Multiclass Posterior Probability Support Vector Machines

Mehmet Gönen, Ayşe Gönül Tanuğur, and Ethem Alpaydın, *Senior Member, IEEE*

***Abstract*—Tao *et al.* have recently proposed the posterior probability support vector machine (PPSVM) which uses soft labels derived from estimated posterior probabilities to be more robust to noise and outliers. Tao *et al.*'s model uses a window-based density estimator to calculate the posterior probabilities and is a binary classifier. We propose a neighbor-based density estimator and also extend the model to the multiclass case. Our bias–variance analysis shows that the decrease in error by PPSVM is due to a decrease in bias. On 20 benchmark data sets, we observe that PPSVM obtains accuracy results that are higher or comparable to those of canonical SVM using significantly fewer support vectors.**

***Index Terms*—Density estimation, kernel machines, multiclass classification, support vector machines (SVMs).**

I. Introduction

SUPPORT VECTOR MACHINE (SVM) is the optimal margin linear discriminant trained from a sample of l independent and identically distributed instances

$$(\boldsymbol{x}_1, z_1), (\boldsymbol{x}_2, z_2), \ldots, (\boldsymbol{x}_l, z_l) \tag{1}$$

where $\boldsymbol{x}_i$ is the d-dimensional input and $z_i \in \{-1, 1\}$; its label in a two-class problem is $z_i = +1$ if x_i is a positive $(+)$ example, and $z_i = -1$ if x_i is a negative example. The basic idea behind SVM is to solve the following model:

$$\min \quad \frac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}) + C \sum_{i=1}^{l} \xi_i \tag{2}$$

$$\text{s.t.} \quad z_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0 \tag{3}$$

which is a C-soft margin algorithm where $\boldsymbol{w}$ and b are the weight coefficients and bias term of the separating hyperplane, C is a predefined positive real number and ξ_i are slack variables [1]. The first term of the objective function given in (2) ensures the regularization by minimizing the l_2 norm of the weight coefficients. The second term tries to minimize the classification errors by using slack variables: A nonzero slack variable means that the classifier introduced some error on the corresponding instance. The constraint given in (3) is the separation inequality which tries to put each instance on the correct side of the separating hyperplane.

Once $\boldsymbol{w}$ and b are optimized, during test, the discriminant is used to estimate the labels

$$\hat{z} = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b) \tag{4}$$

and we choose the positive class if $\hat{z} = +1$, and choose the negative class if $\hat{z} = -1$. This model is generalized to learn nonlinear discriminants by using kernel functions, which correspond to defining nonlinear basis functions to map $\boldsymbol{x}$ to a new space and learning a linear discriminant there.

Tao *et al.* [2] in their proposed posterior probability SVM (PPSVM) modify the canonical SVM discussed previously to utilize class probabilities instead of using hard $-1/+1$ labels. These "soft labels" are calculated from estimated posterior probabilities as

$$y_i = 2\hat{P}(+|\boldsymbol{x}_i) - 1 \tag{5}$$

and Tao *et al.* rewrite (3) as

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq y_i^2 - y_i^2 \xi_i. \tag{6}$$

Note that (3) can also be rewritten as (6) with hard z_i in place of soft y_i. In other words, z_i are equal to y_i when the posterior probability estimates of (5) are 0 or 1.

The advantage of using soft labels derived from posterior probabilites y_i instead of hard class labels z_i in (6) is twofold. Because the posterior probability at a point is the combined effect of a number of neighboring instances, first, it gives a chance to correct the error introduced by wrongly labeled/noisy points due to correctly labeled neighbors; this can be seen as a smoothing of labels and, therefore, of the induced boundary. Second, an instance which is surrounded by a number of instances of the same class becomes redundant and this decreases the number of stored supports.

This paper is organized as follows: The different approaches for multiclass SVM are discussed in Section II. Section III contains our proposed mathematical model for multiclass posterior probability SVM. Experiments and results obtained are summarized in Section IV and Section V concludes this paper.

Manuscript received November 17, 2006; revised March 26, 2007; accepted May 1, 2007. This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, the Boğaziçi University Scientific Research Project 05HA101, and the Turkish Scientific Technical Research Council (TÜBİTAK) under Grant EEEAG 104E079. The work of M. Gönen was supported by the Ph.D. scholarship (2211) from TÜBİTAK. The work of A. G. Tanuğur was supported by the M.Sc. scholarship (2210) from TÜBİTAK.

M. Gönen and E. Alpaydın are with the Department of Computer Engineering, Boğaziçi University, 34342 Istanbul, Turkey (e-mail: gonen@boun.edu.tr).

A. G. Tanuğur is with the Department of Industrial Engineering, Boğaziçi University, 34342 Istanbul, Turkey.

Digital Object Identifier 10.1109/TNN.2007.903157

II. Multiclass SVMs

In a multiclass problem, we have a sample as given in (1) where an instance $\boldsymbol{x}_i$ can belong to one of $k > 2$ classes and the class label is given as $z_i \in \{1, \ldots, k\}$.

There are two basic approaches in the literature to solve multiclass pattern recognition problems. In the *multimachine approach*, the original multiclass problem is converted to a number of independent, uncoupled two-class problems. In the *single-machine approach*, the constraints due to having multiple classes are coupled in a single formulation.

A. Multimachine Approaches

In *one-versus-all*, k distinct binary classifiers are trained to separate one class from all others [3], [4]. During test, the class label which is obtained from the binary classifier with the maximum output value is assigned to a test instance. Each binary classifier uses all training samples, and for each class, we have an l-variable quadratic programming problem to be solved. There are two basic concerns about this approach. First, binary classifiers are trained on many more negative examples than positive ones. Second, real valued outputs of binary classifiers may be in different scales and direct comparison between them is not applicable [5].

Another approach is the *all-versus-all* or *pairwise* decomposition [6], [7], where there are $k(k-1)/2$ binary classifiers for each possible pair of classes. Classifier count is generally much larger than *one-versus-all* but when separating class i from j instances of all classes except i and j are ignored and hence the quadratic programs in each classifier are much smaller, making it possible to train the system very fast. This approach has the disadvantage of potential variance increase due to the small training set size for each classifier [8]. The test procedure should utilize a voting scheme to decide which class a test point belongs to, and a modified testing procedure to speed up by using directed acyclic graph traversals instead of evaluating all $k(k-1)/2$ classifiers has also been proposed [9]. In [10], a binary tree of SVMs is constructed in order to decrease the number of binary classifiers needed, where the idea is to use the same pairwise classifier for more than a single pair. Total training time can be greatly reduced in problems with large number of classes.

Both one-versus-all and pairwise methods are special cases of the error-correcting output codes (ECOC) [11] which decompose a multiclass problem to a set of two-class problems and ECOC has also been used with SVM [12]. The main issue in this approach is to construct a good ECOC matrix.

B. Single-Machine Approaches

A more natural way than using the *multimachine approach* is to construct a decision function by considering all classes at once, as proposed by Vapnik [1], Weston and Watkins [13], and Bredensteiner and Bennett [14]

$$\min \quad \frac{1}{2}\sum_{m=1}^{k}(\boldsymbol{w}_m \cdot \boldsymbol{w}_m) + C\sum_{i=1}^{l}\sum_{m=1}^{k}\xi_i^m \tag{7}$$

$$\text{s.t.} \quad (\boldsymbol{w}_{z_i} \cdot \boldsymbol{x}_i) + b_{z_i} \geq (\boldsymbol{w}_m \cdot \boldsymbol{x}_i) + b_m + 2 - \xi_i^m, \qquad \xi_i^m \geq 0 \tag{8}$$

where z_i contains the index of the class $\boldsymbol{x}_i$ belongs to and $\boldsymbol{w}_m$ and b_m are the weight coefficients and bias term of separating hyperplane for class m. This gives the decision function

$$f(\boldsymbol{x}) = \arg\max_n \left[(\boldsymbol{w}_n \cdot \boldsymbol{x}) + b_n\right]. \tag{9}$$

The objective function of (7) is also composed of the two regularization and classification error terms. The regularization term tries to minimize the l_2 norm of all separating hyperplanes simultaneously. The classification errors for each class are treated equally and their sum is added to the objective function. There are also modifications to this approach by using different C values for errors of different classes according to some loss criteria or prior probabilities. The constraint of (8) aims to place each instance on the negative side of the separating hyperplane for all classes except the one it belongs to.

The solution to this optimization problem can be found by finding the saddle point of the Lagrangian dual. After writing the Lagrangian dual, differentiating with respect to the decision variables, we get

$$\begin{aligned}
c_i^n &= \begin{cases} 1, & \text{if } z_i = n \\ 0, & \text{if } z_i \neq n \end{cases} \\
A_i &= \sum_{m=1}^{k}\alpha_i^m \\
\max \quad & 2\sum_{i=1}^{l}\sum_{m=1}^{k}\alpha_i^m - \sum_{i=1}^{l}\sum_{j=1}^{l}\frac{1}{2}c_j^{z_i}A_iA_j(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \\
& \quad + \sum_{i=1}^{l}\sum_{j=1}^{l}\sum_{m=1}^{k}\left(\alpha_i^m\alpha_j^{z_i} - \frac{1}{2}\alpha_i^m\alpha_j^m\right)(\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \\
\text{s.t.} \quad & \sum_{i=1}^{l}\alpha_i^n = \sum_{i=1}^{l}c_i^nA_i, \qquad 0 \leq \alpha_i^m \leq C \\
& \alpha_i^{z_i} = 0.
\end{aligned} \tag{10}$$

This gives the decision function

$$f(\boldsymbol{x}) = \arg\max_n \left[\sum_{i=1}^{l}(c_i^nA_i - \alpha_i^n)(\boldsymbol{x}_i \cdot \boldsymbol{x}) + b_n\right]. \tag{11}$$

The main disadvantage of this approach is the enormous size of the resulting quadratic programs. For example, *one-versus-all* method solves k separate l-variable quadratic problems, but the formulation of (10) has $l \times k$ variables.

In order to tackle such large quadratic problems, different decomposition methods and optimization algorithms are proposed for both two-class and multiclass SVMs. Sequential minimal optimization (SMO) is the most widely used decomposition algorithm for two-class SVMs [15]. Asymptotic convergence proof of SMO and other SMO-type decomposition variants are discussed in [16]. For multiclass SVMs, Crammer and Singer developed a mathematical model that reduces the variable size from $l \times k$ to l and proposed an efficient algorithm to solve the *single-machine* formulation [17].

Another possibility to decrease the training complexity is to preselect a subset of training data as support vectors and solve a smaller optimization problem; this is called a reduced SVM (RSVM) [18]. Statistical analysis of RSVMs can be found in

[19]. The same strategy can also be applied for multiclass problems in *single-machine* formulation to decrease the size of optimization problem.

III. Proposed Multiclass PPSVM Model

The multiclass extension to SVM [1], [13], [14] can also be modified to include posterior probability estimates instead of hard labels. The constraint of (8) tries to place each instance on the correct side of each hyperplane with at least two-units distance. In the canonical formulation, this comes from the fact that the true class label is $+1$ and the wrong class label is -1.

In the PPSVM formulation, the label of an instance for class m is defined as $2\hat{P}(m|\boldsymbol{x}_i) - 1$, and the required difference between instances on the two sides of the hyperplane becomes $[2\hat{P}(z_i|\boldsymbol{x}_i) - 1] - [2\hat{P}(m|\boldsymbol{x}_i) - 1]$. So, the constraint of (8) in the primal formulation is replaced by the following constraint:

$$(\boldsymbol{w}_{z_i} \cdot \boldsymbol{x}_i) + b_{z_i} \geq (\boldsymbol{w}_m \cdot \boldsymbol{x}_i) + b_m + 2\hat{P}(z_i|\boldsymbol{x}_i) - 2\hat{P}(m|\boldsymbol{x}_i) - \xi_i^m. \quad (12)$$

The objective function of (10) in the dual formulation becomes

$$\begin{aligned} \max \quad & 2\sum_{i=1}^{l}\sum_{m=1}^{k} \alpha_i^m \left[\hat{P}(z_i|\boldsymbol{x}_i) - \hat{P}(m|\boldsymbol{x}_i)\right] \\ & - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} c_j^{z_i} A_i A_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \\ & + \sum_{i=1}^{l}\sum_{j=1}^{l}\sum_{m=1}^{k} \left(\alpha_i^m \alpha_j^{z_i} - \frac{1}{2}\alpha_i^m \alpha_j^m\right)(\boldsymbol{x}_i \cdot \boldsymbol{x}_j). \quad (13) \end{aligned}$$

Classical kernel trick can be applied by replacing $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$ with $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in (10) and (13).

We show the difference of canonical and posterior probability SVMs on a toy data set in Fig. 1. We see that the canonical SVM stores the outliers as support vectors and this shifts the induced boundary. With PPSVM, because the neighbors of the outliers belong to a different class, the posterior probabilities are small and the outliers are effectively cancelled resulting in a reduction in bias (as will be discussed in Section IV-C); they are not chosen as support vectors and, therefore, they do not affect the boundary. Though this causes some training error, the generalization is improved, and the number of support vectors decreases from six to four.

Weston and Watkins [13] showed that a *single-machine* multiclass SVM optimization problem reduces to a classical two-class SVM optimization problem for binary data sets. Using a similar analogy, if we have two classes, (12) becomes

$$\begin{aligned} (\boldsymbol{w}_+ - \boldsymbol{w}_-) \cdot \boldsymbol{x}_i + b_+ - b_- &\geq 2y_i - \xi_i^+, \quad \text{if} \quad z_i = +1 \\ (\boldsymbol{w}_+ - \boldsymbol{w}_-) \cdot \boldsymbol{x}_i + b_+ - b_- &\leq 2y_i + \xi_i^-, \quad \text{if} \quad z_i = -1 \end{aligned}$$

and the resulting optimization problem is equivalent to Tao *et al.*'s [2] formulation in terms of the obtained solution.

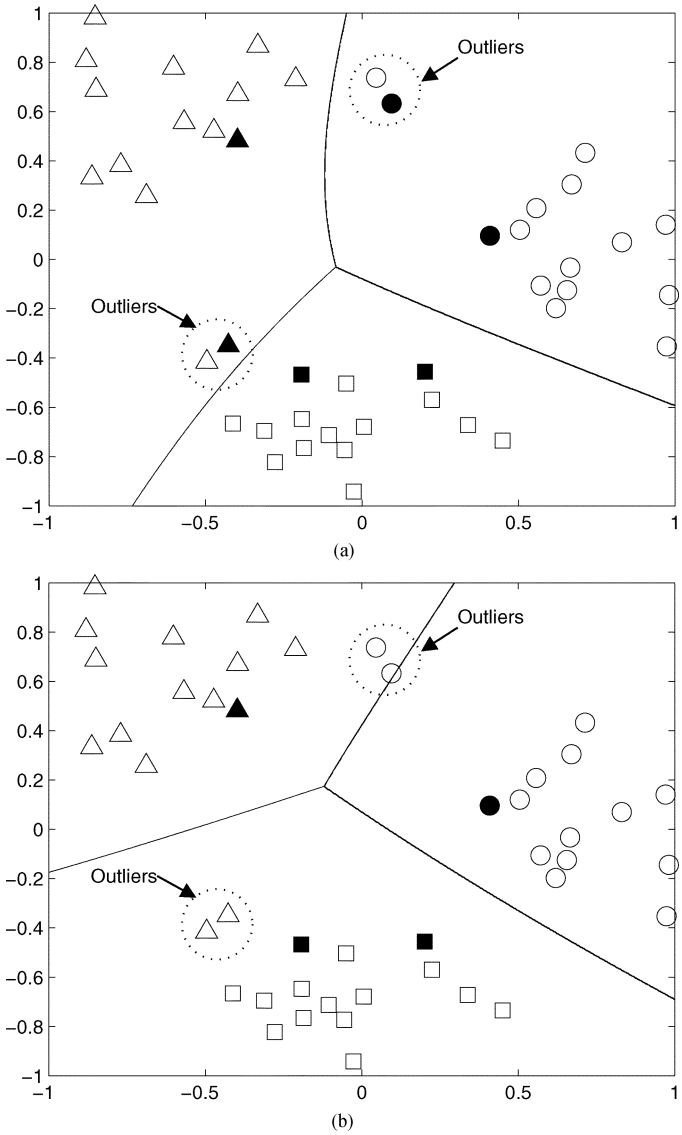


Fig. 1. Separating hyperplanes (solid lines) and support vectors (filled points) on a toy data set. Canonical SVM stores the outliers as support vectors which shift the class boundary. PPSVM ignores the outliers which reduces bias and leads to better generalization and fewer support vectors. (a) Canonical SVM. (b) PPSVM.

IV. Experiments and Results

A. Estimating the Posteriors

To be able to solve (13), we need to be able to estimate $\hat{P}(j|\boldsymbol{x}_i), j = 1, \ldots, k$. We can use any density estimator for this purpose. We will report results with two methods, *windows method* and k*-nearest neighbor method* (k-NN). The advantage of using such nonparametric methods, as opposed to a parametric approach of, for example, assuming Gaussian $p(\boldsymbol{x}|j)$, is that they make less assumptions about the data, and hence, their applicability is wider.

1) Windows Method: Tao *et al.*'s method [2], when generalized to multiple classes, estimates the posterior probability as

$$\hat{P}(j|\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i : \|\boldsymbol{x} - \boldsymbol{x}_i\| \leq r \text{ AND } z_i = j\}}{\#\{\boldsymbol{x}_i : \|\boldsymbol{x} - \boldsymbol{x}_i\| \leq r\}}. \quad (14)$$

That is, given input $\boldsymbol{x}_i$, we find all training instances that are at most r away and count the proportion belonging to class j among them. r defines the size of the neighborhood and as such is the smoothing parameter. We call this the *windows method* and use a simple heuristic to determine r: On the training set, we calculate the average of the distance from each instance to its nearest neighbor and name this r_0. We use $r \in \{r_0, 2r_0, 4r_0\}$ in the experiments.

2) The k-Nearest Neighbor Method: The correct value of r is data dependent and may be difficult to fine tune it on a new data set. We also use the k-NN estimator which is similar except that instead of fixing a window width r and checking how many instances fall in there, we fix the number of neighbors as k. If among these k, k_j of them belong to class j, the posterior probability estimate is

$$\hat{P}(j|\boldsymbol{x}) = \frac{k_j}{k}. \tag{15}$$

We use $k \in \{3, 5, 7, 9, 11\}$ in the experiments.

B. Kernels

The following three different kernel functions are used in this paper:

1) linear kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y})$;
2) polynomial kernel of degree b: $K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^b$ where $b \in \{2, 3, 4, 5\}$;
3) radial basis function (RBF) kernel with width s: $K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2 / s^2)$ where $s \in \{r_0, 2r_0, 4r_0, 8r_0, 16r_0\}$, with r_0 calculated as in the windows method.

C. Synthetic Data Sets

We use four synthetic data sets to illustrate the differences between canonical and posterior probability SVMs (Table I). Total of 400 instances are generated for each case where 200 instances are reserved for testing and the remaining part is divided into two for training and validation. We use the validation set to optimize C and the kernel and density estimator parameters. We compare canonical and posterior probability SVMs in terms of their accuracy and the number of support vectors stored; note that the latter determines both the space complexity (number of parameters stored) and the time complexity (number of kernel calculations).

Table II shows the percent changes in accuracy and support vector count of posterior probability SVM compared to the canonical SVM on these four problems, using the two density estimators and kernel types. We see that with the nonparametric density estimators, windows, and k-NN, PPSVMs achieve greater accuracy than canonical SVM with fewer support vectors.

In Fig. 2, we see how the boundary and the support vectors change with the density estimator used by PPSVM compared with the canonical SVM. PPSVM variants induce good class boundaries storing much fewer support vectors. A small number of instances is sufficient to generate correct posterior probabilities in a large neighborhood effectively covering for a large number of instances. In Fig. 3, we see how the boundary and the support vectors change as k of k-NN is increased; the canonical

TABLE I
SYNTHETIC PROBLEMS USED IN THE EXPERIMENTS

Dataset	Prior Probabilities	Mean Vectors	Covariance Matrices
R1	$P(z_1) = 0.7$	$\mu_1 = (0,0)^T$	$\Sigma_1 = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$
	$P(z_2) = 0.3$	$\mu_2 = (2,2)^T$	$\Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$
R2	$P(z_1) = 0.7$	$\mu_1 = (0,0)^T$	$\Sigma_1 = \begin{pmatrix} 0.25 & 0 \\ 0 & 1 \end{pmatrix}$
	$P(z_2) = 0.3$	$\mu_2 = (2,0)^T$	$\Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 0.25 \end{pmatrix}$
R3	$P(z_1) = 0.5$	$\mu_1 = (0,0)^T$	$\Sigma_1 = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$
	$P(z_2) = 0.3$	$\mu_2 = (2,0)^T$	$\Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$
	$P(z_3) = 0.2$	$\mu_3 = (2,2)^T$	$\Sigma_3 = \begin{pmatrix} 2 & 0 \\ 0 & 4 \end{pmatrix}$
R4	$P(z_1) = 0.5$	$\mu_1 = (0,0)^T$	$\Sigma_1 = \begin{pmatrix} 0.25 & 0 \\ 0 & 1 \end{pmatrix}$
	$P(z_2) = 0.3$	$\mu_2 = (2,0)^T$	$\Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 0.25 \end{pmatrix}$
	$P(z_3) = 0.2$	$\mu_3 = (2,2)^T$	$\Sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

TABLE II
PERCENT CHANGES IN ACCURACY AND CORRESPONDING SUPPORT VECTOR COUNT OF PPSVM COMPARED TO CANONICAL SVM

		k-NN		Windows	
	Kernel	Acc.	SV	Acc.	SV
R1	Linear	−2.0	−86.0	−1.0	−10.5
	Poly	+3.5	−23.0	+6.0	−14.5
	RBF	+0.5	+6.5	+3.0	+3.0
R2	Linear	+2.5	0.0	0.0	−65.5
	Poly	+7.0	−22.0	+6.0	−19.5
	RBF	+1.0	−28.5	+1.0	−29.5
R3	Linear	+2.0	−38.0	+5.0	−18.5
	Poly	+5.0	−39.0	+1.5	+30.0
	RBF	+12.0	−67.5	+12.0	−72.5
R4	Linear	+10.0	−42.5	+7.5	−68.0
	Poly	+9.0	−33.0	+9.5	−31.0
	RBF	+2.0	−29.0	+2.0	−11.5

SVM corresponds to k-NN with $k = 1$. We see that as k increases, instances cover larger neighborhoods in the input space and fewer support vectors suffice.

To see the reason of decrease in error with PPSVM, we do a bias–variance analysis. We do simulations to see the effect of using soft labels derived from posterior probabilities on the bias–variance decomposition of error. On R4 of Table I, we create 100 training sets and 50 test sets, composed of 200 and 1000 instances, respectively. Each training set is first divided into two as training proper and validation sets and the best C is found on the validation set after training on the training set proper for different C values (2). After finding the best C, the whole training set is used for training the SVM and evaluation is done on the test set.

To convert SVM outputs to posterior probabilities, we use the softmax function (for trained models $t = 1, \ldots, 100$)

$$\hat{P}^{(t)}(j|\boldsymbol{x}) = \frac{\exp\left[\sum_{i=1}^{l} (c_i^j A_i^{(t)} - \alpha_i^{j(t)})(\boldsymbol{x}_i \cdot \boldsymbol{x}) + b_j^{(t)}\right]}{\sum_{m=1}^{k} \exp\left[\sum_{i=1}^{l} (c_i^m A_i^{(t)} - \alpha_i^{m(t)})(\boldsymbol{x}_i \cdot \boldsymbol{x}) + b_m^{(t)}\right]}$$

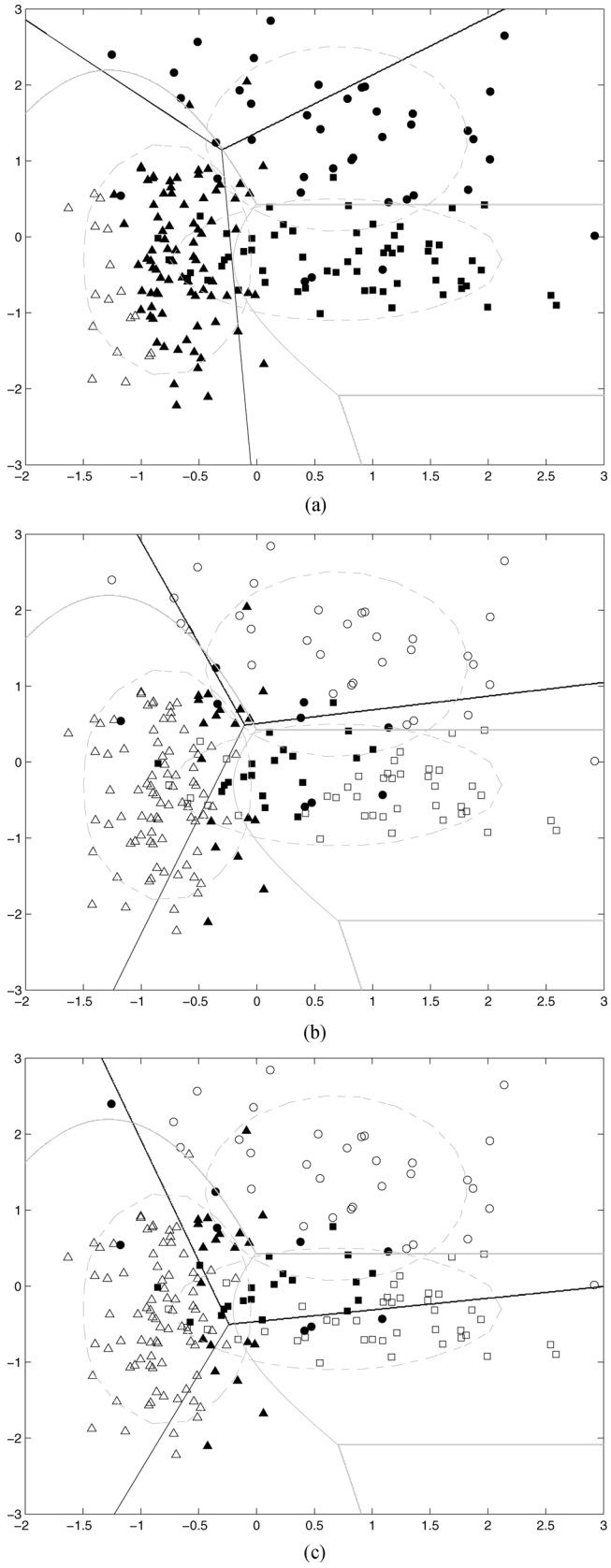


Fig. 2. Separating hyperplanes (solid lines) and support vectors (filled points) on R4 data set with linear kernel. Gray dashed lines show the Gaussians from which the data are sampled and gray solid lines show the optimal Bayes' discriminant. We see that the PPSVM variants use much fewer support vectors while inducing the correct boundaries. (a) Canonical SVM. (b) PPSVM with windows method ($r = r_0$). (c) PPSVM with k-NN ($k = 3$).

which we average to get the expected value

$$\hat{P}(j|\boldsymbol{x}) = \frac{1}{100}\sum_{t=1}^{100}\hat{P}^{(t)}(j|\boldsymbol{x}).$$

We use the bias–variance–noise decomposition of error given by Breiman [20]

$$\text{Bias} = \frac{1}{n}\sum_{i=1}^{n}\left[P(z_i^*|\boldsymbol{x}_i) - P(\widehat{z}_i^*|\boldsymbol{x}_i)\right]\widehat{P}(\widehat{z}_i^*|\boldsymbol{x}_i)$$

$$\text{Variance} = \frac{1}{n}\sum_{i=1}^{n}\sum_{m\neq\widehat{z}_i^*}^{k}\left[P(z_i^*|\boldsymbol{x}_i) - P(m|\boldsymbol{x}_i)\right]\widehat{P}(m|\boldsymbol{x}_i)$$

$$\text{Noise} = \frac{1}{n}\sum_{i=1}^{n}\left[1 - P(z_i^*|\boldsymbol{x}_i)\right] \tag{16}$$

where n is the number of test instances, z_i^* denotes the correct class label ($z_i^* = \arg\max_j P(j|\boldsymbol{x}_i)$), $\widehat{P}(j|\boldsymbol{x}_i)$ denotes the mean value of the estimated probabilities obtained from 100 SVMs trained on training sets, and $\widehat{z}_i^*$ denotes the estimated class, i.e., $\widehat{z}_i^* = \arg\max_j \widehat{P}(j|\boldsymbol{x}_i)$. We repeat (16) 50 times on the 50 test sets and look at average values to get smoother estimates.

We see in Fig. 4 that as we go from the canonical SVM to PPSVM, it is the bias that decreases while variance does not change. With PPSVM, increasing k of the k-NN density estimator further decreases the bias. This is true also for increasing r with the windows method. Even with small k or r, the nonparametric estimators return a smooth estimate that further smoothing (by increasing k or r) does not decrease variance; using more instances makes the average estimate closer to the real discriminant and, therefore, reduces bias.

D. Real Data Sets

We perform experiments on several two-class and multiclass benchmark data sets from the University of California at Irvine (UCI) Repository [21] and Statlog Collection [22] (Table III).

Given a data set, a random one-third is reserved as the test set and then the remaining two-thirds is resampled using 5×2 cross validation to generate ten training and validation sets, with stratification. These ten normalized folds are processed to obtain posterior probability labels by the two density estimators and used by PPSVM, whereas the canonical SVM uses the hard labels. To solve the quadratic optimization problems, we use CPLEX 9.1 C Callable Library. In each fold, the validation set is used to optimize C (by trying all values between 10^{-6} and 10^{+6} in log scale), kernel type and parameters (linear, polynomials of degree 2, 3, 4, 5, and RBF kernel with width r_0, $2r_0$, $4r_0$, $8r_0$, and $16r_0$), and for PPSVM, the parameters of the density estimators (k-NN with $k = 3, 5, 7, 9, 11$, windows with $r = r_0$, $2r_0$, and $4r_0$). The best configuration (the one that has the highest average accuracy on the ten validation folds) is used to train the final SVM on the whole two-thirds and its performance (accuracy and support vector percentage) is measured over the test set (the remaining one-third). So, for each data set, we have ten validation sets and one test set results.

As examples, Figs. 5 and 6 show the effect of density estimators on accuracy and support vector percentages for validation

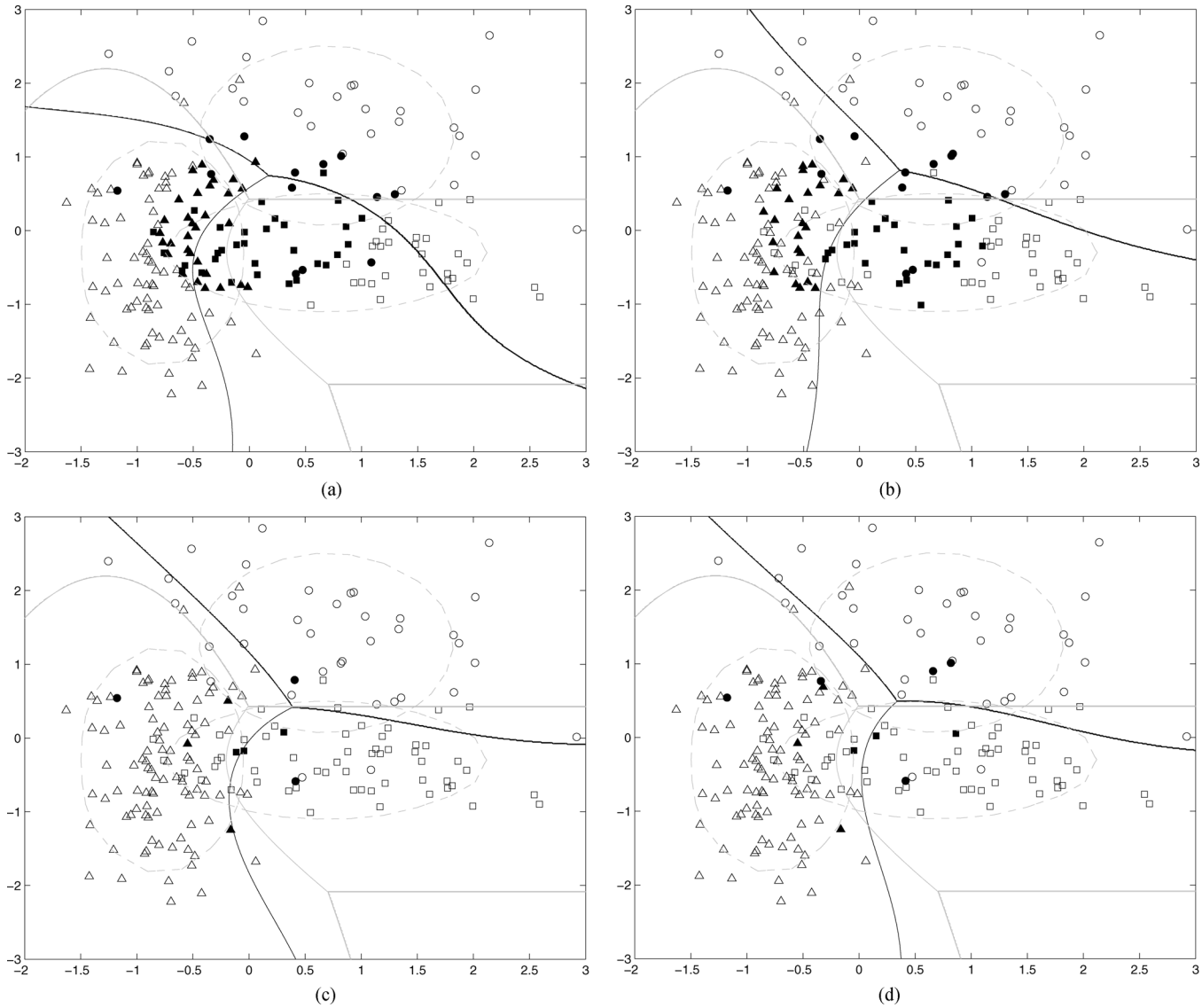


Fig. 3. Separating hyperplanes (solid lines) and support vectors (filled points) on R4 data set with polynomial kernel ($b = 3$). We see that as the neighborhood gets larger, instances cover larger regions, and more instances become redundant, effectively decreasing the number of support vectors while still having a good approximation of the boundary. (a) Canonical SVM. (b) PPSVM with k-NN ($k = 3$). (c) PPSVM with k-NN ($k = 7$). (d) PPSVM with k-NN ($k = 11$).

and test sets of *spambase* and *glass*. On *spambase*, which is a two-class problem, we see that PPSVM uses fewer support vectors and achieves higher accuracy both on validation and test sets. The improvement in both complexity and accuracy increases as the neighborhood (k of k-NN or r of windows) increases. We see the same type of behavior on *glass* as well, which has six classes. We see in the latter data set that with windows density estimator with very large width, the accuracy drops (and number of support vectors go up) indicating that it is important to fine tune the density estimator for PPSVM to work well.

On all 20 data sets, the comparison of results by canonical and posterior probability SVM are given in Tables IV–VI. Table IV shows our results on two-class data sets and Tables V and VI show the results on multiclass data sets for the *single-machine* PPSVM and the *multimachine* PPSVM utilizing the *one-versus-all* approach, respectively. In all three tables, for the canonical SVM, we report the kernel type and parameter that has the highest average accuracy on the validation folds, its average accuracy and support vector percentage on the validation set, and the accuracy and support vector count of that model over the test set when it is trained over the whole two-thirds. Similarly, for the PPSVM, we check for the best kernel type, parameter, and the density estimator over the validation set and report similarly its performance on validation and test sets. Below the PPSVM results, we also report the count of wins–ties–losses of PPSVM over canonical SVM using different tests. To compare accuracies on the validation folds, we use the 5×2 cross-validated (cv) paired F test [23]; to compare accuracies over the test sets (there is a single value for each), we use McNemar's test. For the support vector percentages, there is no test that can be used and we just compare two values. Direct comparison compares averages without checking for significance. The number of wins, ties, and losses over ten at the bottom of the table is made bold if the number of wins out of ten is significant using sign test.

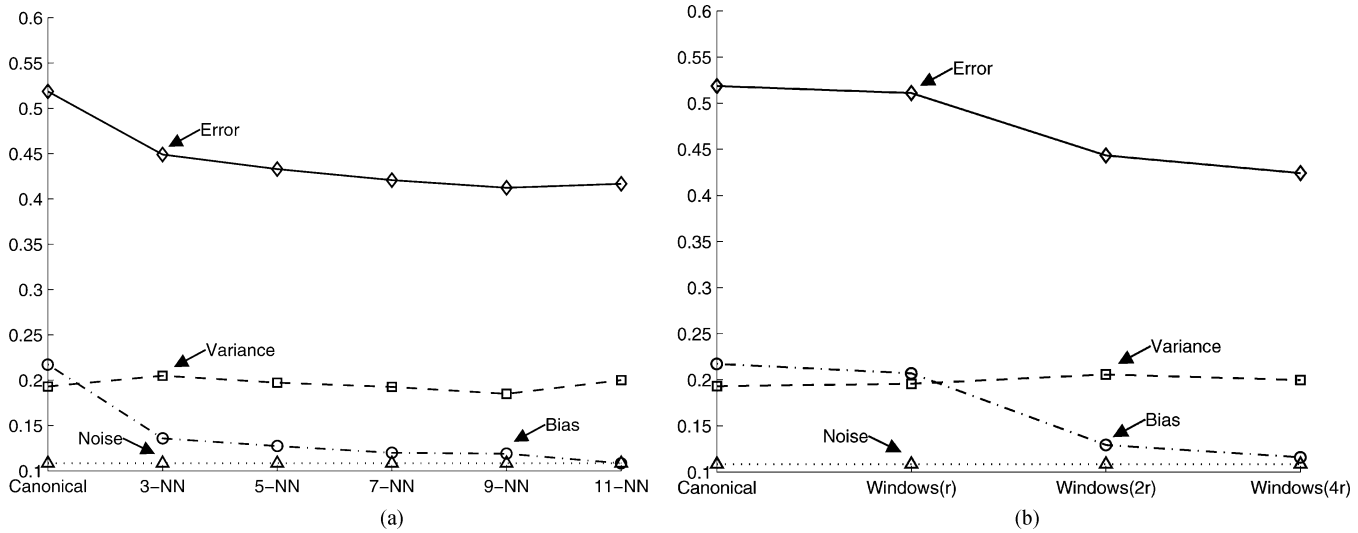


Fig. 4. Effect of the density estimation method over the decomposition of error on R4 data set with polynomial kernel ($b = 3$). PPSVM decreases the bias and hence error while variance stays unchanged. With PPSVM, increasing the neighborhood (by increasing k of k-NN or r of windows) further decreases the bias. (a) The$k$-NN method. (b) Windows method.

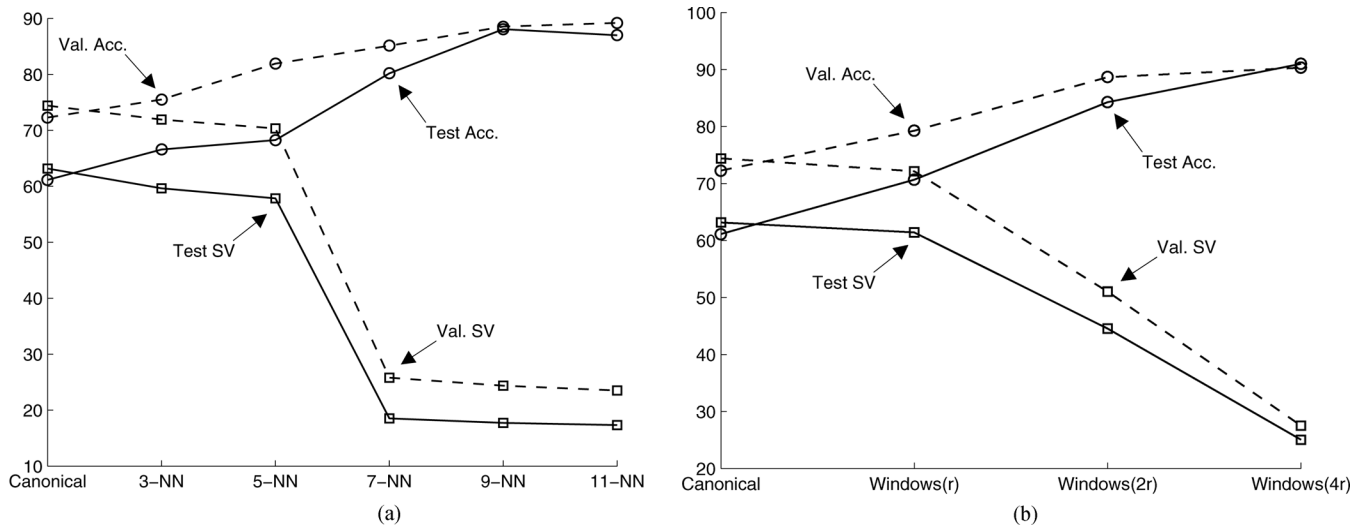


Fig. 5. Effect of the density estimation method over accuracy and support vector percentages for validation and test sets on *spambase* with linear kernel. (a) The$k$-NN method. (b) Windows method.

Wilcoxon's signed rank test is also used as a nonparametric test to compare algorithms in terms of their accuracy and support vector percentages over validation and test sets, and its result is shown as follows: W—win, T—tie, or L—loss.

PPSVM can be thought of as a postprocessor after the density estimator and, to check for what SVM adds, we compare its accuracy with the accuracy of the density estimator directly used as a classifier.[1] The comparison values are reported in the last two columns and the counts are given in the following; again, a win indicates a win for PPSVM over the density estimator, and is made bold if significant.

On the two-class data sets, we see in Table IV that PPSVM obtains accuracy and support vector results comparable to those of canonical SVM. The 5×2 cv paired F test finds only two wins and eight ties and, on the test set, McNemar's test finds five wins and seven ties. In terms of averages, PPSVM has higher average accuracy on validation folds which is significant using sign test and also using Wilcoxon's signed rank test; the differences are not significant over test sets. In terms of the support vectors stored, PPSVM seems to store fewer support vectors than the canonical SVM (five wins versus two losses on validation folds and seven wins versus two losses on the test) but the difference is not statistically significant. Note in the last two columns that PPSVM achieves significantly higher accuracy than the density estimator used as classifier on both validation and test sets.

On the multiclass data sets using the *single-machine* approach, as we see in Table V, PPSVM does not win significantly in terms of accuracy over the canonical SVM but wins significantly in terms of support vector percentages. On many data sets, *carevaluation, contraceptive, iris, waveform*, and *wine*, support vector percentage is decreased to half or one-third of what is stored by the canonical SVM. Table VI gives the results for PPSVM utilizing the *multimachine* approach. Similar to the

[1]We would like to thank an anonymous reviewer for suggesting this comparison.

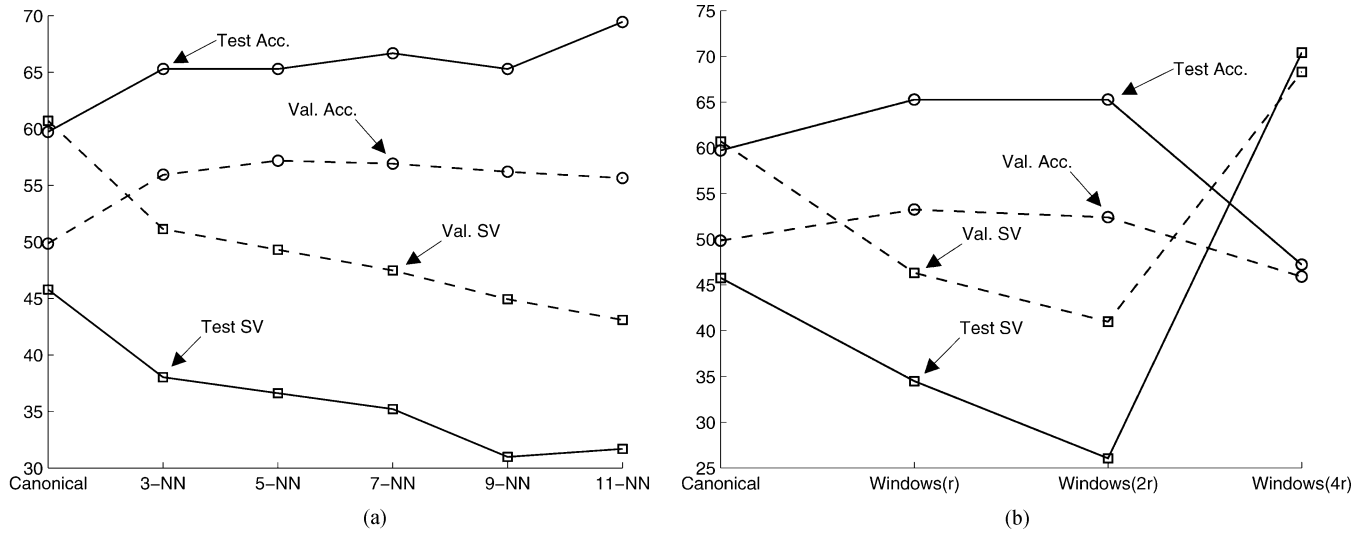


Fig. 6. Effect of the density estimation method over accuracy and support vector percentages for validation and test sets on *glass* with polynomial kernel ($b = 4$). (a) The k-NN method. (b) Windows method.

TABLE III
BENCHMARK DATA SETS USED IN THE EXPERIMENTS

Name	No. of Instances	No. of Classes	No. of Attributes
banana	5300	2	2
germannumeric	1000	2	24
heart	270	2	13
ionosphere	351	2	34
liverdisorder	345	2	6
pima	768	2	8
sonar	208	2	60
spambase	4601	2	57
twonorm	7400	2	20
wdbc	570	2	30
balancescale	625	3	4
carevaluation	1728	4	21
contraceptive	1473	3	24
dna	3190	3	240
glass	214	6	9
iris	150	3	4
vehicle	946	4	18
vowel	990	11	10
waveform	5000	3	21
wine	178	3	13

single-machine case, we see a decrease in the support vector percentages without sacrificing the accuracy. Both *single-machine* and *multimachine* approaches have significantly higher accuracy results on validation and test sets than the density method used as a classifier. Note that the density method used as a classifier (without the SVM that follows it) is not as accurate as the canonical SVM, indicating that it is the SVM part that is more important, and not the density estimation.

Looking at Tables V and VI , we see that *single-machine* and *multimachine* approaches choose similar kernels and density estimators for both canonical and posterior probability SVM. Canonical SVM chooses the same (family) kernel for both approaches on five (eight) data sets, and PPSVM chooses the same (family) kernel six (seven) times out of ten data sets. Wee also see that *single-machine* and *multimachine* PPSVM use the same (family) density estimator on four (six) data sets.

Table VII summarizes the comparison of performance results of *single-machine* and *multimachine* approaches for the *multiclass* case, where the wins are reported for the *multimachine* approach. There does not seem to be a significant difference in accuracy or support vector percentage between the two using any test. As the only difference, we notice that the *single-machine* approach uses fewer support vectors on validation sets according to Wilcoxon's signed rank test. If we compare running times, we see that solving k separate l-variable quadratic problems (*multimachine*) instead of solving one $l \times k$-variable quadratic problem (*single-machine*) significantly decreases the training time on validation and test sets for both canonical and posterior probability SVM. On the other hand, the *single-machine* approach has significantly less testing time than the *multimachine* approach for canonical SVM but the differences are not significant for PPSVM.

To summarize, we see that on both validation folds and test sets, PPSVM is as accurate as canonical SVM for both *two-class* and *multiclass* problems. Wilcoxon's signed rank test finds that PPSVM has higher accuracy on validation folds of *two-class* problems. PPSVM uses fewer support vectors on validation folds and test sets of multiclass data sets for both *single-machine* and *multimachine* approaches; this decrease is significant according to both 5×2 cv paired F test and Wilcoxon's signed rank test. The number of support vectors seems also to decrease on two-class problems though the difference is not statistically significant (at 0.95 confidence; it would have been significant using the Wilcoxon's signed rank test had the confidence been 0.85). We also see that PPSVM uses the k-NN estimator in many cases proving that it is an accurate density estimation method that can be used along with the windows-based estimator. We believe that more than the improvement in accuracy, it is the decrease in the percentage of stored support vectors that is the main advantage of the PPSVM. On many multiclass data sets, the percentage is decreased to half or one-third of what is stored by the canonical SVM.

V. CONCLUSION

This paper extends the posterior probability SVM idea to the multiclass case. The effect of outliers and noise in the data is

TABLE IV
COMPARISON BETWEEN CANONICAL SVM AND POSTERIOR PROBABILITY SVM ON TWO-CLASS DATA SETS

	Canonical SVM					Posterior Probability SVM					Density	
		Validation		Test		Density		Validation		Test		Val.	Test
Dataset	Kernel	Acc.	SV	Acc.	SV	Method	Kernel	Acc.	SV	Acc.	SV	Acc.	Acc.
banan.	R ($4r_0$)	89.62	86.23	90.20	82.60	k-NN(7)	R ($8r_0$)	89.88	14.99	91.17	34.35	88.79	90.49
germa.	L	72.13	99.67	70.06	97.60	Win. ($2r_0$)	L	73.36	36.40	78.44	34.08	69.67	71.26
heart	L	84.67	53.56	82.22	43.89	Win. ($2r_0$)	R ($16r_0$)	87.56	77.56	80.00	75.56	80.00	77.78
ionos.	R (r_0)	93.08	64.36	96.58	54.27	Win. (r_0)	R (r_0)	94.10	62.39	95.73	53.42	66.67	65.81
liver.	L	64.52	98.35	69.57	100.00	k-NN(7)	L	65.83	100.00	70.43	100.00	54.78	63.48
pima	L	74.88	99.61	70.31	96.48	k-NN(7)	L	77.30	38.59	72.66	34.57	71.48	73.44
sonar	P (3)	77.10	87.97	80.00	76.09	Win. (r_0)	P (3)	78.12	82.90	81.43	73.19	71.01	74.29
spamb.	P (2)	87.08	19.86	88.14	12.13	Win. ($4r_0$)	L	90.33	27.51	91.00	25.04	65.10	68.25
twono.	P (3)	97.96	57.95	97.12	57.79	k-NN(11)	P (3)	97.96	51.96	97.32	51.91	97.28	96.80
wdbc	L	96.26	22.00	95.24	16.05	k-NN(3)	L	97.00	12.84	95.24	7.11	94.21	97.35
					5×2 cv Paired F Test		W-T-L	2-8-0	5-3-2			5-5-0	
					McNemar's Test		W-T-L			3-7-0			3-7-0
					Direct Comparison		W-T-L	**10-0-0**	7-0-3	7-1-2	7-1-2	**10-0-0**	**8-0-2**
					Wilcoxon's Signed Rank Test		W/T/L	W	T	T	T	W	W

TABLE V
COMPARISON BETWEEN CANONICAL SVM AND POSTERIOR PROBABILITY SVM ON MULTICLASS DATA SETS FOR SINGLE-MACHINE CASE

	Canonical SVM					Posterior Probability SVM					Density	
		Validation		Test		Density		Validation		Test		Val.	Test
Dataset	Kernel	Acc.	SV	Acc.	SV	Method	Kernel	Acc.	SV	Acc.	SV	Acc.	Acc.
balan.	L	86.73	100.00	90.91	100.00	Win. (r_0)	L	85.91	98.56	86.12	100.00	56.25	60.29
carev.	P (4)	89.05	95.14	94.97	90.97	Win. (r_0)	P (3)	83.70	59.51	82.64	49.83	73.96	69.62
contr.	R ($2r_0$)	44.95	81.51	44.40	88.90	Win. ($4r_0$)	R ($2r_0$)	46.76	33.99	51.53	38.70	41.96	49.49
dna	L	93.72	93.50	96.05	84.02	Win. (r_0)	L	93.72	93.03	96.14	84.11	64.76	69.90
glass	R (r_0)	56.20	83.24	65.28	77.46	Win. (r_0)	R (r_0)	61.83	72.39	65.28	71.13	56.34	59.72
iris	R ($2r_0$)	96.60	62.00	92.00	64.00	Win. (r_0)	P (5)	96.60	20.60	92.00	11.00	72.00	72.00
vehic.	R (r_0)	68.48	93.09	70.21	90.78	k-NN(3)	R (r_0)	68.12	86.81	72.70	83.87	68.09	71.28
vowel	R (r_0)	86.88	95.85	84.85	96.06	Win. (r_0)	R (r_0)	86.88	95.70	84.85	96.06	54.85	63.33
wavef.	R (r_0)	85.60	41.93	85.36	41.66	k-NN(7)	P (3)	86.49	16.89	85.24	16.15	82.00	81.76
wine	R (r_0)	94.75	88.47	98.33	77.12	k-NN(11)	R ($2r_0$)	95.93	41.86	98.33	24.58	98.31	96.67
					5×2 cv Paired F Test		W-T-L	0-9-1	**6-4-0**			**7-3-0**	
					McNemar's Test		W-T-L			1-7-2			**6-4-0**
					Direct Comparison		W-T-L	5-2-3	**10-0-0**	3-4-3	**7-2-1**	**10-0-0**	**10-0-0**
					Wilcoxon's Signed Rank Test		W/T/L	T	W	T	W	W	W

TABLE VI
COMPARISON BETWEEN CANONICAL SVM AND POSTERIOR PROBABILITY SVM ON MULTICLASS DATA SETS FOR MULTIMACHINE CASE

	Canonical SVM					Posterior Probability SVM					Density	
		Validation		Test		Density		Validation		Test		Val.	Test
Dataset	Kernel	Acc.	SV	Acc.	SV	Method	Kernel	Acc.	SV	Acc.	SV	Acc.	Acc.
balan.	L	84.90	100.00	89.00	100.00	k-NN (5)	L	86.87	79.42	89.47	77.40	70.19	70.33
carev.	P (3)	85.94	89.65	89.06	87.07	Win. ($2r_0$)	L	81.16	25.16	82.29	11.20	69.97	70.14
contr.	R (r_0)	47.54	88.55	22.61	100.00	Win. ($4r_0$)	L	46.80	75.21	49.08	77.19	41.96	49.49
dna	L	89.83	96.11	86.55	98.64	k-NN (9)	L	90.43	54.97	90.59	43.72	77.82	73.19
glass	R (r_0)	61.69	83.94	66.67	78.87	Win. (r_0)	R (r_0)	62.11	79.01	62.50	78.87	56.34	59.72
iris	R ($2r_0$)	96.00	62.00	92.00	60.00	Win. (r_0)	P (3)	95.60	22.40	84.00	23.00	72.00	72.00
vehic.	L	72.48	99.61	73.40	99.65	Win. (r_0)	L	70.00	98.55	70.92	98.40	50.71	49.29
vowel	R (r_0)	86.12	100.00	93.64	95.30	Win. (r_0)	R (r_0)	86.12	100.00	93.64	95.30	54.85	63.33
wavef.	L	86.49	100.00	85.90	100.00	k-NN (9)	P (3)	86.71	23.49	85.06	37.74	82.48	82.54
wine	R ($2r_0$)	96.10	100.00	83.33	27.97	Win. (r_0)	R ($2r_0$)	96.10	100.00	83.33	27.97	91.53	90.00
					5×2 cv Paired F Test		W-T-L	1-8-1	**6-4-0**			**8-2-0**	
					McNemar's Test		W-T-L			1-7-2			6-3-1
					Direct Comparison		W-T-L	4-2-4	**8-2-0**	3-2-5	**7-3-0**	**10-0-0**	**8-0-2**
					Wilcoxon's Signed Rank Test		W/T/L	T	W	T	W	W	W

TABLE VII
COMPARISON BETWEEN SINGLE-MACHINE SVM AND MULTIMACHINE (ONE-VERSUS-ALL) SVM

		Canonical SVM								Posterior Probability SVM							
		Validation				Test				Validation				Test			
		Acc.	SV	Train Time	Test Time	Acc.	SV	Train Time	Test Time	Acc.	SV	Train Time	Test Time	Acc.	SV	Train Time	Test Time
5×2 cv Paired F Test	W-T-L	0-8-2	1-4-5	**9-1-0**	0-5-5					0-9-1	3-1-6	**10-0-0**	2-5-3				
McNemar's Test	W-T-L					1-5-4								2-6-2			
Direct Comparison	W-T-L	5-0-5	**1-2-7**	**10-0-0**	**0-1-9**	4-1-5	4-1-5	**9-0-1**	**1-2-7**	6-0-4	3-0-7	**10-0-0**	3-1-6	**2-0-8**	4-0-6	**9-0-1**	3-2-5
Wilcoxon's Signed Rank Test	W/T/L	T	L	W	L	T	T	T	L	T	T	W	T	T	T	W	T

diminished by considering soft labels as inputs to the SVM algorithm instead of hard $-1/+1$ labels and the calculated discriminants become more robust. Our bias–variance analysis shows that the effect of PPSVM is on decreasing the bias rather than variance. Experiments on 20 data sets, both two-class and multiclass, show that PPSVM achieves similar accuracy results storing fewer support vectors. The decrease in the support vector count decreases both the space complexity in that fewer data need to be stored and also the time complexity in that fewer kernel calculations are necessary in computing the discriminant.

REFERENCES

[1] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[2] Q. Tao, G. Wu, F. Wang, and J. Wang, "Posterior probability support vector machines for unbalanced data," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1561–1573, Nov. 2005.

[3] C. Hsu and C. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

[4] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, 2004.

[5] E. Mayoraz and E. Alpaydın, "Support vector machines for multi-class classification," in *Lecture Notes in Computer Science*, J. Mira and J. V. S. Andres, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1607, pp. 833–842.

[6] M. Schmidt and H. Gish, "Speaker identification via support vector classifiers," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 1996, pp. 105–108.

[7] U. Krefsel, "Pairwise classification and support vector machines," in *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1998.

[8] Y. Lee, Y. Lin, and G. Wahba, "Multicategory support vector machines," Dept. Statistics, Univ. Wisconsin, Madison, WI, Tech. Rep. 1043, 2001.

[9] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, , S. A. Solla, T. K. Leen, and K.-R. Müller, Eds., "Large margin DAGs for multiclass classification," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2000, vol. 12.

[10] B. Fei and J. Liu, "Binary tree of SVM: A new fast multiclass training and classification algorithm," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 696–704, May 2006.

[11] T. G. Dietterich and G. Bakiri, "Solving multi-class learning problems via error-correcting output codes," *J. Artif. Intell. Res.*, vol. 2, pp. 263–286, 1995.

[12] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *J. Mach. Learn. Res.*, pp. 113–141, 2000.

[13] J. Weston and C. Watkins, "Multi-class support vector machines," Dept. Comput. Sci., Univ. London, Royal Holloway, U.K., Tech. Rep. CSD-TR-98-04, 1998.

[14] E. J. Bredensteiner and K. P. Bennett, "Multicategory classification by support vector machines," *Comput. Optim. Appl.*, vol. 12, no. 1–3, pp. 53–79, 1999.

[15] J. C. Platt, , B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*. Cambridge, MA: MIT Press, 1998.

[16] P. H. Chen, R. E. Fan, and C. J. Lin, "A study on SMO-type decomposition methods for support vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 893–908, Jul. 2006.

[17] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, 2001.

[18] K. M. Lin and C. J. Lin, "A study on reduced support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1449–1459, Nov. 2003.

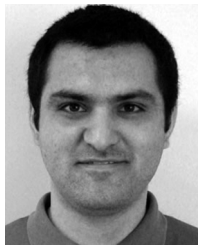
[19] Y. J. Lee and S. Y. Huang, "Reduced support vector machines: A statistical theory," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 1–13, Jan. 2007.

[20] L. Breiman, "Combining predictors," in *Combining Artificial Neural Nets*. London, U.K.: Springer-Verlag, 1999, pp. 31–50.

[21] C. L. Blake and C. J. Merz, "UCI Repository of machine learning databases" Dept. Inf. Comput. Sci., Univ. California, Tech. Rep., 1998 [Online]. Available: http://www.ics.uci.edu/ mlearn/MLRepository.html

[22] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[23] E. Alpaydın, "Combined 5 x 2 cv F test for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 11, pp. 1885–1892, 1999.



Mehmet Gönen received the B.Sc. degree in industrial engineering and the M.Sc. degree in computer engineering from Boğaziçi University, Istanbul, Turkey, in 2003 and 2005, respectively, where he is currently working towards the Ph.D. degree at the Computer Engineering Department.

He is a Teaching Assistant at the Computer Engineering Department, Boğaziçi University. His research interests include support vector machines, kernel methods, and real-time control and simulation of flexible manufacturing systems.

Ayşe Gönül Tanuğur received the B.Sc. degree in industrial engineering from Boğaziçi University, Istanbul, Turkey, in 2005, where she is currently working towards the M.Sc. degree at the Industrial Engineering Department.

She is a Teaching Assistant at the Industrial Engineering Department, Boğaziçi University. Her research interests include reverse logistics, metaheuristics, and machine learning.

Ethem Alpaydın (SM'04) received the Ph.D. degree in computer science from Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 1990.

He did his Postdoctoral work at the International Computer Science Institute (ICSI), Berkeley, CA, in 1991. Since then, he has been teaching at the Department of Computer Engineering, Boğaziçi University, Istanbul, Turkey, where he is now a Professor. He had visiting appointments at the Massachusetts Institute of Technology (MIT), Cambridge, in 1994, ICSI (as a Fulbright scholar) in 1997, and IDIAP, Switzerland, 1998. He is the author of the book *Introduction to Machine Learning* (Cambridge, MA: MIT, 2004).

Dr. Alpaydın received the Young Scientist award from the Turkish Academy of Sciences in 2001 and the scientific encouragement award from the Turkish Scientific and Technical Research Council in 2002.