Large-Scale Maximum Margin Discriminant Analysis Using Core Vector Machines

Ivor Wai-Hung Tsang, András Kocsor, and James Tin-Yau Kwok

***Abstract*—Large-margin methods, such as support vector machines (SVMs), have been very successful in classification problems. Recently, maximum margin discriminant analysis (MMDA) was proposed that extends the large-margin idea to feature extraction. It often outperforms traditional methods such as kernel principal component analysis (KPCA) and kernel Fisher discriminant analysis (KFD). However, as in the SVM, its time complexity is cubic in the number of training points m, and is thus computationally inefficient on massive data sets. In this paper, we propose an $(1+\epsilon)^2$-approximation algorithm for obtaining the MMDA features by extending the core vector machine. The resultant time complexity is only linear in m, while its space complexity is independent of m. Extensive comparisons with the original MMDA, KPCA, and KFD on a number of large data sets show that the proposed feature extractor can improve classification accuracy, and is also faster than these kernel-based methods by over an order of magnitude.**

***Index Terms*—Feature extraction, support vector machines (SVMs), core vector machines, scalability.**

I. Introduction

IN MANY real-world problems, the presence of superfluous features often deteriorates the classification performance. It is thus worthwhile to perform dimensionality reduction that maps the original features to a lower dimensional space while still ensuring that the data's overall structure remains intact. Over the past few decades, many linear and nonlinear dimensionality reduction methods have been proposed. In particular, if a linear method only involves dot products of the inputs, a nonlinear version can be readily obtained by using the well-known "kernel trick" [1].

A popular kernel-based dimensionality reduction method is the kernel principal component analysis (KPCA) [2]. While KPCA is unsupervised, the use of supervised information like that in the kernel Fisher discriminant analysis (KFD) [3] can lead to even better features. In the special case where the two classes are normally distributed (in the kernel-induced feature space) with the same covariance, the direction found by KFD is Bayes optimal. However, when this is not the case, the KFD directions may be far from optimal. Recently, several nonparametric methods have been proposed that correct the between-class covariance matrix by using points near the class boundary [4] or support vectors identified by a support vector machine (SVM) [5].

However, the *margin*, which is an important ingredient in the success of the SVM [1], is not used in these approaches. On some high-dimensional gene data sets, Guyon *et al.* [6] showed that the use of features extracted from the SVM's weight vector can yield promising results. However, although the SVM often performs superbly, it is not always perfect, especially when a single hyperplane does not fit the data well. To overcome this problem, Mangasarian *et al.* [7] proposed a multisurface version that uses multiple hyperplanes to fit the data with large margin and small variance. In addition, ensemble methods (such as bagging and boosting) that combine multiple SVMs have also been proposed [8], [9]. However, these methods usually involve a lot of SVMs and are thus computationally expensive.

Based on the SVM approach, maximum margin discriminant analysis (MMDA) [10] is a recent feature extraction method which finds a sequence of orthogonal hyperplanes that best separate the classes. The corresponding normal vectors of the hyperplanes are taken as new features and the data is projected onto them. The first MMDA feature is obtained by simply using the standard SVM. Then, after obtaining s orthogonal MMDA features, the $(s+1)$st feature is found by optimizing the SVM in the remaining feature subspace. In contrast to KFD, MMDA does not require normality assumptions on the data. Experimentally, MMDA has been successfully used in tasks such as face recognition [11].

From the computational point of view, feature extraction in MMDA is formulated as a quadratic programming (QP) problem similar to that for the SVM. This has the important computational advantage of not suffering from the problem of local minima. However, given m training patterns, a naive QP solver requires $O(m^3)$ time and $O(m^2)$ space [12]. Thus, a major challenge is how to scale this up for massive data sets. A common approach is to use decomposition methods [13], [14], such as the well-known sequential minimal optimization (SMO) algorithm [14], which break the original QP problem into a series of smaller QP problems. Experimentally, these schemes typically have an empirical training time complexity of $O(m) - O(m^{2.3})$ [14].

Recently, the core vector machine (CVM) approach was proposed that exploits "approximateness" in the design of SVM

Manuscript received December 2, 2006; revised July 9, 2007; accepted September 4, 2007. This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region under Grant 614907. The work of A. Kocsor was supported by the János Bolyai Fellowship of the Hungarian Academy of Sciences.

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

A. Kocsor is with the Research Group on Artificial Intelligence, the Hungarian Academy of Sciences and University of Szeged, H-6720 Szeged, Hungary (e-mail: kocsor@inf.u-szeged.hu)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.911746

implementations [15]. By making use of an approximation algorithm for the minimum enclosing ball (MEB) problem in computational geometry, the CVM has an asymptotic time complexity that is *linear* in m and a space complexity that is *independent* of m. Experiments on large classification [15] and regression [16], [17] data sets demonstrate that the CVM is as accurate as other state-of-the-art SVM implementations, but is much faster and can handle much larger data sets than existing scaleup methods.

In this paper, we attempt to scale up MMDA by integrating it with the CVM algorithm. However, as the original CVM does not involve orthogonality constraints, MMDA's QP is not of the form required by the CVM. Thus, we propose an extension of the MEB problem that imposes multiple projection constraints on the MEB's center. By adapting the CVM and its associated optimization problem, we can then perform MMDA on massive data sets in an efficient manner. A preliminary version of this paper has appeared earlier in [18].

The rest of this paper is organized as follows. Sections II and III briefly introduce the MMDA and CVM approaches, respectively. Section IV then describes the proposed MEB and CVM extensions. Experimental results are presented in Section V, and the last section gives some concluding remarks. Proofs are shown in the Appendices I–IV.

In the sequel, $\mathbf{A}\succ 0$ (resp., $\mathbf{A}\succeq 0$) means that the matrix $\mathbf{A}$ is symmetric and positive definite (pd) [resp., positive semidefinite (psd)]. Moreover, the transpose of a vector/matrix (in both the input and feature spaces) will be denoted by the superscript $'$, and $\mathbf{0},\mathbf{1}\in\mathbb{R}^m$ denote the zero vector and the vector of all ones, respectively. The inequality $\mathbf{v}=[v_1,\ldots,v_k]'\geq\mathbf{0}$ means that $v_i\geq 0$ for $i=1,\ldots,k$. In addition, $\mathbb{R}_+^k$ denotes the set of nonnegative vectors in $\mathbb{R}^k$.

II. Maximum Margin Discriminant Analysis

In this section, we first review MMDA [10]. Our main focus will be on binary classification problems. For a multiclass classification problem, we use the traditional one-versus-all approach [19] to decompose it into multiple binary problems. MMDA features are then extracted from each of these pairwise classifiers.

Suppose that we are given a training set $\mathcal{D}=\{(\mathbf{x}_i,y_i)\}_{i=1}^m$, where $\mathbf{x}_i$ is the input and $y_i\in\pm 1$ is the corresponding class label. When the data is linearly separable in the (kernel-induced) feature space, the SVM separates $\mathcal{D}$ with maximum margin [1]. When it is not the case, the SVM tries to find a hyperplane with large margin and small error. In this paper, we focus on an SVM variant called the Lagrangian SVM (LSVM) [20] that penalizes the 2-norm errors as

$$\begin{aligned}\min_{\mathbf{w},b,\xi_i}\quad & \|\mathbf{w}\|^2+b^2+C\sum_{i=1}^m\xi_i^2\\ \text{s.t.}\quad & y_i(\mathbf{w}'\varphi(\mathbf{x}_i)+b)\geq 1-\xi_i,\qquad i=1,\ldots,m.\end{aligned}\tag{1}$$

Experimentally, the generalization performance of the LSVM is often comparable to that of the standard SVM using 1-norm error [21], [22], [20]. Here, $\mathbf{w}'\varphi(\mathbf{x}_i)+b$ is the hyperplane to be learned, φ is the nonlinear feature associated with a given kernel k, $\boldsymbol{\xi}=[\xi_1,\ldots,\xi_m]'\in\mathbb{R}_+^m$ are slack variables for the errors, and C is a regularization parameter. It can be easily shown that the constraints $\xi_i\geq 0$ (for $i=1,\ldots,m$) are automatically satisfied at the optimal solution and so they are dropped here.

As mentioned earlier, MMDA extracts the features one by one. The first MMDA feature $(\mathbf{w}_1)$ is simply the weight vector of the SVM. Let $\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_s$ be the features that have already been extracted. To avoid potential scaling problems, we replace each $\mathbf{w}_q$ by the corresponding unit vector $\mathbf{u}_q=\mathbf{w}_q/\|\mathbf{w}_q\|$. As in other feature extraction methods [23], MMDA requires that these features are orthogonal to each other. To find a new feature $\mathbf{w}$, we thus solve the following optimization problem:

$$\begin{aligned}\min_{\mathbf{w},b,\xi_i}\quad & \|\mathbf{w}\|^2+b^2+C\sum_{i=1}^m\xi_i^2\\ \text{s.t.}\quad & y_i(\mathbf{w}'\varphi(\mathbf{x}_i)+b)\geq 1-\xi_i,\qquad i=1,\ldots,m\\ & \mathbf{u}_q'\mathbf{w}=0,\qquad q=1,\ldots,s.\end{aligned}\tag{2}$$

Introducing Lagrange multipliers $\boldsymbol{\alpha}=[\alpha_1,\ldots,\alpha_m]'\in\mathbb{R}_+^m$ and $\boldsymbol{\gamma}=[\gamma_1,\ldots,\gamma_s]'\in\mathbb{R}^s$ for the inequality and equality constraints, and by using the method of Lagrange multipliers, we arrive at the dual

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\gamma}}\quad & 2\boldsymbol{\alpha}'\mathbf{1}-\boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha}-2\boldsymbol{\alpha}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma}-\boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma}\\ \text{s.t.}\quad & \boldsymbol{\alpha}\geq\mathbf{0}\end{aligned}\tag{3}$$

where

$$\mathbf{U}_{d\times s}=[\mathbf{u}_1,\ldots,\mathbf{u}_s]\tag{4}$$

with d being the dimensionality of the kernel-induced feature space

$$\mathbf{Y}_{m\times m}=\operatorname{diag}(y_1,\ldots,y_m)\tag{5}$$

and

$$\hat{\mathbf{K}}_{m\times m}=\mathbf{Y}\left(\mathbf{K}+\mathbf{1}\mathbf{1}'+\frac{1}{C}\mathbf{I}\right)\mathbf{Y}\tag{6}$$

with $\mathbf{K}_{m\times m}=\boldsymbol{\Phi}'\boldsymbol{\Phi}$ (where $\boldsymbol{\Phi}=[\varphi(\mathbf{x}_1),\ldots,\varphi(\mathbf{x}_m)]$). Note that $\hat{\mathbf{K}}$ can be regarded as a kernel matrix of the transformed kernel $\tilde{k}(\mathbf{z}_i,\mathbf{z}_j)=y_iy_jk(\mathbf{x}_i,\mathbf{x}_j)+y_iy_j+\delta_{ij}y_iy_j/C$ defined on $\mathbf{z}_i=(\mathbf{x}_i,y_i)$ and $\mathbf{z}_j=(\mathbf{x}_j,y_j)$, where $\delta_{ij}=1$ when $i=j$ and 0 otherwise. The feature map corresponding to $\tilde{k}$ is

$$\hat{\varphi}(\mathbf{z}_i)=[y_i\varphi(\mathbf{x}_i)',y_i,y_i/\sqrt{C}\mathbf{e}_i']'\tag{7}$$

where $\mathbf{e}_i\in\mathbb{R}^m$ is zero except for the ith entry of one. Obviously, (3) is also a QP problem.

By using the Karush–Kuhn–Tucker (KKT) conditions, the primal variables can be recovered from the optimal values of $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ via the relations

$$\begin{aligned}\mathbf{w}&=\sum_{i=1}^m\alpha_iy_i\varphi(\mathbf{x}_i)+\sum_{q=1}^s\gamma_q\mathbf{u}_q\\ b&=\sum_{i=1}^m\alpha_iy_i\quad \xi_i=\frac{\alpha_i}{C}.\end{aligned}\tag{8}$$

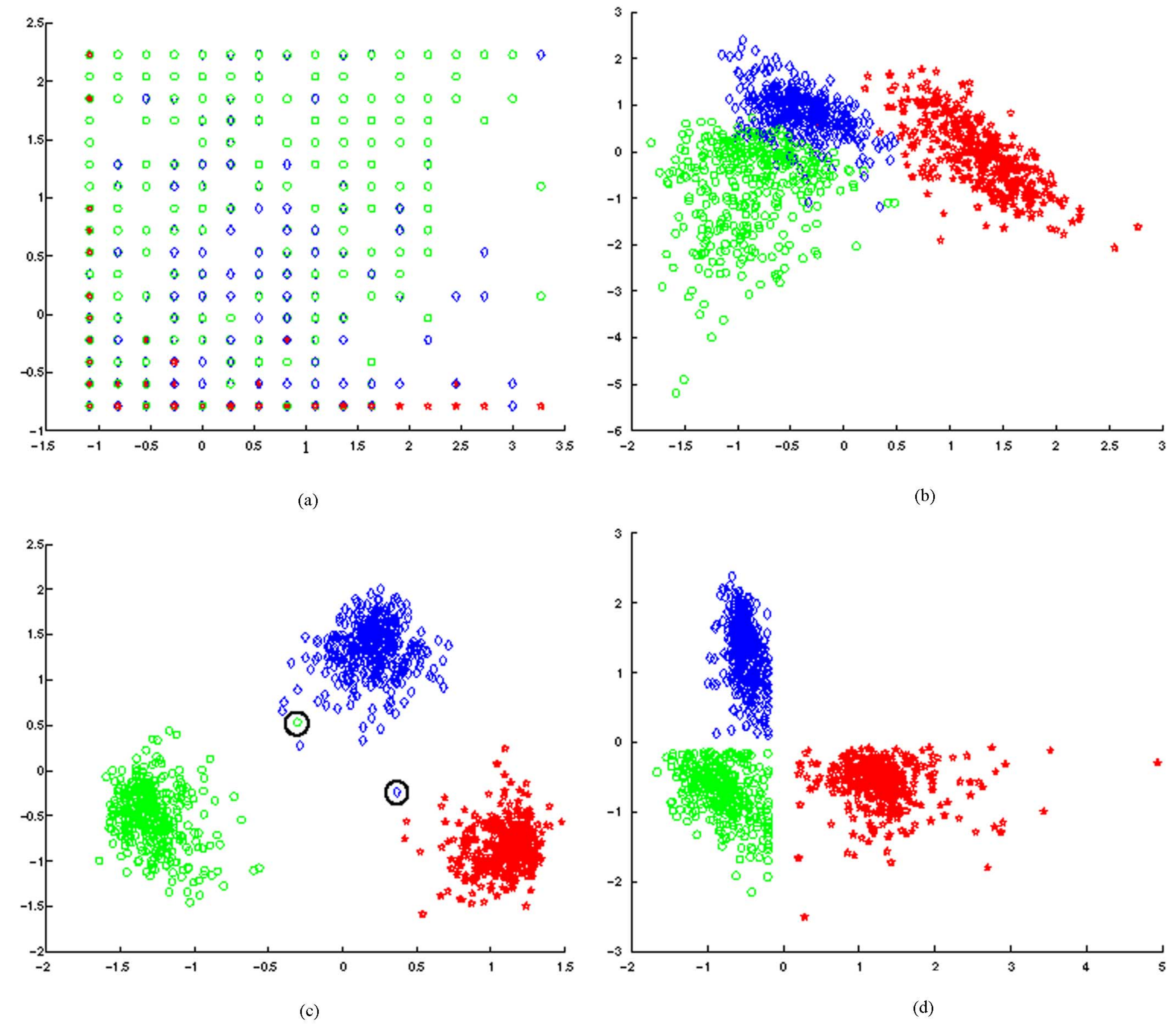


Fig. 1. Digits 0, 6, and 9 in the 2-D feature spaces extracted by KPCA, KFD, and MMDA. (a) Original. (b) KPCA. (c) KFD. (d) MMDA.

Let $\boldsymbol{\alpha}_s = [\alpha_{s1}, \ldots, \alpha_{sm}]'$ be the optimal $\boldsymbol{\alpha}$ obtained at the sth iteration. Recall that the first MMDA feature $\mathbf{w}_1$ is simply the weight vector of the SVM, i.e.,

$$\mathbf{w}_1 = \sum_{i=1}^{m} \alpha_{1i} y_i \varphi(\mathbf{x}_i) = \boldsymbol{\Phi}\mathbf{Y}\boldsymbol{\alpha}_1. \tag{9}$$

It is thus a linear combination of $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m)$. For the second MMDA feature, we have from (8) and (9)

$$\mathbf{w}_2 = \boldsymbol{\Phi}\mathbf{Y}\boldsymbol{\alpha}_2 + \gamma_1 \mathbf{u}_1 = \boldsymbol{\Phi}\mathbf{Y}\left(\boldsymbol{\alpha}_2 + \gamma_1 \frac{\boldsymbol{\alpha}_1}{\|\boldsymbol{\Phi}\mathbf{Y}\boldsymbol{\alpha}_1\|}\right)$$

which is again a linear combination of $\varphi(\mathbf{x}_i)$s. By induction, it is easy to see that $\mathbf{w}$ in (8) for each iteration can always be expressed as a linear combination of $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m)$.

As an illustration, Fig. 1 shows the features extracted by KPCA, KFD, and MMDA on three-digit classes from the Optdigits data set.[1] As can be seen, both KFD (except for the two circled points) and MMDA separate the classes well.

III. Core Vector Machines

Now we review the CVM algorithm in [15]–[17], [24]. The key idea is to transform the SVM training problem into an equivalent MEB problem in computational geometry, which is then solved by an efficient $(1+\epsilon)$-approximation algorithm.[2] As the approximation ratio (which is $(1+\epsilon)$ here) is known, the quality of the approximate solution obtained can thus be guaranteed. In

[1]Details of this data set and the experimental setup will be given in Section V.

[2]Let C be the cost of the solution returned by an approximate algorithm, and C^* be the cost of the optimal solution. An approximation algorithm has *approximation ratio* $\rho(n)$ for an input size n if $\max((C)/(C^*), (C^*)/(C)) \leq \rho(n)$. 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 (roughly the same as) the optimal solution. If the ratio does not depend on n, we can just write ρ and call the algorithm a *ρ-approximation algorithm.*

the following, we denote the ball with center $\mathbf{c}$ and radius R by $B(\mathbf{c}, R)$. Moreover, given a ball B, its center and radius will be denoted by $\mathbf{c}_B$ and r_B, respectively.

A. Original MEB Problem

Given a set of points $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, the MEB of $\mathcal{S}$, denoted MEB$(\mathcal{S})$, is the smallest ball that contains all the points in $\mathcal{S}$. Denote the feature map associated with a given kernel k by φ. Finding the MEB $B(\mathbf{c}, R)$ in the feature space induced by k leads to the following optimization problem:

$$\begin{aligned} \min_{R,\mathbf{c}} \quad & R^2 \\ \text{s.t.} \quad & \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 \le R^2, \qquad i = 1, \ldots, m. \end{aligned} \tag{10}$$

Its dual is the QP problem

$$\max_{\boldsymbol{\alpha}} \quad \boldsymbol{\alpha}' \mathrm{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \tag{11}$$

$$\text{s.t.} \quad \boldsymbol{\alpha} \ge \mathbf{0} \quad \boldsymbol{\alpha}'\mathbf{1} = 1 \tag{12}$$

where $\boldsymbol{\alpha} = [\alpha_i, \ldots, \alpha_m]'$. In [15], we assumed that k satisfies

$$k(\mathbf{x}, \mathbf{x}) = \kappa \tag{13}$$

a constant, for any pattern $\mathbf{x}$. This is satisfied when either the isotropic kernel $K(\|\mathbf{x} - \mathbf{y}\|)$ (e.g., Gaussian kernel), or the dot product kernel $K(\mathbf{x}'\mathbf{y})$ (e.g., polynomial kernel) with normalized inputs, or any normalized kernel $(K(\mathbf{x}, \mathbf{y}))/(\sqrt{K(\mathbf{x}, \mathbf{x})}\sqrt{K(\mathbf{y}, \mathbf{y})})$ is used. As will be shown in Section III-B, this assumption can also be dropped by using the extension proposed in [16] and [17].

Using the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$ in (12), we have $\boldsymbol{\alpha}'\mathrm{diag}(\mathbf{K}) = \kappa$. Dropping this constant from the objective in (11), we obtain a simpler QP problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha} \ge \mathbf{0} \quad \boldsymbol{\alpha}'\mathbf{1} = 1. \end{aligned} \tag{14}$$

Conversely, whenever the kernel k satisfies (13), any QP problem of the form (14) can be regarded as an MEB problem. This reveals an important connection between the MEB problem and kernel methods. For example, in the classification setting (with training patterns $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$), it can be shown that the dual of the two-class Lagrangian SVM is given by [15]

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha} \ge \mathbf{0}, \quad \boldsymbol{\alpha}'\mathbf{1} = 1 \end{aligned}$$

where

$$[\hat{\mathbf{K}}]_{ij} = \hat{k}(\mathbf{z}_i, \mathbf{z}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}. \tag{15}$$

If k satisfies (13), then the kernel function $\hat{k}$ corresponding to $\hat{\mathbf{K}}$ satisfies $\hat{k}(\mathbf{z}, \mathbf{z}) = \kappa + 1 + (1/C)$, which is also a constant. Hence, training this SVM is the same as finding the MEB in the feature space associated with $\hat{k}$ [15].

Once the learning problem has been formulated as an MEB problem, one can use state-of-the-art MEB algorithms to find its solution. In particular, Bădoiu and Clarkson [25] proposed an efficient $(1+\epsilon)$-approximation algorithm using the idea of *core sets*. To see how it works, first denote the estimate of the MEB at the tth iteration by $B(\mathbf{c}_t, R_t)$. This MEB is then expanded by including the furthest point outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$, and the process is repeated until all the points in $\mathcal{S}$ are covered. By using this approximation algorithm, the resultant CVM procedure is much faster and almost as accurate as existing SVM implementations. Experimentally, it also generates fewer support vectors (and thus leads to faster testing) on large data sets [15], [24]. Moreover, for a fixed ϵ, its asymptotic time complexity is only *linear* in the training set size m while its space complexity is *independent* of m. Instead of finding the furthest point at each iteration, one can use the probabilistic speedup method in [26] to efficiently obtain a point that is probably furthest away from the current MEB estimate. As shown in [24], the time complexity can then be reduced to become *independent* of m for a fixed value of ϵ.

B. Center-Constrained MEB Problem

In Section III-A, we assumed that the kernel function k satisfies condition (13) and that the QP corresponding to the kernel method has the form of (14). In this section, these conditions are relaxed by extending the original MEB problem in (10) to a *center-constrained* MEB problem that places additional constraints on the MEB's center [16], [17]. To be more specific, we first augment an extra $\delta_i \in \mathbb{R}$ to each $\varphi(\mathbf{x}_i)$ in Section III-A, forming $\left[\begin{smallmatrix}\varphi(\mathbf{x}_i)\\ \delta_i\end{smallmatrix}\right]$. 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. The primal in (10) is thus changed to

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

The corresponding dual is again a QP problem

$$\begin{aligned} \max \quad & \boldsymbol{\alpha}'(\mathrm{diag}(\mathbf{K}) + \boldsymbol{\Delta}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}'\mathbf{1} = 1 \quad \boldsymbol{\alpha} \ge \mathbf{0} \end{aligned} \tag{17}$$

where $\boldsymbol{\Delta} = [\delta_1^2, \ldots, \delta_m^2]' \ge \mathbf{0}$. 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

$$\begin{aligned} \max \quad & \boldsymbol{\alpha}'(\mathrm{diag}(\mathbf{K}) + \boldsymbol{\Delta} - \eta\mathbf{1}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha}'\mathbf{1} = 1 \quad \boldsymbol{\alpha} \ge \mathbf{0}. \end{aligned} \tag{18}$$

Using the same argument as in Section III-A, any QP problem of the form (18), with $\boldsymbol{\Delta} \ge \mathbf{0}$, can be regarded as an MEB problem (16). Note that (18) has a linear term in the objective, which allows us to extend the CVM to kernel methods like support vector regression and the ranking SVM [16], [17]. Besides, unlike that in Section III-A, the derivation does not require the kernel to satisfy condition (13). Moreover, this extension inherits from the original CVM algorithm its low time and space complexities. For more details, interested readers are referred to [16] and [17].

IV. Integrating MMDA With CVM

Note that MMDA's QP problem in (3) does not take the same form as (14) or (17). Hence, it can neither be formulated as a standard MEB (in Section III-A) nor a center-constrained MEB problem (in Section III-B). However, just as we can extend the standard MEB problem (10) to the center-constrained MEB problem (16) so as to accommodate more kernel methods, we now consider another extension of the MEB problem (Section IV-A). It will then be shown that this extension is related to the optimization problem of MMDA (Section IV-B). Moreover, efficient solution of this new MEB problem can again be obtained with the use of core sets (Section IV-C). Finally, some properties of the algorithm are discussed in Section IV-D.

A. New Constrained MEB Problem

In this section, let the kernel be $\tilde{k}$ and the corresponding feature map be $\tilde{\varphi}$. We will see in Section IV-B how this kernel $\tilde{k}$ is related to the kernel k used in MMDA. As in Section III-B, we first augment an extra $\delta_i \in \mathbb{R}$ to each $\tilde{\varphi}(\mathbf{x}_i)$, giving $[\begin{smallmatrix}\tilde{\varphi}(\mathbf{x}_i)\\ \delta_i\end{smallmatrix}]$. Then, we find the MEB for these augmented points, while at the same time constraining the ball's center such that the following hold:

1) its last coordinate is equal to zero (i.e., of the form $[\mathbf{c}', 0]'$ as in Section III-B);
2) $\mathbf{c}$ vector is orthogonal to a given set of orthogonal vectors $\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \ldots, \tilde{\mathbf{u}}_s \in \mathbb{R}^{\tilde{d}}$, where $\tilde{d}$ is the dimensionality of $\tilde{\varphi}(\mathbf{x}_i)$.

Following the derivation as in Section III-B, the primal in (10) can be modified to

$$\begin{aligned}\min_{R,\mathbf{c}} \quad & R^2 \\ \text{s.t.} \quad & \left\| \begin{bmatrix}\mathbf{c}\\ 0\end{bmatrix} - \begin{bmatrix}\tilde{\varphi}(\mathbf{x}_i)\\ \delta_i\end{bmatrix} \right\|^2 \le R^2, \qquad i = 1, \ldots, m, \\ & \tilde{\mathbf{u}}_q'\mathbf{c} = 0, \qquad q = 1, \ldots, s. \end{aligned} \tag{19}$$

Note that the second set of constraints ($\tilde{\mathbf{u}}_q'\mathbf{c} = 0$) is not used in Section III-B.

After introducing Lagrange multipliers $\tilde{\boldsymbol{\alpha}} = [\tilde{\alpha}_1, \ldots, \tilde{\alpha}_m]' \in \mathbb{R}_+^m$ and $\tilde{\boldsymbol{\gamma}} = [\tilde{\gamma}_1, \ldots, \tilde{\gamma}_s]' \in \mathbb{R}^s$ for the inequality and equality constraints respectively, it can be readily shown that the dual is

$$\begin{aligned}\max_{\tilde{\boldsymbol{\alpha}},\tilde{\boldsymbol{\gamma}}} \quad & \tilde{\boldsymbol{\alpha}}'(\text{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} \\ \text{s.t.} \quad & \tilde{\boldsymbol{\alpha}} \ge \mathbf{0} \quad \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1 \end{aligned} \tag{20}$$

where $\boldsymbol{\Delta} = [\delta_1^2, \ldots, \delta_m^2]'$, $\tilde{\mathbf{U}}_{\tilde{d}\times s} = [\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_s]$, and $\tilde{\mathbf{K}}_{m\times m} = \tilde{\boldsymbol{\Phi}}'\tilde{\boldsymbol{\Phi}}$ is the kernel matrix with $\tilde{\boldsymbol{\Phi}}_{\tilde{d}\times m} = [\tilde{\varphi}(\mathbf{x}_1), \ldots, \tilde{\varphi}(\mathbf{x}_m)]$. As $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$, an arbitrary multiple of $\tilde{\boldsymbol{\alpha}}'\mathbf{1}$ can be added to the objective without affecting its solution. In other words, for an arbitrary $\eta \in \mathbb{R}$, (20) yields the same optimal solution as

$$\begin{aligned}\max_{\tilde{\boldsymbol{\alpha}},\tilde{\boldsymbol{\gamma}}} \quad & \tilde{\boldsymbol{\alpha}}'(\text{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta} - \eta\mathbf{1}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} \\ \text{s.t.} \quad & \tilde{\boldsymbol{\alpha}} \ge \mathbf{0} \quad \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1. \end{aligned}$$

On setting $\boldsymbol{\Delta} = \eta\mathbf{1} - \text{diag}(\tilde{\mathbf{K}})$, we have $\boldsymbol{\Delta} \ge \mathbf{0}$ for a large enough η. The first term in the objective can then be dropped, leading to

$$\begin{aligned}\max_{\tilde{\boldsymbol{\alpha}},\tilde{\boldsymbol{\gamma}}} \quad & -\tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} \\ \text{s.t.} \quad & \tilde{\boldsymbol{\alpha}} \ge \mathbf{0} \quad \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1. \end{aligned} \tag{21}$$

Conversely, any QP problem of this form corresponds to the constrained MEB problem (19). Note that, as in Section III-B, this does not place any restriction on the kernel function and so the method can be used with any linear/nonlinear kernel. From the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$, the radius R and the center $\mathbf{c}$ can be recovered via

$$R = \sqrt{\tilde{\boldsymbol{\alpha}}'(\text{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}}} \tag{22}$$

and

$$\mathbf{c} = \sum_{i=1}^{m} \tilde{\alpha}_i\tilde{\varphi}(\mathbf{x}_i) + \sum_{q=1}^{s} \tilde{\gamma}_q\tilde{\mathbf{u}}_q. \tag{23}$$

B. Correspondence Between the Duals of MMDA and the Constrained MEB Problem

Note that MMDA's dual [in (3)] is slightly different from the constrained MEB problem's dual [in (21)]. The $\tilde{\boldsymbol{\alpha}}'\mathbf{1}$ term, which appears in a constraint of (21), is part of the objective function in (3). In this section, we show how the optimal solution of (3) can be obtained from that of (21).

1) KKT Conditions for the Optimization Problems (3) and (21): Recall that the KKT conditions have to be satisfied at optimality. Hence, we will first consider the KKT conditions for the two optimization problems. For the problem in (3), after introducing a Lagrange multiplier $\mu_i \ge 0$ for each $\alpha_i \ge 0$ constraint, its Lagrangian is

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\mu}) = 2\boldsymbol{\alpha}'\mathbf{1} - \boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} - \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma} + 2\boldsymbol{\alpha}'\boldsymbol{\mu}$$

where $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_m]' \in \mathbb{R}_+^m$. At optimality, the KKT conditions require that the derivatives of $\mathcal{L}$ with respect to (w.r.t.) $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ be zero, i.e.,

$$\mathbf{G}\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\gamma}\end{bmatrix} = \begin{bmatrix}\mathbf{1}+\boldsymbol{\mu}\\ \mathbf{0}\end{bmatrix}$$

where

$$\mathbf{G}_{(m+s)\times(m+s)} = \begin{bmatrix}\hat{\mathbf{K}} & \mathbf{Y}\Phi'\mathbf{U}\\ \mathbf{U}'\Phi\mathbf{Y} & \mathbf{U}'\mathbf{U}\end{bmatrix} \tag{24}$$

and also that

$$\boldsymbol{\alpha} \ge \mathbf{0} \quad \boldsymbol{\mu} \ge \mathbf{0}, \qquad \alpha_i\mu_i = 0, \qquad \text{for } i = 1, \ldots, m.$$

Property 1: $\mathbf{G}$ in (24) is pd.

Proof is in Appendix I. Consequently, $\mathbf{G}$ is invertible, and, from (24), we have

$$\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\gamma}\end{bmatrix} = \mathbf{G}^{-1}\begin{bmatrix}\mathbf{1}+\boldsymbol{\mu}\\ \mathbf{0}\end{bmatrix}. \tag{25}$$

Similarly, for the optimization problem in (21), we introduce a Lagrange multiplier $\tilde{\mu}_i \geq 0$ for each nonnegative constraint and β for the equality constraint. Then, its Lagrangian is

$$\mathcal{L}(\tilde{\boldsymbol{\alpha}}, \tilde{\gamma}, \tilde{\boldsymbol{\mu}},) = -\tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\gamma} - \tilde{\gamma}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\gamma} + 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\mu}} + 2(\tilde{\boldsymbol{\alpha}}'\mathbf{1} - 1)$$

where $\tilde{\boldsymbol{\mu}} = [\tilde{\mu}_1, \ldots, \tilde{\mu}_m]' \in \mathbb{R}_+^m$. At optimality, the KKT conditions require that

$$\tilde{\mathbf{G}} \begin{bmatrix} \tilde{\boldsymbol{\alpha}} \\ \tilde{\boldsymbol{\gamma}} \end{bmatrix} = \begin{bmatrix} \mathbf{1} + \tilde{\boldsymbol{\mu}} \\ \mathbf{0} \end{bmatrix}$$

where

$$\tilde{\mathbf{G}}_{(m+s)\times(m+s)} = \begin{bmatrix} \tilde{\mathbf{K}} & \tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}} \\ \tilde{\mathbf{U}}'\tilde{\boldsymbol{\Phi}} & \tilde{\mathbf{U}}'\tilde{\mathbf{U}} \end{bmatrix} \tag{26}$$

and also that

$$\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1 \quad \tilde{\boldsymbol{\alpha}} \geq \mathbf{0} \quad \tilde{\boldsymbol{\mu}} \geq \mathbf{0}, \qquad \tilde{\alpha}_i\tilde{\mu}_i = 0, \qquad \text{for } i = 1, \ldots, m. \tag{27}$$

In Section IV-B2, we will show that $\tilde{\mathbf{G}} \succ 0$ for our particular choice of $\tilde{\mathbf{G}}$. Hence, (26) can also be written as

$$\begin{bmatrix} \tilde{\boldsymbol{\alpha}} \\ \tilde{\boldsymbol{\gamma}} \end{bmatrix} = \tilde{\mathbf{G}}^{-1} \begin{bmatrix} \mathbf{1} + \tilde{\boldsymbol{\mu}} \\ \mathbf{0} \end{bmatrix}. \tag{28}$$

2) Obtaining the Optimal Solution of (21) from the Optimal Solution of (3): We now show how the optimal solutions of the two optimization problems (3) and (21) are related. In the following, we set $\tilde{\mathbf{K}}$ in (21) to $\hat{\mathbf{K}}$ in (3), i.e.,

$$\tilde{\mathbf{K}} = \hat{\mathbf{K}}. \tag{29}$$

Using (7), the corresponding matrix $\tilde{\boldsymbol{\Phi}}$ of $\tilde{\varphi}$-mapped feature vectors is

$$\tilde{\boldsymbol{\Phi}} = \begin{bmatrix} \boldsymbol{\Phi}\mathbf{Y} \\ \mathbf{y}' \\ \frac{1}{\sqrt{C}}\mathbf{Y}\mathbf{I}_{m\times m} \end{bmatrix}$$

where $\mathbf{y} = [y_1, \ldots, y_m]'$ and $\mathbf{I}_{m\times m}$ is the identity matrix of size $m \times m$. As for $\tilde{\mathbf{U}}$, we set $\tilde{\mathbf{U}} = [\begin{smallmatrix} \mathbf{U} \\ \mathbf{0}_{(m+1)\times s} \end{smallmatrix}]$, where $\mathbf{0}_{(m+1)\times s}$ is the $(m+1) \times s$ matrix of all zeroes. Then

$$\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}} = \mathbf{Y}\boldsymbol{\Phi}'\mathbf{U} \quad \text{and} \quad \tilde{\mathbf{U}}'\tilde{\mathbf{U}} = \mathbf{U}'\mathbf{U}. \tag{30}$$

Hence, $\tilde{\mathbf{G}}$ in (26) becomes identical to $\mathbf{G}$ in (24). Recall from Property 1 that $\mathbf{G} \succ 0$, hence $\tilde{\mathbf{G}}$ is also pd as mentioned in Section IV-B1.

Denote the upper left $m \times m$ submatrix of $\mathbf{G}^{-1}(= \tilde{\mathbf{G}}^{-1})$ by $\mathbf{H}$. From (25), we have $\boldsymbol{\alpha} = \mathbf{H}(\mathbf{1} + \boldsymbol{\mu})$, and, similarly, from (28), $\tilde{\boldsymbol{\alpha}} = \mathbf{H}(\mathbf{1} + \tilde{\boldsymbol{\mu}})$. Combining, we have

$$\tilde{\boldsymbol{\alpha}} - \mathbf{H}\tilde{\boldsymbol{\mu}} = \beta(\boldsymbol{\alpha} - \mathbf{H}\boldsymbol{\mu}). \tag{31}$$

Note that[3] $\boldsymbol{\alpha}'\mathbf{1} > 0$. Take the dot product with $\mathbf{1}$ on both sides of (31). Then, by using the constraint $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$ in (20), we obtain

$$\beta = \frac{1 - \mathbf{1}'\mathbf{H}\tilde{\boldsymbol{\mu}}}{\boldsymbol{\alpha}'\mathbf{1} - \mathbf{1}'\mathbf{H}\boldsymbol{\mu}} = \frac{1}{\boldsymbol{\alpha}'\mathbf{1}} \frac{\boldsymbol{\alpha}'\mathbf{1} - \mathbf{1}'\mathbf{H}\tilde{\boldsymbol{\mu}}\boldsymbol{\alpha}'\mathbf{1}}{\boldsymbol{\alpha}'\mathbf{1} - \mathbf{1}'\mathbf{H}\boldsymbol{\mu}} = \frac{1}{\boldsymbol{\alpha}'\mathbf{1}} \tag{32}$$

[3]Suppose to the contrary that $\boldsymbol{\alpha}'\mathbf{1} = 0$. We have $\boldsymbol{\alpha} = \mathbf{0}$ as $\boldsymbol{\alpha} \geq \mathbf{0}$. From (8), the weight vector $\mathbf{w}$ obtained then lies in the span of $\mathbf{u}_q$'s. However, from (2), $\mathbf{w}$ has to be orthogonal to all the $\mathbf{u}_q$'s; so $\boldsymbol{\gamma} = \mathbf{0}$, and subsequently, $\mathbf{w} = \mathbf{0}$, $b = 0$, and $\xi_i = 0$, which is not a feasible solution for (2).

on setting

$$\tilde{\boldsymbol{\mu}} = \frac{\boldsymbol{\mu}}{\boldsymbol{\alpha}'\mathbf{1}} \geq \mathbf{0}. \tag{33}$$

Thus

$$\tilde{\boldsymbol{\mu}} = \beta\boldsymbol{\mu}. \tag{34}$$

We then obtain the following relationship between $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\gamma}})$ and $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ at optimality:

$$\begin{aligned} \begin{bmatrix} \tilde{\boldsymbol{\alpha}} \\ \tilde{\boldsymbol{\gamma}} \end{bmatrix} &= \beta\tilde{\mathbf{G}}^{-1} \begin{bmatrix} \mathbf{1} + \boldsymbol{\mu} \\ \mathbf{0} \end{bmatrix} \quad \text{[from (28) and (34)]} \\ &= \frac{1}{\boldsymbol{\alpha}'\mathbf{1}} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \end{bmatrix} \quad \text{[from (25) and (32)].} \end{aligned} \tag{35}$$

Note that $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = (\boldsymbol{\alpha}'\mathbf{1})/(\boldsymbol{\alpha}'\mathbf{1}) = 1$, $\tilde{\boldsymbol{\alpha}} \geq \mathbf{0}$, and $\tilde{\alpha}_i\tilde{\mu}_i = (\alpha_i)/(\boldsymbol{\alpha}'\mathbf{1})(\mu_i)/(\boldsymbol{\alpha}'\mathbf{1}) = 0$, and so (27) is satisfied. In other words, the optimal solution $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\gamma}})$ of (20) can be recovered from the optimal solution $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ of (3) by the simple normalization in (35). The converse is also true, i.e., the optimal solution $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ can also be recovered from the optimal solution $(\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\gamma}})$. However, this result seems less useful here and so the derivation will be postponed to Appendix II.

C. Finding the Constrained MEB Defined on the Core Set

Now, given that the optimal solutions of (3) and (21) are related to each other, we can solve the optimization problem (3) associated with MMDA by solving the corresponding constrained MEB problem (21) instead. Based on the encouraging results using the CVM algorithm to solve various large-scale MEB problems [15], [16], here we will apply the same CVM algorithm (Section III) to solve this constrained MEB problem. For completeness, the algorithm is shown in Algorithm 1. The only change required is in the implementation of step 4, which now requires finding the constrained MEB of the core set $\mathcal{S}_{t+1}$ instead of the standard MEB in [15] (or the center-constrained MEB in [16]).

Algorithm 1: The CVM algorithm

```
1: Initialize $\mathcal{S}_0, \mathbf{c}_0$ and $R_0$.
2: Terminate if there is no training point $\mathbf{z}$ falling outside
   the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$.
3: Find $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ is furthest away from $\mathbf{c}_t$. Set
   $S_{t+1} = \mathcal{S}_t \cup \{\mathbf{z}\}$ .
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 $R_{t+1} = r_{\mathrm{MEB}(\mathcal{S}_{t+1})}$.
5: Increment $t$ by 1 and go back to step 2.
```

To find the constrained MEB of $\mathcal{S}_t$ in step 4, a straightforward approach is to make use of (28). Denote the $\tilde{\mathbf{G}}$ corresponding to the core set $\mathcal{S}_t$ by $\tilde{\mathbf{G}}_t$. As shown in (26), $\tilde{\mathbf{G}}_t$ is defined in terms of $\tilde{\mathbf{K}}_t$ and $\tilde{\boldsymbol{\Phi}}_t$, which in turn are defined using the patterns in $\mathcal{S}_t$. However, (28) requires inverting the matrix $\tilde{\mathbf{G}}$, which takes $O((t+s)^3)$ time and becomes costly when the core set is large. In addition, the optimal values for β and $\tilde{\boldsymbol{\mu}}$ are not known beforehand.

Alternatively, one can solve (21), which is a QP problem, directly by using some QP solver. In the implementation of [15], we used an efficient decomposition method called SMO [14] as the internal QP solver to find the standard MEB in step 4. However, for the constrained MEB problem here, the Lagrange multipliers $\boldsymbol{\gamma}_t$ are not involved in the equality constraint of (21) and this hinders the use of SMO.

Recall that the optimal solutions of (3) and (21) are related to each other, so we can solve the MEB subproblem in step 4 by solving the corresponding MMDA subproblem (3) instead. This is advantageous because unlike (21), (3) does not have the equality constraint $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$, and can then be solved by a very efficient optimization method called successive overrelaxation (SOR) [27], [28].

1) SOR Algorithm: In the following, we first give a brief introduction to SOR as detailed in [28]. SOR is an iterative procedure that employs the Gauss–Seidel (GS) iterations with the extrapolation factor $\omega \in (0,2)$ to accelerate the solving of the linear system $\mathbf{Hx} = -\mathbf{a}$, where $\mathbf{x} = [x_1, \ldots, x_n]' \in \mathbb{R}^n$, $\mathbf{a} \in \mathbb{R}^n$, and $\mathbf{H} \succeq 0 \in \mathbb{R}^{n\times n}$. In particular, GS is the special case of SOR when $\omega = 1$. In [28], SOR is further extended to solve QP problems with linear convergence. Consider QPs of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}'\mathbf{Hx} + \mathbf{a}'\mathbf{x} \\ \text{s.t.} \quad & L_i \le x_i \le U_i, \qquad i = 1, \ldots, n \end{aligned} \tag{36}$$

where $U_i, L_i \in \mathbb{R}$ with $U_i > 0$ and $L_i \le 0$. Following [27] and [28], $\mathbf{x}$ is initialized to some given feasible solution. At each iteration, an index $i \in \{1, \ldots, n\}$ is selected and the change to the corresponding x_i is

$$\triangle x_i = -(\mathbf{Hx} + \mathbf{a})_i / \mathbf{H}_{ii}. \tag{37}$$

At the next iteration, x_i is obtained by projecting $x_i + \omega\triangle x_i$ to the feasible range $[L_i, U_i]$ in (36). Because x_i is just 1-D, this projection can be easily achieved by simple clipping. The process is repeated with another index i until the KKT conditions are satisfied [27], or $\|\mathbf{x}_{t-1} - \mathbf{x}_t\|$ is less than some prescribed tolerance [28], where $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ are two consecutive solutions. The SOR algorithm is summarized in Algorithm 2.

Algorithm 2: The SOR algorithm

```
1: Initialize $\mathbf{x} = [x_1, \ldots, x_n]'$ and $\omega \in (0,2)$.
2: For $i = 1, \ldots, n$
3: Compute $x_i \leftarrow x_i + \omega\triangle x_i$, where $\triangle x_i$ is given by (37).
4: Project $x_i$ to the feasible range $[L_i, U_i]$.
5: End For
6: Go back to step 2 until convergence.
```

2) Finding MEB($\mathcal{S}_t$) in (3) Using SOR: In this section, we use the SOR procedure to solve the optimization problem of MEB($\mathcal{S}_t$) in (3). We first set $\boldsymbol{\alpha}_t$ and $\boldsymbol{\gamma}_t$ in (3) to some initial feasible solution. Here, the subscript t denotes that the variable is defined w.r.t. the core set $\mathcal{S}_t$. For $t = 1$, there is only one pattern in the core set $\mathcal{S}_1$ and we can simply set $\boldsymbol{\alpha}_1 = [1]$ and $\boldsymbol{\gamma}_1 = \mathbf{0}$. Otherwise, we use the solution $(\boldsymbol{\alpha}_{t-1}, \boldsymbol{\gamma}_{t-1})$ of the previously obtained MEB for warm start, and set $\boldsymbol{\alpha}_t = [\boldsymbol{\alpha}'_{t-1}\ 0]' \in \mathbb{R}^t_+$ (recall that $\boldsymbol{\alpha}_{t-1} \in \mathbb{R}^{t-1}_+$) and $\boldsymbol{\gamma}_t = \boldsymbol{\gamma}_{t-1} \in \mathbb{R}^s$. An index i is then selected from 1 to $t+s$. Note that (3) can be written as

$$\begin{aligned} \min_{\boldsymbol{\alpha}_t, \boldsymbol{\gamma}_t} \quad & \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_t \\ \boldsymbol{\gamma}_t\end{bmatrix}' \mathbf{G}_t \begin{bmatrix}\boldsymbol{\alpha}_t \\ \boldsymbol{\gamma}_t\end{bmatrix} + \begin{bmatrix}-\mathbf{1}_{t\times 1} \\ \mathbf{0}_{s\times 1}\end{bmatrix}' \begin{bmatrix}\boldsymbol{\alpha}_t \\ \boldsymbol{\gamma}_t\end{bmatrix} \\ \text{s.t.} \quad & \boldsymbol{\alpha}_t \ge \mathbf{0} \end{aligned}$$

where

$$\mathbf{G}_t = \begin{bmatrix} \hat{\mathbf{K}}_t & \mathbf{Y}_t\boldsymbol{\Phi}'_t\mathbf{U} \\ \mathbf{U}'\boldsymbol{\Phi}_t\mathbf{Y}_t & \mathbf{U}'\mathbf{U} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{K}}_t & \tilde{\boldsymbol{\Phi}}'_t\tilde{\mathbf{U}} \\ \tilde{\mathbf{U}}'\tilde{\boldsymbol{\Phi}}_t & \tilde{\mathbf{U}}'\tilde{\mathbf{U}} \end{bmatrix} \tag{38}$$

by using (24) and the substitutions [(29) and (30)] needed to establish the relationship between (3) and (21) in Section IV-B2. At each iteration, the change in $\begin{bmatrix}\boldsymbol{\alpha}_t \\ \boldsymbol{\gamma}_t\end{bmatrix}_i$ can thus be computed according to (37) as

$$\begin{bmatrix}\triangle\boldsymbol{\alpha}_t \\ \triangle\boldsymbol{\gamma}_t\end{bmatrix}_i = -\frac{1}{[\mathbf{G}_t]_{ii}}\left(\mathbf{G}_t\begin{bmatrix}\boldsymbol{\alpha}_t \\ \boldsymbol{\gamma}_t\end{bmatrix} - \begin{bmatrix}\mathbf{1} \\ \mathbf{0}\end{bmatrix}\right)_i \tag{39}$$

where $(\,\cdot\,)_i$ denotes the ith entry of the vector argument. Substituting (38) into (39), we obtain

$$\triangle\alpha_i^{(t)} = -(\tilde{\mathbf{K}}_t\boldsymbol{\alpha}_t + \tilde{\boldsymbol{\Phi}}'_t\tilde{\mathbf{U}}\boldsymbol{\gamma}_t - \mathbf{1})_i / [\tilde{\mathbf{K}}_t]_{ii} \tag{40}$$

$$\triangle\gamma_i^{(t)} = -(\tilde{\mathbf{U}}'\tilde{\boldsymbol{\Phi}}_t\boldsymbol{\alpha}_t + \tilde{\mathbf{U}}'\tilde{\mathbf{U}}\boldsymbol{\gamma}_t)_i / [\tilde{\mathbf{U}}'\tilde{\mathbf{U}}]_{ii} \tag{41}$$

where $\triangle\alpha_i^{(t)}$ and $\triangle\gamma_i^{(t)}$ are the ith components of $\triangle\boldsymbol{\alpha}_t$ and the $(i-t)$th components of $\triangle\boldsymbol{\gamma}_t$, respectively. The next iterate $\alpha_i^{(t+1)}$ is then obtained by clipping $\alpha_i^{(t)} + \triangle\alpha_i^{(t)}$ to the feasible region $\{\boldsymbol{\alpha} : \boldsymbol{\alpha} \ge \mathbf{0}\}$. This updating process is repeated with the next $i \in \{1, \ldots, t+s\}$ until the KKT conditions [27] defined on $\mathcal{S}_t$

$$\begin{aligned} \tilde{\mathbf{K}}_t\boldsymbol{\alpha}_t + \tilde{\boldsymbol{\Phi}}'_t\tilde{\mathbf{U}}\boldsymbol{\gamma}_t - \mathbf{1} &\ge \mathbf{0} \\ (\boldsymbol{\alpha}_t)_i(\tilde{\mathbf{K}}_t\boldsymbol{\alpha}_t + \tilde{\boldsymbol{\Phi}}'_t\tilde{\mathbf{U}}\boldsymbol{\gamma}_t - \mathbf{1})_i &= 0 \\ \tilde{\mathbf{U}}'\tilde{\boldsymbol{\Phi}}_t\boldsymbol{\alpha}_t + \tilde{\mathbf{U}}'\tilde{\mathbf{U}}\boldsymbol{\gamma}_t &= \mathbf{0} \end{aligned}$$

are satisfied. With the converged solution $(\boldsymbol{\alpha}_t, \boldsymbol{\gamma}_t)$, the corresponding values of $(\tilde{\boldsymbol{\alpha}}_t, \tilde{\boldsymbol{\gamma}}_t)$ can be recovered by using the normalization factor in (35). Finally, the radius and the center of the constrained MEB of $\mathcal{S}_t$ can be determined from (22) and (23).

It should be mentioned here that while standard GS iterations select the index i sequentially for update, here we improve its efficiency by selecting the point with the largest changes as in [27] or the point that violates the KKT condition the most as in [29]. The whole procedures is summarized in Algorithm 3. Other strategies, such as caching of the kernel entries [14], [29], can also be used to further improve memory usage and reduce the number of kernel evaluations.

TABLE I
DATA SETS USED IN THE EXPERIMENTS

	optdigits	satimage	pendigits	letters	mnist	usps	face
# classes (N_c)	10	6	10	26	10	2	2
# attributes	64	36	16	16	780	676	361
# training patterns	3,823	4,435	7,494	16,000	60,000	266,079	346,260
# testing patterns	1,797	2,000	3,498	4,000	10,000	75,383	24,045

Algorithm 3: Algorithm for solving MEB ($\mathcal{S}_t$)

```
1: Initialize $\boldsymbol{\alpha}_t, \boldsymbol{\gamma}_t, \omega \in (0, 2)$.
2: Repeat
3: For $i = 1, \ldots, t$, compute $\delta\alpha_i^{(t)} \leftarrow \max(0, \alpha_i^{(t)} + \omega\triangle\alpha_i^{(t)}) - \alpha_i^{(t)}$, where $\triangle\alpha_i^{(t)}$ is given by (40).
4: For $i = t+1, \ldots, s$, compute $\delta\gamma_i^{(t)} \leftarrow \omega\triangle\gamma_i^{(t)}$, where $\triangle\gamma_i^{(t)}$ is given by (41).
5: Pick $i = \arg\max(\max_{i=1}^{t}\{|\delta\alpha_i^{(t)}|, \max_{i=t+1}^{s}|\delta\gamma_i^{(t)}|\})$.
6: If $i \le t, \alpha_i^{(t)} \leftarrow \alpha_i^{(t)} + \delta\alpha_i^{(t)}$; otherwise, $\gamma_i^{(t)} \leftarrow \gamma_i^{(t)} + \delta\gamma_i^{(t)}$.
7: Go back to step 2 until it converges.
```

D. Properties of the Proposed Algorithm

The proposed MEB algorithm has properties analogous to those of the original CVM in [15]. In particular, recall from Section I that the main motivation for using an approximation algorithm is that its resultant time and space complexities are much lower than other procedures for finding an exact solution. Indeed, the complexities required in extracting one MMDA feature can be computed in a manner similar to that in [15] (details are shown in Appendix III). When probabilistic speedup is not used in step 3, it can be shown that the total time required for computing the $(s+1)$st projection is $O((m)/(\epsilon^2) + (1/\epsilon)((1/\epsilon) + s)^3)$, which is also linear in m for a fixed value of ϵ. When probabilistic speedup is used, this changes to $O((1/\epsilon^2)((1/\epsilon^2) + s)^3)$, which is even independent of m for a fixed ϵ. For the space complexity, the whole algorithm requires a space of $O(1/\epsilon^2)$, independent of m for a fixed ϵ. Here, we ignore the $O(m)$ space requirements for storing the m training patterns as they may be stored outside the core memory.

As for the convergence rate, we have the following.

Theory 1: There exists a subset $\mathcal{S}_t$, with size $2/\epsilon$, of the whole training set $\mathcal{S} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$ such that the distance between $\mathbf{c}_{\mathrm{MEB}(\mathcal{S}_t)}$ and any point $\mathbf{z}_i$ of $\mathcal{S}$ is at most $(1+\epsilon)r_{\mathrm{MEB}(\mathcal{S})}$.

Its proof is similar to that of [17, Th. 1], and a sketch is shown in Appendix IV. Recall that, in step 3 of Algorithm 1, one point from $\mathcal{S}$ is included in the MEB estimate at each iteration. This property thus ensures that the proposed method converges in at most $2/\epsilon$ iterations, independent of the feature dimensionality and the size of $\mathcal{S}$. Moreover, it can be shown that all the training patterns satisfy loose KKT conditions at the end of the extraction process for each MMDA feature.

Moreover, the algorithm converges to the (approximately) optimal solution, as shown by Theorem 2. Its proof is similar to that in [15, Sec. 4.2], and a sketch is shown in Appendix V.

Theorem 2: When $\epsilon = 0$, Algorithm 1 finds the exact solution of the center-constrained MEB problem in (19). When $\epsilon > 0$ and the algorithm terminates at the τth iteration, we have

$$\max\left(\frac{R_\tau^2}{p^*}, \frac{p^*}{R_\tau^2}\right) \le (1+\epsilon)^2$$

where p^* is the optimal value of the objective in (19).

In other words, the proposed method is a $(1+\epsilon)^2$-approximation algorithm. As ϵ is usually very small, the approximate solution obtained is thus very close to the exact optimal solution.

V. EXPERIMENTS

In this section, we report experiments on a number of real-world data sets[4] (Table I). The following feature extraction algorithms are compared:

1) the original MMDA, denoted by MMDA(SVM);
2) the proposed formulation,[5] denoted by MMDA(CVM);
3) kernel Fisher discriminant (KFD);
4) kernel PCA (KPCA).

For KFD, recall that the rank of its between-class scatter matrix is at most $N_c - 1$ [3]. Hence, for the sake of convenience, we always set the number of KFD features to $N_c - 1$. We use the Gaussian kernel $\exp(-\|\mathbf{x} - \mathbf{z}\|^2/)$, where $\beta = (1)/(m^2)\sum_{i,j=1}^{m}\|\mathbf{x}_i - \mathbf{x}_j\|^2$ is the average squared distance between the training patterns. For MMDA, the C parameter is always set to 1. All the algorithms are implemented in MATLAB and run on an AMD Athlon 4400+ PC with 4 GB of RAM. Moreover, as KFD and KPCA only require the leading eigenvectors, the MATLAB function eigs is used in the implementations.

A. Varying the Number of Extracted Features

First, we investigate how the performance depends on the number of features extracted. A feedforward artificial neural network (ANN) is used as the classifier for the extracted features. It has a single layer of ten hidden units and is trained via standard backpropagation. Here, experiments are only performed on the three smaller data sets (optdigits, satimage, and pendigits) in Table I. As all these have $N_c > 2$ classes, we use the one-versus-all scheme for MMDA and so MMDA is run N_c

[4]The first five data sets are from the University of California at Irvine (UCI) machine learning repository, while the last two are from http://www.cse.ust.hk/~ivor/cvm.html.

[5]The value of ϵ was fixed at 0.001. Preliminary results showed that this value of ϵ often leads to fast training and good extracted features.

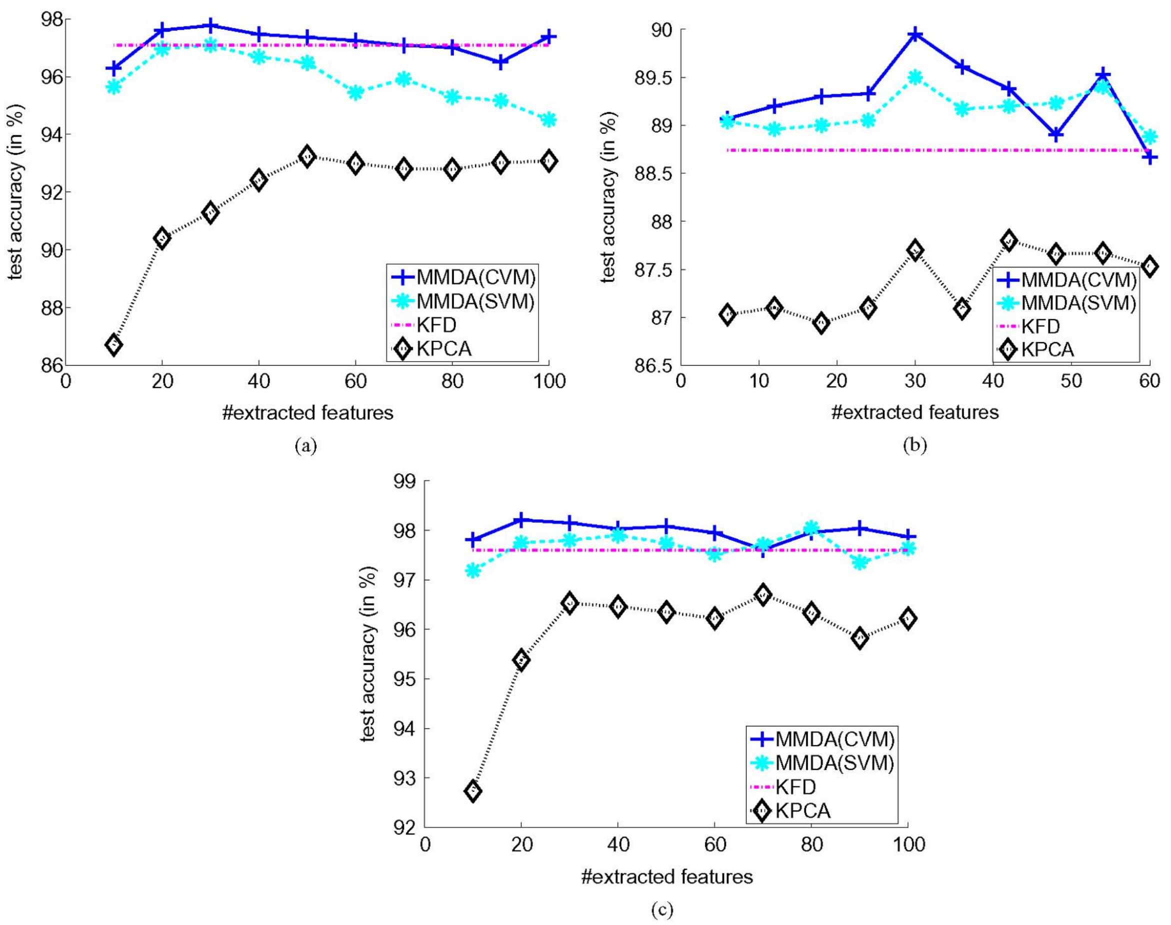


Fig. 2. Testing accuracies with different numbers of extracted features: (a) optdigits, (b) satimage, and (c) pendigits.

times. For KFD, this is easier as we can use its standard multiclass version [30]. For KPCA, it is unsupervised and so patterns from all the classes can be directly used for feature extraction.

Fig. 2 shows the testing accuracies when the ANN is trained on different numbers of features extracted by the various algorithms. As can be seen, the use of more features can initially improve performance. However, as more and more features are added, the accuracy no longer increases and sometimes even drops as features with little classification information are included. For both MMDA implementations, the testing accuracies are better than the others when $3N_c$ to $5N_c$ features are used.

Fig. 3 shows the time taken for each feature extraction algorithm. As expected, this increases with the number of extracted features for both MMDA implementations. On the other hand, the one for KPCA is almost constant, as most of the time there is spent on computing the constant-sized $m \times m$ kernel matrix, regardless of how many features were to be extracted.

As mentioned in Section II, the MMDA features (i.e., the $\mathbf{w}$'s) can always be expressed as a linear combination of $\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m)$. This is also true for KFD and KPCA. Fig. 4 compares the total numbers of kernel evaluations involved in all the extracted features. As can be seen, this increases steadily with the number of extracted features for all the feature extractors. Furthermore, as expected, MMDA(CVM) can identify a small number of points when solving the constrained MEB problem. Thus, the features obtained by MMDA(CVM) involve a smaller number of kernel evaluations than the other feature extractors, and are thus computationally less expensive in the testing phase.

B. Experimental Results on All the Data Sets

In the previous section, we showed that MMDA performs well when the number of features to be extracted lies around $3N_c$ to $5N_c$. Now we describe experiments performed on all the data sets in Table I, with the number of MMDA features set to $N_c, 3N_c$ or $5N_c$, while the number of KPCA features here is fixed at $5N_c$. The extracted features are used with the following classifiers:

1) SVM using the LIBSVM implementation;[6]
2) 1-nearest neighbor (1-NN) classifier;
3) ANN with the same settings as in Section V-A.

As a baseline, we also compare with the case that does not perform feature extraction. Note that when m is large, KPCA and KFD become expensive in terms of both time and space. To alleviate this problem, we only use a random sample of size 3500 when these two methods are used on the letters, mnist, usps, and face data sets. The sampling procedure is performed such that the sizes of all the classes in the sample are the same.

[6]Downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvm/

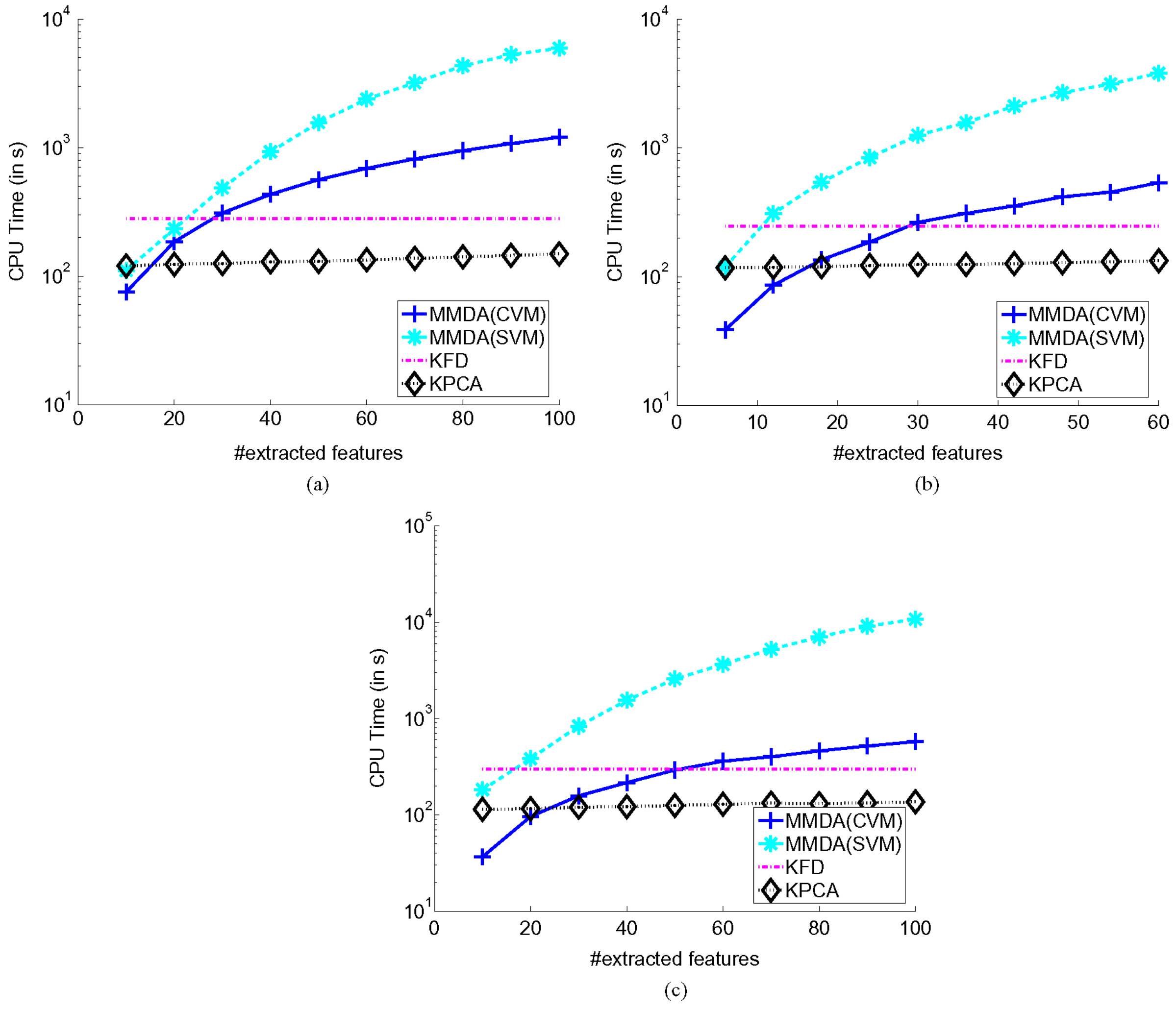


Fig. 3. CPU time (in seconds) taken by the various algorithms in extracting different numbers of features: (a) optdigits, (b) satimage, and (c) pendigits.

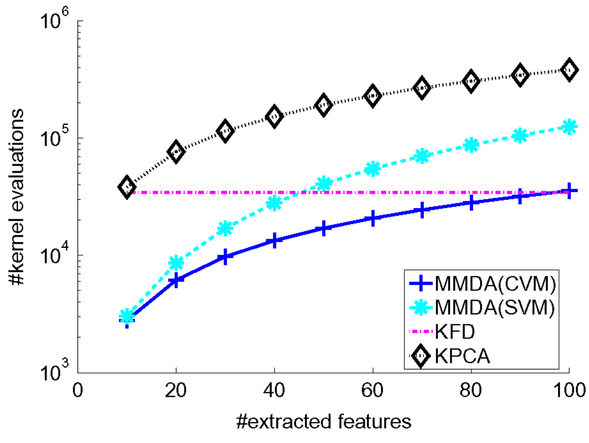
Table II shows the results of the testing accuracies. Methods that do not finish in 24 h are indicated by "—." The following remarks can be made. First, feature extraction can evidently improve classification accuracy. In particular, the use of MMDA features can outperform SVM that uses no feature extraction. Second, both the original and new implementations of MMDA have better accuracies than the other feature extraction methods. Third, as also demonstrated in Section V-A, the use of $3N_c$ to $5N_c$ features for MMDA is often beneficial. But note that KPCA and KFD sometimes perform miserably on the large data sets. This is because we only used a random sample of size 3500 for these two methods. Thus, in general, random sampling is not a good approach for reducing computational complexity.

Table III shows the central processing unit (CPU) time required in the feature extraction process. As expected, MMDA(CVM) is always faster than the original MMDA implementation, and the improvement can sometimes be of two orders of magnitude. In addition, on the three largest data sets, the original MMDA implementation cannot even converge in 24 h, while MMDA(CVM) successfully extract good features in just several hundreds of thousands seconds. MMDA(CVM) is also often faster than KPCA and KFD on the small data sets. On the larger data sets, recall that we have used random sampling for KPCA and KFD and this explains why MMDA appears slower. However, we should also not forget to say that such a random sampling scheme always leads to a poor generalization performance of KPCA/KFD in our experiments.

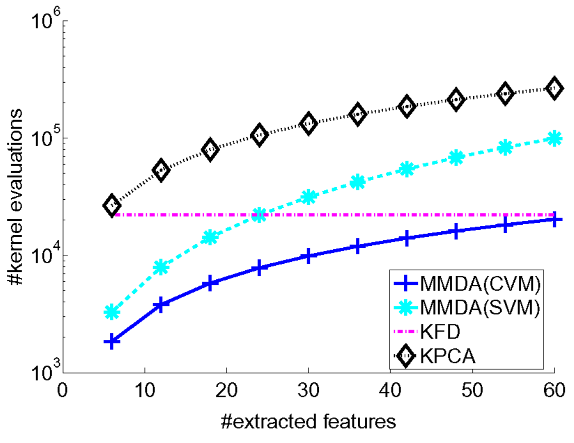
Table IV gives a comparison of the average numbers of kernel evaluations for each extracted feature. As can be seen, the MMDA(CVM) features are much sparser than the others, including the original MMDA features. As kernel evaluations dominate the computational cost in testing, MMDA(CVM) is thus much faster and offers clear benefits over the usual approaches.

C. MMDA Features Lead to Smaller Decision Trees

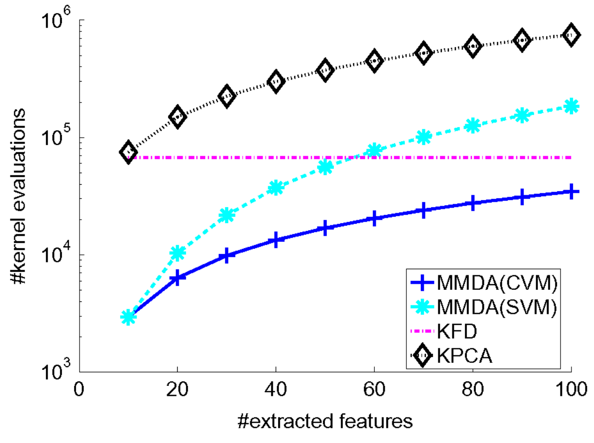
In this section, we feed the extracted features to the C4.5 decision tree classifier [31]. Intuitively, we expect that the "better" features extracted by MMDA(CVM) can lead to smaller decision trees. This is confirmed by the results in Table V. As can be seen, MMDA(CVM) often leads to smaller decision trees for a comparable performance. Moreover, in line with the results reported in the previous sections, the features extracted by MMDA(CVM) lead to better testing accuracy. Note again that KPCA and KFD perform poorly on letters, mnist, usps, and face because of the random sampling problem mentioned in Section V-B.



(a)



(b)



(c)

Fig. 4. Total number of kernel evaluations required for different numbers of extracted features: (a) optdigits, (b) satimage, and (c) pendigits.

VI. Conclusion

In this paper, we investigated the problem of kernel feature extraction in large-scale classification tasks. A good feature extractor should, ideally, do the following: 1) produce features that can lead to high classification accuracy and 2) be computationally efficient during both training and testing. MMDA was recently proposed to use large margin for feature extraction and has shown promising results over KFD. However, it is computationally inefficient on large data sets. In this paper, by introducing a new constrained MEB problem, we extended the CVM algorithm in [15] and proposed an $(1+\epsilon)^2$-approximation algorithm for extracting MMDA features. We examined the theoretical aspects of the method and demonstrated its efficiency through various experiments. In practice, it is 10–100 times faster than the original MMDA implementation. The features extracted by the proposed method are also sparser, and involve fewer kernel evaluations. This in turn allows new features to be computed much faster during testing.

Instead of using the orthogonality constraints, one might also consider using uncorrelated constraints as suggested in [32]. The primal of MMDA in (2) can then be changed to

$$\begin{aligned}\min_{\mathbf{w},b,\xi_i} \quad & \|\mathbf{w}\|^2 + b^2 + C\sum_{i=1}^{m}\xi_i^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}'\varphi(\mathbf{x}_i)+b) \geq 1-\xi_i, \qquad i=1,\ldots,m \\ & \mathbf{u}_q'\mathbf{\Phi}\mathbf{\Phi}'\mathbf{w}=0, \qquad q=1,\ldots,s.\end{aligned}$$

It can be shown that the corresponding dual has the same form as in (3). The possibilities it offers together with the possible use of other types of constraints for MMDA feature extraction will be investigated in the future.

Appendix I
Proof of Property 1

First, we introduce two lemmas.

Lemma 1: $\hat{\mathbf{K}} \succ 0$.

Proof: From the definition of $\hat{\mathbf{K}}$ in (6), for any nonzero vector $\mathbf{v} \in \mathbb{R}^m$, we have

$$\begin{aligned}\mathbf{v}'\hat{\mathbf{K}}\mathbf{v} &= \mathbf{v}'\left(\mathbf{Y}\left(\mathbf{K}+\mathbf{1}\mathbf{1}'+\frac{1}{C}\mathbf{I}\right)\mathbf{Y}\right)\mathbf{v} \\ &= (\mathbf{Y}\mathbf{v})'\mathbf{K}(\mathbf{Y}\mathbf{v}) + (\mathbf{Y}\mathbf{v})'\mathbf{1}\mathbf{1}'(\mathbf{Y}\mathbf{v}) + \frac{1}{C}(\mathbf{Y}\mathbf{v})'(\mathbf{Y}\mathbf{v}) \\ &\geq \frac{1}{C}(\mathbf{Y}\mathbf{v})'(\mathbf{Y}\mathbf{v}) \quad (\text{as } \mathbf{K}\succeq 0 \text{ and } \mathbf{1}\mathbf{1}'\succeq 0) \\ &= \frac{1}{C}\mathbf{v}'\mathbf{v} \quad (\text{as } \mathbf{Y}'\mathbf{Y}=\mathbf{I}) \\ &> 0.\end{aligned}$$

■

TABLE II
TESTING ACCURACIES ON THE VARIOUS DATA SETS. METHODS THAT DID NOT FINISH IN 24 h ARE INDICATED BY "—"

features		classifier	optdigits	satimage	pendigits	letters	mnist	usps	face
MMDA(CVM)	#features= N_c	SVM	97.16	89.75	97.71	94.85	94.54	–	–
	$3N_c$		97.10	89.50	97.74	94.60	95.73	–	–
	$5N_c$		97.05	89.30	97.79	94.35	**95.92**	–	–
	#features= N_c	1-NN	97.44	88.65	97.74	**96.78**	93.42	99.43	–
	$3N_c$		96.44	89.00	97.71	96.53	94.70	**99.43**	–
	$5N_c$		95.94	89.50	97.68	96.28	95.18	99.41	–
	#features= N_c	ANN	96.27	89.75	97.22	80.97	92.99	99.30	98.28
	$3N_c$		97.77	89.30	97.85	82.45	93.28	99.33	**98.39**
	$5N_c$		97.36	**89.95**	**98.08**	82.67	93.34	99.34	98.34
MMDA(SVM)	#features= N_c	SVM	96.93	89.45	97.82	92.85	–	–	–
	$3N_c$		96.93	88.55	97.85	93.95	–	–	–
	$5N_c$		96.93	89.65	97.91	94.02	–	–	–
	#features= N_c	1-NN	97.16	87.25	97.66	96.55	–	–	–
	$3N_c$		95.66	88.95	97.91	96.05	–	–	–
	$5N_c$		95.38	89.90	98.03	95.42	–	–	–
	#features= N_c	ANN	95.65	88.95	97.19	80.20	–	–	–
	$3N_c$		97.09	89.65	97.80	82.02	–	–	–
	$5N_c$		96.48	88.70	97.74	82.65	–	–	–
KFD		SVM	97.77	89.10	98.05	91.25	11.35	–	–
		1-NN	**97.94**	85.85	98.03	91.42	11.35	96.87	–
		ANN	97.82	88.75	97.68	85.20	10.28	96.80	1.96
KPCA		SVM	93.65	84.05	92.13	67.92	10.28	–	–
		1-NN	94.94	87.95	97.37	88.45	9.58	99.38	–
		ANN	93.65	87.70	96.05	76.90	10.28	98.60	83.61
no feature extraction		SVM	96.66	89.60	96.74	90.95	95.12	–	–
		1-NN	96.38	89.35	97.43	95.20	94.34	–	–
		ANN	94.37	87.40	95.19	70.95	90.39	99.12	97.40

TABLE III
CPU TIME (IN SECONDS) REQUIRED IN THE FEATURE EXTRACTION PROCESS

		optdigits	satimage	pendigits	letters	mnist	usps	face
MMDA(CVM)	#features= N_c	41	23	20	92	1,610	2,359	105
	$3N_c$	181	78	95	301	4,928	6,585	337
	$5N_c$	332	136	174	512	8,179	10,630	556
MMDA(SVM)	#features= N_c	84	121	127	1,911	–	–	–
	$3N_c$	476	421	570	9,646	–	–	–
	$5N_c$	1,495	900	1,674	20,860	–	–	–
KFD		282	246	298	298	312	471	391
KPCA		150	130	137	147	226	371	325

TABLE IV
AVERAGE NUMBER OF KERNEL EVALUATIONS INVOLVED IN EACH EXTRACTED FEATURE

		optdigits	satimage	pendigits	letters	mnist	usps	face
MMDA(CVM)	#features= N_c	279	308	292	351	434	423	403
	$3N_c$	359	334	349	357	434	398	409
	$5N_c$	367	342	353	360	427	396	411
MMDA(SVM)	#features= N_c	303	548	293	870	–	–	–
	$3N_c$	841	1,056	1,149	1,772	–	–	–
	$5N_c$	1,278	1,553	1,875	2,420	–	–	–
KFD		3,823	4,435	7,494	3,500	3,500	3,500	3,500
KPCA		3,823	4,435	7,494	3,500	3,500	3,500	3,500

Lemma 2: The Schur complement of $\mathbf{G}$, i.e., $\hat{\mathbf{K}} - \mathbf{Y}\Phi'\mathbf{U}(\mathbf{U}'\mathbf{U})^{-1}\mathbf{U}'\Phi\mathbf{Y}$, is pd.

Proof: Recall that $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_s]$ in (4); so $\mathbf{U}'\mathbf{U} = \mathbf{I}$ as the $\mathbf{u}_i$'s are orthonormal. From the definition of $\hat{\mathbf{K}}$ in (6), we have

$$\begin{aligned}
&\hat{\mathbf{K}} - \mathbf{Y}\Phi'\mathbf{U}(\mathbf{U}'\mathbf{U})^{-1}\mathbf{U}'\Phi\mathbf{Y}\\
&\quad = \mathbf{Y}\Phi'\Phi\mathbf{Y} + \mathbf{Y}\mathbf{1}\mathbf{1}'\mathbf{Y} + \frac{1}{C}\mathbf{I} - \mathbf{Y}\Phi'\mathbf{U}(\mathbf{U}'\mathbf{U})^{-1}\mathbf{U}'\Phi\mathbf{Y}\\
&\quad = \mathbf{Y}(\Phi'\Phi - \Phi'\mathbf{U}(\mathbf{U}'\mathbf{U})^{-1}\mathbf{U}'\Phi)\mathbf{Y} + \mathbf{Y}\mathbf{1}\mathbf{1}'\mathbf{Y} + \frac{1}{C}\mathbf{I}\\
&\quad = \mathbf{Y}(\Phi'\Phi - \Phi'\mathbf{U}\mathbf{U}'\Phi)\mathbf{Y} + \mathbf{Y}\mathbf{1}\mathbf{1}'\mathbf{Y} + \frac{1}{C}\mathbf{I}.
\end{aligned}$$

Now, for any nonzero $\mathbf{z}$

$$\mathbf{z}'(\mathbf{I} - \mathbf{U}\mathbf{U}')\mathbf{z} = \|\mathbf{z}\|^2 - \|\mathbf{U}'\mathbf{z}\|^2 = \|\mathbf{U}_\perp\mathbf{z}\|^2 \geq 0$$

where $\mathbf{U}_\perp\mathbf{z}$ is the projection of $\mathbf{z}$ in the subspace orthogonal to the span of $\mathbf{u}_1, \ldots, \mathbf{u}_s$. In other words, $\mathbf{I} - \mathbf{U}\mathbf{U}' \succeq 0$

TABLE V
TESTING ACCURACIES (IN PERCENT) AND SIZES OF THE RESULTANT DECISION TREES (IN BRACKETS) WHEN C4.5 IS USED AS THE CLASSIFIER. THE NUMBERS OF EXTRACTED FEATURES FOR BOTH KPCA AND MMDA ARE ALWAYS FIXED AT $5N_c$. METHODS THAT DO NOT FINISH IN 24 h ARE INDICATED BY "—"

	optdigits	satimage	pendigits	letters	mnist	usps	face
MMDA (CVM)	**95.40** (19)	87.80 (79)	97.10 (19)	**90.30** (285)	**93.30** (619)	**99.30** (29)	**98.40** (25)
MMDA (SVM)	94.90 (19)	87.90 (119)	**97.20** (19)	85.30 (449)	-	-	-
KFD	94.40 (19)	**88.60** (43)	95.90 (21)	79.30 (109)	10.30 (23)	92.50 (3)	65.70 (3)
KPCA	81.40 (117)	87.10 (113)	89.30 (75)	58.90 (323)	10.10 (197)	97.00 (43)	80.70 (51)
no feature extraction	82.80 (143)	86.30 (109)	89.70 (161)	81.00 (777)	36.20 (1,631)	98.00 (876)	96.90 (759)

and so $\mathbf{Y}(\Phi'\Phi - \Phi'\mathbf{U}\mathbf{U}'\Phi)\mathbf{Y}\succeq 0$. Moreover, $\mathbf{Y}\mathbf{1}\mathbf{1}'\mathbf{Y}\succeq 0$, while $(1/C)\mathbf{I}\succ 0$. Combining all these, we thus have $\hat{\mathbf{K}} - \mathbf{Y}\Phi'\mathbf{U}(\mathbf{U}'\mathbf{U})^{-1}\mathbf{U}'\Phi\mathbf{Y}\succ 0$. ■

Using [33, Fact 11], we thus have $\mathbf{G}\succ 0$.

APPENDIX II
OBTAINING THE OPTIMAL $(\boldsymbol{\alpha},\boldsymbol{\gamma})$ FROM THE OPTIMAL $(\tilde{\boldsymbol{\alpha}},\tilde{\boldsymbol{\gamma}})$

By equating the optimal objective values of the primal in (2) and the dual in (3), and by using (8), it is easy to show that

$$\boldsymbol{\alpha}'\mathbf{1} = \boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha} + 2\boldsymbol{\alpha}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} + \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma} \tag{42}$$

since

$$\begin{aligned}&\tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} + 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} + \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}}\\ &\quad= \tilde{\boldsymbol{\alpha}}'\hat{\mathbf{K}}\tilde{\boldsymbol{\alpha}} + 2\tilde{\boldsymbol{\alpha}}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\tilde{\boldsymbol{\gamma}} + \tilde{\boldsymbol{\gamma}}'\mathbf{U}'\mathbf{U}\tilde{\boldsymbol{\gamma}} \quad \text{[from (30)]}\\ &\quad= \frac{\boldsymbol{\alpha}'\hat{\mathbf{K}}\boldsymbol{\alpha} + 2\boldsymbol{\alpha}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} + \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma}}{(\boldsymbol{\alpha}'\mathbf{1})^2} \quad \text{[from (35)]}\\ &\quad= \frac{1}{\boldsymbol{\alpha}'\mathbf{1}} \quad \text{[from (42)].}\end{aligned} \tag{43}$$

From (35) and (43), we then have the following relation:

$$\begin{bmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\gamma}\end{bmatrix} = \frac{1}{\tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} + 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}} + \tilde{\boldsymbol{\gamma}}'\tilde{\mathbf{U}}'\tilde{\mathbf{U}}\tilde{\boldsymbol{\gamma}}}\begin{bmatrix}\tilde{\boldsymbol{\alpha}}\\ \tilde{\boldsymbol{\gamma}}\end{bmatrix}.$$

APPENDIX III
TIME AND SPACE COMPLEXITIES

The time and space complexities can be computed in a manner similar to that in [15]. In the following, we assume that a simple QP implementation, which takes $O((m+s)^3)$ time and $O((m+s)^2)$ space for $(s+1)$st projection, is used for the QP subproblem in step 3. We first consider the case where probabilistic speedup is not used in step 3. From Theorem 1, the algorithm converges in at most $2/\epsilon$ iterations. Consequently, the size of the final core set is $O(1/\epsilon)$, though in practice it has often been observed to be much smaller than this worst-case upper bound [34]. As only one core vector is added at each iteration, $|S_t| = t+2$. Distance computations in steps 2 and 3 take $O((t+s)m+(t+2+s)^2) = O((t+s)m+(t+s)^2)$ time. Finding the MEB in step 3 takes $O((t+2+s)^3) = O((t+s)^3)$ time, and the other operations take constant time. Hence, the tth iteration takes a total of $O((t+s)m+(t+s)^3)$ time. The overall time required for computing the $(s+1)$st projection in $\tau = O(1/\epsilon)$ iterations is

$$\begin{aligned}T &= \sum_{t=1}^{\tau} O((t+s)m+(t+s)^3)\\ &= O(\tau(\tau+s)m + \tau(\tau+s)^3)\\ &= O\left(\frac{m}{\epsilon}\left(\frac{1}{\epsilon}+s\right) + \frac{1}{\epsilon}\left(\frac{1}{\epsilon}+s\right)^3\right)\end{aligned}$$

which is linear in m for a fixed ϵ.

When probabilistic speedup is used, distance computations in steps 2 and 3 take $O((t+2+s)^2) = O((t+s)^2)$ time. The number of iterations τ may be larger than $2/\epsilon$, though it can still be bounded by $O(1/\epsilon^2)$ [35]. Thus, the overall time changes to $O((1)/(\epsilon^2)(\frac{1}{\epsilon^2}+s)^3)$, which is even independent of m for a fixed ϵ.

For the space complexity, the whole algorithm requires a space of $O(1/\epsilon^2+s)$, independent of m for a fixed ϵ. Here, we ignore the $O(m)$ space requirements for storing the m training patterns, as they may be stored outside the core memory.

APPENDIX IV
PROOF OF THEOREM 1

Our Theorem 1 here is identical to [17, Th. 1] and the proofs are also the same, except that we have to reestablish [17, Th. 1, Lemma 1]. Its proof in our new context is shown in the following. Note that as the ball's center is constrained in (19) to be of the form $\tilde{\mathbf{c}} = \begin{bmatrix}\mathbf{c}\\ 0\end{bmatrix}$; this is equivalent to requiring that $[\mathbf{0}'1]\tilde{\mathbf{c}} = 0$. Moreover, the constraint $\tilde{\mathbf{u}}_q'\mathbf{c} = 0$ in (19) is also the same as $[\tilde{\mathbf{u}}_q'0]\tilde{\mathbf{c}} = 0$. Thus, the constrained MEB problem in (19) is the same as an MEB problem with multiple projection constraints on the center. In the following, we will denote the ball's center $\tilde{\mathbf{c}}$ simply as $\mathbf{c}$.

Lemma 3: There exists a point $\mathbf{z}$ on the boundary of the constrained MEB$(\mathcal{S}_t)$ such that the angle between the two vectors $\mathbf{c}_{t+1} - \mathbf{c}_t$ and $\mathbf{c}_t - \mathbf{z}$ is $\geq 90°$.

Proof: Let $\mathbf{z}$ be the point inside the constrained MEB$(\mathcal{S}_t)$ that is furthest away from $\mathbf{c}_t$. First, consider the special case where this $\mathbf{z} = \mathbf{c}_t + \sum_p \lambda_p\tilde{\mathbf{u}}_p$ for not all λ_p's are zeros, such that $\mathbf{z}-\mathbf{c}_t$ lies on the span of $\mathrm{Span}\{\tilde{\mathbf{u}}_p\}$. Since $\tilde{\mathbf{u}}_p'\mathbf{c}_t = \tilde{\mathbf{u}}_p'\mathbf{c}_{t+1} = 0$ for all p, then $\tilde{\mathbf{u}}_p'(\mathbf{c}_{t+1}-\mathbf{c}_t) = 0$, and so $\mathbf{c}_{t+1}-\mathbf{c}_t$ is orthogonal to $\mathbf{z}-\mathbf{c}_t = \sum_p \lambda_p\tilde{\mathbf{u}}_p$. In other words, the angle between these two vectors is $90°$.

On the other hand, if $\mathbf{z}-\mathbf{c}_t$ is not lying on the $\mathrm{Span}\{\tilde{\mathbf{u}}_p\}$, then, as in [34, Proof of Lemma 2], any closed half-space bounded by a hyperplane that contains $\mathbf{c}_t$ and orthogonal to the normal vector $\tilde{\mathbf{u}}_p$ of the plane $\tilde{\mathbf{u}}_p'\mathbf{c}=0$, must also contain a point $\mathbf{z}$ at the boundary. Otherwise, we can shift the center $\mathbf{c}_t$, and construct a smaller constrained $\mathrm{MEB}(\mathcal{S}_t)$, leading to a contradiction. Therefore, we can choose this point in the half-space 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 $\geq 90°$. □

Appendix V Proof of Theorem 2

Proof: When $\epsilon = 0$, as the number of core vectors increases in each iteration and the training set size is finite, the algorithm must terminate in a finite number (say, τ) of iterations. Using the same argument as in [15], $\mathrm{MEB}(\mathcal{S}_\tau)$ must be the exact MEB enclosing all the whole training set on termination. Thus, Algorithm 1 finds the exact solution of the center-constrained MEB problem in (19). On the other hand, when $\epsilon > 0$, and the algorithm terminates at the τth iteration, we have $R_\tau \leq r_{\mathrm{MEB}(\mathcal{S})} \leq (1+\epsilon)R_\tau$ by definition. □

References

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

[2] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, 1998.

[3] S. Mika, G. Rätsch, J. Weston, B. Schoölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Proc. Neural Netw. Signal Process. IX*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds., 1999, pp. 41–48.

[4] Z. Li, W. Liu, D. Lin, and X. Tang, "Nonparametric subspace analysis for face recognition," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, Jun. 2005, vol. 2, pp. 961–966.

[5] B. Zhang, X. Chen, S. Shan, and W. Gao, "Nonlinear face recognition based on maximum average margin criterion," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, Jun. 2005, vol. 1, pp. 554–559.

[6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, no. 1–3, pp. 389–422, 2002.

[7] O. Mangasarian and E. Wild, "Multisurface proximal support vector machine classification via generalized eigenvalues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 1, pp. 69–74, Jan. 2006.

[8] G. Valentini and T. Dietterich, "Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods," *J. Mach. Learn. Res.*, vol. 5, pp. 725–775, 2004.

[9] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Bang, "Constructing support vector machine ensemble," *Pattern Recognit.*, vol. 36, no. 12, pp. 2757–2767, 2003.

[10] A. Kocsor, K. Kovács, and C. Szepesvári, "Margin maximizing discriminant analysis," in *Proc. 15th Eur. Conf. Mach. Learn.*, Pisa, Italy, Sep. 2004, pp. 227–238.

[11] K. Kovacs, A. Kocsor, and C. Szepesvari, "Maximum margin discriminant analysis based face recognition," in *Proc. Joint Hungarian-Austrian Conf. Image Process. Pattern Recognit.*, 2005, pp. 71–78.

[12] S. Boyd and L. Vandenberghe*, Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[13] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184.

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

[15] 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.

[16] 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.

[17] I. Tsang, J. Kwok, and J. Zurada, "Generalized core vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1126–1140, Sep. 2006.

[18] I. W. Tsang, A. Kocsor, and J. T. Kwok, "Efficient kernel feature extraction for massive data sets," in *Proc. Int. Conf. Knowl. Disc. Data Mining*, Philadelphia, PA, Aug. 2006, pp. 724–729.

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

[20] O. Mangasarian and D. Musicant, "Lagrangian support vector machines," *J. Mach. Learn. Res.*, vol. 1, pp. 161–177, 2001.

[21] Y.-J. Lee and O. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. 1st SIAM Int. Conf. Data Mining*, 2001, pp. 184–200.

[22] O. Mangasarian and D. Musicant, "Active set support vector machine classification," in *Advances in Neural Information Processing Systems 13*, T. Leen, T. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, pp. 577–583.

[23] J. Ye, "Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems," *J. Mach. Learn. Res.*, vol. 6, pp. 483–502, Apr. 2005.

[24] I. Tsang, J. Kwok, and P.-M. Cheung, "Very large SVM training using core vector machines," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, Barbados, Jan. 2005, pp. 349–356.

[25] M. Bădoiu and K. L. Clarkson, "Optimal core-sets for balls," in *Proc. DIMACS Workshop Comput. Geometry*, 2002.

[26] A. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proc. 17th Int. Conf. Mach. Learn.*, Stanford, CA, Jun. 2000, pp. 911–918.

[27] W. Kienzle and B. Schölkopf, "Training support vector machines with multiple equality constraints," in *Proc. Eur. Conf. Mach. Learn.*, Porto, Portugal, Oct. 2005, pp. 182–193.

[28] O. Mangasarian and D. Musicant, "Successive overrelaxation for support vector machines," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1032–1037, Sep. 1999.

[29] C.-J. Lin, "Asymptotic convergence of an SMO algorithm without any assumptions," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 248–250, Jan. 2002.

[30] M. Aizerman, E. Braverman, and L. Rozonoer, "Generalized discriminant analysis using a kernel approach," *Neural Comput.*, vol. 12, pp. 2385–2404, 2000.

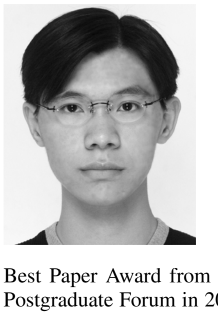
[31] J. Quinlan*, C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[32] J. Ye, T. Li, T. Xiong, and R. Janardan, "Using uncorrelated discriminant analysis for tissue classification with gene expression D," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 1, no. 4, pp. 181–190, Oct. 2004.

[33] M. Todd, "Semidefinite optimization," *Acta Numerica*, vol. 10, pp. 515–560, 2001.

[34] P. Kumar, J. Mitchell, and A. Yildirim, "Approximate minimum enclosing balls in high dimensions using core-sets," *ACM J. Experiment. Algorithm.*, vol. 8, p. 1.1, Jan. 2003.

[35] M. Bădoiu, S. Har-Peled, and P. Indyk, "Approximate clustering via core-sets," in *Proc. 34th Annu. ACM Symp. Theory Comput.*, Montréal, QC, Canada, 2002, pp. 250–257.



Ivor Wai-Hung Tsang received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2007.

Currently, he is a Postdoctoral Researcher at the Department of Computer Science, HKUST. His scientific interests include machine learning, large scale optimization, and kernel methods.

Dr. Tsang was awarded the Microsoft Fellowship in 2005, the IEEE Transactions on Neural Networks Outstanding 2004 Paper Award, and the Best Paper Award from the IEEE Hong Kong Chapter of Signal Processing Postgraduate Forum in 2006.

András Kocsor received the Ph.D. degree in computer science from the University of Szeged, Szeged, Hungary, in 2003.

Currently, he is a Senior Researcher at the Research Group on Artificial Intelligence, Hungarian Academy of Sciences, University of Szeged. His current research interests include kernel-based machine learning, similarity measures, speech recognition, speech synthesis, inequalities, and mathematics techniques applied in artificial intelligence.

James Tin-Yau Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, 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 the Hong Kong University of Science and Technology in 2000, where he is currently an Associate Professor at the Department of Computer Science and Engineering. His research interests include kernel methods, machine learning, pattern recognition, and artificial neural networks.

Dr. Kwok is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and the *Neurocomputing* journal. He received the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006.