MINIMAX SUPPORT VECTOR MACHINES

Mark A. Davenport, Richard G. Baraniuk *

Rice University
Department of Electrical and Computer Engineering

Clayton D. Scott

University of Michigan
Department of Electrical Engineering and Computer Science

ABSTRACT

We study the problem of designing support vector machine (SVM) classifiers that minimize the maximum of the false alarm and miss rates. This is a natural classification setting in the absence of prior information regarding the relative costs of the two types of errors or true frequency of the two classes in nature. Examining two approaches – one based on shifting the offset of a conventionally trained SVM, the other based on the introduction of class-specific weights – we find that when proper care is taken in selecting the weights, the latter approach significantly outperforms the strategy of shifting the offset. We also find that the magnitude of this improvement depends chiefly on the accuracy of the error estimation step of the training procedure. Furthermore, comparison with the minimax probability machine (MPM) illustrates that our SVM approach can outperform the MPM even when the MPM parameters are set by an oracle.

1. INTRODUCTION

In a typical classification setting we are given a sample of *training vectors* $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ each belonging to one of two classes, along with corresponding *labels* $y_1, \ldots, y_n \in \{-1, +1\}$ indicating the class for each training vector. Our task is then to find a function $f : \mathbb{R}^d \to \{+1, -1\}$ that "accurately" predicts the label when presented with a new sample, where accuracy is usually equated with having a small probability of error. However, there is a significant problem with this approach: in assuming that a classifier which has a small error rate on the training data will have a small probability of error when applied to the larger population we are implicitly assuming that the class frequencies in the training data accurately reflect the true prior probabilities in nature, which is frequently not the case. Moreover, in many cases the training data is *unbalanced* – meaning that we have many more samples from one class than from the other. Without knowledge of the true prior probabilities *and* the relative costs of the two types of errors, we have no reason to favor one class over the other. However, an algorithm that attempts to minimize the probability of error will tend to emphasize a class in proportion to its representation is the training set, which may reflect neither the true prior probability of the class nor the actual cost of the errors from that class.

For these reasons, many researchers prefer to use classifiers operating at the the *break even point* (BEP) or *equal error rate* (EER), meaning that $P_F(f) = P_M(f)$, where

$$P_F(f) = \Pr(f(\mathbf{x}) = +1 | y = -1) \quad \text{and} \tag{1}$$

$$P_M(f) = \Pr(f(\mathbf{x}) = -1 | y = +1) \tag{2}$$

denote the *false alarm* and *miss* rates of f, respectively. See, for example, [1, 2]. Of course, if an algorithm is sufficiently flexible, it is possible that many classifiers will satisfy $P_F(f) = P_M(f)$. We seek the best one possible, which we shall denote the *minimax* classifier.[1] Specifically, the minimax classifier is defined as

$$f^*_{mm} = \arg\min_f \max \left(P_F(f), P_M(f)\right). \tag{3}$$

Traditional wisdom holds that we should be able to estimate the minimax classifier by simply applying the "cost-sensitive" extensions that exist for many common classification algorithms. These algorithms seek to minimize a more general "misclassification cost" rather than the probability of error. However, assigning the costs appropriately in practical settings is often difficult (see [5, 6]). In this work we emphasize the importance of accurate error estimation in solving these problems – it is precisely the ability of an algorithm to leverage accurate error estimation techniques to tune the free parameters appropriately that will determine whether an algorithm will be able to give us the desired performance.

This paper studies support vector machines (SVMs) for minimax classification. We evaluate two approaches for adjusting the false alarm and miss rates. One involves shifting an offset parameter, resulting in an affine shift of the decision boundary, and the other entails introducing an additional parameter to control the relative weight given to each class. It is clear that both approaches affect the desired tradeoff between false alarms and misses. As might be expected, we find that shifting the offset does not perform as well as introducing an additional parameter to control the relative weights. However, optimizing over this additional parameter significantly increases the training time, and thus we also evaluate a method for greatly reducing the complexity of the expanded search with no significant loss in performance. We also suggest a method for decreasing the variance of the error estimates – which significantly affects the performance of our algorithms – highlighting the importance of accurate error estimation in minimax classification. We then compare the proposed algorithms to the minimax probability machine, showing that the 2ν-SVM can outperform the MPM even when the MPM parameters are set by an oracle. Our code (based on the LIBSVM packaged [7]) is available at www.dsp.rice.edu/software.

*Supported by NSF, DARPA, AFOSR, ONR, and the Texas Instruments Leadership University Program.

Email: {md,richb}@rice.edu, cscott@eecs.umich.edu Web: dsp.rice.edu

[1] Somewhat related to minimax classification is the *Neyman-Pearson* (NP) framework, in which the user sets a target false alarm rate α, and the goal of the algorithm is to minimize the miss rate subject to the condition that the false alarm rate is no greater than α. This is quite natural in many settings, especially when one class is more or less important than the other. In a sense, we can view minimax classification as NP classification where α is automatically chosen to obtain equal false alarm and miss rates. For more information on NP classification see [3, 4].

2. SUPPORT VECTOR MACHINES

SVMs are among the most effective methods for classification [8]. Conceptually, we construct a support vector classifier in a two step process. In the first step, we transform the training vectors $\mathbf{x}_i$ via a mapping $\Phi : \mathbb{R}^d \to \mathcal{H}$ where $\mathcal{H}$ is a high (possibly infinite) dimensional Hilbert space. Our intuition is that we should be able to more easily separate the two classes in $\mathcal{H}$ than in $\mathbb{R}^d$. For algorithmic reasons, Φ is chosen so that we can compute inner products in $\mathcal{H}$ without explicitly evaluating Φ through the use of a *kernel* operator $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}$.

In the second step, we determine a hyperplane in the induced feature space according to the max-margin principle. In the case where we can separate the two classes by a hyperplane, the SVM chooses the hyperplane that maximizes the *margin* – the distance between the decision boundary and the closest point to the boundary. When we cannot separate the classes by a hyperplane, we relax the constraints through the introduction of slack variables ξ_i. If $\xi_i > 0$, this means that the corresponding $\mathbf{x}_i$ lies inside the margin and is called a *margin error.* If $\mathbf{w} \in \mathcal{H}$ and $b \in \mathbb{R}$ are the normal vector and affine shift defining the max-margin hyperplane, then the support vector classifier is given by $f_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sgn}(k(\mathbf{w}, \mathbf{x}) + b)$. The offset parameter b is often called the *bias.*

One possible SVM formulation is the so-called ν-SVM [9]:

$$
\begin{aligned}
(P_\nu) \qquad & \min_{\mathbf{w},b,\boldsymbol{\xi},\rho} \quad \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{n}\sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad & y_i(k(\mathbf{w}, \mathbf{x}_i) + b) \geq \rho - \xi_i \qquad \text{for } i = 1, 2, \ldots, n \\
& \xi_i \geq 0 \qquad \text{for } i = 1, 2, \ldots, n \\
& \rho \geq 0
\end{aligned}
$$

where $\nu \in [0, 1]$ is a parameter set by the user. This formulation implicitly penalizes any error equally regardless of its class. However, as described in the introduction, this is not desirable in minimax classification. To address this issue, we consider the cost-sensitive extension of the ν-SVM – the 2ν-SVM [10]:

$$
\begin{aligned}
(P_{2\nu}) \qquad & \min_{\mathbf{w},b,\boldsymbol{\xi},\rho} \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{\gamma}{n}\sum_{i\in I_+} \xi_i + \frac{1-\gamma}{n}\sum_{i\in I_-} \xi_i \\
\text{s.t.} \quad & y_i(k(\mathbf{w}, \mathbf{x}_i) + b) \geq \rho - \xi_i \qquad \text{for } i = 1, 2, \ldots, n \\
& \xi_i \geq 0 \qquad \text{for } i = 1, 2, \ldots, n \\
& \rho \geq 0
\end{aligned}
$$

where $\gamma \in [0, 1]$ is a parameter for trading off the two types of errors. An equivalent parametrization replaces ν and γ with $\nu_+ = \nu n/(2\gamma|I_+|)$ and $\nu_- = \nu n/(2(1-\gamma)|I_-|)$. This parametrization has the benefit that $(P_{2\nu})$ is feasible if and only if $\nu_+ \leq 1$ and $\nu_- \leq 1$ (see [11]), and thus we focus on this parametrization.

3. MINIMAX LEARNING

We now return to the problem of minimax classification. We consider two main strategies for controlling false alarms. The first is to use the 2ν-SVM to achieve the desired false alarm and miss rates by adjusting ν_+ and ν_- appropriately. The second approach is to train a ν-SVM and then shift b (the bias) to achieve the desired false alarm and miss rates. In what follows $\widehat{P}_F(f)$ and $\widehat{P}_M(f)$ denote empirical estimates of $P_F(f)$ and $P_M(f)$.

3.1. 2ν-SVM Approach to Minimax Classification

As described in Section 2, the 2ν-SVM has two possible parameterizations. The (ν_+, ν_-) parameterization has the benefit that the dual formulation of $(P_{2\nu})$ is feasible if and only if $\nu_+ \leq 1$ and $\nu_- \leq 1$, with a trivial solution if $\nu_+ \leq 0$ or $\nu_- \leq 0$ (see [11]). Therefore, to search over the parameters of the 2ν-SVM it suffices to conduct a search over a uniform grid of (ν_+, ν_-) in $[0, 1]^2$. Hence, the full algorithm for minimax classification with the 2ν-SVM is to search over ν_+, ν_-, and any kernel parameters, obtain estimates of $P_F(f)$ and $P_M(f)$ using an error estimation technique such as cross-validation, and select the parameter combination minimizing $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$.

Smoothing the error estimates. We have observed across a wide range of datasets that $\widehat{P}_F(f)$ and $\widehat{P}_M(f)$ tend to display a slowly varying (low-frequency) trend when plotted as functions of (ν_+, ν_-). However, these estimates also appear somewhat "noisy". Without smoothing, some grid points will look much better than they actually are, due to chance variation. Thus, a heuristic offering potential improvement for the full grid search over (ν_+, ν_-) is to smooth both $\widehat{P}_F(f)$ and $\widehat{P}_M(f)$ with a low-pass filter after estimating the error at each point on the grid. In our experiments we consider two smoothing strategies: we can either apply a two-dimensional smoothing filter (we use a simple Gaussian window) to the error estimates for $(\nu_+, \nu_-) \in [0, 1]^2$ separately for each value of the kernel parameter, or we can apply a three-dimensional smoothing filter to the error estimates, smoothing across different kernel parameter values. Both strategies effectively reduce the variance of the error estimates. This approach is especially effective for high variance estimates like cross-validation. This technique illustrates another advantage of the (ν_+, ν_-) parametrization since the ability to discretize the parameter space of the 2ν-SVM with a uniform grid plays a key role in justifying this heuristic.

Coordinate descent: Speeding up the 2ν-SVM. The additional parameter in the 2ν-SVM renders a full grid search somewhat time consuming, especially for large data sets. Fortunately, a simple speed-up is possible. Again inspired by the smoothness of $P_F(f)$ and $P_M(f)$ as functions of (ν_+, ν_-), instead of conducting a full grid search over (ν_+, ν_-) we propose a coordinate descent search. Several variants are possible, but the ones we employ run as follows: For a fixed value of the kernel parameter, find the best parameters on grids placed along the lines $\nu_+ = 1/2$ and $\nu_- = 1/2$. From then on, conduct a line search in the direction orthogonal to the previous line search, at each step selecting the parameters minimizing $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, repeating this procedure for each kernel parameter value. Just as before, a simple three-dimensional extension of this algorithm is also considered, along with various approaches to smoothing $\widehat{P}_F(f)$ and $\widehat{P}_M(f)$.

3.2. Alternative Approaches to Minimax Classification

Bias-shifting. A potential advantage of the *bias-shifting* strategy is the ability to separate the training into two stages. First, we search over the parameters of the SVM (ν and any kernel parameters). Using an error estimation method such as cross-validation (CV), we then select the parameters that minimize either the misclassification rate or $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$. Second, once ν has been selected we shift the bias of the corresponding classifier and, again using some form of error estimation, select the bias that further minimizes $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$. In our experiments, we use the resubstitution estimate to select the bias. Resubstitution is generally a poor

estimate when the set of classifiers is complex; however, once we fix a normal vector $\mathbf{w}$, the set of possible shifted hyperplanes is a class with low complexity, and so resubstitution is in fact a reasonable error estimate. Note that we can apply the same technique to an SVM trained using the 2ν-SVM approach described above in the hope that it will improve the performance of that method as well.

Balanced ν-SVM. A common motivation for minimax classification is that some datasets are unbalanced in the sense that they have many more samples from one class than from the other. In this scenario, another possible algorithm is to apply the strategy described above for the ν-SVM, but instead to use a 2ν-SVM with $\nu_+ = \nu_-$. We refer to this method as the *balanced ν-SVM*. Since ν_+ and ν_- are upper bounds on the fractions of margin errors from their respective classes, we might expect that this method will be superior to the traditional ν-SVM. Note that this method has the same computational complexity as the traditional ν-SVM.

Minimax probability machine. The *minimax probability machine* (MPM) is a kernel-based alternative to the SVM that is specifically designed for minimax classification [12]. The general idea is to use the training data to estimate the mean and covariance matrices for each of the two classes, and then select the (hyperplane) classifier that minimizes $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$ for the worst-case over all possible choices of class-conditional densities whose (class-conditional) means and covariance matrices match those estimated from the training data. Since the means and covariance matrices estimated from the training data will be subject to some error, the user must set up to four parameters that reflect the uncertainty in these estimates. The MPM can be kernelized in a similar manner to the SVM, and the two algorithms have similar computational complexity, although the MPM has a greater number of free parameters to tune.

4. EXPERIMENTS

4.1. Experimental setup

We ran our algorithms on a collection of benchmark datasets that are available online with documentation.[2] The datasets comprise a mixture of synthetic datasets and datasets based on real data collected from various repositories on the web. The datasets are summarized in Table 1. For each of the first 9 data sets, we have 100 permutations of the training and test data, and for the last two ("image" and "splice") we have 20 permutations.

In all of our experiments we used a radial basis function (Gaussian) kernel and searched for the bandwidth parameter σ over a logarithmically spaced grid of 50 points from 10^{-4} to 10^4. For the ν-SVM method we searched over a uniform grid of 50 points of the parameter ν, and for the balanced ν-SVM we searched over a uniform grid of 50 points of the parameter $\nu_+ = \nu_-$. For the 2ν-SVM methods we considered a 50×50 regular grid of $(\nu_+, \nu_-) \in [0, 1]^2$. For each parameter combination, we estimated $P_F(f)$ and $P_M(f)$ using 5-fold cross-validation. In adjusting the bias for any of these methods we selected the optimal bias according to the resubstitution estimate. We applied a 3×3 Gaussian window to the error estimates to implement 2-dimensional smoothing, and we applied a $3 \times 3 \times 3$ Gaussian window to the error estimates to implement 3-dimensional smoothing. The standard deviation of the Gaussian window was set to the length of one grid interval. Different window sizes and widths were tried, but without much change in performance. For the coordinate descent methods we used the same grid structure described above. In our experiments with the ν-SVM we used the LIBSVM package [7]. For the 2ν-SVM we implemented our own version that is available online at www.dsp.rice.edu/software.

[2] http://ida.first.fhg.de/projects/bench/

Table 1. *Description of benchmark datasets used in our experiments: d denotes the dimension of the feature vectors, n (Training/Testing) the number of feature vectors in the training and test sets, and $\bar{n}_+$ ($\bar{n}_-$) the average number of feature vectors in the training set from the positive (negative) class.*

Dataset	d	n (Training)	$\bar{n}_+$	$\bar{n}_-$	n (Testing)
banana	2	400	182	218	4900
cancer	9	200	59	141	77
diabetes	8	468	164	304	300
flare-solar	9	666	368	298	400
heart	13	170	76	94	100
ringnorm	20	400	199	201	7000
thyroid	5	140	43	97	75
twonorm	20	400	202	198	7000
waveform	21	400	132	268	4600
image	18	1300	746	554	1010
splice	60	1000	483	517	2175

For each permutation of each dataset we ran our algorithms on the training data and estimated the false alarm and miss rates using the test data. On any given permutation, our performance metric is $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, where $\widehat{P}_F(f)$ and $\widehat{P}_M(f)$ now denote the false positive and miss rates estimated using the *test data*. To generate a more reliable performance estimate, we repeat this for each permutation and then average the minimax scores over all permutations – a procedure known as Monte Carlo cross-validation [13]. To evaluate performance on unbalanced datasets, we repeated these experiments retaining only 10% of the negatively labeled training data.

We use two main statistical tests to compare the algorithms described above, as advocated in [14]. In the case where we want to make a direct comparison between only two algorithms, we use the Wilcoxon signed-ranks test, which ranks the differences in performances of the two classifiers over the 11 datasets, and then compares the ranks for the positive and negative differences to test if the observed differences between the two algorithms is statistically significant. When reporting results from the Wilcoxon signed-ranks test, we will give the p-value, or probability of obtaining the observed differences by chance.

When we wish to compare more than two algorithms on multiple datasets, we use a two-step procedure. First we use the Friedman test, which is a statistical test similar to the Wilcoxon signed-ranks test in that it allows us to determine the probability of obtaining the observed performances by chance. Next, once we have rejected the null-hypothesis (that the differences have occurred by chance) we apply the Nemenyi test which involves computing a ranking of the algorithms for each dataset, and then an average ranking for each algorithm. Along with these rankings, we provide the so-called critical difference for a significance level of 0.05. (If the average ranking of two algorithms differs by more than this value, the performance of the two algorithms is significantly different with a p-value of 0.05.) See [14] for a more thorough discussion of and motivation for these techniques.

4.2. Results

Preliminary results. We begin by evaluating the performance of the 2ν-SVM. Perhaps somewhat surprisingly, bias-shifting actually

Table 2. *The effect of bias-shifting on the 2ν-SVM methods for minimax classification applied to both balanced and unbalanced datasets. In every case bias-shifting leads to worse performance. The table lists the p-values calculated using the Wilcoxon signed-ranks test indicating the significance of this difference in performance are listed for each 2ν-SVM method.*

Smoothing	Coordinate Descent	Balanced	Unbalanced
None	None	.148	.042
2-D	None	.001	.001
3-D	None	.005	.001
None	2-D	.107	.007
None	3-D	.032	.007
2-D	2-D	.001	.001
3-D	2-D	.002	.001
3-D	3-D	.032	.001

Table 3. *Comparison of smoothing methods for the 2ν-SVM for minimax classification. The table lists the average ranking for each approach. (Friedman test yields p-values of .001 for both balanced and unbalanced experiments. The critical difference for the Nemenyi test at 0.05 is 1.10.)*

Smoothing	Balanced	Unbalanced
None	2.91	2.91
2-D	1.73	1.64
3-D	1.36	1.45

results in uniformly worse performance for every 2ν-SVM-based method, with p-values below 0.05 in almost every case (see Table 2). As we will see again in our discussion of the balanced ν-SVM and the ν-SVM, bias-shifting only leads to improved performance when the SVM parameters have been selected to minimize the error rate. When the SVM parameters are selected to minimize $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, bias-shifting has a negative impact on overall performance.

On the other hand, smoothing and coordinate descent are extremely effective. The results of smoothing are shown in Table 3, and they clearly indicate that 2-D and 3-D smoothing offer a statistically significant gain in performance, with 3-D smoothing offering a slight edge. Similarly, the results in Table 4 show that 3-D smoothing combined with either 2-D or 3-D coordinate descent offer gains in performance as well, which is particularly helpful since these methods speed up the parameter selection process considerably.

Before we directly compare the smoothing and coordinate descent methods, we note that for the balanced ν-SVM and the traditional ν-SVM there are three main strategies: (1) adjust the SVM parameters to minimize $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, and *do not* adjust the bias, (2) adjust the SVM parameters to minimize $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, and *do* adjust the bias, or (3) adjust the SVM parameters to minimize the misclassification rate, and adjust the bias to minimize $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$. We refer the reader to [11] for a detailed comparison of these approaches. In general we find that strategy (2) always performs worst. For the traditional ν-SVM, we find that strategy (3) is most effective, while for the balanced ν-SVM the results indicate that the best method is to follow strategy (1). In all cases it is again beneficial to smooth the error estimates. Thus, we will compare the 2ν-SVM methods to the balanced ν-SVM *without* bias-shifting and the ν-SVM *with* bias-shifting.

Table 4. *Comparison of coordinate descent methods for the 2ν-SVM for minimax classification. The table lists the average ranking for each approach. (Friedman test yields p-values of .002 for balanced and .001 for unbalanced experiments. The critical difference for the Nemenyi test at 0.05 is 1.92.)*

Smoothing	Coordinate Descent	Balanced	Unbalanced
None	2-D	4.18	4.18
None	3-D	3.91	4.00
2-D	2-D	2.73	2.82
3-D	2-D	2.00	2.00
3-D	3-D	2.18	2.00

Table 5. *Minimax rates on the* balanced *datasets for the best 2ν-SVM methods, the balanced ν-SVM, and the ν-SVM with bias-shifting. Scores reported are $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, averaged over all 100 (or 20) permutations.*

Dataset	3D-GS	2D-CD	3D-CD	Bν-SVM	ν-SVM
banana	.129	.129	.129	.133	.132
cancer	.414	.419	.431	.490	.425
diabetes	.302	.301	.303	.304	.289
flare-solar	.355	.352	.433	.415	.365
heart	.226	.219	.224	.231	.221
ringnorm	.024	.023	.021	.022	.027
thyroid	.075	.078	.070	.076	.081
twonorm	.032	.031	.030	.029	.034
waveform	.119	.117	.116	.123	.135
image	.043	.050	.065	.039	.040
splice	.114	.118	.118	.113	.157

Overall results. We are now in a position to compare the 2ν-SVM strategies to the balanced ν-SVM and traditional ν-SVM. In Tables 5 and 6 we give the minimax error rates for the 3-D smoothing approach (labeled 3D-GS), the 2-D and 3-D coordinate descent methods (labeled 2D-CD and 3D-CD – both use 3-D smoothing), the balanced ν-SVM (labeled Bν-SVM), and the traditional ν-SVM with bias-shifting (labeled ν-SVM). Table 5 shows the performance of each algorithm on each dataset averaged over the permutations, and Table 6 shows the same for the unbalanced datasets. Table 7 gives the results of the Nemenyi test for these algorithms. In the balanced dataset experiments, the 2ν-SVM methods appear to exhibit stronger performance, but according to the Nemenyi test this difference is not statistically significant. However, for the *un*balanced datasets, there is a statistically significant difference with the 2ν-SVM methods being clearly superior. The 3D-GS method appears to be the best performing overall, but the coordinate descent methods exhibit very similar performance.

Finally, we also compare the 2ν-SVM with the MPM. Recall that the parameters for the MPM represent the uncertainty in our knowledge of the class-dependent means and covariance matrices. We can calculate this uncertainty *exactly* by calculating the differences between the means and covariances based on the training set and those based on the test set, which allows us to realize the best performance possible with the MPM. To make a fair comparison, we only set two free parameters – we follow [12] and assume that the uncertainty in the means and covariances is the same for both classes, and thus the MPM and 2ν-SVM would require roughly the same

Table 6. *Minimax rates on the* unbalanced *datasets for the best 2ν-SVM methods, the balanced ν-SVM, and the ν-SVM with bias-shifting. Scores reported are $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, averaged over all 100 (or 20) permutations.*

Dataset	3D-GS	2D-CD	3D-CD	Bν-SVM	ν-SVM
banana	.193	.194	.189	.226	.218
cancer	.451	.460	.477	.564	.737
diabetes	.340	.340	.338	.455	.449
flare-solar	.410	.412	.425	.595	.548
heart	.271	.286	.275	.413	.490
ringnorm	.048	.049	.040	.055	.088
thyroid	.133	.139	.126	.126	.135
twonorm	.060	.060	.058	.079	.099
waveform	.168	.171	.168	.210	.181
image	.134	.133	.157	.151	.097
splice	.195	.196	.200	.379	.335

Table 7. *Comparison of best 2ν-SVM methods for minimax classification, the balanced ν-SVM (without bias-shifting), and the ν-SVM (with bias-shifting). The table lists the average ranking for each approach. (Friedman test yields p-values of .502 for balanced and .0003 for unbalanced experiments. The critical difference for the Nemenyi test at 0.05 is 1.92.)*

Method	Balanced	Unbalanced
3D-GS	2.73	2.00
2D-CD	2.64	2.64
3D-CD	2.73	2.00
ν-SVM	3.64	4.09
Bal ν-SVM	3.27	4.27

computational complexity to set the parameters in a practical setting. We have also observed that the MPM algorithm is somewhat unstable with regard to kernel parameters, so we limit our experiments to the linear kernel, which also simplifies the comparison somewhat. We then compare this to the performance of the 2ν-SVM where the parameters for the 2ν-SVM are chosen via cross-validation. The results are given in Table 8. In the unbalanced case we do not see a significant difference, with each algorithm doing better on roughly half the datasets, although in this case we do see that when the 2ν-SVM outperforms the MPM it tends to do so by a large amount compared to cases where the MPM outperforms the 2ν-SVM. However, in the balanced case we get a clear difference in performance. Furthermore, in a more practical setting, in which the parameters for the MPM are set imperfectly, the 2ν-SVM would likely out-perform the MPM by an even greater margin.

5. CONCLUSIONS

We have evaluated several strategies for minimax classification with SVMs. As might be expected, the offset-shifting strategy does not perform as well as introducing an additional parameter to control the relative weights. This difference is especially pronounced in the case where the dataset is unbalanced – a particularly natural setting for minimax classification. A key insight from our study is that the performance of these algorithms is significantly affected by the accuracy of the error estimates. As we have demonstrated, simple heuristics for decreasing the variance of these estimates can significantly improve the overall performance. Hence, future work on minimax classification should focus on understanding how to improve these heuristics further.

Table 8. *Minimax rates for the 2ν-SVM with linear kernel (where ν_+ and ν_- are selected through cross validation, with smoothing of the error estimates) and the linear MPM where the parameters are chosen to be optimal for the test data set. Scores reported are $\max\{\widehat{P}_F(f), \widehat{P}_M(f)\}$, averaged over all 100 (or 20) permutations.*

Dataset	Balanced		Unbalanced	
	SVM	MPM	SVM	MPM
banana	.558	.482	.619	.517
cancer	.396	.401	.453	.421
diabetes	.291	.311	.332	.319
flare-solar	.350	.360	.392	.399
heart	.218	.205	.285	.238
ringnorm	.287	.308	.335	.302
thyroid	.197	.387	.253	.392
twonorm	.031	.027	.064	.038
waveform	.141	.180	.175	.218
image	.194	.341	.230	.362
splice	.175	.184	.239	.296

6. REFERENCES

[1] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, pp. 1–47, 2002.

[2] S. Bengio, J. Mariéthoz, and M. Keller, "The expected performance curve," in *Proc. Int. Conf. Machine Learning*, 2005, Bonn, Germany.

[3] C. D. Scott and R. D. Nowak, "A Neyman-Pearson approach to statistical learning," *IEEE Trans. Inform. Theory*, vol. 51, no. 11, pp. 3806–3819, 2005.

[4] M. A. Davenport, R. G. Baraniuk, and C. D. Scott, "Controlling false alarms with support vector machines," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, 2006, Toulouse, France.

[5] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[6] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proc. Int. Conf. Machine Learning*, 1998, San Francisco, CA.

[7] C. C. Chang and C. J. Lin, *LIBSVM: a library for support vector machines*, 2001, See http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[8] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[9] B. Schölkopf, A. J. Smola, R. Williams, and P. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, pp. 1083–1121, 2000.

[10] H. G. Chew, R. E. Bogner, and C. C. Lim, "Dual-ν support vector machine with error rate and training size biasing," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP)*, 2001.

[11] M. A. Davenport, *Error control for support vector machines*, MS thesis, Department of Electrical and Computer Engineering, Rice University, Houston, TX, 2007.

[12] G. R. G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan, "A robust minimax approach to classification," *J. Machine Learning Research*, vol. 3, pp. 555–582, 2002.

[13] J. Shao, "Linear model selection by cross-validation," *J. Amer. Statist. Assoc.*, vol. 88, no. 422, pp. 486–494, 1993.

[14] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Machine Learning Research*, vol. 7, pp. 1–30, 2006.