Reducing SVM Classification Time Using Multiple Mirror Classifiers

Jiun-Hung Chen and Chu-Song Chen, *Member, IEEE*

***Abstract*—We propose an approach that uses mirror point pairs and a multiple classifier system to reduce the classification time of a support vector machine (SVM). Decisions made with multiple simple classifiers formed from mirror pairs are integrated to approximate the classification rule of a single SVM. A coarse-to-fine approach is developed for selecting a given number of member classifiers. A clustering method, derived from the similarities between classifiers, is used for a coarse selection. A greedy strategy is then used for fine selection of member classifiers. Selected member classifiers are further refined by finding a weighted combination with a perceptron. Experiment results show that our approach can successfully speed up SVM decisions while maintaining comparable classification accuracy.**

***Index Terms*—Classification, kernel-based method, multiple classifier system, supervised learning, support vector machine.**

I. Introduction

THE SUPPORT VECTOR machine (SVM) [35], known for being a powerful classification and regression tool, has been to perform successfully in many applications such as: handwritten digit recognition [18], [35], face detection [21] and object recognition [25].[1] In principle, its classification time is proportional to the number of support vectors it has. To speed up SVM decision time (i.e., to reduce its classification time), some studies [3], [22], [29] proposed finding a simplified classifier in which the number of vectors involved in classification is smaller than the number of support vectors. All these works focused on solving a reduced set (RS) problem [3]. An RS will be defined in Section II-A.

Unlike previous works, our approach uses a class of simple classifiers. We define a simple classifier as one in which the number of vectors involved in determining a classification result is small. Decisions of these classifiers are combined to approximate an SVM. This idea combining many simple classifiers is the same as using multiple classifier systems (MCSs) for pattern classifications [13], [31], [36]. In terms of the number of parameters to be optimized, using simple classifiers has the advantage of a shorter computation time for finding each classifier than an RS problem which uses more complex classifiers.

Manuscript received January 16, 2003; revised August 12, 2003. This work was supported in part by the National Science Council of Taiwan, R.O.C., under Grant NSC 90-2213-E-001-033. This paper was recommended by Associate Editor M. Berthold.

J.-H. Chen is with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: jhchen@cs.washington.edu).

C.-S. Chen is with the Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C. (e-mail: song@iis.sinica.edu.tw).

Digital Object Identifier 10.1109/TSMCB.2003.821867

[1]Other applications using the SVM for the training purpose can be found in http://clopinet.com/isabelle/Projects/SVM/applist.html.

In this paper, a simple classifier is devised using mirror points. Only two vectors are required in determining a classification result for a simple classifier from a pair of mirror points. Another additional advantage of using simple classifiers is that simple classifiers can be constructed or selected systematically and in tandem from a training data set. Selecting some classifiers from a set of simple ones is a combinatorial problem and is intractable if a brute force method is used. In order to solve this problem efficiently, a clustering method and a greedy approach are combined for selecting classifiers.

The remainder of this paper is organized as follows. Section II reviews the SVM and some MCSs. Our approach is detailed in Section III. Experiment results are discussed in Section IV. Conclusions and future research directions are given in Section V.

II. SVMs and MCSs

In this section, SVMs and some related work on speeding up SVM classification are reviewed, followed by an introduction to MCSs.

A. SVMs and Related Work on SVM Classification Speed

Consider a two-class classification problem. Let $\Omega = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) | x_i \in R^d, y_i \in \{-1, 1\}, i = 1, 2, \ldots, n\}$ be a set of input–output training data pairs, where R^d space is referred to as the input space and is denoted by Λ herein. The SVM [18], [35] first projects the input vectors onto the feature space F by a nonlinear function $\phi : \Lambda \rightarrow F$, and then finds a linear separating hyperplane $H_{\tilde{w},b} : \tilde{w}^T \tilde{x} + b = 0$ in the feature space, where $\tilde{x}, \tilde{w} \in F$, and $b \in R$. By solving this equation in its dual form [18], [35] $\tilde{w}$ and b can be obtained. The solution of $\tilde{w}$ is

$$\tilde{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i) \tag{1}$$

where $\alpha_i \geq 0, i = 1, \ldots, n$ are Lagrange multipliers. Note that a training vector x_i with nonzero α_i is called a support vector and typically the number of support vectors is smaller than the number of training examples.

The classification rule of SVM is

$$\begin{aligned} f_{H_{\tilde{w},b}}(x) &= \operatorname{sgn}\left(\tilde{w}^T \phi(x) + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(x, x_i) + b\right) \end{aligned} \tag{2}$$

where $K(x, x_i) = \phi(x)^T \phi(x_i)$. Note that $K(.,.)$ is a Mercer's kernel [35]. Commonly used Mercer's kernels include Gaussian RBF, polynomial functions, and sigmoidal functions. We will

denote $f_{H_{\tilde{w},b}}(x)$ as $f(x)$ for brevity in cases in which confusion is not incurred.

As seen in (2), classification time is proportional to the number of support vectors, $n' = card(\{\alpha_i > 0 | i = 1, 2, \ldots, n\})$ where $card(A)$ is the cardinal of set A. To speed up SVM decision, research focuses on solving the *RS problem* [3], which finds Υ^*, defined as follows:

$$\Upsilon^* = \arg\min_{\Upsilon} d(\Upsilon, \tilde{w}) \tag{3}$$

where

$$\Upsilon = \sum_{i=1}^{l} \beta_i \phi(\hat{x}_i) \tag{4}$$

and

$$d(\tilde{x}, \tilde{y}) = ||\tilde{x} - \tilde{y}||$$

is the Euclidean distance between the two vectors $\tilde{x}$, $\tilde{y}$ in the feature space F. l is smaller than the number of support vectors. A set of $\beta_i \in R$ and $\hat{x}_i \in \Lambda$, $i = 1, \ldots, l$ can be found by optimizing (3) (into which (4) is incorporated). The number of parameters in this optimization problem is $l(d+1)$. When l or d is large, the computation time for solving the optimization problem is very high. Iterative methods, including the RS method [3], [29] and the regression method [22], have been developed to solve the RS problem.

B. MCS

MCSs [13], [16], [31], [36] use a group of classifiers, instead of a single classifier, to compromise on a given task. The reason for combining multiple classifiers is to improve their generalization ability [13], [36], tolerate the failure of individual classifiers [13], [36] and to increase training efficiency [6].

Typically, there are two key steps in designing an MCS. The first step is to create member classifiers, and the second is to combine member classifiers. Since it is not useful to combine classifiers if they have similar classification boundaries, the main aim of the first step is to create member classifiers with different generalization abilities. Common heuristics for creating classifiers with different generalizations [31] include using different kinds of classifiers [9], [36], manipulating training parameters [34], and training on different or disjoint training sets [6]. However, the above heuristics do not always guarantee classifiers with independent errors, whereas such classifiers provide a useful criterion for improving the classification accuracy of an MCS [10], [15]. Some methods select a subset formed by the most error-independent classifiers from an initial large set of classifiers [9], [23], [32]. This is referred to as an *overproduce and select* strategy. Some other methods [20], [28] can directly create classifiers with independent errors, which are called *direct* strategy methods. For further details on these methods see [20], [28]. For combining classifiers, methods like weighted averaging [11], [34], [36], majority voting [36], and rank-based approaches [13] have been proposed. A theoretical study of combination strategies has recently been given in [16] but the study employs a restricted underlying assumption that the estimates of classifiers are independently and identically distributed (normal or uniform distribution) which cannot be immediately applied to real life situations.

Similar to an MCS system, Boosting [17] combines simple member classifiers with weights to form an ensemble such that the performance of each single ensemble member is improved. Both member classifiers and weights will be learned within the Boosting procedure. Kearns and Valiant [14] theoretically proved that weak classifiers, which perform only slightly better than random, can be combined to form an arbitrarily good ensemble classifier when enough data is available. There are a number of practical Boosting algorithms. The most popular, AdaBoost [8], allows the designers to continue adding weak learners until a desired low training error has been achieved. However, in early Boosting literature, there is a misconception that Boosting would not overfit even when working with a large number of iterations. Simulations on data sets with more noise content clearly show overfitting effects which can be avoided by regularizing Boosting so as to limit the complexity of the function class. For example, AdaBoostReg [26], BrownBoost [7], and SmoothBoost [30] are designed to adapt the applicability of boosting to noisy cases.

In this paper, we design a MCS to approximate the decision of a classifier (especially, an SVM). A coarse-to-fine approach that takes classification accuracy and classification efficiency into account is used to select a given number of member classifiers. Clustering is performed according to classifier similarities as a coarse member classifier selection. A greedy strategy, which exploits the signed-distance differences between the input training vectors and the mirror pairs, performs a fine selection. In fixing a number of member classifiers to be selected, the greedy strategy finds a subset from the coarsely selected member classifiers that approximates the original classifier.

III. MCS Constructed From Approximate Mirror Classifiers

We adopt the *overproduce and select* strategy [9] to create member classifiers with errors that are as independent as possible. A set of simple classifiers, referred to as mirror classifiers [4], are overproduced from both the SVM and the training data set. Then, some candidate classifiers are selected. The selected candidate classifiers are linearly combined with weights and a bias both obtained by training a perceptron. Its inputs are the selected candidate classifiers' outputs. More details about the proposed method are given in the following subsections.

A. Mirror Classifiers

Assume that a linear separating hyperplane $H_{\tilde{w},b}$ in the feature space F is used as a classification plane where $\tilde{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)$ and $||\tilde{w}|| = (\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j))^{1/2} > 0$. Given $v \in \Lambda$, the distance from its image $\phi(v)$ to a hyperplane $H_{\tilde{w},b}$, denoted by $d(\phi(v), H_{\tilde{w},b})$, is

$$\begin{aligned} d(\phi(v), H_{\tilde{w},b}) &= \frac{\left|\tilde{w}^T \phi(v) + b\right|}{||\tilde{w}||} \\ &= \frac{\left|\sum_{i=1}^{n} \alpha_i y_i K(v, x_i) + b\right|}{\left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j)\right)^{\frac{1}{2}}}. \end{aligned} \tag{5}$$

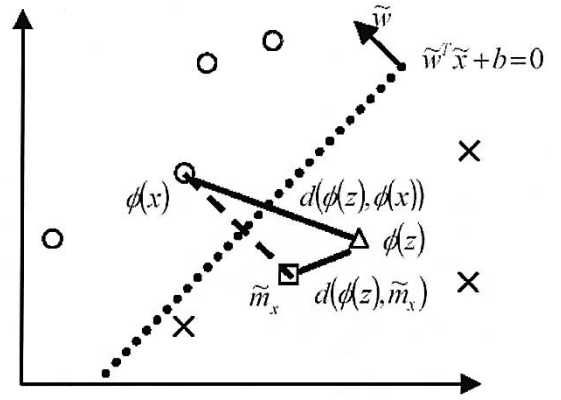


Fig. 1. Linear classifiers formed with mirror points. Dotted line $\tilde{w}^T\tilde{x}+b=0$ is a linear separating hyperplane in the feature space. The arrow points to the region in which data are classified with positive labels. Os and Xs are data points belonging to positive and negative classes, respectively. The mirror point of $\phi(x)$ is $\tilde{m}_x$. z can be classified according to the distances $d(\phi(z),\phi(x))$ and $d(\phi(z),\tilde{m}_x)$. If $d(\phi(z),\phi(x))$ is not larger than $d(\phi(z),\tilde{m}_x)$, z is classified to the same class of point x. If not, z is classified to the opposite class of point x.

Furthermore, its mirror vector $\tilde{m}_v$, associated with $H_{\tilde{w},b}$ in the feature space F, is defined as

$$\tilde{m}_v = \phi(v) - 2f(v)d\left(\phi(v), H_{\tilde{w},b}\right)\frac{\tilde{w}}{||\tilde{w}||}. \tag{6}$$

Given a pair of mirror points $(\phi(v),\tilde{m}_v)$ with $d(\phi(v),H_{\tilde{w},b}) > 0$, let us define a classification rule $g_{\phi(v),\tilde{m}_v}(z)$ as

$$g_{\phi(v),\tilde{m}_v}(z) = \begin{cases} f(v), & \text{if } d\left(\phi(z),\phi(v)\right) \le d\left(\phi(z),\tilde{m}_v\right) \\ -f(v), & \text{otherwise} \end{cases} \tag{7}$$

where $z \in \Lambda$. Then the following property holds.

Property 1: $g_{\phi(v),\tilde{m}_v}(z) = f(z)$ for all $z \in \Lambda$.

Proof: This property holds because $\phi(v)$ and $\tilde{m}_v$ form a mirror pair and the distances from $\phi(v)$ and $\tilde{m}_v$ to the hyperplane $H_{\tilde{w},b}$ in the feature space are the same.

From Property 1, the classification results of $f(z)$ and $g_{\phi(v),\tilde{m}_v}(z)$ are the same for all $z \in \Lambda$. This concept is illustrated in Fig. 1.

B. SVM Approximation From Combined Decisions of Multiple Mirror Classifiers

Assume that the pre-image [29] of $\tilde{m}_v$, (i.e., a point $q \in \Lambda$ such that $\tilde{m}_v = \phi(q)$) can be clearly identified. Then, a single pair of mirror points, $(\phi(v),\tilde{m}_v)$, can be used to construct an equivalent classifier of the SVM. However, the pre-image of $\tilde{m}_v$ may either not exist or require a complex representation. Let us consider a case in which $\tilde{m}_v$ is approximated by $\tilde{u}$, an approximate mirror point of v. An approximate classification rule $AM^{\tilde{u}}_{\phi(v)}(z)$, which results in a linear classifier in the feature space, is defined as

$$AM^{\tilde{u}}_{\phi(v)}(z) = \begin{cases} f(v), & \text{if } d\left(\phi(z),\phi(v)\right) \le d\left(\phi(z),\tilde{u}\right) \\ -f(v), & \text{otherwise.} \end{cases} \tag{8}$$

Note that $AM^{\tilde{u}}_{\phi(v)}$ uses a single pair of approximate mirror points to approximate f.

Below, we define a procedure **CAM** that uses a weighted combination of L AM-type classifiers $(AM^{\tilde{u}_j}_{\phi(v_j)}(z), j = 1,\ldots,L)$ to approximate f more accurately. Its pseudocodes are as follows.

Procedure CAM(input: z, **output**: result)

Step 1 Let v_j, $\tilde{u}_j$, $j = 1,\ldots,L$.
Step 2 result $\leftarrow 0$.
Step 3 **For** ($j = 1$ to L)
Step 3.1 Compute $d(\phi(z),\tilde{u}_j)$ and $d(\phi(z),\phi(v_j))$.
Step 3.2 result $\leftarrow$ result $+ \Xi^{\tilde{u}_j}_{\phi(v_j)}(z)$
Step 4 **END FOR**
Step 5 result $\leftarrow \mathrm{sgn}(\text{result})$,

$\Xi^{\tilde{u}}_{\phi(v)}(z)$ is defined as

$$\Xi^{\tilde{u}}_{\phi(v)}(z) = f(v)D^z_{\tilde{u},v} \tag{9}$$

where $D^z_{\tilde{u},v} = d(\phi(z),\tilde{u})^2 - d(\phi(z),\phi(v))^2$. CAMs use the sum of signed-distance differences of the squared distances associated along with the input vectors and all their corresponding mirror pairs in order to classify the input vector.

Two problems remain. One is finding approximate pre-images of mirror points for creating mirror classifiers, and the other is finding a suitable combination of classifiers. These two problems are addressed in the following two subsections.

C. Finding an Approximate Pre-Image of a Mirror Point

If $\tilde{m}_v$ is given, we want to find (β^*, x^*) that is satisfied by

$$(\beta^*, x^*) = \arg\min_{\beta\in R, x\in\Omega_I} d\left(\beta\phi(x),\tilde{m}_v\right) \tag{10}$$

where $\Omega_I = \{x_1,\ldots,x_n\}$ is the input training data set. The mirror point $\tilde{m}_v$ can then be approximated as $\tilde{u} = \beta^*\phi(x^*)$. We find (β^*, x^*) by investigating the training data set. First, consider each $x_i \in \Omega_I$. Let β^*_i be defined as

$$\begin{aligned}\beta^*_i &= \arg\min_{\beta_i\in R} d\left(\beta_i\phi(x_i),\tilde{m}_v\right) \\ &= \arg\min_{\beta_i\in R}\left(\beta_i^2\phi(x_i)^T\phi(x_i) - 2\beta_i\phi(x_i)^T\tilde{m}_v + \tilde{m}_v{}^T\tilde{m}_v\right)^{\frac{1}{2}} \\ &= \arg\min_{\beta_i\in R}\left(\beta_i^2\phi(x_i)^T\phi(x_i)\right. \\ &\qquad \left. -2\beta_i\phi(x_i)^T\tilde{m}_v + \tilde{m}_v{}^T\tilde{m}_v\right). \end{aligned} \tag{11}$$

Since (11) a bilinear when x_i is given, analytic solutions of β^*_i can be obtained as follows:

$$\beta^*_i = \frac{C_2}{C_1} \tag{12}$$

where

$$\begin{aligned} C_1 &= \phi(x_i)^T\phi(x_i) = K(x_i,x_i) \\ C_2 &= \phi(x_i)^T\tilde{m}_v \\ &= K(v,x_i) - 2f(v)d\left(\phi(v),H_{\tilde{w},b}\right) \\ &\quad \times \frac{\sum_{j=1}^n \alpha_j y_j K(x_i,x_j)}{\left(\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j K(x_i,x_j)\right)^{\frac{1}{2}}}. \end{aligned} \tag{13}$$

After finding β^*_i for each x_i contained in the training set, $(\beta^*_i,\ x_i)$ with the minimal $d(\beta^*_i\phi(x_i),\tilde{m}_v)$ among all x_i, $i = 1,\ldots,n$ is set to (β^*, x^*). The above concept is illustrated in Fig. 2. To generalize (10), we can consider that

$$(\beta^{**}, x^{**}) = \arg\min_{\beta\in R, x\in\Lambda} d\left(\beta\phi(x),\tilde{m}_v\right) \tag{14}$$

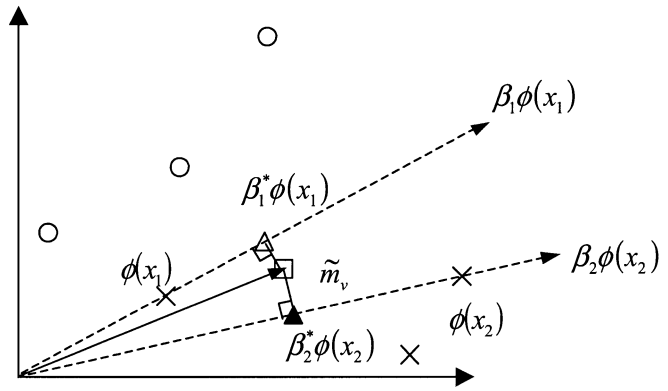


Fig. 2. (β^*, x^*) is found by investigating the training data set. Assume that $\tilde{m}_v$ is the mirror point to be approximated and Os and Xs are some training data points that belong to positive and negative classes, respectively. In the first step, we find β_i^* for all $i = 1, \ldots, n$ by using (12). For example, for $\phi(x_1)$ and $\phi(x_2)$, $\beta_1^*\phi(x_1)$, a vector starting from the origin and ending with $\triangle$, and $\beta_2^*\phi(x_2)$, a vector starting from the origin and ending with $\blacktriangle$, can be obtained by using (12). (β_i^*, x_i), with the minimal $d(\beta_i^*\phi(x_i), \tilde{m}_v)$ among all $i = 1, \ldots, n$, is set at (β^*, x^*). In this example, $(\beta^*, x^*) = (\beta_1^*, x_1)$.

where $\tilde{u} = \beta^{**}\phi(x^{**})$. To minimize the objection function (14), the solution of (10) can serve as the initial estimate to be inserted in the iterative method developed in [29].

D. Finding a Better Combination of Classifiers

A coarse-to-fine approach is developed for selecting and combining a given number of member classifiers. It takes into account both classification accuracy and classification efficiency (closely related to the number of selected classifiers). The steps in developing this approach follow.

1) Coarse Selection of Classifiers: By varying v in the space Λ, many approximate mirror classifiers [e.g., $(\phi(v), \tilde{u})$] can be generated (as introduced in Sections III-B and III-C), and a pool of classifiers can be purposely over-produced. Two possible ways for creating such a pool include choosing v to be any of the input training examples or merely choosing v randomly in space Λ. In this paper, the former approach is used. Since AM-type classifiers can be over-produced, the goal of coarse selection is selecting some representative classifiers that have the most independent errors. To do this, we adopted a classifier clustering method [9] in which classifiers are clustered according to a distance measure that measures dissimilarities of the classification results. We adopt the K-means clustering principle to cluster the classifiers. Centers of classifier clusters are selected. The distance between two AM-type classifiers AM_1 and AM_2, $\mathrm{dis}(AM_1, AM_2)$, is chosen to be

$$\mathrm{dis}(AM_1, AM_2) = ||R_1 - R_2||_1 \tag{15}$$

where $R_1 = [AM_1(x_1), \ldots, AM_1(x_n)]$, $R_2 = [AM_2(x_1), \ldots, AM_2(x_n)]$, and $||x||_1$ is the one-norm of x. The center $\bar{c}$ of a cluster consisting of n' classifiers $U = \{AM_1, \ldots, AM_{n'}\}$ is defined as

$$\bar{c} = \arg \min_{AM_j \in U} \sum_i \mathrm{dis}(AM_i, AM_j). \tag{16}$$

The pseudocodes for coarse selection are as follows.

Procedure Coarse-Selection by K-Means Clustering (**input**: a set of AM type classifiers, **output**: a set of K centers of clusters $\{\bar{c}_1, \ldots, \bar{c}_K\}$)

Step 1 Initialize K centers of clusters $\{\bar{c}_1, \ldots, \bar{c}_K\}$.
Step 2 **Do**
Step 2.1 For each mirror classifier AM_i, find its nearest center among $\{\bar{c}_1, \ldots, \bar{c}_K\}$ by (15). Thus K clusters are formed.
Step 2.2 Update $\{\bar{c}_1, \ldots, \bar{c}_K\}$ by (16).
Step 3 **UNTIL** there is no change in $\{\bar{c}_1, \ldots, \bar{c}_K\}$

2) Creating a CAM by a Greedy Method: Assume that after K-means clustering, K AM-type classifiers are obtained whose signed-difference functions [defined in (9)] are $\Xi_1(.), \ldots, \Xi_K(.)$. The following classification rule is used to classify a given z in the procedure CAM:

$$f_{\mathrm{CAM}}(z) = \mathrm{sgn}\left(\sum_{i=1}^{L} \Xi_{\phi(v_i)}^{\tilde{u}_i}(z)\right). \tag{17}$$

Finding an intermediate CAM that better approximates f is addressed below.

The problem is formulated as finding a suitable subset containing at most L classifiers among the K classifiers $(L < K)$ which has the best classification performance. Let P be a mapping from $\{1, \ldots, L\}$ to $\{1, \ldots, K\}$. Then, $\{\Xi_{P(1)}, \Xi_{P(2)}, \ldots, \Xi_{P(L)}\}$ corresponds to a set of at most L classifiers chosen from $\Psi = \{\Xi_1, \ldots, \Xi_K\}$. To choose a set of at most L classifiers among the initial K ones is equivalent to finding a P (note that some elements of $\{\Xi_{P(1)}, \Xi_{P(2)}, \ldots, \Xi_{P(L)}\}$ may be the same). Note that $\mathrm{sgn}(\sum_{i=1}^{L} \Xi_{P(i)}(x_j)) = y_j$ for all $j = 1, \ldots, n$, is expected in the ideal case. A performance index is defined as

$$g(P) = \sum_{j=1}^{n} y_j \left(\mathrm{sgn}\left(\sum_{i=1}^{L} \Xi_{P(i)}(x_j)\right)\right), \tag{18}$$

and we hope to find a P^* maximizing the performance index:

$$P^* = \arg \max_P g(P). \tag{19}$$

Note that in (18), the performance index is defined according to the ground-truth of the classification problem. Alternatively, if we change y_j to be the output of the SVM being approximated, we can define a performance index based on how accurately the SVM can be approximated. We choose the former performance index since the resultant CAM will more likely have a better generalization ability than that of the latter.

As finding the global optimal P^* is difficult, we use the greedy method described below to find a suboptimal solution. Initially, for each of the K classifiers, we construct a MCS that contains only this classifier. We call these MCSs *current*. New MCSs are formed by iteratively *adding* a classifier to *current* MCSs. We design the following iterative procedure that has three steps per iteration. The first step for each classifier is that K new MCSs are constructed by adding it to each *current* MCS. The second step for each classifier is that the MCS with the best classification performance among these K new MCSs is saved. In the third step, we replace the K *current* MCSs with the K saved MCSs in the second step. By

repeating this procedures $L-1$ times, we can get K *current* MCSs with at most L different classifiers. The MCS with the best classification performance among these K *current* MCSs is our final suboptimal solution.

In the following, we show the details of the above greedy method. Let $S(m,l)$ and $\Delta(m,l)$ be recursively defined for $l=1,2,\ldots,L$ and $m=1,2,\ldots,K$ as (20) and (21), shown at the bottom of the page, respectively. Note that $S(i,l-1)_j$ is the jth component of $S(i,l-1)$.

To clarify the above formulations, we will explain them step by step. Initially, for classifier $\Xi_m, m=1,\ldots,K$, we construct a MCS that contains only this classifier and use $S(m,1)$ to store this classifier's classification results for all data $x_1,x_2,\ldots,x_n$ (i.e., $S(m,1)=[\Xi_m(x_1),\Xi_m(x_2),\ldots,\Xi_m(x_n)]$). We call these K MCSs, $S(1,1),S(2,1),\ldots,S(K,1)$, *current* MCSs. In addition, we let $\Delta(m,1)=m$ for $m=1,\ldots,K$.

Then, a new MCS can be formed by adding a classifier to a *current* MCS. For example, by adding classifier Ξ_m to *current* MCS $S(i,1)$, $i\in\{1,2,\ldots,K\}$, a new MCS can be constructed under a condition that for data x_j its classification result is $S(i,1)_j+\Xi_m(x_j)$. If we combine Ξ_m to each *current* MCS in the above way, K new MCSs can be constructed. Among these K new MCSs we find the MCS with the best classification performance (i.e., $\max_{i\in 1,2,\ldots,K}\sum_{j=1}^{n} y_j(\mathrm{sgn}(S(i,1)_j+\Xi_m(x_j)))$). In addition, we use $\Delta(m,2)$ to save one of the *current* MCS indexes in combination with Ξ_m that creates the best MCS among these K new MCSs. The best classification results for all data are stored at $S(m,2)$ (i.e, $S(\Delta(m,2),1)+[\Xi_m(x_1),\Xi_m(x_2),\ldots,\Xi_m(x_n)]$). $\Delta(m,2)$ and $S(m,2)$ for all $m=1,2,\ldots,K$ can be similarly calculated. We replace the K *current* MCSs $S(m,1)$, $m=1,2,\ldots,K$ with the K saved MCSs $S(m,2)$, $m=1,2,\ldots,K$. By repeating above procedures $L-1$ times, we can get K *current* MCSs at most L different classifiers, $S(1,L),S(2,L),\ldots,S(K,L)$. Then, the MCS with the best classification performance among these K *current* MCSs is our final suboptimal solution and can be found as follows.

Let

$$\hat{m}=\arg\max_{m\in 1,2,\ldots,K}\sum_{j=1}^{n} y_j\mathrm{sgn}\left(S(m,L)_j\right). \tag{22}$$

$S(\hat{m},L)$ is the best MCS (found by our method), which consists of at most L different classifiers, and serves as our resultant MCS. We then backtrack to find member classifiers of the resultant MCS by finding $\hat{P}(l)$ as defined in the following:

$$\hat{P}(l)=\begin{cases}\hat{m}, & \text{if } l=L,\\ \Delta\left(\hat{P}(l+1),l+1\right), & \text{if } l=1,2,\ldots,L-1.\end{cases} \tag{23}$$

The obtained classifiers associated with $\hat{P}$ are then employed by the CAM we described in Section III-B (that is, $\Xi_{\hat{P}(j)}$ substitutes for $\Xi^{\tilde{u}_j}_{\phi(v_j)}$ in **Procedure CAM**).

Step 1 Initialize $S(m,1)=[\Xi_m(x_1),\ldots,\Xi_m(x_n)]$ and $\Delta(m,l)=m$ for $m=1,\ldots,K$.
Step 2 **For** ($l=2$ to L)
Step 2.1 **For** ($m=1$ to K)
Step 2.1.1 find $\Delta(m,l)$ by (21) and compute $S(m,l)$ by (20).
Step 2.2 **END FOR**
Step 3 **END FOR**
Step 4 Compute $\hat{m}$ by (22).
Step 5 **For** ($l=L$ to 1)
Step 5.1 Compute $\hat{P}(l)$ by (23).
Step 6 **END FOR**

The procedure of fine selection is as follows.

Procedure Fine-Selection(**input**: a set of K classifiers $\Xi_1(.),\ldots,\Xi_K(.)$ and the number L, **output**: $\hat{P}$) .

Note that the above method can obtain the global optimum if $\mathrm{sgn}(.)$ is modified to be an identity function in (18), (21) and the definition of $\hat{m}$. When $\mathrm{sgn}(.)$ is replaced by an identity function, the above method finds a path with the largest additive score if the score is defined as $\sum_{j=1}^{n}\Xi_i(x_j)$. In this condition, Bellman's principle of optimality [1] for dynamic programming is satisfied so that the optimal solution can be found using our described greedy method. However, this greedy method can not always yield P^* defined in (19) since Bellman's principle of optimality is not exactly satisfied when $\mathrm{sgn}(.)$ is used. Nevertheless, according to our experience, this greedy method can find a good sub-optimal solution in practice.

3) Finding a Weighted Combination for a CAM: Since the signed differences of squared distances, $\Xi^{\tilde{u}_j}_{\phi(v_j)}(.)$, $j=1,\ldots,L$, are all measured in the feature space F, they are not necessarily proportional to the distances or differences measured in the input space Λ. In this subsection, we further introduce a weight associated with each $\Xi^{\tilde{u}_j}_{\phi(v_j)}(.)$ to compensate this effect. The CAM is therefore further refined as a weighted CAM (WCAM) as introduced below.

Procedure WCAM (**input**: z, **output**: result)

Step 0 Let v_j, w_j, $\tilde{u}_j$, $j=1,\ldots,L$ and b_0 be given
Step 1 result $\leftarrow 0$.
Step 2 **For** (j $=1$ to L)
Step 2.1 Compute $d(\phi(z),\tilde{u_j})$ and $d(\phi(z),\phi(v_j))$.
Step 2.2 result $\leftarrow$ result $+\, w_j\Xi^{\tilde{u}_j}_{\phi(v_j)}(z)$
Step 3 **END FOR**
Step 4 result $\leftarrow \mathrm{sgn}(\text{result}+b_0)$,

$$S(m,l)=\begin{cases}[\Xi_m(x_1),\Xi_m(x_2),\ldots,\Xi_m(x_n)], & \text{if } l=1\\ S\left(\Delta(m,l),l-1\right)+[\Xi_m(x_1),\Xi_m(x_2),\ldots,\Xi_m(x_n)], & \text{if } l=2,3,\ldots,L\end{cases} \tag{20}$$

$$\Delta(m,l)=\begin{cases}m, & \text{if } l=1\\ \arg\max_{i\in 1,2,\ldots,K}\sum_{j=1}^{n} y_j\left(\mathrm{sgn}\left(S(i,l-1)_j+\Xi_m(x_j)\right)\right), & \text{if } l=2,3,\ldots,L\end{cases} \tag{21}$$

Let $\underline{\Xi}(x) = [\Xi^{\tilde{u}_1}_{\phi(v_1)}(x), \ldots, \Xi^{\tilde{u}_L}_{\phi(v_L)}(x)]^T$ and make $\underline{\Lambda} = \{\underline{\Xi}(x) | x \in \Lambda\}$. We can verify that a WCAM is a linear classifier in $\underline{\Lambda}$ because

$$f_{\mathrm{WCAM}}(z) = \mathrm{sgn}\left(\sum_{i=1}^{L} w_i \Xi^{\tilde{u}_i}_{\phi(v_i)}(z) + b_0\right). \quad (24)$$

A CAM is a special case of WCAMs when $w_j = 1$, $j = 1, \ldots, L$ and $b_0 = 0$ in WCAMs. A WCAM with better classification accuracy can be accomplished by using the obtained CAM as an initial estimation. The optimal weight parameters and bias in WCAM can be found by solving a new classification problem that discriminates the following input–output training data pairs $\underline{\Omega}$.

$$\underline{\Omega} = \{(\underline{\Xi}(x_1), y_1), (\underline{\Xi}(x_2), y_2), \ldots, (\underline{\Xi}(x_N), y_n)\}. \quad (25)$$

In this paper, we use a method similar to those introduced in [11] and [34] to refine these parameters. A standard perceptron [12] is trained to discriminate $\underline{\Omega}$ and we let the resultant separating hyperplane be $\underline{w}^T \underline{x} + \underline{b} = 0$, where $\underline{w}$, $\underline{x} \in R^L$ and $\underline{b} \in R$ with the initialization being $\underline{w} = [1\ 1 \ldots 1]^T$ and $\underline{b} = 0$. After the separating hyperplane is obtained, we set w_j to be the jth component of $\underline{w}$ for all $j = 1, \ldots, L$ and b_0 to be $\underline{b}$. Fig. 3 shows the state-flow of the perceptron used for finding the optimal parameters.

E. Implementation and Complexity Analysis for WCAM With Kernel Functions

The procedure WCAM can be implemented with Mercer's kernels as shown below. Assume that a set of $\{v_1, v_2, \ldots, v_L\}$ is given. For each v_i, $(i = 1, \ldots, L)$, we can get $\dot{\beta}_i$ and $\dot{x}_i$ $(i = 1, \ldots, L)$ by solving the following equation (see (10)) as introduced in Section III-C.

$$(\dot{\beta}_i, \dot{x}_i) = \arg \min_{\beta \in R, x \in \Omega_I} d(\beta\phi(x), \tilde{m_{v_i}}),$$

In addition, let $\tilde{u}_i = \dot{\beta}_i \phi(\dot{x}_i)$ for $i = 1, \ldots, L$.

From (24), the following derivations can be obtained.

$$\begin{aligned}
&f_{\mathrm{WCAM}}(z) \\
&= \mathrm{sgn}\left(\sum_{i=1}^{L} w_i \Xi^{\tilde{u}_i}_{\phi(v_i)}(z) + b_0\right) \\
&= \mathrm{sgn}\left(\sum_{i=1}^{L} w_i f(v_i) \left(d\left(\phi(z), \dot{\beta}_i \phi(\dot{x}_i)\right)^2 \right.\right. \\
&\qquad \left.\left. - d(\phi(z), \phi(v_i))^2\right) + b_0\right) \\
&= \mathrm{sgn}\left(\sum_{i=1}^{L} w_i f(v_i) \left(K(z,z) - 2\dot{\beta}_i K(z, \dot{x}_i) + \dot{\beta}_i^2 K(\dot{x}_i, \dot{x}_i) \right.\right. \\
&\qquad \left.\left. - (K(z,z) - 2K(z, v_i) + K(v_i, v_i))\right) + b_0\right) \\
&= \mathrm{sgn}\left(\sum_{i=1}^{L} w_i f(v_i) \left(-2\left(\dot{\beta}_i K(z, \dot{x}_i) - K(z, v_i)\right) \right.\right. \\
&\qquad \left.\left. + \dot{\beta}_i^2 K(\dot{x}_i, \dot{x}_i) - K(v_i, v_i)\right) + b_0\right). \quad (26)
\end{aligned}$$

In (26), the terms $w_i f(v_i)$, $\dot{\beta}_i^2 K(\dot{x}_i, \dot{x}_i) - K(v_i, v_i)$ $i = 1, \ldots, L$ can be obtained off-line. For simplicity, we let

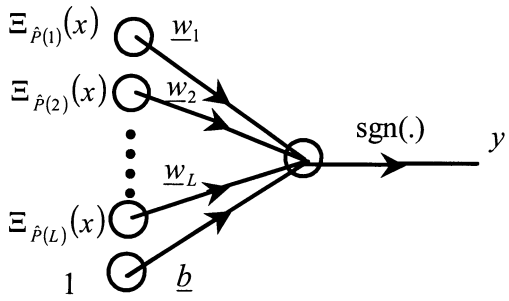


Fig. 3. Refining weight parameters for WCAM by training a perceptron.

$W_i = w_i f(v_i)$ and $A_i = \dot{\beta}_i^2 K(\dot{x}_i, \dot{x}_i) - K(v_i, v_i)$, for $i = 1, \ldots, L$ herein. By substituting W_i, A_i, $i = 1, \ldots, L$ into (26), we obtain

$$\begin{aligned}
&f_{\mathrm{WCAM}}(z) \\
&= \mathrm{sgn}\left(\sum_{i=1}^{L} W_i \left(-2\left(\dot{\beta}_i K(z, \dot{x}_i) - K(z, v_i)\right) + A_i\right) + b_0\right) \\
&= \mathrm{sgn}\left(\sum_{i=1}^{L} W_i' K(z, \dot{x}_i) + \sum_{i=1}^{L} W_i'' K(z, v_i) + B\right) \quad (27)
\end{aligned}$$

where $W_i' = -2W_i\dot{\beta}_i$, $W_i'' = 2W_i$ and $B = \sum_{i=1}^{L} W_i A_i + b_0$ is a constant.

Let us refer to one computation of $K(.,.)$ as a kernel operation. The time complexity of (27) can be described as approximately the sum of $2L$ kernel operations and $2L$ multiplications. Note the time complexity of a standard SVM decision as shown in (2), which requires approximately the sum of n kernel operations and n multiplications. In practice, the kernel operations are a major bottleneck for SVM decision. For example, if the kernel is selected to be a Gaussian RBF [35]

$$K(z_1, z_2) = \exp\left(-\gamma \|z_1 - z_2\|^2\right) \quad (28)$$

where z_1, $z_2 \in \Lambda$, then there are approximately d subtractions, d multiplications, d additions, a square root operation, and an exponential operation in a kernel operation. If the kernel is selected to be a polynomial function [35]

$$K(z_1, z_2) = \left(z_1^T z_2 + c_1\right)^{c_2} \quad (29)$$

where c_1, $c_2 \in R$, then there are approximately d multiplications, d additions, and a c_2 power operation in a kernel operation. Hence, if the kernel operation is viewed as the *major operation* in (27) and (2), their time complexities are $O(2L)$ and $O(n')$, respectively. Therefore, the speedup ratio of WCAM to SVM decision is approximately $(n'/2L)$. When L is selected to be smaller than $(n'/2)$, speedup is achieved.

IV. Experiment Results and Discussions

Since operations described in Section III-C and Section III-D (finding approximate mirror points and constructing a WCAM) can be done off-line, we focus on the on-line classification efficiencies and accuracy. The adopted kernel function is RBF in all experiments.

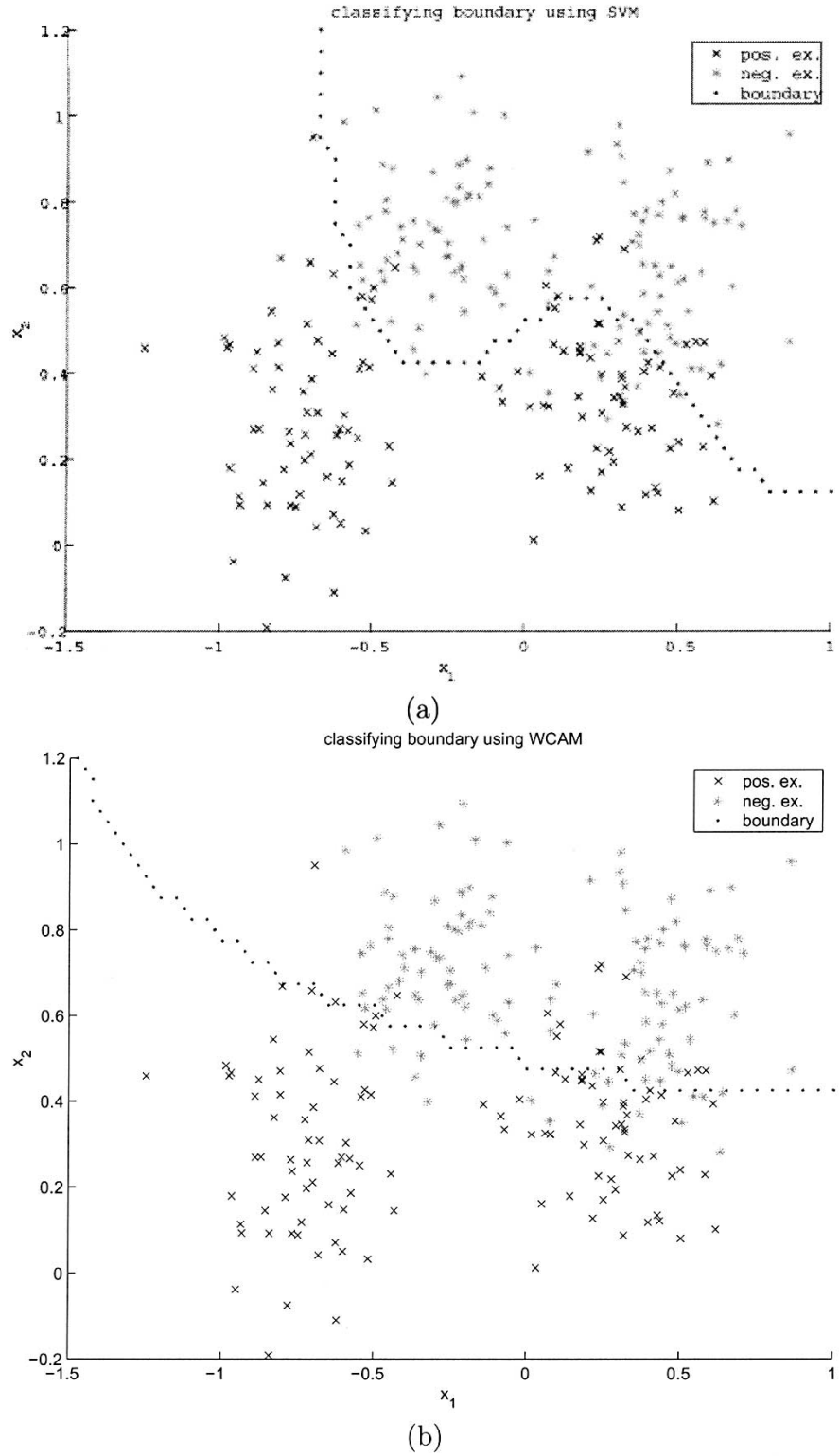


Fig. 4. Classification boundaries obtained using SVM and WCAM for the Ripley data set. (a) Classification boundary obtained using SVM. (b) Classification boundary obtained using WCAM in which two mirror pairs (i.e., four vectors) were used.

A. Two-Class Classification Results

1) Synthetic Data Set Results: This section presents the result of a classification problem using a synthetic data set, Ripley data set,[2] that was used in experiments in [22]. In this data set, the number of attributes is 2 and the numbers of training and testing examples are 250 and 1000, respectively. The parameters γ and C of the kernel function RBF used for training the SVM are 1 and 100, respectively (which are the same as in [22]). The LIBSVM[3] software was used to train an SVM for this classification problem. For the trained SVM, the number of support

[2]Available: ftp://markov.stats.ox.ac.uk/pub/neural/papers.

[3][Online] Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

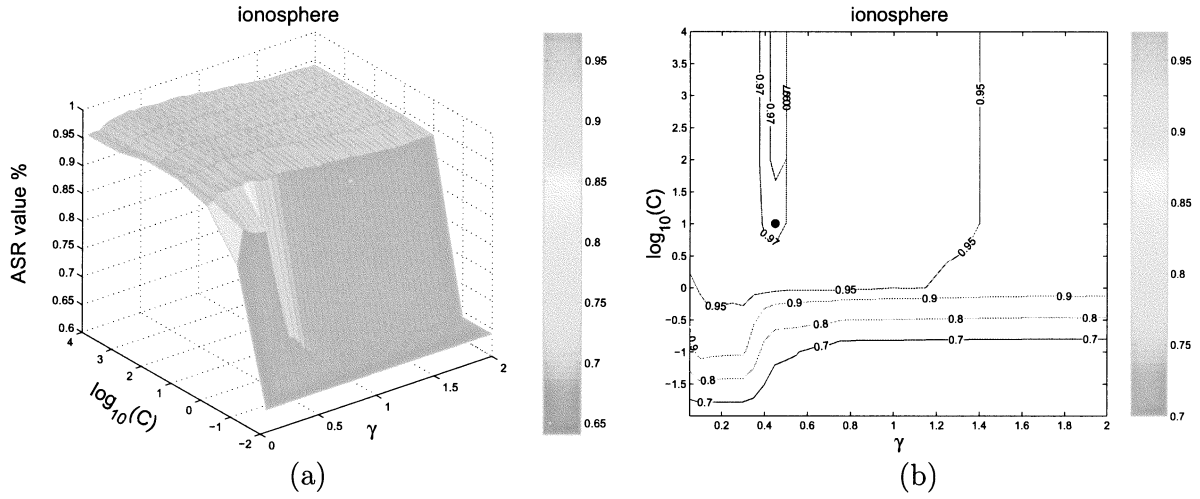


Fig. 5. Model selection process for SVM over the ionosphere dataset. A Black dot shows the positions of the best parameters. (a) ASR versus the parameters (r,C) for the ionosphere dataset. (b) The level-curve diagram of (a).

vectors is 78, and the training correct rate and the testing correct rate are 89.6%, and 89.7%, respectively. Fig. 4(a) shows the classifying boundaries of the Ripley data set using the SVM.

We use the approach introduced in Section III to find a WCAM to approximate the SVM decision. The number of mirror pairs L is set to be 2. The obtained classification boundary is shown in Fig. 4(b). The correct rates for training and testing data sets with the WCAM are 88% and 89%, respectively. These rates are comparable to the classification results rates, 89.6% and 89.7%, obtained by SVM. From experimental results in [22], 14 vectors and 33 vectors (i.e., about 5.5 and 2.33 times speedup) were used to approximate $\tilde{w}$.[4] The WCAM obtained by our method achieves $78/(2*2) = 19.5$ times faster than the SVM decision and has a comparable classification performance.

2) Real Data Set Results: In the first experiment, we use ten-fold cross-validation to train a classifier of the ionosphere dataset,[5] and in this dataset the numbers of attributes and training examples are respectively 32 and 351. To find an SVM with a good classification performance, model selection was performed to find the best pair of parameters (γ, C) within a given range, and the average of the training and testing correct rates (ASR) is chosen to be the performance measure. The parameters γ and C were respectively investigated within [0,2] and $[10^{-2}, 10^{4}]$ in our model selection process. Fig. 5 shows model selection results. The best (γ, C) found is (0.45,10). In this parameter setting, the average number of support vectors, the average training correct rate and the average testing correct rate are 171.8, 100% and 94.6%, respectively.

A WCAM was built with the proposed method to approximate the SVM in this experiment. Average classification and speedup performances of all folds were computed for both the training and testing data sets. Fig. 6 shows the experiment results. As is shown, when the number of mirror pairs in use is 8.55 (i.e., the number of vectors in use is $17.1 = 2*8.55$), the speedup ratio is ten. The obtained correct ratios for training and testing data are respectively 91.9% and 91.5%. Note that for testing data WCAM can achieve better correct rates than SVM when the number of vectors involved in determining classifications is larger than 44.2 (i.e., the speedup ratio is $3.89 = 171/44.2$). The best correct rates achieved by the approximation of WCAM for training and testing data are 98.6% and 96.3%, respectively. Meanwhile, the number of vectors involved in determining classifications is 74.4 (i.e., the speedup ratio is $2.31 = 171/74.4$). In this example, the ASRs of WCAM and SVM are 97.45% and 97.3%, respectively. In term of ASRs, the performance of WCAM is slightly better than that of SVM even when parameters of the SVM have been extensively investigated for this dataset. Note that WCAM takes less than half the classification time of that required for SVM to achieve this performance.

A digit recognition experiment that discriminates between digit 0 and all other digits is performed on the USPS data set.[6] In this USPS data set, the number of attributes is 256 and the numbers of training and testing examples are 7291 and 2007, respectively. The (γ, C) is set to be (1/128,10)[7] for training SVM. The number of support vectors is 230 and the training and the testing correct rates are 100%, and 99.25%, respectively.

Fig. 7 shows the experiment results using WCAM. For example, when the number of vectors in use is 20, the speedup ratio is $11.5 = 230/20$. The obtained training and testing correct ratios are 99.51% and 98.85%, respectively. In this experiment, the best training and testing correct rates achieved by the approximation of WCAM are 99.67% and 98.95%, respectively. In the WCAM with the best training and testing correct rates, the number of vectors involved in determining classifications is 64 (i.e., the speedup ratio is $3.59 = 230/64$).

The USPS data set has also been used in [29] which introduces two techniques in RS methods. One is called the RS selection technique, and the other is called the reduced set construction (RSC) technique. Selection via kernel PCA (SK) and as well as L_1 penalization (SP) are both introduced in the RS selection technique. Table I shows the comparisons of the num-

[4]However, they did not report the classification results and performances.

[5]Available: UCI Repository of machine learning databases http://www.ics.uci.edu/~mlearn/MLRepository.html.

[6]Available: ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data.

[7]This setting of parameters is the same as those in [29].

TABLE I
COMPARISON OF THE NUMBERS OF MISCLASSIFIED DIGITS AND THE CORRECT RATE OF THE TESTING DATA (SEPARATED WITH /) FOR EACH METHOD UNDER GIVEN SPEEDUP RATIOS. FOR SK, SP, AND RSC, ALL NUMBERS ARE ESTIMATED FROM TABLES II TO IV IN [29]

	SK	SP	RSC	WCAM
Speedup ratio(25)	154/92.33%	>319/<84.11%	26/98.70%	37.5/98.13%
Speedup ratio(11.5)	134.5/93.30%	>319/<84.11%	24/98.80%	23/98.85%
Speedup ratio(8.2)	102.5/94.90	314.3/84.34%	20.3/98.99%	21.1/98.95%

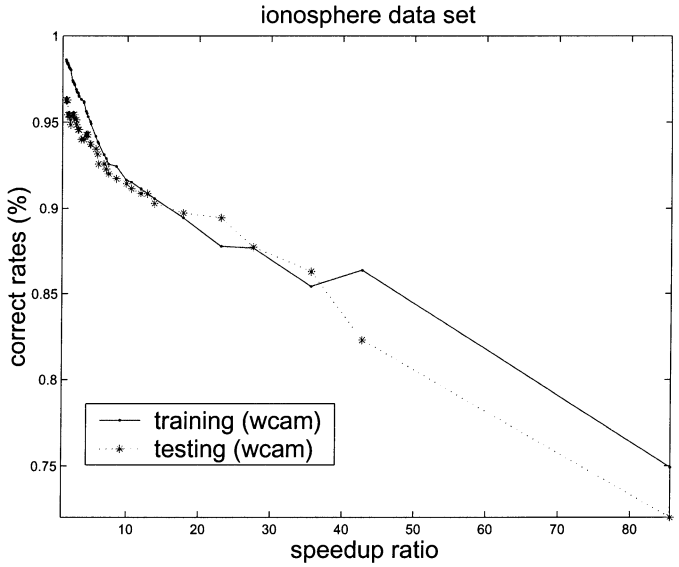


Fig. 6. Speedup performance of WCAM for the ionosphere data set. Correct rates versus the speedup ratios.

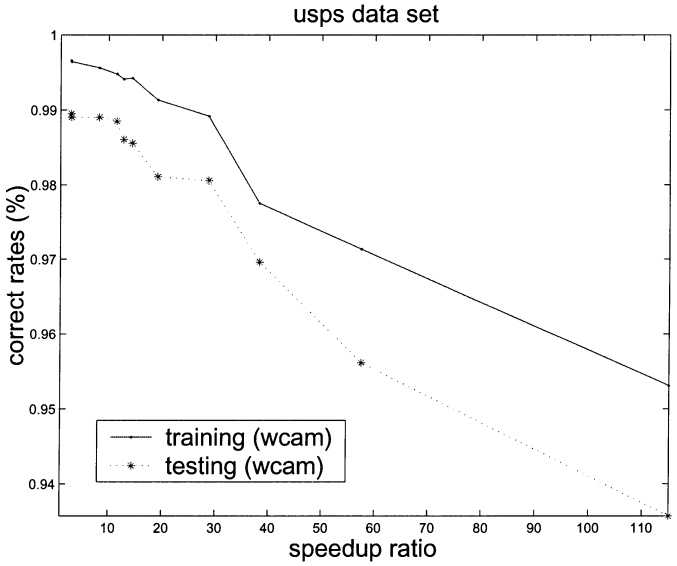


Fig. 7. Speedup performance of WCAM for the USPS data set. Correct rates versus the speedup ratios.

bers of misclassified digits under some given speedup ratios for each different method. When the number of misclassified digits is fixed at 23, the speedup rates for SK, SP, RSC and WCAM are 3.39, 2.24, 11.29, and 11.5, respectively. Both Table I and the above statistics show that our method is superior to SK and SP, and approximately comparable to RSC.

B. Multiclass Classification Results

We test our method on a multiclass classification problem using the COIL20 database [19]. Since to the best of our knowledge, there are no other related papers using SVM on this database, we design a simple setting for this database. For each object, we use 54 of its images as training images and its other remaining 18 images as testing images. To deal with this multiclass classification problem, we use a standard one-against-one strategy. That is, for each pair of two different labels, we train a binary classifier to determine the label of a test image. We then use majority voting among these $C(20,2) = 190$ classifiers to determine the label of a test image. After performing parameter selection for SVM, we can achieve perfect 100% correct rates for both training and testing images. Based on this SVM, our method is used to speed up each of $C(20,2)$ binary SVM classifiers. The average speedup rates versus correct rates for $C(20,2)$ classifiers are reported below. For total $C(20,2)$ binary classification, the best average of training and testing correct rates is 97.64% and the average speedup is 31.7 times. However, using these binary classifiers for multiclass classification, we obtain a very worse performance: the training correct rate is 87.87%, the testing correct rate is 54.72% and the speedup rate is 2.71. For multiclass classification, the best performance is that the training correct rate is 84%, the testing correct rate is 80% and the speedup rate is 1.17.

C. Discussions

We justify our method from the following two viewpoints: 1) we use clustering to speed up the greedy search process since the computational cost of our greedy search process is proportional to the number of classifiers that make up the CAM and 2) we provide some experiment results to support greedy search and perceptron learning. For example, in ionosphere's experiment results, the best average training and testing correct rates for a single mirror classifier over tenfold cross validation are 67.0% and 65.7%, respectively. After perceptron learning, these correct rates for a single mirror classifier increase to 74.9% and 72.0%, respectively. Furthermore, after combining greedy search and perceptron learning, when the speedup ratio is 10, the average training and testing correct rates become 91.9% and 91.5%. For the USPS dataset, after combining greedy search and perceptron learning, similar performance improvement is also shown.

From the above multiclass results, some points need to be discussed. For binary classification, even though our method can achieve about 31.7 times speedup, for multiclass classification, the speedup is slight. Extending a binary classification speedup method to a multiclass classification speedup method (that is, using a binary classification speedup method to speed up each binary classifier and then combining them by through a voting

method) cannot work well since the speedup for binary classification diminishes once all binary classifiers are combined. A practical extension should account for the number of common vectors in all binary classifiers so that the speedup for binary classification can be maintained after all binary classifiers are combined. One future research direction is to extend our method for multiclass classification speedup. To the best of our knowledge, there have not been any research papers on this topic.

V. Conclusions

We propose a new approach to speed up SVM decision. Compared with the existing methods [3], [22], [29], our method combines the decisions of multiple simple classifiers for approximating the decision of an SVM. By using the concept of mirror points, a pool of classifiers can be generated in a simple manner and each classifier is an approximation of the SVM. Such a useful property makes simple mirror classifiers easily be incorporated into an MCS scheme for generating an approximate decision.

Some issues merit further study. In the method proposed in this paper, each mirror point is approximated by a vector in the feature space using a single pre-image. A mirror point can also be better approximated with more than one pre-image, and finding such a better approximation for a mirror point is equivalent to solving a RS problem. Although less speedup is achieved when more pre-images are used, more pre-images induces an extension from our original scheme and can achieve a better classification performance. In addition to SVMs, our method can also be used to speed up some other classifiers whose classification functions can be expressed as a linear combination of kernel functions (e.g., (1) for an SVM). For example, the radial-basis-function network [2], [24], the relevance vector machine [33], and the fuzzy kernel perceptron [5] all have the classification rules with the same form as SVM. So our method can be applied to reduce their classification time.

Developing an automated way to find a trade-off between classification accuracy and classification speed would be a very useful future direction. However, defining an objective function which balances classification accuracy and classification speed is still task dependent and currently we do this selection manually.

Acknowledgment

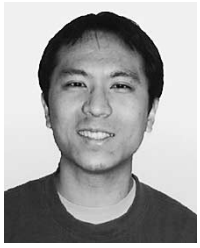
The authors thank anonymous reviewers for their valuable comments.

References

[1] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
[2] D. D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–335, 1988.
[3] C. Burges, "Simplified support vector decision rules," in *Proc. 13th Int. Conf. Machine Learning*, Bari, Italy, 1996, pp. 71–77.
[4] J.-H. Chen and C.-S. Chen, "Speeding up SVM decision based on mirror points," in *Proc. 16th Int. Conf. Pattern Recognition, ICPR 2002*, Quebec City, QB, Canada.
[5] ——, "Fuzzy kernel perceptron," *IEEE Trans. Neural Networks*, vol. 13, pp. 1364–1373, 2002.
[6] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVM's for very large scale problems," *Neural Comput.*, vol. 14, pp. 1105–1114, 2002.
[7] Y. Freund, "An adaptive verision of the boost by majority algorithm," *Mach. Learn.*, vol. 43, pp. 293–318, 2001.
[8] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an pplication to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp. 119–139, 1997.
[9] G. Giacinto, F. Roli, and G. Fumera, "Design of effective multiple classifier systems by clustering of classifiers," in *Proc. 15th Int. Conf. Pattern Recognition*, vol. 2, Barcelona, Spain, 2000, pp. 160–163.
[10] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 993–1001, Oct. 1990.
[11] S. Hashem, "Optimal linear combinations of neural networks," *Neural Networks*, vol. 10, pp. 599–614, 1997.
[12] S. Haykin, *Neural Networks a Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.
[13] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 66–75, Jan. 1994.
[14] M. Kearns and L. Valiant, "Cryptographic limitations on learning Boolean formuale and finite automata," *J. ACM*, vol. 41, pp. 67–95, 1994.
[15] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation and active learning," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, G. Tesauro, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 231–238.
[16] L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 281–286, Feb. 2002.
[17] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," in *Advanced Lectures on Machine Learning*, S. Mendelson and A. Smola, Eds. New York: Springer, 2003, pp. 119–184.
[18] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Network*, vol. 12, pp. 181–201, Mar. 2001.
[19] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-d objects from appearance," *Int. J. Comput. Vis.*, vol. 14, pp. 5–24, 1995.
[20] D. Optiz and J. Shavlik, "A genetic algorithm approach for creating neural network ensembles," in *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, A. J. C. Sharkey, Ed. NY: Springer-Verlag, 1999, pp. 79–99.
[21] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Proc. Computer Vision and Pattern Recognition*, San Juan, PR, 1997, pp. 130–136.
[22] E. Osuna and F. Girosi, "Reducing the run-time complexity of support vector machines," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 271–284.
[23] D. Partridge and W. B. Yates, "Engineering multiversion neural-net system," *Neural Comput.*, vol. 8, pp. 869–893, 1996.
[24] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, pp. 1481–1497, 1990.
[25] M. Pontil and A. Verri, "Support vector machines for 3-D object recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 637–646, June 1998.
[26] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for AdaBoost," *Mach. Learn.*, vol. 43, pp. 287–320, 2001.
[27] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake, "Computationally efficient face detection," in *Proc. 8th IEEE Int. Conf. Computer Vision*, Vancouver, BC, Canada, 2001, pp. 695–700.
[28] B. E. Rosen, "Ensemble learning using decorrelated neural network," *Connect. Sci.*, vol. 8, pp. 373–383, 1996.
[29] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Networks*, vol. 10, pp. 1000–1017, Sept. 1999.
[30] R. A. Servedio, "Smooth boosting and learning wtih malicious noise," in *Proc. 14th Annu. Conf. Computational Learning Theory*, 1999, pp. 278–285.
[31] A. J. C. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*. New York: Springer-Verlag, 1999.
[32] A. J. C. Sharkey and N. E. Sharkey, "Combining diverse neural nets," *Knowl. Eng. Rev.*, vol. 12, pp. 231–247, 1997.
[33] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.
[34] N. Ueda, "Optimal linear combination of neural networks for improving classification performance," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 207–213, Feb. 2000.

[35] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[36] L. Xu, A. Krzyzák, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 418–435, 1992.



Jiun-Hung Chen received the B.S. and M.S. degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1997 and 1999, respectively.

He was a Research Assistant in the Institute of Information Science, Academia Sinica, Taipei, until October 2003. Currently, he is with the Department of Computer Science and Engineering, University of Washington, Seattle. His research interests include pattern recognition, image processing and computer vision.

Chu-Song Chen (M'93) received the B.S. degree in control engineering from National Chiao-Tung University, Hsing-Chu, Taiwan, R.O.C., in 1989, and the M.S. and Ph.D degrees in 1991 and 1996, respectively, both from the Department of Computer Science and Information Engineering, National Taiwan University, Taipei.

From 1997 to 1999, he was a postdoctoral Fellow of the Institute of Information Science, Academia Sinica, and was promoted to Assistant Research Fellow in 1999. Currently, he is an Associate Research Fellow. His research interests include pattern recognition, computer vision, and signal/image processing. He has authored more than 50 technical papers in these fields.

Dr. Chen has received both the outstanding paper award of the Image Processing and Pattern Recognition (IPPR) Society and the best paper award of the Image Processing and Application Association (IPAA) of Taiwan, R.O.C. in 1997. He has received the outstanding paper award in the field of computer applications in ICS'2000 held in Chiayi, Taiwan.