal symmetric "noise" matrix to each of the input matrices:

$$C^k = AD^kA^T + \sigma^2(R^k)(R^k)^T,$$

where the elements of R^k are drawn from a standard normal distribution. The parameter σ allows one to control the impact of the non-diagonalizable component. Another example, with a more realistic noise model, will be presented in subsection 6.5.

Figure 6 shows the convergence plots of FFDIAG for various values of σ. The experimental setup is the same as in Section 6.1, apart from the additive noise. The impact of the latter can be quantified by computing the off$(\cdot)$ function on the noise terms only (averaged over all runs), which is shown by the dotted line in Figure 6. One can see that the algorithm converges quadratically to the level determined by the noise factor.

Similar to the second scenario in Section 6.2, the previously mentioned algorithms are tested on the problem of approximate joint diagonalization with non-orthogonal transforms. (Only the extended Jacobi algorithm had to be excluded from the comparison since it is not designed to work with non-orthogonal diagonalizers.) However, in contrast to Section 6.2, positive-definite target matrices were generated in order to enable a comparison with Pham's method.

Furthermore, we introduce another measure to assess the algorithms' performance in the non-orthogonal, non-diagonalizable case. In synthetic experiments with artifical data the distance from the true solution is a good evaluation criterion. To be meaningful, this distance has to be invariant w.r.t. the irrelevant scaling and permutation ambiguities. For this reason, we choose a performance index that is commonly used in the context of ICA/BSS where the same invariances exist (see e.g. in Amari and Cichocki, 1998; Cardoso, 1999). Following the formulation of Moreau (2001) a suitable performance index is defined on the normalized "global" matrix $G \stackrel{\text{def}}{=} VA$ according to

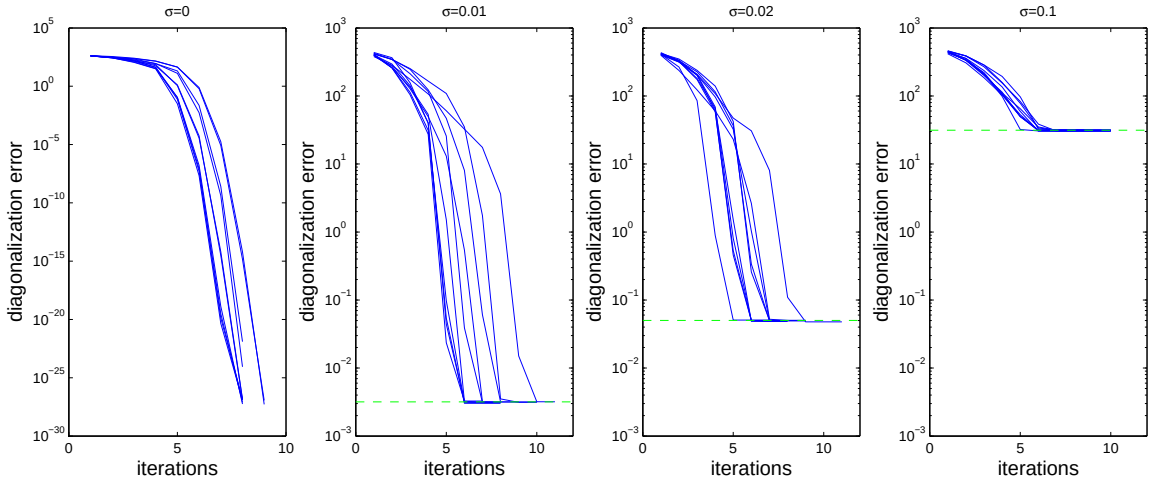


Figure 6: Diagonalization errors of the FFDIAG algorithm on non-diagonalizable matrices.

$$\text{score}(G) = \frac{1}{2}\left[\sum_i \left(\sum_j \frac{|G_{ij}|^2}{\max_l |G_{il}|^2} - 1\right) + \sum_j \left(\sum_i \frac{|G_{ij}|^2}{\max_l |G_{lj}|^2} - 1\right)\right]. \tag{24}$$

Clearly, this non-negative index attains zero iff G is a product of an invertible diagonal matrix D and of a permutation matrix P, i.e., $G = DP$.

The results of the comparison of the FFDIAG, Pham's and AC-DC algorithms on a non-orthogonal positive-definite problem (5 matrices of dimension 5×5) at various noise levels are shown in Figure 7 for three typical runs. The graphs illustrate some interesting ascpects of the convergence behavior of the algorithms. Both the FFDIAG and Pham's algorithm converge within a small number of iterations to approximately the same error level. The AC-DC algorithm converges linearly, and occasionally convergence can be very slow, as can be seen in each of the plots in Figure 7. However, when AC-DC converges, it exhibits better performance as measured by the score function; the higher the noise level, the stronger the difference.

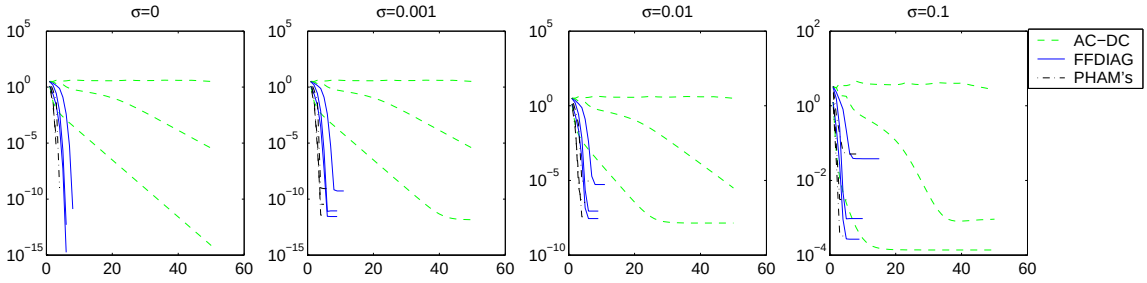


Figure 7: Comparison of the FFDIAG, Pham's and AC-DC algorithms in the non-diagonalizable, non-orthogonal, positive-definite case at various noise levels: performance index as measured by the score function (24).

6.5 Blind Source Separation

Finally, we apply our method to a blind source separation task. The signal matrix S contains seven audio signals containing 10000 points recorded at 8kHz and one Gaussian noise source of the same length. These signals are mixed by a 8×8 Hadamard matrix,

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

This scaled orthogonal matrix produces a complete mixture, in the sense that each observation contains a maximal contribution from each source. We compute 50 symmetrized, time-lagged correlation matrices according to Equation (23) with $\Phi^{\tau}(t) = \delta_{t\tau}$ and apply the FFDIAG algorithm with $V_{(0)} = I$. Figure 8 shows the evolution of performance measure defined in (24) and of the entries of the (normalized) global system $V_{(n)}A$. One can see that the difference from the true solution, in terms of the score function, approaches zero and that $V_{(n)}A$ converges to a permutation matrix (as shown in the middle and the right panels).

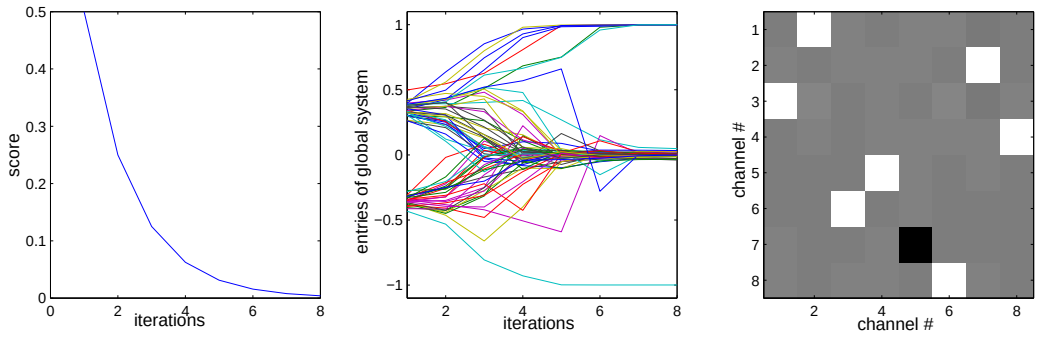


Figure 8: Convergence progress of the FFSEP algorithm on the BSS task. The middle panel shows the evolution of the entries of the normalized global matrix G. The right panel shows those entries for the final (8th) iteration in matrix form and indicates that the cross-talk is minimized since the matrix G resembles a scaled and permutated identity matrix. Here, black, white and gray squares correspond to values -1, 1 and 0, respectively.

In order to study the behavior of the FFDIAG algorithm in a more realistic noisy scenario the following experiment is conducted. The data is generated by mixing three stationary, time-correlated sources with the fixed matrix $A = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$. The sources are generated by feeding an i.i.d. random noise signal into a randomly chosen, auto-regressive (AR) model of order 5 whose coefficients are drawn from a standard normal distribution and are sorted in decreasing order (to ensure stability). The generated signals have a total length of 50000 samples. To separate the sources we estimate 10 symmetrized, time-lagged correlation matrices of the mixed signals according to Equation (23) with $\Phi^{\tau}(t) = \delta_{t\tau}$ and perform simultaneous diagonalization of these matrices.

Clearly, the quality of the estimation depends on the number T of samples used to estimate these correlation matrices. By varying T we can simulate different noise levels corresponding to

the variance of the estimates, which is more realistic than corrupting the target matrices with small i.i.d. additive noise.

The results of the experiment are shown in Figure 9. Performance of the FFDIAG and the AC-DC algorithms, as measured by the score (24), is displayed for four different sample sizes, the smaller samples corresponding to the higher noise level. 100 repetitions are performed for each sample size, and the 25%, 50% and 75% quantiles of the log-score are shown in the plots. Two observations can be made from Figure 9: FFDIAG converges much faster than AC-DC, and when converged, FFDIAG yields a better score (on average), with the difference more pronounced for samples sizes 10000 and 30000 in our experiment.

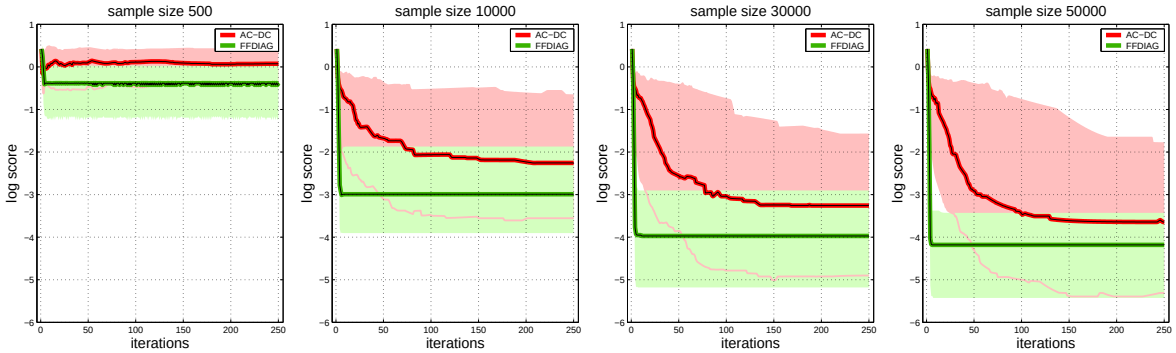


Figure 9: Performance of FFDIAG and AC-DC measured by the log of the score (24) for different sample sizes and 100 trials each. 25% (lower edge of the shaded region), 50% (thick line in the middle) and 75% quantiles (upper edge of the shaded region) are shown.

7. Discussion and Conclusions

We have presented a new algorithm FFDIAG for simultaneous diagonalization of a set of matrices. The algorithm is based on the Frobenius norm formulation of the simultaneous diagonalization problem and provides an efficient means of diagonalization in the absence of additional constraints, such as orthogonality or positive-definiteness. The important feature of our algorithm is the direct enforcement of invertibility of the diagonalizer; in previous work this was usually achieved by an orthogonality constraint which reduces the space of solutions.

The efficiency of the FFDIAG algorithm lies in the special second-order approximation of the cost function, which yields a block-diagonal Hessian and thus allows for highly efficient computation of the Newton update step. Although, theoretically, such approximation can be seen as a weakness of the approach—and raise the question of whether the point of the algorithm's convergence is indeed an optimizer of the full cost function—we have empirically observed that the solution found by the algorithm is of good quality for practical applications.

A series of comparisons of the FFDIAG algorithm with state-of-the-art diagonalization algorithms is presented under a number of conditions that can or cannot be handled by other algorithms. The main conclusions of this comparative evaluation is that our algorithm is competitive with the best algorithms (i.e. Jacobi-based and Pham's algorithm) that impose additional constraints either on the class of solutions or the type of input data. FFDIAG is significantly more efficient—as far as

both the scaling factors and the absolute constants are concerned—than the AC-DC algorithm, the only general algorithm applicable under the same conditions as ours. The FFDIAG algorithm can be applied to matrices of dimensions in the hundreds of rows/columns, under no additional assumptions. It also performs reliably on non-diagonalizable data, for which only an approximate solution is possible.

Several interesting research topics can be anticipated. From a theoretical point of view, convergence analysis could yield further insights into the numerical behavior of FFDIAG as well as a better understanding of the general techniques for optimization over nonholonomic manifolds that the algorithm belongs to. Further investigation of the robustness of joint diagonalization algorithms in the presence of various forms of noise is a very interesting practical issue. Numerous applications of the algorithm to real-life problems can be clearly foreseen.

Acknowledgments

The authors thank Benjamin Blankertz, Steven Lemm, Christin Schäfer, Sebastian Mika, Stefan Harmeling, Frank Meinecke, Guido Dornhege, Motoaki Kawanabe, David Tax, Julian Laub, Matthias Krauledat, Marcel Joho, Michael Zibulevsky and Arie Yeredor for sharing their insights and expertise in many fruitful discussions. Furthermore, the in-depth comments and valuable suggestions of the three anonymous reviewers are highly appreciated. This helped us to improve the initial version of the manuscript.

A.Z., P.L. and K.-R.M. acknowledge partial funding in the EU project (IST-1999-14190 – BLISS), by BMBF under contract FKZ 01IBB02A, the SFB 618 and the PASCAL Network of Excellence (EU #506778). G.N. has been supported by a grant from the National Foundation for Functional Brain Imaging.

References

J. P. Airoldi and B. Flury. An application of common principal component analysis to cranial morphometry of microtus californicus and m. ochrogaster (mammalia, rodentia). *Journal of Zoology*, 216:21–36, 1988.

T. Akuzawa and N. Murata. Multiplicative nonholonomic newton-like algorithm. *Chaos, Solitons & Fractals*, 12(4):785–793, 2001.

S.-I. Amari, T.-P. Chen, and A. Cichocki. Nonholonomic orthogonal learning algorithms for blind source separation. *Neural Computation*, 12:1463–1484, 2000.

S.-I. Amari and A. Cichocki. Adaptive blind signal processing – neural network approaches. *Proceedings of the IEEE*, 9:2026–2048, 1998.

A. Belouchrani, K. Abed Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique based on second order statistics. *IEEE Trans. on Signal Processing*, 45(2):434–444, 1997.

A. Bunse-Gerstner, R. Byers, and V. Mehrmann. Numerical methods for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications*, 14(4):927–949, 1993.

J.-F. Cardoso. On the performance of orthogonal source separation algorithms. In *Proc. EUSIPCO*, pages 776–779, 1994a.

J.-F. Cardoso. Perturbation of joint diagonalizers. ref# 94d027. Technical report, Télécom Paris, 1994b.

J.-F. Cardoso. High-order contrasts for independent component analysis. *Neural Computation*, 11 (1):157–192, January 1999.

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings F*, 140(6):362–370, 1993.

J.-F. Cardoso and A. Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM J. Mat. Anal. Appl.*, 17(1):161–164, January 1996.

S. Choi, A. Cichocki, and A. Belouchrani. Blind separation of second-order nonstationary and temporally colored sources. In *Proc. IEEE Workshop on Statistical Signal Processing (IEEE SSP 2001)*, pages 444–447, Singapore, 2001.

P. Comon. Independent component analysis, a new concept? *Signal Processing, Elsevier*, 36(3): 287–314, 1994.

M. R. Fengler, W. Härdle, and C. Villa. The dynamics of implied volatilities: A common principal components approach. Technical Report Discussion paper 2003-38, SFB 373, Humboldt-Universität zu Berlin, 2001.

B. Flury. *Common Principal Components and Related Multivariate Models*. Wiley, New York, 1988.

B. Flury and W. Gautschi. An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form. *SIAM Journal on Scientific and Statistical Computing*, 7(1):169–184, January 1986.

G. H. Golub and C. F. van Loan. *Matrix Computation*. The Johns Hopkins University Press, London, 1989.

S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15:1089–1124, 2003.

S. Haykin, editor. *Unsupervised adaptive filtering, Volume 1, Blind Source Separation*. John Wiley & Sons, New York, 2000.

G. Hori. Joint diagonalization and matrix differential equations. In *Proc. of NOLTA'99*, pages 675–678. IEICE, 1999.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press., Cambridge, 1985.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, New York, 2001.

M. Joho and K. Rahbar. Joint diagonalization of correlation matrices by using Newton methods with application to blind signal separation. In *Proc. of IEEE Sensor Array and Multichannel Signal Processing Workshop SAM*, pages 403–407, 2002.

C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

L. V. Kantorovich. On Newton's method. In *Trudy Mat. Inst. Steklov*, volume 28, pages 104–144. Akad. Nauk SSSR, 1949. Translation: Selected Articles in Numerical Analysis by C. D. Benster.

B. Laheld and J.-F. Cardoso. Equivariant adaptive source separation. *IEEE Trans. on Signal Processing*, 44(12):3017–3030, 1996.

T.-W. Lee, A. Ziehe, R. Orglmeister, and T. J. Sejnowski. Combining time-delayed decorrelation and ICA: Towards solving the cocktail party problem. In *Proc. ICASSP98*, volume 2, pages 1249–1252, Seattle, 1998.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, pages 164–168, 1944.

H. Lütkepohl. *Handbook of matrices*. John Wiley & Sons, 1996.

S. Makeig, A. J. Bell, T.-P. Jung, and T. J. Sejnowski. Independent component analysis of electroencephalographic data. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems (NIPS'95)*, volume 8, pages 145–151. The MIT Press, 1996.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of SIAM*, 11(2):431–441, jun 1963.

K. Matsuoka, M. Ohya, and M. Kawamoto. A neural net for blind separation of nonstationary signals. *Neural Networks*, 8:411–419, 1995.

L. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters*, 72(23):3634–3637, 1994.

J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In G. Watson, editor, *Lecture Notes in Mathematics*, volume 630, pages 105–116. Springer-Verlag, 1978.

E. Moreau. A generalization of joint-diagonalization criteria for source separation. *IEEE Trans. on Signal Processing*, 49(3):530–541, March 2001.

N. Murata, S. Ikeda, and A. Ziehe. An approach to blind source separation based on temporal structure of speech signals. *Neurocomputing*, 41(1-4):1–24, August 2001.

B. Noble and W. Daniel. *Applied matrix algebra*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1977.

M. R. Osborne. Nonlinear least squares - the Levenberg algorithm revisited. *Journal of Australian Mathematical Society, Series B*, 19:343–357, 1976.

L. Parra and C. Spence. Convolutive blind source separation based on multiple decorrelation. In *Proc. IEEE Workshop on Neural Networks for Signal Processing (NNSP'97)*, Cambridge, UK, 1998.

D.-T. Pham. Joint approximate diagonalization of positive definite matrices. *SIAM J. on Matrix Anal. and Appl.*, 22(4):1136–1152, 2001.

D.-T. Pham and J.-F. Cardoso. Blind separation of instantaneous mixtures of non-stationary sources. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 187–193, Helsinki, Finland, 2000.

D.-T. Pham and J.-F. Cardoso. Blind separation of instantaneous mixtures of non-stationary sources. *IEEE Trans. Sig. Proc.*, 49(9):1837–1848, 2001.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C*. Cambridge University Press., Cambridge, 1992.

L. Tong, V. C. Soon, and Y. Huang. Indeterminacy and identifiability of identification. *IEEE Trans. on Circuits and Systems*, 38(5):499–509, 1991.

A.-J. van der Veen. Joint diagonalization via subspace fitting techniques. In *Proc. ICASSP*, volume 5, 2001.

A. J. van der Veen, P. B. Ober, and E. F. Deprettere. Azimuth and elevation computation in high resolution DOA estimation. *IEEE Trans. Signal Processing*, 40(7):1828–1832, 1992.

A. J. van der Veen, M. C. Vanderveen, and A. Paulraj. Joint angle and delay estimation using shift-invariance techniques. *IEEE Trans. Signal Processing*, 46(2):405–418, 1998.

H. A. Van der Vorst and G. H. Golub. 150 years old and still alive: Eigenproblems. In *The State of the Art in Numerical Analysis*, volume 63, pages 93–120. Oxford University Press, 1997. URL `citeseer.nj.nec.com/vandervorst97years.html`.

R. Vigário, V. Jousmäki, M. Hämäläinen, R. Hari, and E. Oja. Independent component analysis for identification of artifacts in magnetoencephalographic recordings. In Jordan, Kearns, and Solla, editors, *Proc. NIPS 10*. The MIT Press, 1998.

C. von der Malsburg and W. Schneider. A neural cocktail-party processor. *Biological Cybernetics*, 54:29–40, 1986.

H.-C. Wu and J. C. Principe. Simultaneous diagonalization in the frequency domain (SDIF) for source separation. In *Proc. Int. Workshop on Independent Component Analysis and Blind Source Separation (ICA'99)*, pages 245–250, Aussois, France, January 11–15, 1999.

G. Wübbeler, A. Ziehe, B.-M. Mackert, K.-R. Müller, L. Trahms, and G. Curio. Independent component analysis of non-invasively recorded cortical magnetic DC-fields in humans. *IEEE Transactions on Biomedical Engineering*, 47(5):594–599, 2000.

A. Yeredor. Blind source separation via the second characteristic function. *Signal Processing*, 80 (5):897–902, 2000.

A. Yeredor. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Trans. on Sig. Proc.*, 50(7):1545–1553, July 2002.

M. Zibulevsky. Relative Newton method for quasi-ML blind source separation. In *Proc. 4th Intern. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 897–902, Nara, Japan, 2003.

A. Ziehe, P. Laskov, K.-R. Müller, and G. Nolte. A linear least-squares algorithm for joint diagonalization. In *Proc. 4th Intern. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 469–474, Nara, Japan, 2003.

A. Ziehe and K.-R. Müller. TDSEP–an efficient algorithm for blind separation using time structure. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN'98)*, pages 675–680, Skövde, Sweden, 1998.

A. Ziehe, K.-R. Müller, G. Nolte, B.-M. Mackert, and G. Curio. Artifact reduction in magnetoneurography based on time-delayed second-order correlations. *IEEE Trans Biomed Eng.*, 47(1): 75–87, January 2000a.

A. Ziehe, G. Nolte, G. Curio, and K.-R. Müller. OFI: Optimal filtering algorithms for source separation. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 127–132, Helsinki, Finland, 2000b.