Generalized Core Vector Machines

Ivor Wai-Hung Tsang, James Tin-Yau Kwok, and Jacek M. Zurada, *Fellow, IEEE*

***Abstract*—Kernel methods, such as the support vector machine (SVM), are often formulated as quadratic programming (QP) problems. However, given m training patterns, a naive implementation of the QP solver takes $O(m^3)$ training time and at least $O(m^2)$ space. Hence, scaling up these QPs is a major stumbling block in applying kernel methods on very large data sets, and a replacement of the naive method for finding the QP solutions is highly desirable. Recently, by using approximation algorithms for the minimum enclosing ball (MEB) problem, we proposed the core vector machine (CVM) algorithm that is much faster and can handle much larger data sets than existing SVM implementations. However, the CVM can only be used with certain kernel functions and kernel methods. For example, the very popular support vector regression (SVR) cannot be used with the CVM. In this paper, we introduce the center-constrained MEB problem and subsequently extend the CVM algorithm. The *generalized CVM* algorithm can now be used with any linear/nonlinear kernel and can also be applied to kernel methods such as SVR and the ranking SVM. Moreover, like the original CVM, its asymptotic time complexity is again linear in m and its space complexity is independent of m. Experiments show that the generalized CVM has comparable performance with state-of-the-art SVM and SVR implementations, but is faster and produces fewer support vectors on very large data sets.**

***Index Terms*—Approximation algorithms, core vector machines (CVMs), kernel methods, minimum enclosing ball (MEB), quadratic programming, support vector machines (SVMs).**

I. Introduction

KERNEL methods have been highly successful in various machine learning and pattern recognition problems. Examples include the support vector machine (SVM) and support vector regression (SVR), which are especially prominent for classification and regression tasks [1]. Often, kernel methods are formulated as quadratic programming (QP) problems, which have the important computational advantage of not suffering from the problem of local minima. However, given m training patterns, a naive implementation of the QP solver takes $O(m^3)$ training time and at least $O(m^2)$ space [2]. Hence, a major stumbling block in applying kernel methods on large data sets is how to scale up these QPs, and a replacement of the naive method for finding QP solutions posed by an SVM is highly desirable.

In recent years, a variety of approaches have been proposed in the context of kernel methods. A popular technique is to replace the kernel matrix by some low-rank approximation. These approximations can be obtained by various methods including: the Nyström method [3], greedy approximation [4], sampling [5], or matrix decompositions [6]. Another approach to scale up kernel methods is by chunking [7] or more sophisticated decomposition methods [8]–[11]. Going to the extreme, the well-known sequential minimal optimization (SMO) algorithm [10], [12] breaks the original QP into a series of smallest possible QPs, each involving only two variables. Similar in spirit to decomposition algorithms are methods that scale down the training data before inputting to the SVM (such as [2], [13], and the reduced SVM [14]). Other scaleup methods include the kernel adatron [15], SimpleSVM [11], and [16]. For a more complete survey (for works before 2003), interested readers are referred to [17] and [1, Ch. 10]. By incorporating these speed-up strategies, state-of-the-art SVM implementations have an empirical training time complexity that scales between $O(m)$ and $O(m^{2.3})$ in practice [10]. This can be further driven down to $O(m)$ with the use of parallel mixtures [2]. However, these time complexities are only empirical observations and not theoretical guarantees.

Recently, Tsang *et al.* proposed the core vector machine (CVM) [18] by exploiting the "approximateness" in the design of SVM implementations. The key observation is that practical SVM implementations, as in many numerical routines, only *approximate* the optimal solution by an iterative strategy. Typically, the stopping criterion utilizes either the precision of the Lagrange multipliers or the duality gap. For example, in SMO, SVM$^{\text{light}}$ and SimpleSVM, training stops when the Karush–Kuhn–Tucker (KKT) conditions are fulfilled within a tolerance parameter ϵ. Experience with this software indicates that near-optimal solutions are often good enough in practical applications. By utilizing an approximation algorithm for the *minimum enclosing ball* (MEB) problem in computational geometry, the CVM algorithm achieves an asymptotic[1] time complexity that is *linear* in m and a space complexity that is *independent* of m. Experiments on large classification data sets also demonstrated that the CVM is as accurate as existing SVM implementations, but is much faster and can handle much larger data sets than existing scale-up methods. For example, CVM with the Gaussian kernel produces superior results on the KDDCUP-99 intrusion detection data, which has about five million training patterns, in only 1.4 s on a 3.2 GHz Pentium-4 PC.

However, as mentioned in [18], applicability of the CVM algorithm depends critically on the following two requirements being satisfied:

1) the kernel function k satisfies $k(\mathbf{x}, \mathbf{x}) = $constant for any pattern $\mathbf{x}$;

Manuscript received September 2, 2005; revised January 19, 2006. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region under Grant 615005.

I. W.-H. Tsang and J. T.-Y. Kwok are with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: ivor@cs.ust.hk; jamesk@cs.ust.hk).

J. M. Zurada is with the Department of Electrical and Computer Engineering, the University of Louisville, Louisville, KY 40292 USA (e-mail: jacek.zurada@louisville.edu).

Color versions of Figs. 2–7 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2006.878123

[1]As we are interested in handling very large data sets, an algorithm that is asymptotically more efficient (in time and space) will be the best choice. However, on smaller problems, this may be outperformed by algorithms that are not as efficient asymptotically.

2) the QP problem for the kernel method is of a special form. In particular, there can be no linear term in the QP's objective function.

The first condition is satisfied for a variety of kernels, including the following:

1) the isotropic kernel $k(\mathbf{x}, \mathbf{y}) = K(||\mathbf{x}-\mathbf{y}||)$ (e.g., Gaussian kernel);
2) the dot product kernel $k(\mathbf{x}, \mathbf{y}) = K(\mathbf{x}'\mathbf{y})$ (e.g., polynomial kernel) with normalized inputs;
3) any normalized kernel $k(\mathbf{x},\mathbf{y})=K(\mathbf{x},\mathbf{y})/\sqrt{K(\mathbf{x},\mathbf{x})K(\mathbf{y},by)}$.

As for the second condition, it has been shown in [18] that this holds for kernel methods including the one-class SVM [19] and two-class SVM.

However, there are still some popular kernel methods that violate these conditions and so cannot be used with the CVM. For example, when the SVM is applied on imbalanced data sets, each class (or sometimes even each individual training pattern) is often penalized differently. As will be shown in Section IV-A, although the resultant QP problem can be rewritten in the required form, the corresponding "kernel" does not satisfy condition 1). Similarly, this is also the case with the ranking SVM commonly used in information retrieval [20] (Section IV-B). Likewise, there are kernel methods whose QPs simply cannot be written into the required form. One such example is SVR [21] as will be discussed in Section III-A.

In this paper, we propose an extension of the CVM that allows for a more general QP formulation. As the CVM is closely related to the MEB problem in computational geometry, the key here is to extend the original MEB problem to what will be called the *center-constrained* MEB problem in Section III-B. The resultant QP problem then allows a linear term in the dual objective. While so far the original CVM has only been shown to be applicable to the one-class and two-class SVMs [18], the proposed relaxed QP formulation allows a significantly larger repertoire of kernel methods to be cast as (center-constrained) MEB problems. Moreover, it turns out that this also allows the (first) condition on the kernel function to be lifted. In other words, the proposed algorithm can now be used with *any* linear/nonlinear kernel.

The rest of this paper is organized as follows. Section II gives a review on the original CVM algorithm and the MEB problem. Section III then describes the proposed extension (generalized CVM), and shows how this can be used to scale up SVR. Section IV provides more kernel method examples, besides SVR, that can also benefit from this extension. These include the SVM for imbalanced data and the ranking SVM [20]. Experimental results are presented in Section V, and Section VI gives some concluding remarks. A preliminary and abridged version (without proofs and details) of this paper, which focused only on extending the CVM to the regression setting, has appeared in [22].

II. Core Vector Machine (CVM)

In this section, we first review the CVM algorithm as proposed in [18] and [23]. The CVM utilizes an approximation algorithm for the MEB problem, which will be briefly introduced in Section II-A. The connection between the MEB problem and kernel methods, particularly the one-class and two-class SVMs, will then be described in Section II-B. Finally, the CVM algorithm and some of its properties are summarized in Section II-C.

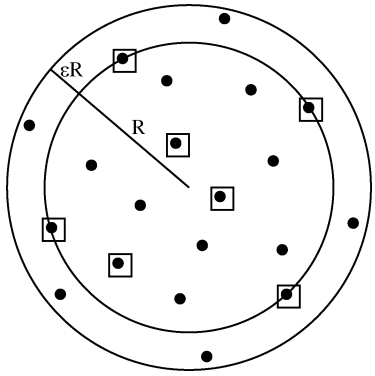

Fig. 1. Inner circle is the MEB of the set of squares and its $(1+\epsilon)$ expansion (the outer circle) covers all the points. The set of squares is a core-set.

A. Approximate Minimum Enclosing Ball

Given a set of points $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where $\mathbf{x}_i \in \mathbb{R}^d$, the minimum enclosing ball of $\mathcal{S}$ [denoted MEB($\mathcal{S}$)] is the smallest ball that contains all the points in $\mathcal{S}$. Traditional algorithms for finding exact MEBs are not efficient for problems with large d (e.g., $d > 30$). Hence, it is of practical interest to study faster *approximation algorithms* that only aim at returning a good approximate solution. Approximation algorithms have long been used in the field of theoretical computer science for tackling computationally difficult problems [24]. For an input size n, an approximation algorithm is said to have an *approximation ratio* $\rho(n)$ if $\max(C/C^*, C^*/C) \leq \rho(n)$, where C is the cost of the solution returned by an approximate algorithm, and C^* is the cost of the optimal solution. Intuitively, this ratio measures how bad the approximate solution is compared with the optimal solution. A large (small) approximation ratio means the solution is much worse than (more or less the same as) the optimal solution. If the ratio does not depend on n, we may just write ρ and call the algorithm an *ρ-approximation algorithm*.

Given an $\epsilon > 0$, a ball[2] $B(\mathbf{c}, (1+\epsilon)R)$ is an *$(1+\epsilon)$-approximation* of MEB($\mathcal{S}$) if $R \leq r_{\mathrm{MEB}(\mathcal{S})}$ and $\mathcal{S} \subset B(\mathbf{c}, (1+\epsilon)R)$. In many shape-fitting problems (which include the MEB problem), it is found that solving the problem on a subset, called the *core-set*, $\mathcal{Q}$ of points from $\mathcal{S}$ can often give an accurate and efficient approximation [25]. More formally, a subset $\mathcal{Q} \subseteq \mathcal{S}$ is a core-set of $\mathcal{S}$ if an expansion by a factor $(1+\epsilon)$ of its MEB contains $\mathcal{S}$, i.e., $\mathcal{S} \subset B(\mathbf{c}, (1+\epsilon)R)$, where $B(\mathbf{c}, R) = \mathrm{MEB}(\mathcal{Q})$ (Fig. 1). A breakthrough on achieving such an $(1+\epsilon)$-approximation was recently obtained by Bădoiu and Clarkson [26]. They used a simple iterative scheme: At the tth iteration, the current estimate $B(\mathbf{c}_t, R_t)$ is expanded by including the furthest point outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$. This is repeated until all the points in $\mathcal{S}$ are covered by $B(\mathbf{c}_t, (1+\epsilon)R_t)$. Despite its simplicity, a surprising property is that the number of iterations, and thus the size of the final core-set, depends only on ϵ but *not* on d or m.

[2]In this paper, we denote the ball with center $\mathbf{c}$ and radius R by $B(\mathbf{c}, R)$. Also, the center and radius of a ball B are denoted by $\mathbf{c}_B$ and r_B, respectively. Moreover, vector/matrix (in both the input and feature spaces) transpose is denoted by the superscript $'$.

B. Kernel Methods as MEB Problems

In [18], we showed that the MEB problem is closely related to many kernel-related problems. In particular, the exact MEB problem is equivalent to the technique of hard-margin support vector data description (SVDD) [27] used in novelty detection. Moreover, it can also be straightforwardly used to find the radius component of the radius-margin bound [7]. Besides these two obvious connections, we showed in [18] that the soft-margin one-class and two-class SVMs can also be viewed as MEB problems, and these will be briefly reviewed in this section.

1) Hard-Margin SVDD: First, consider the hard-margin SVDD. Its primal is

$$\begin{aligned} &\min R^2 \\ &\text{subject to } \|\mathbf{c}-\varphi(\mathbf{x}_i)\|^2 \le R^2, \qquad i=1,\ldots,m \end{aligned} \tag{1}$$

where φ denotes the feature map associated with a given kernel k, and $B(\mathbf{c}, R)$ is the desired MEB in the kernel-induced feature space. Its dual is the QP problem

$$\max\ \boldsymbol{\alpha}'\mathrm{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \tag{2}$$

$$\text{subject to } \boldsymbol{\alpha}'\mathbf{1}=1, \qquad \boldsymbol{\alpha}\ge\mathbf{0} \tag{3}$$

where $\boldsymbol{\alpha} = [\alpha_i,\ldots,\alpha_m]'$ is the vector of Lagrange multipliers, $\mathbf{0} = [0,\ldots,0]'$ and $\mathbf{1} = [1,\ldots,1]'$ are m-vectors, and $\mathbf{K}_{m\times m} = [k(\mathbf{x}_i,\mathbf{x}_j)] = [\varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j)]$ is the corresponding kernel matrix. Assuming that k satisfies

$$k(\mathbf{x},\mathbf{x}) = \kappa \tag{4}$$

a constant, we have $\boldsymbol{\alpha}'\mathrm{diag}(\mathbf{K}) = \kappa$ using the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$ in (3). Dropping this constant term from the objective function in (2), we obtain the simpler QP problem

$$\begin{aligned} &\max\ -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \\ &\text{subject to } \boldsymbol{\alpha}'\mathbf{1}=1, \qquad \boldsymbol{\alpha}\ge\mathbf{0}. \end{aligned} \tag{5}$$

Conversely, whenever the kernel k satisfies (4), any QP problem of the form (5) can be regarded as a MEB problem. As shown in [18], this establishes an important connection between the MEB problem and kernel methods. In Section II-B2, we will consider two such examples, the two-class SVM and the one-class SVM.

2) Two-Class SVM: In a two-class classification problem, denote the training set by $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$, where $y_i \in \{\pm 1\}$ is the class label. The L2-SVM constructs a hyperplane $\mathbf{w}'\varphi(\mathbf{x}) + b$ for which the separation (i.e., the so-called *margin*[3] $\rho/\|\mathbf{w}\|$) between the positive and negative examples is maximized, while at the same time the training errors (represented by the slack variables ξ_i's) is minimized. Formulating as an optimization problem, this is achieved by solving the following primal problem:

$$\begin{aligned} &\min \|\mathbf{w}\|^2 + b^2 - 2\rho + C\sum_{i=1}^m \xi_i^2 \\ &\text{subject to } y_i(\mathbf{w}'\varphi(\mathbf{x}_i)+b) \ge \rho - \xi_i, \qquad i=1,\ldots,m \end{aligned} \tag{6}$$

where $C > 0$ is a parameter that controls the tradeoff between the margin and training error. Using the standard method of Lagrange multipliers, its dual can be obtained as

$$\begin{aligned} &\max\ -\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \\ &\text{subject to } \boldsymbol{\alpha}'\mathbf{1}=1, \qquad \boldsymbol{\alpha}\ge\mathbf{0} \end{aligned} \tag{7}$$

where $\tilde{\mathbf{K}} = [y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) + y_i y_j + \delta_{ij}/C]$ is an $m\times m$ matrix and $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Obviously, $\tilde{\mathbf{K}}$ can be regarded as the kernel matrix of the modified kernel function $\tilde{k}(\mathbf{z}_i,\mathbf{z}_j) = y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) + y_i y_j + \delta_{ij}/C$ defined on the m training examples. If k satisfies (4), then $\tilde{k}$ satisfies $\tilde{k}(\mathbf{z},\mathbf{z}) = \kappa + 1 + 1/C$, a constant. Hence, the two-class L2-SVM corresponds to a MEB problem.

3) One-Class SVM: Another example discussed in [18] is the one-class L2-SVM, which has the dual

$$\begin{aligned} &\max\ -\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \\ &\text{subject to } \boldsymbol{\alpha}'\mathbf{1}=1, \qquad \boldsymbol{\alpha}\ge\mathbf{0} \end{aligned} \tag{8}$$

where $\tilde{\mathbf{K}} = \tilde{k}(\mathbf{x}_i,\mathbf{x}_j) = [k(\mathbf{x}_i,\mathbf{x}_j) + \delta_{ij}/C]$. Again, if the original kernel function k satisfies (4), we have $\tilde{k}(\mathbf{x},\mathbf{x}) = \kappa + 1/C$, a constant. Thus, the one-class L2-SVM can also be regarded as a MEB problem.

Note that the two-norm error has been used. This allows a soft-margin L2-SVM to be transformed to a hard-margin one and so the MEB algorithm does not need to consider outliers. Fitting the MEB with outliers is possible [28], though a major limitation is that the number of outliers has to be moderately small in order for the fitting to be computationally effective. In theory, the use of the two-norm error could be less robust in the presence of outliers. However, experimentally, its generalization performance is often comparable to that of the L1-SVM [18], [29].

C. The CVM Algorithm

The CVM algorithm is shown in Algorithm 1. Here, the coreset, the ball's center, and radius at the tth iteration are denoted by $\mathcal{S}_t$, $\mathbf{c}_t$, and R_t, respectively. Moreover, $\tilde{\varphi}$ denotes the feature map corresponding to the transformed kernel $\tilde{k}$. Similar to many SVM solvers such as SMO, the CVM algorithm requires the input of a termination parameter ϵ. Note that while Steps 2) and 3) appear to make use of $\tilde{\varphi}(\mathbf{z})$ in the (transformed) feature space which is possibly infinite-dimensional, we never need to explicitly compute $\tilde{\varphi}(\mathbf{z})$ in practice thanks to the kernel trick. For details of the algorithm, interested readers are referred to [18]. A nice property of the CVM is that its approximation ratio can be obtained. In particular, it can be shown that when $\epsilon = 0$,

[3]The standard SVM has a margin of $2/\|\mathbf{w}\|$, i.e., ρ is equal to 2. In [18], we have slightly altered its primal to (6), so that the resultant dual is of the desired form. Note that a large margin is still favored as in the original SVM, as we minimize $\|\mathbf{w}\|^2$ and maximize ρ in (6).

CVM outputs the exact solution of the kernel problem. When $\epsilon > 0$, it becomes an $(1+\epsilon)^2$-approximation algorithm.

Algorithm 1 CVM **[18]**.

Step 1): Initialize $\mathcal{S}_0$, $\mathbf{c}_0$, and R_0.

Step 2): Terminate if there is no training point $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ falls outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$.

Step 3): Find (core vector) $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ is furthest away from $\mathbf{c}_t$. Set $S_{t+1} = \mathcal{S}_t \cup \{\mathbf{z}\}$. This can be made more efficient by using the probabilistic speedup method in **[4]** that finds a $\mathbf{z}$ which is only approximately the furthest.

Step 4): Find the new $\mathrm{MEB}(\mathcal{S}_{t+1})$ and set $\mathbf{c}_{t+1} = \mathbf{c}_{\mathrm{MEB}(\mathcal{S}_{t+1})}$ and $R_{t+1} = r_{\mathrm{MEB}(\mathcal{S}_{t+1})}$.

Step 5): Increment t by 1 and go back to Step 2).

III. The Generalized CVM Algorithm

Although the CVM allows some kernel methods, such as the two-class SVM, to be run on very large data sets with great success [18], its applicability to other kernel methods is limited by the following two requirements:

1) $k(\mathbf{x}, \mathbf{x})$ is a constant for any pattern $\mathbf{x}$;
2) the QP problem is of the form in (5).

As mentioned in Section I, unfortunately there are some popular kernel methods that do not satisfy these requirements. As an example, we will demonstrate in Section III-A that the dual objective of SVR contains a linear term and thus violates requirement 2). In Section III-B, we propose an extension of the MEB problem, and consequently a generalized CVM algorithm, that allows for the inclusion of a linear term in the QP problem. It turns out that this has the added benefit of allowing requirement 1) on the kernel function to be lifted.

A. Motivating Example: L2-SVR

In this section, we show that SVR cannot benefit from the original CVM algorithm. Recall that in a regression problem, we are given a training set $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$ with input $\mathbf{x}_i$ and output $y_i \in \mathbb{R}$. SVR constructs a linear function $f(\mathbf{x}) = \mathbf{w}'\varphi(\mathbf{x}) + b$ in the kernel-induced feature space so that it deviates least from the training data according to the $\bar{\varepsilon}$-insensitive loss function[4]

$$|y - f(\mathbf{x})|_{\bar{\varepsilon}} = \begin{cases} 0, & \text{if } |y - f(\mathbf{x})| \le \bar{\varepsilon} \\ |y - f(\mathbf{x})| - \bar{\varepsilon}, & \text{otherwise} \end{cases} \tag{9}$$

while at the same time is as "flat" as possible (i.e., $\|\mathbf{w}\|$ is as small as possible). Using slack variables ξ_i's and ξ_i^*'s, we adopt the following primal which is similar to that of ν-SVR [1]:

$$\begin{aligned} \min \|\mathbf{w}\|^2 + b^2 + \frac{C}{\mu m}\sum_{i=1}^m (\xi_i^2 + \xi_i^{*2}) + 2C\bar{\varepsilon} \\ \text{subject to } y_i - (\mathbf{w}'\varphi(\mathbf{x}_i) + b) \le \bar{\varepsilon} + \xi_i \\ (\mathbf{w}'\varphi(\mathbf{x}_i) + b) - y_i \le \bar{\varepsilon} + \xi_i^* \end{aligned} \tag{10}$$

[4]To avoid confusion with the ϵ in $(1+\epsilon)$-approximation, we add a bar to the ε in the ε-insensitive loss function.

for $i = 1, \ldots, m$. Here, $\mu > 0$ is a parameter (which is analogous to the ν parameter in ν-SVR) that controls the size of $\bar{\varepsilon}$. Also, as in Section II, the bias b is penalized and the two-norm errors (ξ_i^2 and ξ_i^{*2}) are used. Note that the constraints $\xi_i, \xi_i^* \ge 0$ are automatically satisfied. Standard application of the method of Lagrange multipliers yields the dual

$$\begin{aligned} \max \begin{bmatrix} \boldsymbol{\lambda}' & \boldsymbol{\lambda}^{*\prime} \end{bmatrix} \begin{bmatrix} \frac{2}{C}\mathbf{y} \\ -\frac{2}{C}\mathbf{y} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\lambda}' & \boldsymbol{\lambda}^{*\prime} \end{bmatrix} \tilde{\mathbf{K}} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^* \end{bmatrix} \\ \text{subject to } \begin{bmatrix} \boldsymbol{\lambda}' & \boldsymbol{\lambda}^{*\prime} \end{bmatrix} \mathbf{1} = 1, \qquad \boldsymbol{\lambda}, \boldsymbol{\lambda}^* \ge \mathbf{0} \end{aligned} \tag{11}$$

where $\mathbf{y} = [y_1, \ldots, y_m]'$, and $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_m]'$, $\boldsymbol{\lambda}^* = [\lambda_1^*, \ldots, \lambda_m^*]'$ are the $2m$ Lagrange multipliers (dual variables), and

$$\tilde{\mathbf{K}} = [\tilde{k}(\mathbf{z}_i, \mathbf{z}_j)] = \begin{bmatrix} \mathbf{K} + \mathbf{1}\mathbf{1}' + \frac{\mu m}{C}\mathbf{I} & -(\mathbf{K} + \mathbf{1}\mathbf{1}') \\ -(\mathbf{K} + \mathbf{1}\mathbf{1}') & \mathbf{K} + \mathbf{1}\mathbf{1}' + \frac{\mu m}{C}\mathbf{I} \end{bmatrix} \tag{12}$$

is a 2m×2m "kernel matrix."

After solving for the dual variables $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^*$, the primal variables can be recovered as

$$\mathbf{w} = C\sum_{i=1}^m (\lambda_i - \lambda_i^*)\varphi(\mathbf{x}_i) \quad b = C\sum_{i=1}^m (\lambda_i - \lambda_i^*) \tag{13}$$

$$\xi_i = \lambda_i \mu m \quad \xi_i^* = \lambda_i^* \mu m \tag{14}$$

and

$$\bar{\varepsilon} = \begin{bmatrix} \boldsymbol{\lambda}' & \boldsymbol{\lambda}^{*\prime} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \end{bmatrix} - C \begin{bmatrix} \boldsymbol{\lambda}' & \boldsymbol{\lambda}^{*\prime} \end{bmatrix} \tilde{\mathbf{K}} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^* \end{bmatrix}. \tag{15}$$

By plugging the constraint $\sum_{i=1}^m (\lambda_i + \lambda_i^*) = 1$ in (11) into (14), we obtain

$$\mu = \frac{1}{m}\sum_{i=1}^m (\xi_i + \xi_i^*).$$

Recall that ξ_i and ξ_i^* are slack variables for the regression error $|y_i - f(\mathbf{x}_i)|_{\bar{\varepsilon}}$. Thus, the μ parameter, analogous to the ν parameter in ν-SVR, can be interpreted as the expected error $\mathbf{E}(|y - f(\mathbf{x})|_{\bar{\varepsilon}})$.

Note that although (11) is a standard QP problem, it is not of the required form in (5) because of the presence of the linear term. Consequently, the original CVM algorithm cannot be applied. In Section III-B, we will see how the algorithm can be extended to cover this case.

B. The Center-Constrained MEB Problem

The MEB problem in (1) finds the smallest ball containing all $\varphi(\mathbf{x}_i) \in \mathcal{S}$ in the feature space. In this section, we first augment an extra $\delta_i \in \mathbb{R}$ to each $\varphi(\mathbf{x}_i)$, forming $\begin{bmatrix} \varphi(\mathbf{x}_i) \\ \delta_i \end{bmatrix}$. Then, we find the MEB for these augmented points, while at the same time constraining the last coordinate of the ball's center to be zero (i.e., of the form $\begin{bmatrix} \mathbf{c} \\ 0 \end{bmatrix}$) (Fig. 2). The primal in (1) is thus changed to

$$\begin{aligned} \min R^2 \\ \text{subject to } \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 + \delta_i^2 \le R^2, \qquad i = 1, \ldots, m. \end{aligned} \tag{16}$$

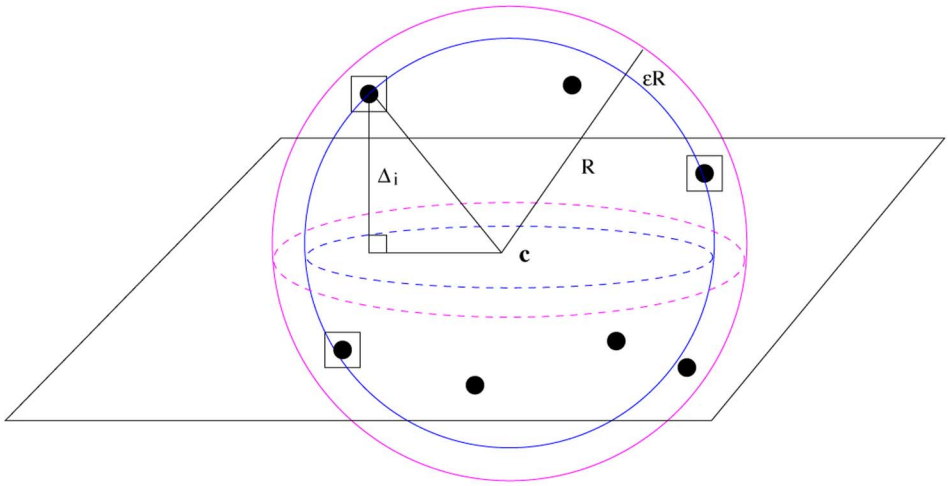

Fig. 2. Center-constrained MEB problem.

It can be easily shown that the corresponding dual is again a QP problem

$$\begin{aligned} &\max \boldsymbol{\alpha}'(\mathrm{diag}(\mathbf{K}) + \boldsymbol{\Delta}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \\ &\text{subject to } \boldsymbol{\alpha}'\mathbf{1} = 1, \qquad \boldsymbol{\alpha} \geq \mathbf{0} \end{aligned} \tag{17}$$

where

$$\boldsymbol{\Delta} = [\delta_1^2, \ldots, \delta_m^2]' \geq \mathbf{0}. \tag{18}$$

From the optimal $\boldsymbol{\alpha}$ solution, we can recover R and $\mathbf{c}$ as

$$R = \sqrt{\boldsymbol{\alpha}'(\mathrm{diag}(\mathbf{K}) + \boldsymbol{\Delta}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}} \quad \mathbf{c} = \sum_{i=1}^{m} \alpha_i \varphi(\mathbf{x}_i). \tag{19}$$

Besides, the squared distance between the center $\begin{bmatrix} \mathbf{c} \\ 0 \end{bmatrix}$ and any point $\begin{bmatrix} \varphi(\mathbf{x}_\ell) \\ \delta_\ell \end{bmatrix}$ is

$$\|\mathbf{c} - \varphi(\mathbf{x}_\ell)\|^2 + \delta_\ell^2 = \|\mathbf{c}\|^2 - 2(\mathbf{K}\boldsymbol{\alpha})_\ell + k_{\ell\ell} + \delta_\ell^2 \tag{20}$$

which does not depend explicitly on the feature map φ.

Because of the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$ in (17), an arbitrary multiple of $\boldsymbol{\alpha}'\mathbf{1}$ can be added to the objective without affecting its $\boldsymbol{\alpha}$ solution. In other words, for an arbitrary $\eta \in \mathbb{R}$, (17) yields the same optimal $\boldsymbol{\alpha}$ as

$$\begin{aligned} &\max \boldsymbol{\alpha}'(\mathrm{diag}(\mathbf{K}) + \boldsymbol{\Delta} - \eta\mathbf{1}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \\ &\text{subject to } \boldsymbol{\alpha}'\mathbf{1} = 1, \qquad \boldsymbol{\alpha} \geq \mathbf{0}. \end{aligned} \tag{21}$$

Using the same argument as in Section II, any QP problem of the form (21), with $\boldsymbol{\Delta} \geq \mathbf{0}$, can also be regarded as a MEB problem (16). Note that (21) allows a linear term in the objective. Besides, unlike that in Section II, it does not require the kernel to satisfy condition (4).

Returning to the QP problem in (11), define $\tilde{\boldsymbol{\alpha}} = [\tilde{\alpha}_1, \ldots, \tilde{\alpha}_{2m}]' = [\boldsymbol{\lambda}' \quad \boldsymbol{\lambda}^{*\prime}]'$ and

$$\boldsymbol{\Delta} = -\mathrm{diag}(\tilde{\mathbf{K}}) + \eta\mathbf{1} + \frac{2}{C}\begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \end{bmatrix} \tag{22}$$

for η large enough such that $\boldsymbol{\Delta} \geq \mathbf{0}$, (11) can then be written as

$$\begin{aligned} &\max \tilde{\boldsymbol{\alpha}}'(\mathrm{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta} - \eta\mathbf{1}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} \\ &\text{subject to } \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1, \qquad \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}. \end{aligned}$$

This is thus of the form in (21). In other words, L2-SVR can now be regarded as a center-constrained MEB problem (16).

Recall that this QP problem involves $2m$ variables. Thus, in the equivalent MEB problem, there are also $2m$ points in the feature space. Indeed, from (12), each training pattern $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ $(i = 1, \ldots, m)$ in the regression problem can be seen to correspond to the two points

$$\begin{bmatrix} \tilde{\varphi}(\mathbf{z}_i) \\ \delta_i \end{bmatrix} = \begin{bmatrix} \varphi(\mathbf{x}_i) \\ 1 \\ \sqrt{\frac{\mu m}{C}}\mathbf{e}_i \\ \delta_i \end{bmatrix}$$

and

$$\begin{bmatrix} \tilde{\varphi}(\mathbf{z}_{i+m}) \\ \delta_{i+m} \end{bmatrix} = \begin{bmatrix} -\varphi(\mathbf{x}_i) \\ -1 \\ \sqrt{\frac{\mu m}{C}}\mathbf{e}_{i+m} \\ \delta_{i+m} \end{bmatrix}$$

where $\mathbf{e}_i$ is the $2m$-dimensional vector with all zeroes except that the ith position is equal to one, and δ_i^2 is the ith component of $\boldsymbol{\Delta}$ as defined in (18) and (22). The CVM algorithm in Section II-C also has to be modified slightly. The only modification is to use (20) for the distance computation between the MEB's center $\begin{bmatrix} \mathbf{c}_t \\ 0 \end{bmatrix}$ and any point $\begin{bmatrix} \tilde{\varphi}(\mathbf{z}_\ell) \\ \delta_\ell \end{bmatrix}$ in Steps 2) and 3).

This extension is also useful in training the one-class and two-class L2-SVMs in Section II-B when the kernel does not satisfy (4). By defining

$$\begin{aligned} \boldsymbol{\Delta} &= \max_i\{\tilde{\mathbf{K}}_{ii}\}\mathbf{1} - \mathrm{diag}(\tilde{\mathbf{K}}) \geq \mathbf{0} \\ \eta &= \max_i\{\tilde{\mathbf{K}}_{ii}\} \end{aligned}$$

the linear term in the objective function of (21) disappears. Both duals (7) and (8) are then of the form in (21) and thus both SVMs are also center-constrained MEB problems.

C. Properties

As the proposed algorithm is an extension of the CVM, one would naturally expect its properties to be analogous to those of the original CVM in [18]. In this section, we confirm this by following the proof procedure in [18]. Note, however, that the detailed proofs (which are in the Appendix) are different because the QP formulations associated with the generalized CVM are much more complicated.

1) Bound on the Number of Iterations: To ensure good scaling behavior of the CVM, a key property is that it converges in at most $2/\epsilon$ iterations, independent of the dimensionality in the kernel-induced feature space and the size of $\mathcal{S}$ [26]. Theorem 1 shows that this still holds for the generalized CVM.

Theorem 1: There exists a set $\mathcal{S}_t \subset \mathcal{S}$ of size $2/\epsilon$ such that the distance between $\mathbf{c}_{\mathrm{MEB}(\mathcal{S})}$ and any point $\mathbf{z}_i$ of $\mathcal{S}$ is at most $(1+\epsilon)r_{\mathrm{MEB}(\mathcal{S})}$.

As a direct consequence of Theorem 1, the generalized CVM algorithm inherits from the original CVM algorithm its low time and space complexities. In particular, when probabilistic speedup is not used in Step 3), the overall time for $\tau = O(1/\epsilon)$ iterations can be shown to be $O\left(m/\epsilon^2 + 1/\epsilon^4\right)$, linear in m for a fixed ϵ. With the use of probabilistic speedup, this can be further reduced to $O\left(1/\epsilon^8\right)$, which is independent of m for a fixed ϵ. As for the space complexity, the whole algorithm requires only $O(1/\epsilon^2)$ space, independent of m for a fixed ϵ (here, we have ignored the $O(m)$ space required for storing the m training patterns, as they may be stored outside the core memory). Detailed derivations are the same as for the CVM and can be found in [18].

2) Convergence to (Approximate) Optimality: Points outside $\mathrm{MEB}(\mathcal{S}_t)$ have zero α_i's and so violate the KKT conditions of the dual problem. As in the original CVM algorithm, we also have the following properties.

3) Property 1: Choosing the point furthest away from the current center in Step 3) is the same as choosing the worst violating pattern corresponding to the KKT constraint.

The proof is very similar to that in [18] and so is skipped here. Moreover, when the generalized CVM is used in the regression setting (Section III-A), we have Property 2.

4) Property 2: In the regression setting, all the training patterns satisfy loose KKT conditions when the generalized CVM terminates.

As for the original CVM algorithm, this generalized version is also an $(1+\epsilon)^2$-approximation algorithm.

5) Property 3: When $\epsilon = 0$, the generalized CVM algorithm outputs the exact solution of the kernel problem. When $\epsilon > 0$ and it terminates at the τth iteration, the optimal primal objective p^* of the kernel problem in (10) satisfies $\max\left(R_\tau^2/(p^*/C^2+\eta), (p^*/C^2+\eta)/R_\tau^2\right) \le (1+\epsilon)^2$.

IV. Example Applications to Other Kernel Methods

In [18], we showed that the one-class SVM (for novelty detection) and the two-class SVM (for classification) are MEB problems. By introducing the center-constrained MEB problem, we showed in Section III-B that SVR (for regression) is also a MEB problem. In this section, we give some more examples that can also be regarded as (center-constrained) MEB problems. This significantly extends the repertoire of kernel methods that can be used with the (generalized) CVM. Of course, this is not intended to cover the full list of such kernel methods. Moreover, in using the CVM algorithm, the primal formulations of the corresponding kernel methods often have to be changed slightly. In particular, as in the previous sections, we will adopt the two-norm error instead of the more commonly used one-norm error.

A. Two-Class SVM for Imbalanced Data

In Section II-B, we considered the two-class L2-SVM where all the slack variables ξ_i's receive the same penalty factor C. However, many real-world data sets are imbalanced and the majority class has much more training patterns than the minority class. As a result, the resultant hyperplane will be shifted towards the majority class. A common remedy is to use different C's for the two classes. In the more general case, each training pattern can have its own penalty factor C_i. The primal in (6) is then changed to

$$\begin{aligned} &\min \|\mathbf{w}\|^2 + b^2 - 2\rho + \sum_{i=1}^{m} C_i \xi_i^2 \\ &\text{subject to } \; y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \ge \rho - \xi_i, \qquad i = 1, \ldots, m \end{aligned}$$

and the corresponding dual is

$$\begin{aligned} &\max \; -\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \\ &\text{subject to } \; \boldsymbol{\alpha}'\mathbf{1} = 1, \qquad \boldsymbol{\alpha} \ge \mathbf{0} \end{aligned} \tag{23}$$

where $\tilde{\mathbf{K}} = [y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \delta_{ij}/C_i]$. Note that as the C_i's are different in general, the diagonal entry $\tilde{\mathbf{K}}_{ii} = k(\mathbf{x}_i, \mathbf{x}_i) + 1 + 1/C_i$ is not constant even if $k(\mathbf{x}_i, \mathbf{x}_i)$ is. Hence, although (23) is of the form in (5), the underlying kernel does not satisfy condition (4). In other words, the original CVM cannot be applied but the generalized formulation, which does not require (4), can still be used.

B. Ranking SVM

A kernel method commonly used in the field of information retrieval is the ranking SVM [20]. Let d_i's be documents in the database and q_k be the kth query in a set of queries Q. The match between query q_k and document d_i is described by a feature map $\varphi(q_k, d_i)$. As an example, $\varphi(q_k, d_i)$ in [20] involves features such as the cosine between d_i and q_k, the cosine between q_k and the words in d_i's title, etc. The desired document ranking for query q_k is represented by a set of document pairs $r_k^* = \{(d_i, d_j)\}$, where $(d_i, d_j) \in r_k^*$ means that document d_i should be ranked higher than document d_j according to the query q_k.

The ranking SVM tries to find a linear ranking function $\mathbf{w}'\varphi(q, d)$ such that $\mathbf{w}'\varphi(q_k, d_i) > \mathbf{w}'\varphi(q_k, d_j)$ for $(d_i, d_j) \in r_k^*$. As for the standard SVM, this condition may be violated and so a slack variable ξ_{ijk} is introduced for the soft constraint associated with query q_k and document pair $(d_i, d_j) \in r_k^*$. Again using the two-norm error, we write the primal as

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{C}{2} \sum_{k:q_k \in Q} \sum_{i,j:(d_i,d_j)\in r_k^*} \xi_{ijk}^2 \\ &\text{subject to } \; \mathbf{w}'(\varphi(q_k, d_i) - \varphi(q_k, d_j)) \ge \rho - \xi_{ijk}, \;\; \xi_{ijk} \ge 0 \\ &\qquad \forall (d_i, d_j) \in r_k^* \qquad \forall q_k \in Q. \end{aligned}$$

TABLE I
BENCHMARK DATA SETS USED

date set	max # training patterns	# validation patterns	# test patterns	# attributes
census housing	18,186	2,273	2,273	121
computer activity	6,554	819	819	21
elevators	13,280	1,660	1,660	18
friedman	200,000	10,000	10,000	10
pole telecomm	12,000	1,500	1,500	48

There are a total of $M \equiv \prod_{k:q_k \in Q} |r_k^*|$ constraints (where $|r_k^*|$ denotes the size of the set r_k^*). Denote the corresponding M-dimensional vector of Lagrange multipliers by $\boldsymbol{\alpha} = [\alpha_{ijk}]$. It can be easily shown that the corresponding dual is

$$\max \; -\frac{1}{2}\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \quad \text{subject to } \boldsymbol{\alpha}'\mathbf{1} = 1, \qquad \boldsymbol{\alpha} \geq \mathbf{0} \tag{24}$$

where $\tilde{\mathbf{K}} = [\tilde{\mathbf{K}}_{ijk,\overline{ijk}}]$ is the $M \times M$ "kernel matrix" with

$$\tilde{\mathbf{K}}_{ijk,\overline{ijk}} = (\varphi(q_k, d_i) - \varphi(q_k, d_j))'(\varphi(q_{\overline{k}}, d_{\overline{i}}) - \varphi(q_{\overline{k}}, d_{\overline{j}})) + \frac{1}{C}\delta_{ijk,\overline{ijk}}.$$

Clearly, the diagonal entry of $\tilde{\mathbf{K}}$

$$\tilde{\mathbf{K}}_{ijk,ijk} = \|\varphi(q_k, d_i) - \varphi(q_k, d_j)\|^2 + \frac{1}{C}$$

is not constant for all (ijk)-triples in general. Hence, (24) is of the form in (5) but the underlying kernel does not satisfy (4), and the generalized formulation in Section III-B again comes to help.

V. EXPERIMENTS

In this section, we use the proposed generalized CVM to implement SVR (Section III-A) and the two-class SVM on imbalanced data (Section IV-A). Our implementation[5] is based on the previous CVM implementation described in [18], which, in turn, is adapted from the LIBSVM software [8]. SMO is used in solving each QP subproblem in Step 4). For simplicity, shrinking [30] is not used in our current implementation. As in LIBSVM, our implementation uses caching and stores all the training patterns in main memory. Besides, we employ the probabilistic speedup method in Step 3). Following [4], a random sample of size 59 is used. The value of ϵ is fixed at 10^{-6} in all the experiments. As in other decomposition methods, the use of a very stringent stopping criterion is not necessary in practice. Our experience on the CVM indicates that $\epsilon = 10^{-6}$ is acceptable for most tasks. Using an even smaller ϵ does not show improved generalization performance, but may increase the training time unnecessarily. All the experiments are performed on a 3.2 GHz Pentium-4 machine with 512 M RAM, running Windows XP.

A. Core Vector Regression

In this section, the following large benchmark regression data sets are used[6] (Table I).

1) Census housing: The task is to predict the median price of the house based on certain demographic information. Following [31], we use 121 features for prediction.
2) Computer activity: Given a number of computer systems activity measures, the task is to predict the portion of time CPUs run in user mode.
3) Elevators: The task is related to an action taken on the elevators of an F16 aircraft.
4) Friedman: This is an artificial data set used in [32]. The input attributes $(x_1, \ldots, x_{10})$ are generated independently, each of which uniformly distributed over [0, 1]. The target is defined by

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0,1) \tag{25}$$

where $\sigma(0,1)$ is the noise term which is normally distributed with mean 0 and variance 1. Note that only $x_1, x_2, \ldots, x_5$ are used in (25) while $x_6, x_7, \ldots, x_{10}$ are noisy irrelevant input attributes.
5) Pole Telecomm: The data describes a telecommunication problem. Details are not available.

Because the Friedman data is a toy problem, we have the liberty of generating a larger training set than the one provided on the web so as to better study the scaling behavior. In the experiment, we use a maximum of 200 000 patterns for training, and generate another 10 000 patterns for validation and another 10 000 patterns for testing. For the other real-world data sets, 10%–80% of the provided set is used for training, another 10% for validation, and the remaining 10% for testing. Our implementation will be denoted CVR in this particular context. For comparison, we also run the latest[7] versions of two state-of-the-art SVR implementations: LIBSVM (version 2.8) and SVM$^{\text{light}}$ (version 6.01).[8] All implementations are in C++. We use the Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/\beta)$, with $\beta = (1/m^2)\sum_{i,j=1}^{m} \|\mathbf{x}_i - \mathbf{x}_j\|^2$. The C and μ parameters in (10) are determined by using a validation set. Parameters in the LIBSVM and SVM$^{\text{light}}$ implementations are tuned in a similar manner.

The following two criteria are used for evaluating the generalization performance based on a test set of n patterns:

1) root-mean-square error

$$\text{RMSE} = \frac{1}{\max_i y_i}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2};$$

[5]It can be downloaded from http://www.cs.ust.hk/~ivor/cvm.html

[6]The census housing data set is from http://www.cs.toronto.edu/~delve/data/census–house/desc.html, while others are from http://www.niaad.liacc.up.pt/~ltorgo/Regression.

[7]Similar results have been reported in [22] based on the older version (2.7) of LIBSVM. The results there show the same trend, though are not exactly identical, as those reported here. Recall that our CVR implementation is adapted from the LIBSVM, and so it is also reimplemented with the latest version (2.8) of the LIBSVM.

[8]LIBSVM and SVM$^{\text{light}}$ can be downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvm/ and http://svmlight.joachims.org/, respectively.

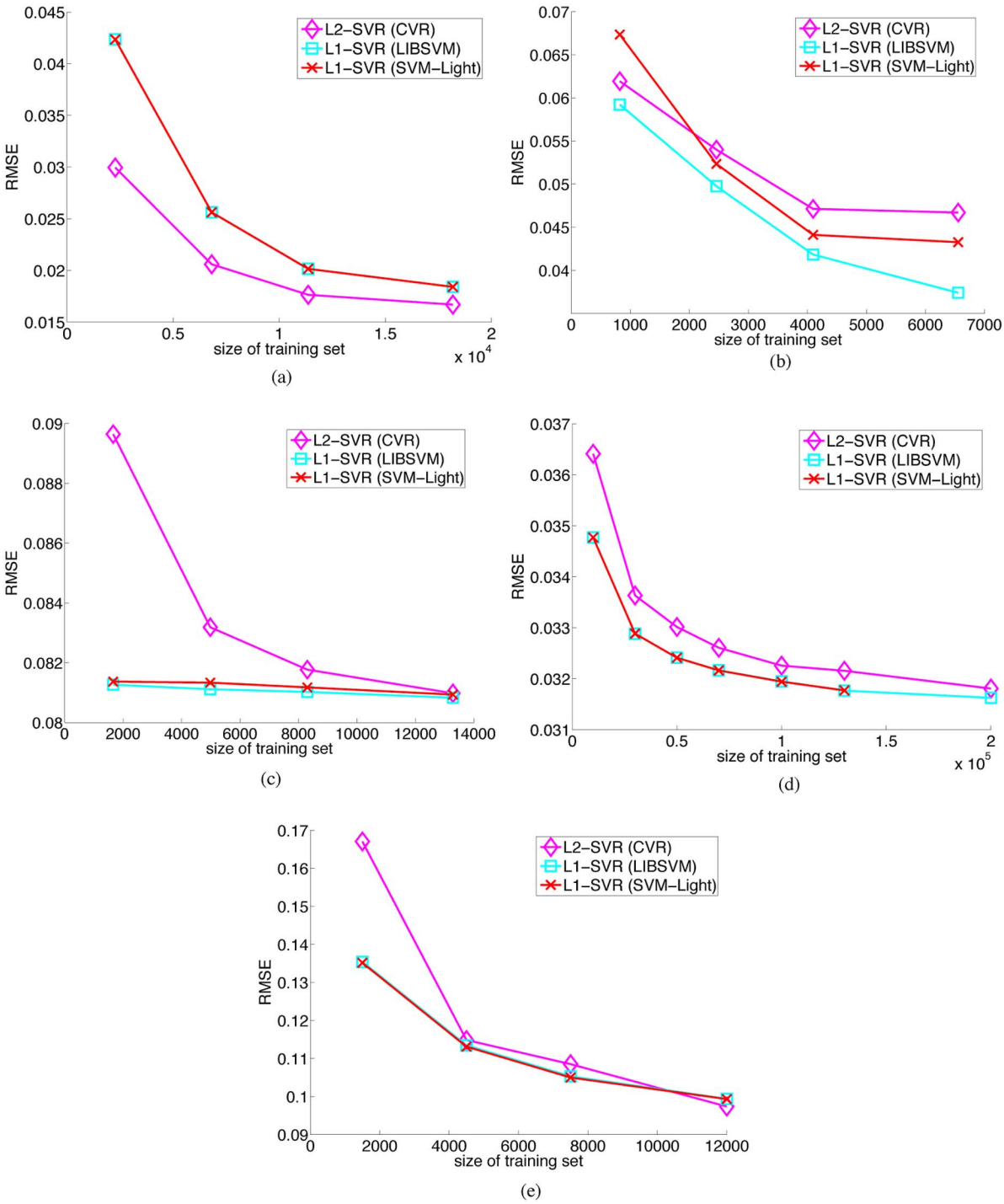

Fig. 3. RMSEs on the five data sets. (a) Census housing. (b) Computer activity. (c) Elevators. (d) Friedman. (e) Pole telecomm.

2) mean relative error

$$\text{MRE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{f(\mathbf{x}_i) - y_i}{y_i}\right|.$$

Figs. 3 and 4 show the RMSEs and MREs, respectively, of the different implementations on the five data sets. As can be seen, although CVR uses the L2-norm error, its generalization performance is comparable to those of the L1-norm-based implementations. CVR is only noticeably inferior when the data set is small. However, as we advocate the use of CVR only on very large data sets, this undesirable effect on small data sets is not relevant. Note also that SVM$^{\text{light}}$ has to be terminated early on the friedman data set because of its excessive training time with 200 000 training patterns.

Fig. 5 compares the training speeds of the various implementations. As can be seen, CVR is much faster (with the only exception on the elevator data set) than the two state-of-the-art solvers, especially when the training set is large. This is also in

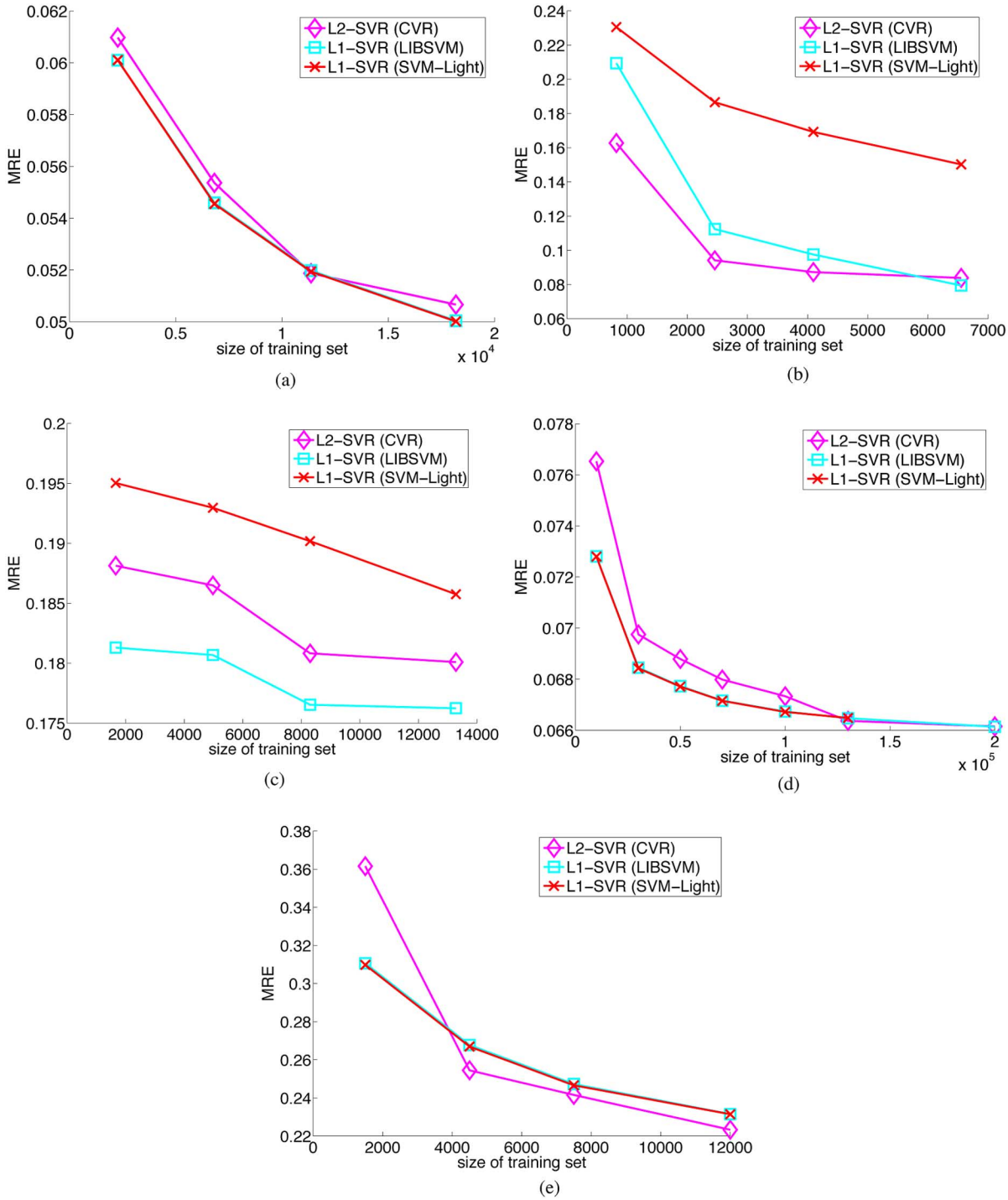

Fig. 4. MREs on the five data sets. (a) Census housing. (b) Computer activity. (c) Elevators. (d) Friedman. (e) Pole telecomm.

line with our experience with the CVM on classification problems [18]. On the elevator data set, CVR is only slightly slower than the LIBSVM implementation when the full training set is used [Fig. 5(c)]. By examining the growth rates of the curves in Fig. 5(c), we expect CVR to be faster than LIBSVM if the training set were larger.

Finally, Fig. 6 compares the numbers of support vectors obtained after the training process. As can be seen, CVR produces far fewer support vectors than the other implementations on all the data sets used in the experiment. Subsequently, CVR is also much faster on testing as the number of kernel evaluations between the test patterns and the support vectors has been significantly reduced.

B. Two-Class SVM for Imbalanced Data

In this section, we use the generalized CVM to implement the two-class SVM in an imbalanced data setting. This will be studied in the context of face detection with an extended version of the Massachusetts Institute of Technology (MIT), Cambridge, face database[9] developed in [33]. The original MIT data

[9]http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html

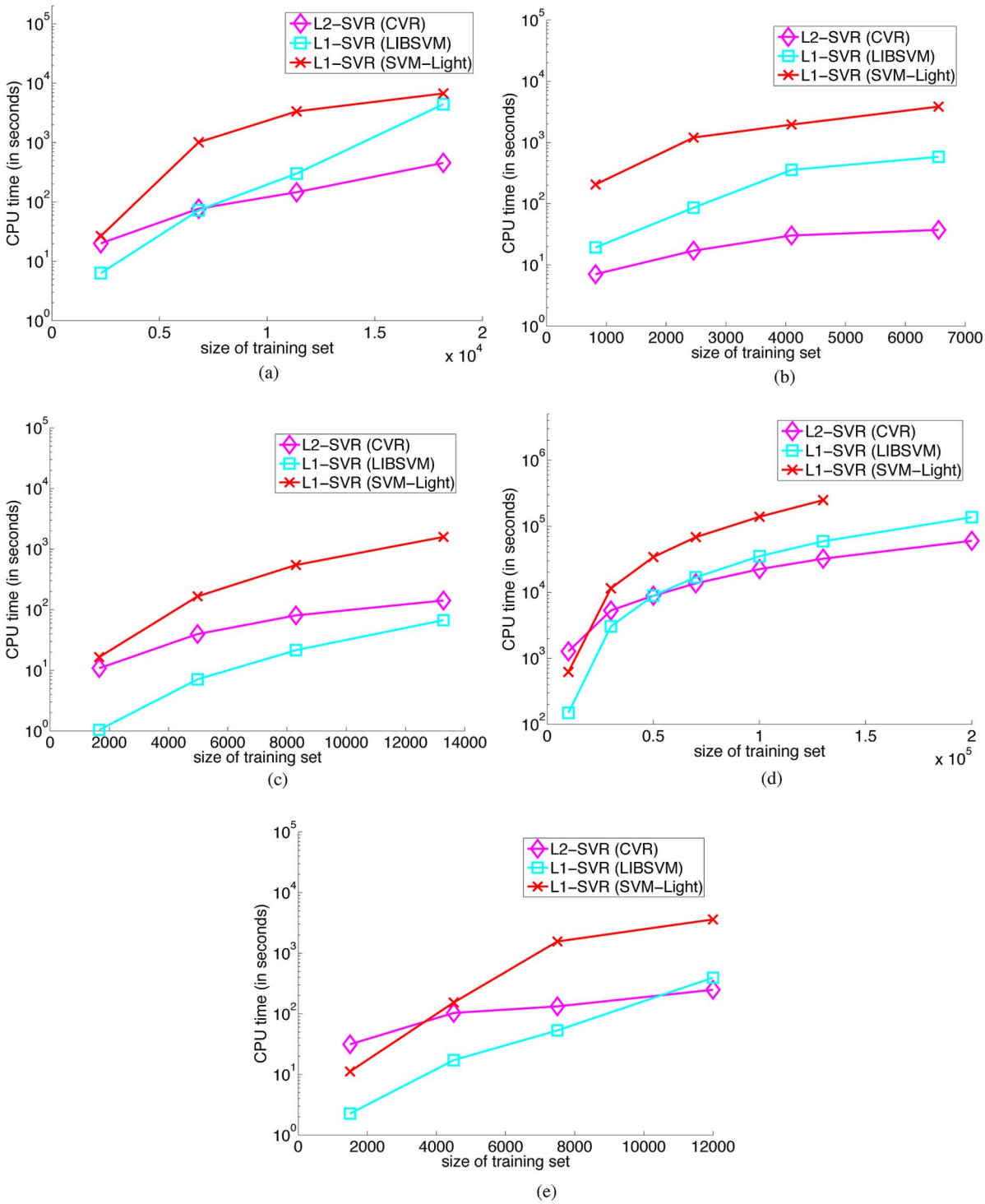

Fig. 5. Training time (in seconds) on the five data sets, shown in log scale. (a) Census housing. (b) Computer activity. (c) Elevators. (d) Friedman. (e) Pole telecomm.

set has 6977 training images (with 2429 faces and 4548 nonfaces) and 24 045 test images (472 faces and 23 573 nonfaces). The training set is enlarged by generating artificial samples (details are in [18]). Finally, three training sets are created and their data characteristics are summarized in Table II. Different Cs are used for these two classes, which are determined by using an independent validation set with 242 faces and 10 000 nonfaces. In this experiment, faces are treated as positives while nonfaces as negatives. Moreover, each image is of size 19×19, leading to a 361-dimensional feature vector which is typically dense. Hence, instead of using a sparse format for the features as in the original LIBSVM, here we have used a dense data format in all the implementations.

Because of the imbalanced nature of this data set, the usual classification error is inappropriate for performance evaluation here. Instead, the following performance measures will be used.

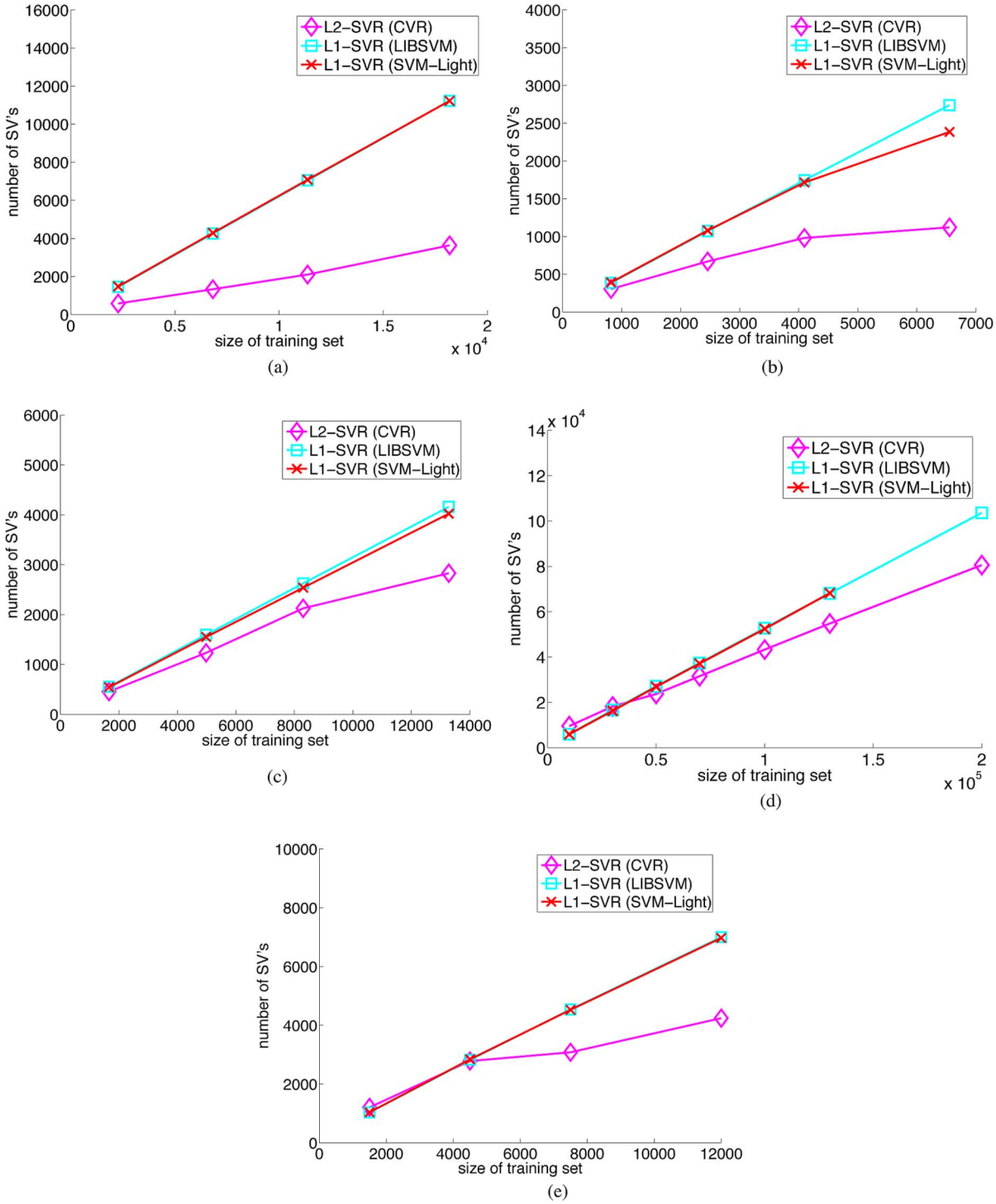

Fig. 6. Numbers of support vectors obtained on the five data sets. (a) Census housing. (b) Computer activity. (c) Elevators. (d) Friedman. (e) Pole telecomm.

TABLE II
NUMBER OF FACES AND NONFACES IN THE FACE DETECTION DATA SETS

training set	# faces	# nonfaces	total
original	2,187	4,548	6,735
set A	2,187	471,914	474,101
set B	17,496	471,914	489,410
set C	367,416	471,914	839,330

1) The balanced loss [34]

$$\ell_{\text{bal}} = 1 - \frac{\text{TP} + \text{TN}}{2}$$

where

$$\text{TP} = \frac{\#\text{positives correctly classified}}{\#\text{total positives}}$$

$$\text{TN} = \frac{\#\text{negatives correctly classified}}{\#\text{total negatives}}$$

are the true positive and true negative rates, respectively.

2) The AUC (area under the ROC curve), which has been commonly used for face detectors. The ROC (receiver op-

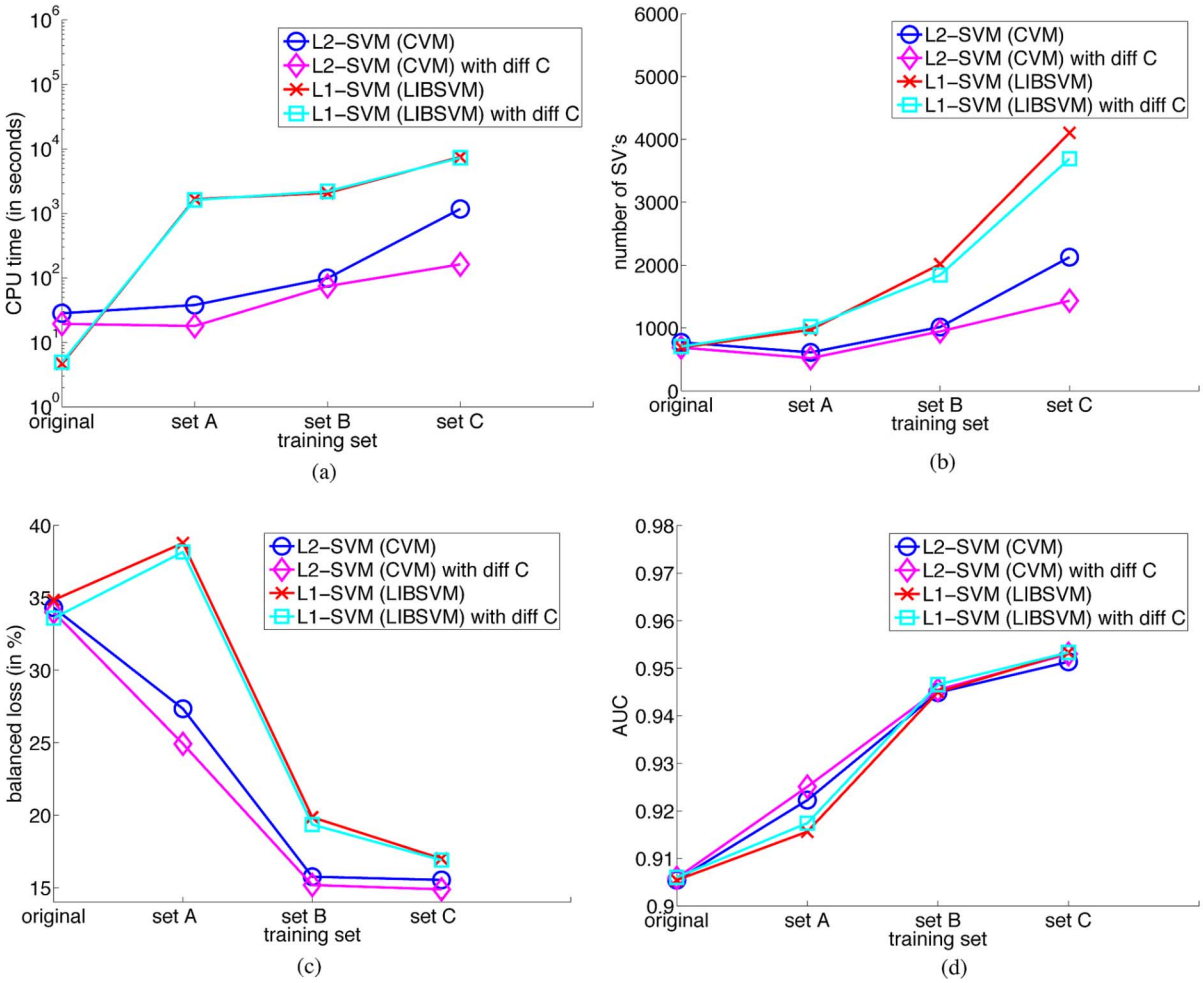

Fig. 7. Results on the extended MIT face data set. Note that the time shown in (a) is in log scale. (a) CPU time. (b) Number of support vectors. (c) Balanced loss. (d) AUC.

erating characteristic) curve [35] plots TP on the Y-axis and the false positive rate

$$\text{FP} = \frac{\#\text{negatives incorrectly classified}}{\#\text{total negatives}}$$

on the X-axis. The AUC is always between 0 and 1. A perfect face detector will have unit AUC, while random guessing will have an AUC of 0.5.

Results are shown in Fig. 7. Again, the generalized CVM implementation is significantly faster than the LIBSVM implementation and has much fewer support vectors. Moreover, the use of different Cs can improve the performance of both L1- and L2-SVMs in general. However, on the highly imbalanced set A, the L1-SVM does not perform well in terms of the balanced loss even with the use of different Cs. On the other hand, the CVM implementation of the L2-SVM shows a steady improvement and is less affected by the skewed distribution.

VI. Conclusion

The original CVM algorithm is restricted to kernels k satisfying $k(\mathbf{x}, \mathbf{x}) =$constant, and to certain kernel methods whose dual objectives do not have a linear term. In this paper, we introduced the center-constrained minimum enclosing ball problem and consequently a generalized CVM formulation that allows for a more general form for the dual objective and also any linear/nonlinear kernel to be handled. In particular, kernel methods such as SVR, ranking SVM, and two-class SVM for imbalanced data can now be performed under this framework. The resultant procedure inherits the simplicity of CVM, and has small asymptotic time and space complexities. Experimentally, it is as accurate as other state-of-the-art implementations, but is much faster and produces far fewer support vectors (and thus faster testing) on large data sets. The ability to perform regression on large data sets is particularly attractive for geometric modeling with implicit surfaces [36]. Encouraging preliminary results on using the proposed extension for implicit surface modeling is recently reported in [22].

While any kernel function can be used with the generalized CVM, additional speedup may be possible by employing kernel-specific computational tricks. For example, Sonnenburg *et al.* [37] recently proposed the use of special data structures and suffix trees for computing string kernels on sequence data. The possible integration of these data structures into the generalized CVM will be studied in the future. Moreover, an experimental evaluation of the ranking SVM proposed in Section IV-B will also be conducted. With the explosion of customer click-through data in online shopping websites, we expect that the ability of the generalized CVM in handling very large data sets will be very useful in

this real-world context. Besides, recall that the two-norm error is used in the CVM formulation. Extending the CVM to handle the one-norm error will be further studied in the future.

APPENDIX
PROOFS

In the following, we denote the center of MEB($\mathcal{S}_t$) by $\bar{\mathbf{c}}_t = \begin{bmatrix} \mathbf{c}_t \\ 0 \end{bmatrix}$.

Lemma 1: There exists a point $\mathbf{z}_i$ on the boundary of the MEB($\mathcal{S}_t$) such that the angle between the two vectors $\bar{\mathbf{c}}_{t+1} - \bar{\mathbf{c}}_t$ and $\bar{\mathbf{c}}_t - \mathbf{z}_i$ is $\geq 90°$.

Proof: Let $\mathbf{z}_i = \begin{bmatrix} \varphi(\mathbf{x}_i) \\ \delta_i \end{bmatrix}$ be the point inside MEB($\mathcal{S}_t$) that is furthest away from $\bar{\mathbf{c}}_t$. First, consider the special case where this $\mathbf{z}_i$ is directly above $\bar{\mathbf{c}}_t$. In other words, $\bar{\mathbf{c}}_t = \begin{bmatrix} \varphi(\mathbf{x}_i) \\ 0 \end{bmatrix}$. As $\mathbf{z}_i$ is on the boundary of MEB($\mathcal{S}_t$), the radius of MEB($\mathcal{S}_t$) is $R_t = |\delta_i| > 0$. From Step 2) of Algorithm 1, if there exists a point $\mathbf{z}_j = \begin{bmatrix} \varphi(\mathbf{x}_j) \\ \delta_j \end{bmatrix}$ such that $\sqrt{||\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j)||^2 + \delta_j^2} \geq (1+\epsilon)|\delta_i|$, then it is added to the core set; otherwise, the algorithm terminates. As $\mathbf{c}_{t+1}$ is constrained to be of the form $\begin{bmatrix} \varphi(\mathbf{x}) \\ 0 \end{bmatrix}$, so $\mathbf{c}_{t+1} - \mathbf{c}_t = \begin{bmatrix} \varphi(\mathbf{x}) - \varphi(\mathbf{x}_i) \\ 0 \end{bmatrix}$ is orthogonal to $\mathbf{z}_i - \bar{\mathbf{c}}_t = \begin{bmatrix} \mathbf{0} \\ \delta_i \end{bmatrix}$. In other words, the angle between these two vectors is 90°.

On the other hand, if $\mathbf{z}_i$ is not directly above $\bar{\mathbf{c}}_t$, then, as in the proof of Lemma 2 in [38], any closed halfspace bounded by a hyperplane that contains $\bar{\mathbf{c}}_t$ and orthogonal to the plane whose last coordinate is 0, i.e., $\begin{bmatrix} \varphi(\mathbf{x}) \\ 0 \end{bmatrix}$, must also contain a point $\mathbf{z}$ at the boundary. Otherwise, we can shift the center $\mathbf{c}_t$ towards the opposite side along $\begin{bmatrix} \varphi(\mathbf{x}) \\ 0 \end{bmatrix}$, and construct a smaller MEB($\mathcal{S}_t$), leading to a contradiction. Therefore, we can choose this point in the halfspace that does not contain $\mathbf{c}_{t+1}$ and the angle between $\mathbf{c}_{t+1} - \mathbf{c}_t$ and $\mathbf{c}_t - \mathbf{z}$ is $\geq 90°$. ■

Theorem 1: There exists a set $\mathcal{S}_t \subset \mathcal{S}$ of size $2/\epsilon$ such that the distance between $\mathbf{c}_{\mathrm{MEB}(\mathcal{S})}$ and any point $\mathbf{z}_i$ of $\mathcal{S}$ is at most $(1+\epsilon) r_{\mathrm{MEB}(\mathcal{S})}$.

Proof: The proof is adapted from that of Theorem 2.2 in [26] with minor modifications.[10] For completeness, we present the whole proof. Let $\tilde{r} = (1+\epsilon) r_{\mathrm{MEB}(\mathcal{S})}$. If all the points in $\mathcal{S}$ are at distances at most $\tilde{r}$ from $\bar{\mathbf{c}}_t = \begin{bmatrix} \mathbf{c}_t \\ 0 \end{bmatrix}$, then we are done. Otherwise, there exists a point $\mathbf{z}_i = \begin{bmatrix} \varphi(\mathbf{x}_i) \\ \delta_i \end{bmatrix} \in \mathcal{S}$ such that $||\mathbf{z}_i - \bar{\mathbf{c}}_t|| = \sqrt{||\varphi(\mathbf{x}_i) - \mathbf{c}_t||^2 + \delta_i^2} > \tilde{r}$. Define $e_t \equiv ||\bar{\mathbf{c}}_{t+1} - \bar{\mathbf{c}}_t||$. This $\mathbf{z}_i$ will be added to MEB($\mathcal{S}_{t+1}$). By the triangle inequality, we have

$$\begin{aligned} R_{t+1} + e_t &\geq ||\mathbf{z}_i - \bar{\mathbf{c}}_{t+1}|| + ||\bar{\mathbf{c}}_{t+1} - \bar{\mathbf{c}}_t|| \\ &\geq ||\mathbf{z}_i - \bar{\mathbf{c}}_t|| \\ &\geq \tilde{r}. \end{aligned} \tag{26}$$

From Lemma 1, there exists a point $\mathbf{z}_i$ on the boundary of MEB($\mathcal{S}_t$) such that the angle between $\bar{\mathbf{c}}_{t+1} - \bar{\mathbf{c}}_t$ and $\bar{\mathbf{c}}_t - \mathbf{z}_i$ is $\geq 90°$. Using the cosine rule

$$R_{t+1} \geq \sqrt{R_t^2 + e_t^2}. \tag{27}$$

Define $\lambda_t \equiv R_t/\tilde{r}$. Then, combining (26) and (27), we have

$$\lambda_{t+1}\tilde{r} \geq \max\left(\tilde{r} - e_t, \sqrt{\lambda_t^2 \tilde{r}^2 + e_t^2}\right). \tag{28}$$

The bound on λ_{t+1} is the smallest with respect to e_t when $\tilde{r} - e_t = \sqrt{\lambda_t^2 \tilde{r}^2 + e_t^2}$. On using (28), this reduces to $\lambda_{t+1} \geq (1 + \lambda_t^2)/2$ or $\lambda_t \geq 1 - 1/(1+t/2)$. Therefore, to get $\lambda_t \geq 1/(1+\epsilon)$, we only need $t \geq 2/\epsilon$. ■

Property 2: In the regression setting, all the training patterns satisfy loose KKT conditions when the generalized CVM terminates.

Proof: At any iteration t, each training point may fall into one of the following three categories.

1) Core vectors: Obviously, they satisfy the loose KKT conditions as they are involved in the QP.
2) Noncore vectors inside/on the ball: Recall that all the α_i's (except those of the initial core vectors) are initialized to zero. Hence, for noncore vectors inside/on the ball, their α_i's are zero and so the KKT conditions are satisfied.
3) Points lying outside $B(\mathbf{c}_t, R_t)$: For any such point $\mathbf{z}_\ell$, its $\tilde{\alpha}_\ell$ is zero (by initialization). We can rewrite (15) as (29), shown at the bottom of the page.

When the algorithm terminates at the τth iteration, all $\mathbf{z}_\ell$'s outside $B(\mathbf{c}_\tau, R_\tau)$ must lie inside/on the $(1+\epsilon)$-ball $B(\mathbf{c}_\tau, (1+$

[10]The only modification required is in the proving of (26) and (27). The other parts in the proof are identical.

$$\begin{aligned} \bar{\varepsilon}_t &= \frac{C}{2}\left(\frac{2}{C}[\boldsymbol{\lambda}' \quad \boldsymbol{\lambda}^{*\prime}]\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}\end{bmatrix} - [\boldsymbol{\lambda}' \quad \boldsymbol{\lambda}^{*\prime}]\tilde{\mathbf{K}}\begin{bmatrix}\boldsymbol{\lambda}\\ \boldsymbol{\lambda}^*\end{bmatrix} - [\boldsymbol{\lambda}' \quad \boldsymbol{\lambda}^{*\prime}]\tilde{\mathbf{K}}\begin{bmatrix}\boldsymbol{\lambda}\\ \boldsymbol{\lambda}^*\end{bmatrix}\right) \\ &= \frac{C}{2}\left(\tilde{\boldsymbol{\alpha}}'(\boldsymbol{\Delta} + \mathrm{diag}(\tilde{\mathbf{K}}) - \eta\mathbf{1}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}}\right) \text{ [using (22)]} \\ &= \frac{C}{2}\left(\tilde{\boldsymbol{\alpha}}'(\boldsymbol{\Delta} + \mathrm{diag}(\tilde{\mathbf{K}})) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - \eta\tilde{\boldsymbol{\alpha}}'\mathbf{1}\right) \\ &= \frac{C}{2}(R_t^2 - ||\mathbf{c}_t||^2 - \eta) \text{ [using (19)]} \\ \Rightarrow R_t^2 &= \frac{2}{C}\bar{\varepsilon}_t + ||\mathbf{c}_t||^2 + \eta \end{aligned} \tag{29}$$

$$
\begin{aligned}
(1+\epsilon)^2 R_\tau^2 &\geq \|\mathbf{c}_\tau - \tilde{\varphi}(\mathbf{z}_\ell)\|^2 + \delta_\ell^2 \\
&= \|\mathbf{c}_\tau\|^2 - 2(\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}})_\ell + \tilde{k}_{\ell\ell} + \delta_\ell^2 \ \text{[using (20)]} \\
&= \|\mathbf{c}_\tau\|^2 + \eta + \frac{2}{C}\left(\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}\end{bmatrix}_\ell - C(\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}})_\ell\right) \ \text{[using (22)]} \\
&= \begin{cases} \|\mathbf{c}_\tau\|^2 + \eta + \frac{2}{C}(y_\ell - (\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell) + b_\tau)), & \text{if } \ell = 1,\ldots,m \\ \|\mathbf{c}_\tau\|^2 + \eta + \frac{2}{C}((\mathbf{w}_\tau'\varphi(\mathbf{x}_{\ell-m}) + b_\tau) - y_{\ell-m}), & \text{if } \ell = m+1,\ldots,2m \end{cases} \\
&\qquad \text{[since } \mathbf{z}_\ell \notin \mathcal{S}_\tau \text{ and on using (12), (13)]} \\
&= \begin{cases} R_\tau^2 - \frac{2}{C}\bar{\varepsilon}_\tau + \frac{2}{C}(y_\ell - (\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell) + b_\tau)), & \text{if } \ell = 1,\ldots,m \\ R_\tau^2 - \frac{2}{C}\bar{\varepsilon}_\tau + \frac{2}{C}((\mathbf{w}_\tau'\varphi(\mathbf{x}_{\ell-m}) + b_\tau) - y_{\ell-m}), & \text{if } \ell = m+1,\ldots,2m \end{cases} \quad \text{[using (29)]} \\
\Rightarrow \frac{C}{2}(2\epsilon+\epsilon^2)R_\tau^2 &\geq \begin{cases} y_\ell - (\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell) + b_\tau) - \bar{\varepsilon}_\tau, & \text{if } \ell = 1,\ldots,m \\ (\mathbf{w}_\tau'\varphi(\mathbf{x}_{\ell-m}) + b_\tau) - y_{\ell-m} - \bar{\varepsilon}_\tau, & \text{if } \ell = m+1,\ldots,2m \end{cases}
\end{aligned}
$$

$\epsilon)R_\tau)$. Hence, we have the equation shown at the top of the page. As this holds for all ℓ's, it can be simplified to

$$\frac{C}{2}(2\epsilon+\epsilon^2)R_\tau^2 \geq |(y_\ell - (\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell) + b_\tau)| - \bar{\varepsilon}_\tau$$

which implies

$$\frac{C}{2}(2\epsilon+\epsilon^2)R_\tau^2 \geq |(y_\ell - (\mathbf{w}_\tau'\varphi(\mathbf{x}_\ell) + b_\tau)|_{\bar{\varepsilon}_\tau} \qquad (30)$$

by using the definition of the ε-insensitive loss function in (9). Using $\boldsymbol{\Delta} \geq \mathbf{0}$, we obtain $\eta\mathbf{1} \geq \operatorname{diag}(\tilde{\mathbf{K}}) - (2/C)\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}\end{bmatrix}$ from (22). Now, all the diagonal elements of a positive–semidefinite (psd) matrix must be nonnegative, so

$$\eta\mathbf{1} \geq -\frac{2}{C}\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}\end{bmatrix} \Rightarrow \eta\mathbf{1} \geq \frac{2}{C}\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}\end{bmatrix}. \qquad (31)$$

Thus

$$
\begin{aligned}
R_\tau^2 &= \tilde{\boldsymbol{\alpha}}'(\operatorname{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} \ \text{[using (19)]} \\
&= \tilde{\boldsymbol{\alpha}}'\left(\eta\mathbf{1} + \frac{2}{C}\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}\end{bmatrix}\right) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} \ \text{[using (22)]} \\
&= \eta + \frac{2}{C}\tilde{\boldsymbol{\alpha}}'\begin{bmatrix}\mathbf{y}\\ -\mathbf{y}\end{bmatrix} - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} \\
&\leq \eta + \eta - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} \ \text{[using (31)]} \\
&\leq 2\eta.
\end{aligned}
$$

Combining with (30), we have

$$|y_i - (\mathbf{w}_\tau'\varphi(\mathbf{x}_i) + b_\tau)|_{\bar{\varepsilon}} \leq \left(2\epsilon+\epsilon^2\right)C\eta \equiv \epsilon_2$$

which is a loose KKT condition on pattern i [which has $\lambda_i = 0$ and $\lambda_i^* = 0$, and thus $\xi_i = 0$ and $\xi_i^* = 0$, respectively, by (14)]. ■

Property 3: When $\epsilon = 0$, the generalized CVM algorithm outputs the exact solution of the kernel problem. When $\epsilon > 0$ and it terminates at the τth iteration, the optimal primal objective p^* of the kernel problem in (10) satisfies $\max\left(R_\tau^2/(p^*/C^2+\eta), (p^*/C^2+\eta)/R_\tau^2\right) \leq (1+\epsilon)^2$.

Proof: When $\epsilon = 0$, as the number of core vectors increases in each iteration and the training set size is finite, so the algorithm must terminate in a finite number (say, τ) of iterations. Using the same argument as in [18], MEB($\mathcal{S}_\tau$) must be the exact MEB enclosing all the whole training set on termination. On the other hand, when $\epsilon > 0$, $R_\tau \leq r_{\text{MEB}(\mathcal{S})} \leq (1+\epsilon)R_\tau$ by definition. It can be easily seen that $p^* = C^2 d^*$, where d^* is the optimal dual objective in (11); d^*, in turn, is related to r_{MEB} as $r_{\text{MEB}}^2 = d^* + \eta$ by comparing (17) and (21). Thus, as $R_\tau \leq r_{\text{MEB}} \leq (1+\epsilon)R_\tau$, we can bound p^* as $R_\tau^2 \leq p^*/C^2 + \eta \leq (1+\epsilon)^2 R_\tau^2$. □

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their constructive comments on an earlier version of this paper.

REFERENCES

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

[2] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVMs for very large scale problems," *Neural Comput.*, vol. 14, no. 5, pp. 1105–1114, May 2002.

[3] C. Williams and M. Seeger, , T. Leen, T. Dietterich, and V. Tresp, Eds., "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2001, vol. 13, pp. 682–688.

[4] A. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proc. 7th Int. Conf. Mach. Learn.*, Stanford, CA, Jun. 2000, pp. 911–918.

[5] D. Achlioptas, F. McSherry, and B. Schölkopf, "Sampling techniques for kernel methods," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, vol. 14, pp. 335–342.

[6] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *J. Mach. Learn. Res.*, vol. 2, pp. 243–264, Dec. 2001.

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

[8] C.-C. Chang and C.-J. Lin, LIBSVM: A Library for Support Vector Machines 2004 [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[9] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: An application to face detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, San Juan, PR, Jun. 1997, pp. 130–136.

[10] J. Platt, , B. Schölkopf, C. Burges, and A. Smola, Eds., "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods – Support Vector Learning*. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[11] S. Vishwanathan, A. Smola, and M. Murty, "SimpleSVM," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, D.C., Aug. 2003, pp. 760–767.

[12] N. Takahashi and T. Nishi, "Rigorous proof of termination of SMO algorithm for support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 774–776, May 2005.
[13] D. Pavlov, J. Mao, and B. Dom, "Scaling-up support vector machines using boosting algorithm," in *Proc. Int. Conf. Pattern Recognit.*, Barcelona, Spain, Sep. 2000, vol. 2, pp. 2219–2222.
[14] Y.-J. Lee and O. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. 1st SIAM Int. Conf. Data Mining*, San Jose, CA, 2001, pp. 184–200.
[15] T. Friess, N. Cristianini, and C. Campbell, "The Kernel-Adatron algorithm: A fast and simple learning procedure for support vector machines," in *Proc. 15th Int. Conf. Mach. Learn.*, Madison, WI, Jul. 1998, pp. 188–196.
[16] W. Chu, C. Ong, and S. Keerthi, "An improved conjugate gradient scheme to the solution of least squares SVM," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 498–501, Mar. 2005.
[17] V. Tresp, "Scaling kernel-based systems to large data sets," *Data Mining Knowl. Discovery*, vol. 5, no. 3, pp. 197–211, 2001.
[18] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets," *J. Mach. Learn. Res.*, vol. 6, pp. 363–392, 2005.
[19] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.
[20] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery and Data Mining*, Edmonton, AB, Canada, 2002, pp. 133–142.
[21] A. Smola and B. Schölkopf, "A tutorial on support vector regression," *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.
[22] I. W. Tsang, J. T. Kwok, and K. T. Lai, "Core vector regression for very large regression problems," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, Aug. 2005, pp. 913–920.
[23] I. Tsang, J. Kwok, and P.-M. Cheung, "Very large SVM training using core vector machines," in *Proc. 10th Int. Workshop Artif. Intell. Stat.*, Barbados, Jan. 2005, pp. 349–356.
[24] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
[25] P. Agarwal, S. Har-Peled, and K. Varadarajan, , E. Welzl, Ed., "Geometric approximation via coresets," in *Current Trends in Combinatorial and Computational Geometry*. Cambridge, U.K.: Cambridge Univ. Press, 2005.
[26] M. Bădoiu and K. Clarkson, "Optimal core-sets for balls," presented at the DIMACS Workshop Comput. Geometry, Piscataway, NJ, Nov. 2002.
[27] D. Tax and R. Duin, "Support vector domain description," *Pattern Recognit. Lett.*, vol. 20, no. 14, pp. 1191–1199, 1999.
[28] S. Har-Peled and Y. Wang, "Shape fitting with outliers," *SIAM J. Comput.*, vol. 33, no. 2, pp. 269–285, 2004.
[29] O. Mangasarian and D. Musicant, "Lagrangian support vector machines," *J. Mach. Learn. Res.*, vol. 1, pp. 161–177, 2001.
[30] T. Joachims, , B. Schölkopf, C. Burges, and A. Smola, Eds., "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods – Support Vector Learning*. Cambridge, MA: MIT Press, 1999, pp. 169–184.
[31] D. Musicant and A. Feinberg, "Active set support vector regression," *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 268–275, Mar. 2004.
[32] J. Friedman, "Multivariate adaptive regression splines (with discussion)," *Ann. Stat.*, vol. 19, no. 1, pp. 1–141, 1991.
[33] K.-K. Sung, "Learning and Example Selection for Object and Pattern Recognition," Ph.D. dissertation, Artif. Intell. Lab. and Cntr. Biol. Comput. Learn., MIT, Cambridge, MA, 1996.
[34] J. Weston, B. Schölkopf, E. Eskin, C. Leslie, and S. Noble, "Dealing with large diagonals in kernel matrices," in *Principles of Data Mining and Knowledge Discovery*. Helsinki, Finland: Springer-Verlag, 2002, vol. 243, Lecture Notes in Computer Science, pp. 494–511.
[35] A. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, 1997.
[36] J. Bloomenthal, *Introduction to Implicit Surfaces*. San Francisco, CA: Morgan Kaufmann, 1997.
[37] S. Sonnenburg, G. Rätsch, and B. Schölkopf, "Large scale genomic sequence SVM classifiers," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, Aug. 2005, pp. 849–856.
[38] P. Kumar, J. Mitchell, and A. Yildirim, "Approximate minimum enclosing balls in high dimensions using core-sets," *ACM J. Exp. Algorithmics*, vol. 8, p. 1.1, Jan. 2003.

Ivor Wai-Hung Tsang received the B.Eng. and M.Phil. degrees in computer science, in 2001 and 2003, respectively, from the Hong Kong University of Science and Technology (HKUST), Hong Kong, where he is currently working toward the Ph.D. degree.

He was an Honor Outstanding Student in 2001, and was awarded the Microsoft Fellowship in 2005, and the Best Paper Award from the Second IEEE Hong Kong Chapter of Signal Processing Postgraduate Forum in 2006. His scientific interests includes machine learning and kernel methods.

James Tin-Yau Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology (HKUST), Hong Kong, in 1996.

He then joined the Department of Computer Science, Hong Kong Baptist University, Hong Kong, as an Assistant Professor. He returned to HKUST in 2000 and is now an Assistant Professor in the Department of Computer Science. His research interests include kernel methods, machine learning, pattern recognition, and artificial neural networks.

Dr. Kwok is an Associate Editor for the IEEETRANSACTIONS ON NEURAL NETWORKS and the *Journal of Neurocomputing*.

Jacek M. Zurada (M'82–SM'83–F'96) received the M.S. and Ph.D. degrees in electrical engineering from the Technical University of Gdansk, Gdansk, Poland.

He is the Samuel T. Fife Alumni Professor, and Chair of the Electrical and Computer Engineering Department, University of Louisville, Louisville, KY. He was the coeditor of *Knowledge-Based Neurocomputing* (MIT Press: 2000), the author of *Introduction to Artificial Neural Systems* (PWS: 1992), contributor to the 1994 and 1995 volumes of *Progress in Neural Networks* (Ablex), and coeditor of *Computational Intelligence: Imitating Life* (IEEE Press: 1994). He is the author or coauthor of more than 250 journal and conference papers in the area of neural networks, computational intelligemce, data mining, image processing, and VLSI circuits. He has delivered numerous invited plenary conference presentations and seminars throughout the world. In March 2003, he was conferred the Title of Professor by the President of Poland, Aleksander Kwasniewski, the Honorary Professorship of Hebei University in China, and, since 2005, he serves as a Foreign Member of the Polish Academy of Sciences.

Dr. Zurada was an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS. From 2001 to 2003, he was a member of the Editorial Board of the IEEE PROCEEDINGS. From 1998 to 2003, he was the Editor-in-Chief of IEEE TRANSACTIONS ON NEURAL NETWORKS. He is an Associate Editor of *Neurocomputing*. He has received a number of awards for distinction in research and teaching, including the 1993 Presidential Award for Research, Scholarship and Creative Activity. In 2001, he received the University of Louisville President's Distinguished Service Award for Service to the Profession. He is the Past President and a Distinguished Speaker of IEEE Computational Intelligence Society.