CONTROLLING FALSE ALARMS WITH SUPPORT VECTOR MACHINES

Mark A. Davenport, Richard G. Baraniuk [*]

Rice University
Department of Electrical and Computer Engineering

Clayton D. Scott [†]

Rice University
Department of Statistics

ABSTRACT

We study the problem of designing support vector classifiers with respect to a Neyman-Pearson criterion. Specifically, given a user-specified level $\alpha \in (0, 1)$, how can we ensure a false alarm rate no greater than α while minimizing the miss rate? We examine two approaches, one based on shifting the offset of a conventionally trained SVM and the other based on the introduction of class-specific weights. Our contributions include a novel heuristic for improved error estimation and a strategy for efficiently searching the parameter space of the second method. We also provide a characterization of the feasible parameter set of the 2ν-SVM on which the second approach is based. The proposed methods are compared on four benchmark datasets.

1. INTRODUCTION

Most approaches to classification attempt to infer a classifier that minimizes the probability of making an error. In many important applications, however, some kinds of errors are more important than others. In tumor classification, for example, the impact of mistakenly classifying a benign tumor as malignant is much less than that of the opposite mistake.

In the Neyman-Pearson (NP) classification paradigm, the goal is to design a classifier (based on training data) that minimizes the probability of a *miss* while constraining the probability of a *false alarm* to be less than some user-specified significance level α. Unlike Neyman-Pearson hypothesis testing in classical detection theory, NP classification relies entirely on training data, placing no parametric assumptions on the data.

The NP framework has two major advantages with respect to conventional classification criteria that seek to minimize the probably of error or, more generally, an expected misclassification cost. First, assigning costs is often less intuitive or reasonable than assigning a false alarm constraint. Second, NP classification does not assume knowledge of the a priori class probabilities. This is extremely important in applications where the class frequencies in the training data do not accurately reflect class probabilities in the larger population. In fact, it could probably be argued that most classification problems of interest fit this description. For a more detailed motivation for the Neyman-Pearson paradigm, see [1].

This paper studies support vector machines (SVMs) for NP classification. In the SVM literature two approaches have been suggested for controlling false alarms. One involves shifting the *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. What is not clear, however, is how to implement these ideas to achieve a *specific* false alarm level α. In other words, the primary challenge is accurate error estimation.

As might be expected, we find that shifting the offset parameter 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. We propose 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 are crucial for NP classification. Furthermore, we offer two contributions regarding the 2ν-SVM, the cost-sensitive SVM we employ. First, we present a theorem that precisely characterizes the feasible set for the defining quadratic program. Second, we have made available at www.dsp.rice.edu/software our code, which is based on the LIBSVM package [2].

2. SUPPORT VECTOR MACHINES

Support vector machines (SVMs) are among the most effective methods for classification [3]. Let $(\mathbf{x}_i, y_i)$, $i = 1, 2, \ldots, n$ denote the training data where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional feature vector and $y_i \in \{+1, -1\}$ indicates the class of $\mathbf{x}_i$. Conceptually, the support vector classifier is constructed in a two step process. In the first step, the $\mathbf{x}_i$ are transformed via a mapping $\Phi : \mathbb{R}^d \to \mathcal{H}$ where $\mathcal{H}$ is a high (possibly infinite) dimensional Hilbert space. The intuition is that the two classes should be more easily separated in $\mathcal{H}$ than in $\mathbb{R}^d$. For algorithmic reasons, Φ must be chosen so that the *kernel* operator $k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}$ is positive definite. This allows us to compute inner products in $\mathcal{H}$ without explicitly evaluating Φ.

In the second step, a hyperplane is determined in the induced feature space according to the max-margin principle. In the case where the two classes can be separated by a hyperplane, the SVM finds the hyperplane that maximizes the *margin* – the distance between the decision boundary and the closest point to the boundary. When the classes cannot be separated by a hyperplane, the constraints are relaxed 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 $f_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle_{\mathcal{H}} + b)$. The offset parameter b is often called the *bias*.

There are two different formulations of the SVM. The original SVM [4], which we will call the C-SVM, can be formulated as the

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

[†]Supported by an NSF VIGRE postdoctoral training grant.

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

following quadratic program:

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

where $C \geq 0$ is a tradeoff parameter that controls overfitting.

For computational reasons, it is often easier to solve (P_C) by solving its dual formulation. This is derived by forming the Lagrangian and then optimizing over the Lagrange multiplier $\boldsymbol{\alpha}$ instead of the primal variables. The primal and the dual are related through $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \Phi(\mathbf{x}_i)$. For most $\mathbf{x}_i$, we will have $\alpha_i = 0$. The $\mathbf{x}_i$ for which $\alpha_i \neq 0$ are called *support vectors.*

An alternative (but equivalent) formulation of the C-SVM is the ν-SVM [5], which replaces C with a different parameter $\nu \in [0,1]$ that serves as an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors. The ν-SVM has the primal formulation:

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

The above formulations implicitly penalize errors in both classes equally. However, as described in the introduction, in many applications there are different costs associated with the two different kinds of errors. To address this issue, cost-sensitive extensions of both the C-SVM and the ν-SVM have been proposed, which we shall denote the $2C$-SVM and the 2ν-SVM, respectively.

First we will consider the $2C$-SVM proposed in [6]. Let $I_+ = \{i : y_i = +1\}$ and $I_- = \{i : y_i = -1\}$. The $2C$-SVM has primal:

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

where $\gamma \in (0,1)$ is a parameter for trading off false alarms and misses.

Similarly, [7] proposed the 2ν-SVM as a cost-sensitive extension of the ν-SVM. The 2ν-SVM has primal:

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

Above we stated that (P_C) and (P_ν) are equivalent. This notion was made precise in [8]. We have extended this result to show that (P_{2C}) and $(P_{2\nu})$ are equivalent in the following sense. The proof is given in a supplemental technical report [9].

Theorem 1. *Let (D_{2C}) and $(D_{2\nu})$ denote the duals of (P_{2C}) and $(P_{2\nu})$ respectively. Fix $\gamma \in [0,1]$ and let $\boldsymbol{\alpha}^*$ be any optimal solution of (D_{2C}). Define*

$$\nu_* = \lim_{C\to\infty}\frac{\sum_{i=1}^{n}\alpha_i^*}{Cn}, \quad \nu^* = \lim_{C\to 0}\frac{\sum_{i=1}^{n}\alpha_i^*}{Cn}.$$

Then

$$0 \leq \nu_* \leq \nu^* = \frac{\min(\gamma n_+, (1-\gamma)n_-)}{n} \leq \frac{1}{2}$$

where $n_+ = |I_+|$ and $n_- = |I_-|$. Then $0 \leq \nu_* \leq \nu^* = \nu_{\max} \leq \frac{1}{2}$. Thus, for any $\nu > \nu^*$, $(D_{2\nu})$ is infeasible. For any $\nu \in (\nu_*, \nu^*]$ the optimal objective value of $(D_{2\nu})$ is strictly positive, thus there exists at least one $C > 0$ such that any $\boldsymbol{\alpha}$ is an optimal solution of (D_{2C}) if and only if $\boldsymbol{\alpha}/(Cn)$ is an optimal solution of $(D_{2\nu})$. For any $\nu \in [0, \nu_*]$, $(D_{2\nu})$ is feasible with zero optimal objective value (and a trivial solution).

3. CONTROLLING FALSE ALARMS

As mentioned in the Introduction, there are two main strategies for controlling false alarms. The first is to train a C-SVM or ν-SVM and then shift the bias (b) to achieve the desired false alarm rate. The second approach is to use the $2C$-SVM or 2ν-SVM and achieve the desired false alarm rate by adjusting γ appropriately. As described is Section 2, (P_{2C}) and $(P_{2\nu})$, as well as (P_C) and (P_ν), are closely related and equivalent in the sense that they explore the same set of possible solutions. For the remainder of this paper we restrict our attention to the ν-SVM and the 2ν-SVM, because their parameter spaces are more conveniently discretized. This is important because it makes the coordinate descent and windowing heuristics we describe below more reasonable.

In what follows $P_F(f)$ and $P_M(f)$ denote the false alarm and miss rates of a classifier f, and $\widehat{P}_F(f)$ and $\widehat{P}_M(f)$ denote estimates of these quantities.

3.1. Bias-shifting approach

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 the estimated probability of error. Second, we shift the bias of that chosen classifier and, again using some form of error estimation, select the bias minimizing $\widehat{P}_M$ such that $\widehat{P}_F \leq \alpha$. Note that the first error estimate must be nested within the second. 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.

Since some datasets are unbalanced, we can also apply the above strategy using a 2ν-SVM with $\nu_+ = \nu_-$ (where ν_+ and ν_- are as defined in Section 3.2) instead of the ν-SVM. This method is referred to as *balanced bias-shifting*.

3.2. 2ν approach

The cost-sensitive extension of the 2ν-SVM proposed in [7] is parameterized in a different manner than $(P_{2\nu})$. Specifically, instead

of parameters ν and γ, $(P_{2\nu})$ is formulated using ν_+ and ν_-, where

$$\nu = \frac{2\nu_+\nu_- n_+ n_-}{(\nu_+ n_+ + \nu_- n_-)n}, \;\; \gamma = \frac{\nu_- n_-}{\nu_+ n_+ + \nu_- n_-} = \frac{\nu n}{2\nu_+ n_+}$$

or equivalently

$$\nu_+ = \frac{\nu n}{2\gamma n_+}, \;\; \nu_- = \frac{\nu n}{2(1-\gamma) n_-}.$$

This parametrization has the benefit that ν_+ and ν_- have a more intuitive meaning, similar to the ν-SVM. Specifically, suppose that the optimal solution of $(P_{2\nu})$ satisfies $\rho > 0$. Then for the optimal solution of $(P_{2\nu})$ ν_+ is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors from class $+1$. Similarly, ν_- is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors from class -1. See [7] for a proof.

Furthermore, from Theorem 1 it follows 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 $\nu_- \leq 0$. Therefore, to search over the parameters of the 2ν-SVM it suffices to conduct a search over a uniform grid of (ν_+, ν_-) in $[0,1]^2$.

In sum, the full algorithm for NP classification with the 2ν-SVM is to conduct a grid search over the SVM parameters, estimate P_F and P_M using some error estimation technique, and select the parameter combination minimizing $\widehat{P}_M$ such that $\widehat{P}_F \leq \alpha$.

3.3. Coordinate descent: Speeding up the 2ν-SVM

The additional parameter in the 2ν-SVM renders a full grid search very time consuming. Fortunately, a simple speed-up is possible. We have observed across a wide range of datasets and kernels that the errors P_F and P_M vary smoothly when plotted as functions of $(\nu_+, \nu_-) \in [0,1]^2$. Thus, instead of conducting a full grid search over (ν_+, ν_-) we propose a kind of coordinate descent search. Several variants are possible, but the one we employ runs as follows: 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 $\widehat{P}_M$ such that $\widehat{P}_F \leq \alpha$. Note that this strategy would be more difficult to justify with the $2C$-SVM because the choice of endpoints and grid spacing would ultimately be arbitrary and data-dependent.

3.4. Windowing the estimated errors

The observation about the smoothness of P_F and P_M as functions of (ν_+, ν_-) leads to another heuristic improvement in all of our methods. For the full grid search over (ν_+, ν_-), after estimating the error at each point on the grid, we low-pass filter both $\widehat{P}_F$ and $\widehat{P}_M$ with a Gaussian window. This effectively reduces the variance of the error estimates. It is especially effective for high variance estimates like cross-validation. Without windowing, some grid points will look much better than they actually are, due to chance variation. For coordinate descent we window along lines in the grid, and for the bias shifting approach, we window the estimates across the ν grid. As with the coordinate descent strategy, the ability to discretize the parameter space of the 2ν-SVM with a uniform grid plays a key role in justifying this heuristic.

4. MEASURING PERFORMANCE

Any experimental comparison of classifiers requires a measure of performance, that is, a scalar criterion that we can evaluate to compare two classifiers head-to-head. In Neyman-Pearson classification, if we only report the observed false alarm and miss probabilities, we will often observe that one classifier has a smaller P_F but larger P_M than another. In this case we cannot make a definitive statement about which classifier is better.

One option for a scalar performance measure is to estimate $P_F(f)$ and $P_M(f)$ and assign a "score" of $\widehat{P}_M(f)$ to the classifier if $\widehat{P}_F(f) \leq \alpha$, and ∞ otherwise, where $\widehat{P}_F$ and $\widehat{P}_M$ are based on an independent test sample. This is problematic, however, because the estimated false alarm rate is based on data and hence susceptible to error. Moreover, in practical settings, it is often acceptable to have $P_F(f)$ be a small amount greater than α.

We evaluate our classifiers using the measure

$$\mathcal{E}(f) = \frac{1}{\alpha} \max\{P_F(f) - \alpha, 0\} + P_M(f). \tag{1}$$

As discussed in [10], this measure satisfies the following desirable properties: (*i*) It is minimized by the classifier $f_\alpha^* = \arg\min\{P_M(f) \,|\, P_F(f) \leq \alpha\}$. (*ii*) It can be accurately estimated from a test sample using the simple plug-in estimate. (*iii*) It has the appealing property that as α draws closer to 0, a stiffer penalty is exacted on classifiers that violate the constraint. In other words, it penalizes the *relative* error $(P_F(f) - \alpha)/\alpha$.

5. EXPERIMENTS

In our experiments with the ν-SVM we used the LIBSVM package [2]. For the 2ν-SVM we implemented our own version that is available online at www.dsp.rice.edu/software.

We ran our algorithms on the benchmark datasets named "banana", "heart", "thyroid", and "breast." The datasets are available online with documentation.[1] The first dataset is synthetic, while the other three are based on real data collected from various repositories on the web. There are 100 permutations of each dataset into training and test data. The dimensions of the datasets are 2, 13, 5, and 9, respectively, and the training sample sizes are 400, 170, 140, and 200, respectively. The targeted α is 0.1.

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 bias-shifting method we searched over a uniform grid of the parameter ν of 50 points. For the 2ν-SVM methods we considered a 50×50 regular grid of $(\nu_+, \nu_-) \in [0,1]^2$.

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. Table 1 reports the average false alarm and miss rates over all 100 permutations, along with standard deviations. For each permutation we also computed the performance measure $\mathcal{E}$, and we show the median values in the table. The table also reports the average training time of each algorithm. The methods compared are bias-shifting with the ν-SVM (BS), balanced bias-shifting using the 2ν-SVM with $\nu_+ = \nu_-$ (BBS), the 2ν-SVM with a full grid search over (ν_+, ν_-) (GS), and the 2ν-SVM with a coordinate descent search over (ν_+, ν_-) (CD). For each of the last two methods we also applied a smoothing window to the error estimates as described in Section 3.4. These two variants are indicated by a "W".

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

Table 1. *Average values of P_F and P_M (over 100 permutations of each data set), the median NP error scores $\mathcal{E}$, and average running times for the six tested methods. Here $\alpha = 0.1$. The tested methods are bias-shifting (BS), balanced bias-shifting (BBS), and 2ν-SVM grid-search (GS), windowed grid search (WGS), coordinate descent (CD), and windowed coordinate descent (WCD).*

		P_F	P_M	$\mathcal{E}$	Time(s)
thyroid	BS	.057 ± .07	.455 ± .47	.637	19
	BBS	.089 ± .06	.059 ± .15	.077	32
	GS	.098 ± .09	.064 ± .09	.127	898
	WGS	.087 ± .06	.032 ± .05	**.051**	898
	CD	.084 ± .06	.039 ± .05	.066	55
	WCD	.093 ± .06	.032 ± .05	.082	55
heart	BS	.086 ± .09	.553 ± .39	1.000	58
	BBS	.113 ± .10	.368 ± .33	.681	66
	GS	.124 ± .06	.219 ± .07	.375	2801
	WGS	.113 ± .05	.231 ± .07	.326	2801
	CD	.106 ± .05	.230 ± .06	**.318**	169
	WCD	.110 ± .05	.231 ± .06	.330	169
breast	BS	.000 ± .00	1.00 ± .00	1.000	45
	BBS	.078 ± .23	.910 ± .24	1.000	83
	GS	.156 ± .09	.668 ± .10	1.122	2084
	WGS	.112 ± .06	.689 ± .10	**.821**	2084
	CD	.114 ± .06	.683 ± .10	.871	121
	WCD	.119 ± .06	.678 ± .10	.906	121
banana	BS	.109 ± .07	.334 ± .40	.628	212
	BBS	.142 ± .04	.104 ± .02	.464	221
	GS	.114 ± .03	.120 ± .02	.255	9727
	WGS	.104 ± .02	.124 ± .02	**.160**	9727
	CD	.104 ± .02	.125 ± .02	.179	541
	WCD	.106 ± .03	.124 ± .02	.198	541

The window size was 3×3 for GS and 1×3 for CD. 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. Windowing for BS and BBS was also tried but lead to no improvements. For all methods, the parameters were selected using leave-one-out cross-validation (LOOCV). For the bias-shifting approaches, the bias was selecting using the resubstitution estimate.

6. CONCLUSION

We have studied two general approaches to controlling the false alarm rate with SVMs. Using the ν-SVM and 2ν-SVM, respectively, we experimentally evaluated the strategies of adjusting the bias and weighting the margin errors. Since the latter approach introduces an additional parameter and is therefore much slower than the former, we also studied a heuristic speed-up based on a greedy coordinate descent search through the 2ν-SVM parameter space.

Based on the results in Table 1, we can definitively conclude that the 2ν-SVM methods (GS, WGS, CD, and WCD) outperform bias-shifting (BS). While balanced bias-shifting (BBS) is also clearly superior to BS, in general it cannot compete with the 2ν-SVM methods either. In part this is to be expected, since the 2ν-SVM offers a more flexible set of classifiers. Yet a more significant problem was the difficulty in enforcing the false alarm constraint; both BS and BBS frequently result in classifiers for which P_F significantly exceeds α, resulting in poor average performance.

The more surprising result is that the full grid search (GS) performed worse than the heuristics (WGS, CD, and WCD). Windowing consistently improved GS, and in general WGS exhibited the best performance. Windowing did not seem to offer any benefit to CD, but CD was remarkably competitive with WGS. Moreover, CD is only 2 to 3 times slower than BS. Given the computational demand of (W)GS, CD seems to be a reliable compromise of computing time and performance.

This paper also contributed a theorem on the feasible parameter set of the 2ν-SVM. This theorem is important for our algorithms because it allows us to discretize the parameter space via a uniform grid in the unit square. This regular grid in turn underlies two heuristics that gave us improved performance: the coordinate descent search and the windowed error estimates.

The performance of classifiers was measured by $\mathcal{E}(f) = (1/\alpha)\max\{P_F(f) - \alpha, 0\} + P_M(f)$. Thus, accurate error estimation is crucial to an algorithm's success. If a learning rule results in an estimated $P_F \leq \alpha$, it stands a much better chance of being competitive with respect to this measure, since errors in excess of α are penalized heavily. In our algorithms we estimated errors using LOOCV, and selected parameters according to the rule $\min\{\widehat{P}_M(f) \,|\, \widehat{P}_F(f) \leq \alpha\}$. Yet in several cases the final test estimate of P_F was in excess of α. We did find some improvements in error estimation by low-pass filtering the error estimates to remove some of the high variance of LOOCV.

Future work on Neyman-Pearson classification should focus on improved methods of error estimation. One possibility is to replace LOOCV with an error estimate with a negative bias, such as the bootstrap zero estimator. Another is to train the classifier as if α is really some number $\alpha' < \alpha$. Yet this quickly enters the realm of ad hoc fixes, and it may take some care to develop rules that perform well across a variety of datasets.

7. REFERENCES

[1] C. Scott and R. Nowak, "A Neyman-Pearson approach to statistical learning," *IEEE Trans. on Information Theory*, November, 2005.

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

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

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

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

[6] E. Osuna, R. Freund, and F. Girosi, "Support vector machines: Training and applications," Tech. Rep. A.I. Memo No. 1602, MIT Artificial Intelligence Laboratory, March 1997.

[7] H. G. Chew, R. E. Bogner, and C. C. Lim, "Dual-ν support vector machine with error rate and training size biasing," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Proc. (ICASSP)*, 2001, pp. 1269–1272.

[8] C. C. Chang and C. J. Lin, "Training ν-support vector classifiers: Theory and algorithms," *Neural Computation*, vol. 13, pp. 2119–2147, 2001.

[9] M. A. Davenport, "The 2ν-SVM: A cost-sensitive extension of the ν-SVM," Tech. Rep. TREE 0504, Rice University, Dept. of Elec. and Comp. Engineering, October, 2005, Available at http://www.ece.rice.edu/~md.

[10] C. Scott, "Performance measures for Neyman-Pearson classification," Tech. Rep., Rice University, Dept. of Statistics, 2005, Available at http://www.stat.rice.edu/~cscott.