Efficient Kernel Feature Extraction for Massive Data Sets

Ivor W. Tsang
Dept. of Computer Science
Hong Kong University of
Science and Technology
Clear Water Bay, Hong Kong
ivor@cs.ust.hk

Andras Kocsor
Dept. of Informatics
University of Szeged
H-6720 Szeged
Aradi vrt. 1., Hungary
kocsor@inf.u-szeged.hu

James T. Kwok
Dept. of Computer Science
Hong Kong University of
Science and Technology
Clear Water Bay, Hong Kong
jamesk@cs.ust.hk

ABSTRACT

Maximum margin discriminant analysis (MMDA) was proposed that uses the margin idea for feature extraction. It often outperforms traditional methods like kernel principal component analysis (KPCA) and kernel Fisher discriminant analysis (KFD). However, as in other kernel methods, 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 machines. 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 more than an order of magnitude.

Categories and Subject Descriptors: I.2.6 [Learning]: Kernel Methods; I.5.2 [Design Methodology]: Feature Extractions.

General Terms: Algorithms.

Keywords: Kernel Feature Extraction, SVM, Scalability.

1. INTRODUCTION

Superfluous features are abundant in real-world data. It is thus often worthwhile to perform dimensionality reduction that maps the original features to a new lower-dimensional feature space, while ensuring that the overall structure of the data points remains intact. A popular example is kernel principal component analysis (KPCA) [9]. While KPCA is unsupervised, the use of supervised information as in kernel Fisher discriminant analysis (KFD) [7] can lead to even better feature extractors. In the special case where the two classes are normally distributed with the same covariance, the direction found by KFD is Bayes optimal. However, when these assumptions are not met, the KFD directions may be far from optimal.

On the other hand, SVM has good generalization performance by separating the classes with a large margin [9]. However, a single SVM is not always perfect, especially when one hyperplane may not fit the data well. Mangasarian *et al.* [6] proposed a multisurface version of the SVM that uses multiple hyperplanes to fit the patterns with both large margin and small variance. Other proposals include the combination of multiple SVMs [4].

Maximum margin discriminant analysis (MMDA) [5] is a recent method that exploits the key ideas of KFD and SVM. In contrast to KFD, MMDA does not require normality assumptions on the data. Its goal is to project the data onto the normal of the hyperplane that best separates the classes. The first MMDA feature is obtained by using the standard SVM. Then, after obtaining d orthogonal MMDA features, the $(d+1)$th feature is found by optimizing the SVM in the remaining feature subspace. As in other feature extractors, this orthogonality constraint reduces the dependence among features, and thus usually only a few features are needed. Computationally, feature extraction in MMDA is formulated as a quadratic program (QP) which is similar to that of the SVM. However, given m training patterns, a naive QP solver requires $O(m^3)$ training time and at least $O(m^2)$ space. Thus, a major challenge is how to scale this up to massive data sets.

Recently, the core vector machines (CVM) is proposed that exploits the "approximateness" in the design of SVM implementations [10]. By utilizing an approximation algorithm for the minimum enclosing ball problem in computational geometry, the CVM has an asymptotic time complexity that is *linear* in m and a space complexity that is even *independent* of m. Experiments on large classification [10] and regression [11] data sets demonstrated that the CVM is as accurate as other state-of-the-art SVM implementations, but is much faster and can handle much larger data.

In this paper, we attempt to integrate MMDA with the CVM. However, as the original CVM does not utilize orthogonal constraints on the weight vectors, the QP of MMDA is not of the form required by the CVM. Thus, we propose an extension of the MEB problem that places orthogonality 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.

The rest of this paper is organized as follows. Sections 2 and 3 provide introductions on MMDA and CVM, respectively. Section 4 then describes the proposed extension of CVM algorithm. Experimental results are presented in Section 5, and the last section gives some concluding remarks.

KDD'06, August 20–23, 2006, Philadelphia,Pennsylvania, USA.

2. MAXIMUM MARGIN DISCRIMINANT ANALYSIS (MMDA)

Given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input and $y_i \in \pm 1$ the class label[1]. The L2-SVM finds the hyperplane that maximizes the margin with minimum squared error. Its primal can be formulated as:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^m \xi_i^2 \quad : \quad y_i(\mathbf{w}'\varphi(\mathbf{x}_i) + b) \geq 1 - \xi_i. \tag{1}$$

Here, φ is the nonlinear feature map associated with kernel k, ξ_i's are the slack variables and C weights the misclassification cost. Note that $\xi_i \geq 0$ is automatically satisfied at optimality. In the following, we will denote this maximum margin separation problem by $\text{MMS}(\mathcal{D}, C)$.

MMDA extracts the features one by one. Let the features that have already been extracted be $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_s$. MMDA (Algorithm 1) requires the new feature $\mathbf{w}$ to be orthogonal to all these previous $\mathbf{w}_q$'s, i.e.,

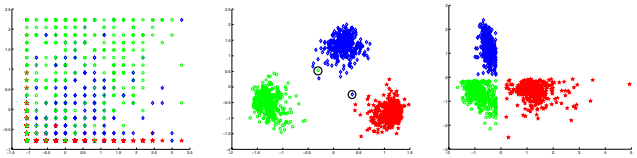
$$\min \text{MMS}(\mathcal{D}, C) \quad : \quad \mathbf{w}_q'\mathbf{w} = 0, \quad q = 1, \ldots, s. \tag{2}$$

Notice that as $\text{MMS}(\mathcal{D}, C)$ is a QP, so is (2).

Algorithm 1 Margin Maximizing Discriminant Analysis

$\mathbf{w}_1 = \text{MMS}(\mathcal{D}, C)$
for $s = 1, \ldots, d-1$ **do**
  $\mathbf{w}_{s+1} = \arg_{\mathbf{w}} \min \text{MMS}(\mathcal{D}, C) : \mathbf{w}_q'\mathbf{w} = 0, q = 1, \ldots, s.$
end for

As an illustration, Figure 1 shows the feature spaces extracted by KFD and MMDA on the Optdigits data set[2]. As can be seen, both KFD (except for the two circled points) and MMDA separate the classes well.



Original. KFD. MMDA.

Figure 1: Digits 0, 6 and 9 in the 2D feature spaces extracted by KFD and MMDA.

3. CORE VECTOR MACHINES (CVM)

Given a set of points $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, the minimum enclosing ball of $\mathcal{S}$ (denoted $\text{MEB}(\mathcal{S})$) is the smallest ball[3] $B(\mathbf{c}, R)$ in the feature space induced by the kernel k that contains all the points in $\mathcal{S}$. Its dual is the QP:

$$\max \boldsymbol{\alpha}'\text{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \quad : \quad \boldsymbol{\alpha} \geq \mathbf{0}, \ \boldsymbol{\alpha}'\mathbf{1} = 1, \tag{3}$$

where $\boldsymbol{\alpha} = [\alpha_i, \ldots, \alpha_m]'$ is the vector of Lagrange multipliers, $\mathbf{0}, \mathbf{1} \in \mathbb{R}^m$ are vectors of zeros and ones, and $\mathbf{K} = \boldsymbol{\Phi}'\boldsymbol{\Phi}$ (where $\boldsymbol{\Phi} = [\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_m)]$) is the kernel matrix. Assume that k satisfies

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

a constant, for any pattern $\mathbf{x}$. Using the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$, we have $\boldsymbol{\alpha}'\text{diag}(\mathbf{K}) = \kappa$. By dropping this constant κ from the objective in (3), we obtain a simpler QP:

$$\max -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \quad \text{s.t.} \quad \boldsymbol{\alpha} \geq \mathbf{0}, \ \boldsymbol{\alpha}'\mathbf{1} = 1. \tag{5}$$

Conversely, whenever k satisfies (4), any QP of the form (5) can be regarded as a MEB problem. This establishes an important relationship between the MEB problem and kernel methods. For example, it can be shown that the two-class L2-SVM, having training data $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$, is equivalent to finding the MEB in the feature space associated with the kernel $\tilde{k}(\mathbf{z}_i, \mathbf{z}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$ [10].

While the learning problem can be formulated as a MEB problem, this offers no immediate computational advantage as traditional algorithms for finding exact MEBs do not scale well with dimensionality. However, recently, Bădoiu and Clarkson proposed a simple yet efficient $(1+\epsilon)$-approximation algorithm using the idea of core sets [1]. Let the MEB estimate at the tth iteration be $B(\mathbf{c}_t, R_t)$. It is then 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 $B(\mathbf{c}_t, (1+\epsilon)R_t)$. By incorporating this approximation algorithm into the CVM (Algorithm 2), the resultant asymptotic time complexity is only *linear* in the training set size m while its space complexity is even *independent* of m [10].

Algorithm 2 Core Vector Machine

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

4. INTEGRATING MMDA WITH CVM

4.1 The Non-Conforming QP

A prerequisite[4] for the applicability of the CVM algorithm is that the QP must be of the form in (5). However, as will be shown in this section, this condition is not met by MMDA.

Consider (2) with the use of the L2-SVM in (1). As in the original CVM [10, 11], we slightly modify the formulation in (1) and write its primal as:

$$\min \quad \|\tilde{\mathbf{w}}\|^2 + \tilde{b}^2 - 2\rho + C\sum_{i=1}^m \tilde{\xi}_i^2 \tag{6}$$

$$\text{s.t.} \quad y_i(\tilde{\mathbf{w}}'\varphi(\mathbf{x}_i) + \tilde{b}) \geq \rho - \tilde{\xi}_i, \ i = 1, \ldots, m, \tag{7}$$

$$\mathbf{u}_q'\tilde{\mathbf{w}} = 0, \ q = 1, \ldots, s, \tag{8}$$

[1]For multi-class problems, we use the standard one-vs-all technique to convert it to multiple binary classification problems. MMDA features are then extracted from each of these pairwise classifiers.
[2]Details of this data set will be given in Table 1.
[3]Here, we denote the ball with center $\mathbf{c}$ and radius R by $B(\mathbf{c}, R)$. Moreover, we denote the center and radius of a ball B by $\mathbf{c}_B$ and r_B.
[4]Recently, this is relaxed to allow QPs of a more general form [11]. However, even this extension cannot cover the type of QPs associated with MMDA.

where $\mathbf{u}_q = \tilde{\mathbf{w}}_q/\|\tilde{\mathbf{w}}_q\|$. Introducing Lagrange multipliers $\tilde{\alpha}_i$'s and $\tilde{\gamma}_i$'s for the constraints in (7) and (8), respectively, we obtain the dual:

$$\max -\tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\mathbf{U}'\mathbf{U}\tilde{\boldsymbol{\gamma}} : \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}, \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1, \quad (9)$$

where $\tilde{\boldsymbol{\alpha}} = [\tilde{\alpha}_1, \ldots, \tilde{\alpha}_m]'$ and $\tilde{\boldsymbol{\gamma}} = [\tilde{\gamma}_1, \ldots, \tilde{\gamma}_s]'$ are the dual variables, $\mathbf{Y} = \text{diag}(y_1, \ldots, y_m)$, $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_s]$, and

$$\tilde{\mathbf{K}} = \mathbf{Y}\left(\mathbf{K} + \mathbf{1}\mathbf{1}' + \frac{1}{C}\mathbf{I}\right)\mathbf{Y}, \quad (10)$$

is the transformed "kernel" matrix. By using the Karush-Kuhn-Tucker (KKT) conditions, the primal variables

$$\tilde{\mathbf{w}} = \sum_{i=1}^{m} \tilde{\alpha}_i y_i \varphi(\mathbf{x}_i) + \sum_{q=1}^{s} \tilde{\gamma}_q \mathbf{u}_q, \;\; \tilde{b} = \sum_{i=1}^{m} \tilde{\alpha}_i y_i, \;\; \tilde{\xi}_i = \frac{\tilde{\alpha}_i}{C} \quad (11)$$

can be recovered from the optimal $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\gamma}}$. Substituting $\mathbf{u}_q = \tilde{\mathbf{w}}_q/\|\tilde{\mathbf{w}}_q\|$ back into (11) recursively, the extracted feature $\tilde{\mathbf{w}}$ is then expressed as a linear combination of $\varphi(\mathbf{x}_i)$'s.

Should it happen that all the $\tilde{\gamma}_q$'s in the optimal solution are zero, then only the first term in (9) remains and so the dual takes the form in (5). Moreover, it is easy to see that the diagonal entry of the "kernel" matrix $\tilde{\mathbf{K}}$ in (10) (which plays the role of $\mathbf{K}$ in (5)) is $[\tilde{\mathbf{K}}]_{ii} = \kappa + 1 + \frac{1}{C}$, and thus (4) is satisfied. Hence, as mentioned in Section 3, this MMDA problem can be regarded as a MEB problem. However, in general, not all $\tilde{\gamma}_q$'s are zero. Thus, (9) will not have the form of (5), and an extension of the CVM is required.

4.2 MEB with Orthogonality Constraints

Consider adding a set of constraints to the MEB problem such that the center $\mathbf{c}$ is required to be orthogonal to the existing $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_s$. We then have

$$\begin{aligned} \min \quad & R^2 && (12)\\ \text{s.t.} \quad & \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 \leq R^2, \;\; i = 1, \ldots, m, && (13)\\ & \mathbf{u}_q'\mathbf{c} = 0, \;\; q = 1, \ldots, s. && (14)\end{aligned}$$

Introducing Lagrangian multipliers α_i, γ_i for the constraints (13) and (14) respectively, the dual can be written as

$$\begin{aligned} \max \quad & \boldsymbol{\alpha}'\text{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}'\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} - \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma}\\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq \mathbf{0}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1, \quad (15)\end{aligned}$$

w.r.t. $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_m]'$ and $\boldsymbol{\gamma} = [\gamma_1, \ldots, \gamma_s]'$. Proceeding as in Section 3, we combine (4) with the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$ and obtain $\boldsymbol{\alpha}'\text{diag}(\mathbf{K}) = \kappa$. Dropping this constant from the objective in (15), we obtain the simpler problem:

$$\max -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}'\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} - \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma} : \boldsymbol{\alpha} \geq \mathbf{0}, \;\; \boldsymbol{\alpha}'\mathbf{1} = 1. \quad (16)$$

The center $\mathbf{c} = \sum_{i=1}^{m} \alpha_i \varphi(\mathbf{x}_i) + \sum_{q=1}^{s} \gamma_q \mathbf{u}_q$ and radius $R = \sqrt{\boldsymbol{\alpha}'\text{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}'\boldsymbol{\Phi}'\mathbf{U}\boldsymbol{\gamma} - \boldsymbol{\gamma}'\mathbf{U}'\mathbf{U}\boldsymbol{\gamma}}$ are then recovered from the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$. Conversely, whenever the kernel k satisfies (4), any QP of the form (16) can be regarded as a MEB problem with orthogonality constraints on the center.

Returning to MMDA's QP in (9), we can rewrite it as:

$$\begin{aligned} \max \quad & -\tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} - 2\tilde{\boldsymbol{\alpha}}'\tilde{\boldsymbol{\Phi}}'[\mathbf{U}'\;\mathbf{0}]'\tilde{\boldsymbol{\gamma}} - \tilde{\boldsymbol{\gamma}}'\mathbf{U}'\mathbf{U}\tilde{\boldsymbol{\gamma}} \quad (17)\\ \text{s.t.} \quad & \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}, \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1,\end{aligned}$$

where $\mathbf{e}_i \in \mathbb{R}^m$ is all zeros except that the ith position is equal to one, and $\tilde{\boldsymbol{\Phi}} = [\tilde{\varphi}(\mathbf{x}_1), \ldots, \tilde{\varphi}(\mathbf{x}_m)]$, with $\tilde{\varphi}(\mathbf{z}_i) = \left[y_i\varphi(\mathbf{x}_i)', y_i, \frac{y_i}{\sqrt{C}}\mathbf{e}_i'\right]$. Note that $\tilde{\boldsymbol{\Phi}}'[\mathbf{U}'\;\mathbf{0}]' = \mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}$, where $\mathbf{0}$ is the $s \times (m+1)$ zero matrix. From (3), as $\hat{k}(\mathbf{z}, \mathbf{z}) = \kappa + 1 + \frac{1}{C} \equiv \tilde{\kappa}$ is a constant, $\tilde{k}$ again satisfies (4). In other words, the optimization problem associated with MMDA in (6) can be viewed as a constrained MEB problem in (12), with φ being replaced by the new feature map $\hat{\varphi}$. Once transformed to a MEB problem, the CVM procedure (Algorithm 2) can be easily adapted to cater for the new set of constraints in (8). Thus, the approximate MEB($\mathcal{S}$) can be obtained by solving for MEB($\mathcal{S}_t$) iteratively.

4.3 Determining MEB($\mathcal{S}_t$)

Recall that finding the MEB($\mathcal{S}_t$) involves a QP. In the implementation of [10], we used an efficient decomposition method called sequential minimal optimization (SMO) [8] as the internal QP solver. However, here, the extra constraints in (14) lead to some Lagrangian multipliers $\tilde{\boldsymbol{\gamma}}_t$ that are not involved in the equality constraint of (17). This hinders the use of SMO. Moreover, the equality constraints in (17) leads to the infeasibility of $\tilde{\boldsymbol{\alpha}}_t = [\tilde{\alpha}_1^{(t)}, \ldots, \tilde{\alpha}_{t+1}^{(t)}]$ when the value of a single $\tilde{\alpha}_i^{(t)}$ is changed from a feasible $\tilde{\boldsymbol{\alpha}}_t$ [3].

Here, we propose instead an efficient incremental update algorithm for finding the MEB($\mathcal{S}_t$). Due to the lack of space, only the major steps will be outlined. By considering its optimality conditions, the dual of (16) can be solved via the kernel adatron (KA) [2], which is essentially a variant of the Gauss-Seidel (GS) iteration approach for solving the linear system [3]: $\begin{bmatrix} \tilde{\mathbf{K}} & \mathbf{Y}\Phi'\mathbf{U} \\ \mathbf{U}'\Phi\mathbf{Y} & \mathbf{U}'\mathbf{U} \end{bmatrix}\begin{bmatrix} \tilde{\boldsymbol{\alpha}} \\ \tilde{\boldsymbol{\gamma}} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix}$, s.t. $\tilde{\boldsymbol{\alpha}} \geq \mathbf{0}$ until the KKT conditions:

$$\mathbf{0} \leq \tilde{\boldsymbol{\alpha}} \perp \tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} + \mathbf{Y}\boldsymbol{\Phi}'\mathbf{U}\tilde{\boldsymbol{\gamma}} - \mathbf{1} \geq \mathbf{0} \text{ and } \mathbf{U}'\boldsymbol{\Phi}\mathbf{Y}\tilde{\boldsymbol{\alpha}} + \mathbf{U}'\mathbf{U}\tilde{\boldsymbol{\gamma}} = \mathbf{0} \quad (18)$$

are met. $[\tilde{\boldsymbol{\alpha}}'\;\tilde{\boldsymbol{\gamma}}']'$ is then normalized such that $\tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$.

To be more specific, this incremental update is used to solve the corresponding QP of the MEB($\mathcal{S}_t$), which is formed with $\tilde{\mathbf{K}}_t, \boldsymbol{\Phi}_t$ and $\mathbf{Y}_t$ defined on the core-set $\mathcal{S}_t$ analogous to $\tilde{\mathbf{K}}, \boldsymbol{\Phi}$ and $\mathbf{Y}$ on the whole training set $\mathcal{S}$. The new feasible solution of the Lagrangian multipliers for constraints in (7) of the patterns from $\mathcal{S}_t$ and the orthogonality constraints in (8): $\tilde{\boldsymbol{\alpha}}_t = [\tilde{\boldsymbol{\alpha}}_{t-1}'\;0]'$ and $\tilde{\boldsymbol{\gamma}}_t = \tilde{\boldsymbol{\gamma}}_{t-1}$ respectively, which are adapted from the optimal $\tilde{\boldsymbol{\alpha}}_{t-1}$ and $\tilde{\boldsymbol{\gamma}}_{t-1}$ of the MEB($\mathcal{S}_{t-1}$) as a warm start. For $\mathcal{S}_0$, we initialize $\tilde{\boldsymbol{\alpha}}_0 = [1]$ and $\tilde{\boldsymbol{\gamma}}_0 = \mathbf{0}$ as a feasible starting point. Then, $\tilde{\boldsymbol{\alpha}}_t$ and $\tilde{\boldsymbol{\gamma}}_t$ of the MEB($\mathcal{S}_t$) are updated by gradient descent for each variable by fixing others as in the GS iteration [3]:

$$\begin{aligned} \triangle\tilde{\gamma}_q^{(t)} &= -(\mathbf{U}'\boldsymbol{\Phi}_t\mathbf{Y}_t\tilde{\boldsymbol{\alpha}}_t + \mathbf{U}'\mathbf{U}\tilde{\boldsymbol{\gamma}}_t)_q/[\mathbf{U}'\mathbf{U}]_{qq},\\ \triangle\tilde{\alpha}_i^{(t)} &= -(\tilde{\mathbf{K}}_t\tilde{\boldsymbol{\alpha}}_t + \mathbf{Y}_t\boldsymbol{\Phi}_t'\mathbf{U}\tilde{\boldsymbol{\gamma}}_t - \mathbf{1})_i/[\tilde{\mathbf{K}}_t]_{ii},\end{aligned}$$

and the new value of $\tilde{\boldsymbol{\alpha}}_t$ are projected into the feasible region which satisfies the box constraints $\tilde{\boldsymbol{\alpha}}_t \geq \mathbf{0}$. This process is repeated until the KKT conditions in (18) defined on $\mathcal{S}_t$ are satisfied. Then, normalize $[\tilde{\boldsymbol{\alpha}}_t'\;\tilde{\boldsymbol{\gamma}}_t']'$ s.t. $\tilde{\boldsymbol{\alpha}}_t'\mathbf{1} = 1$. Afterwards, $\mathbf{c}_t$ and R_t of the MEB($\mathcal{S}_t$) can be used to find the furthest point (Step 3 of Algorithm 2) to construct $\mathcal{S}_{t+1}$.

4.4 Properties

Here, we list some properties of the modified CVM algorithm. The proofs are very similar to those in [10] and so are skipped here.

Bound on the number of iterations: *There exists a subset $\mathcal{S}_t$, with size $2/\epsilon$, of the whole training set $\mathcal{S}$ such*

that the distance between $\mathbf{c}_{MEB(\mathcal{S}_t)}$ *and any point* $\mathbf{z}_i$ *of* $\mathcal{S}$ *is at most* $(1+\epsilon)r_{MEB(\mathcal{S})}$.

Recall that one point from $\mathcal{S}$ is added to the MEB at each iteration of Step 3 (Algorithm 2), this property thus ensures that the proposed method converges in at most $2/\epsilon$ iterations, independent of the feature dimensionality and the size of $\mathcal{S}$.

Convergence to (approximate) optimality: *When* $\epsilon = 0$*, the algorithm finds the exact MMDA solution. When* $\epsilon > 0$ *and the algorithm terminates at the* τ*th iteration, and we have* $\max\left(\frac{R_\tau^2}{p^*+\bar{\kappa}}, \frac{p^*+\bar{\kappa}}{R_\tau^2}\right) \leq (1+\epsilon)^2$*, where* p^* *is the optimal value of MMDA's objective in (6).*

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

Complexities: Recall that the main motivation for using an approximation algorithm is that its time and space complexities are much smaller than those of an exact algorithm. For the proposed algorithm, it can be shown that when probabilistic speedup is used in Step 3, the total time for solving the $(s+1)$th SVM is $O\left(\frac{1}{\epsilon^2}\left(\frac{1}{\epsilon^2}+s\right)^3\right)$, while the whole algorithm takes $O(1/\epsilon^2)$ space, which are 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.

5. EXPERIMENTS

In this section, we perform experiments on a number of real-world data sets[5] (Table 1). The following feature extractors (all implemented in MATLAB) are compared: 1) the original MMDA, which is based on SMO [8]; 2) the proposed method, denoted MMDA(CVM), with probabilistic speedup and ϵ fixed[6] at 0.001; 3) kernel PCA (KPCA); and 4) kernel Fisher discriminant analysis (KFD). Methods that did not finish in 24 hours will be indicated by "–".

data set	#classes	#attribs	#tr patns	#test patns
optdigits	10	64	3,823	1,797
satimage	6	36	4,435	2,000
pendigits	10	16	7,494	3,498
letters	26	16	16,000	4,000
mnist	10	780	60,000	10,000
usps	2	676	266,079	75,383
face	2	361	346,260	24,045

Table 1: Data sets used in the experiments.

In our experiments, the C parameter in (6) is fixed at the value of 1. We use the Gaussian kernel $\exp(-\|\mathbf{x}-\mathbf{z}\|^2/\beta)$, where $\beta = \frac{1}{m^2}\sum_{i,j=1}^{m}\|\mathbf{x}_i - \mathbf{x}_j\|^2$ is the average square distance between patterns. Experiments are performed on an AMD Athlon 4400+ PC with 4GB of RAM.

5.1 Varying the Number of Features

As the performance of classification algorithms critically depend on the input features, we first examine the behavior of classification performance and extraction time of these kernel feature extractors using different numbers of extracted features. Here, experiments are only performed on the first three smaller data sets in Table 1. The classification performance is obtained from the testing accuracy of an artificial neural network (ANN) using the extracted features as input. This ANN is a feed-forward multilayer perceptron with a single layer of 10 hidden units, and training is performed via standard back-propagation. Note that the rank of the between-class matrix in KFD is at most $N_c - 1$ (where N_c is the number of classes) [7], and so the number of features for KFD is always fixed at $N_c - 1$.

Figure 2 shows that the CPU time of MMDA extraction increases with the number of extracted features. On the other hand, the CPU time for KPCA and KFA is almost fixed, as both are dominated by the eigendecomposition of the $m \times m$ kernel matrix, which is always required no matter how many features are to be extracted. Furthermore, the results also confirm that the proposed CVM-based kernel MMDA implementation is often much faster than the other feature extractors.

As for testing accuracy, the performance with KPCA features usually improves at first, and then becomes stabilized or sometimes even degraded as features with lower classification information are included. For both MMDA feature extractors, their classification performance appear to be optimal and better than the others when there are around $3N_c$ to $5N_c$ features. Hence, in the sequel, we will only conduct experiments using N_c, $3N_c$ and $5N_c$ MMDA features.

5.2 Using Decision Tree as Classifier

In this section, we experiment with another classifier, the C4.5 decision tree, on all the data sets in Table 1. Recall that both KPCA and KFD involve solving an $m \times m$ eigensystem, which is computationally expensive on large data sets. To alleviate this problem, we will only use a random sample of size 3,500 when these two methods are used on the letters, mnist, usps and face data sets. Moreover, for simplicity, we fix the number of extracted features at $5N_c$ for both KPCA and MMDA.

Results are shown in Tables 2 and 3. As can be seen, the features extracted by MMDA(CVM) often lead to small trees and high testing accuracy. As decision trees use the extracted features for node splitting. A small tree indicates that the set of extracted features carry useful classification information. Note also that KPCA and KFD perform poorly on letters, mnist and face. This demonstrates that, in general, random sampling is not a good approach to reduce the computational complexity.

data set	w/o feature extraction	KPCA	KFD	MMDA (SVM)	MMDA (CVM)
optdigits	143	117	**19**	**19**	**19**
satimage	109	113	**43**	119	79
pendigits	161	75	21	**19**	**19**
letters	777	323	**109**	449	285
mnist	1,631	197	**23**	–	619
usps	876	43	**3**	–	29
face	759	51	**3**	–	25

Table 2: Sizes of the resultant decision trees.

5.3 Using 1-NN and ANN as Classifier

In this section, we feed the extracted features to the 1-nearest neighbor (1-NN) classifier, and the artificial neural

[5]The first five data sets are from the UCI machine learning repository, while the last two are from http://www.cs.ust.hk/∼ivor/cvm.html.

[6]Preliminary results show that this fixed value of ϵ leads to both fast training and good feature extraction.

data set	w/o feature extraction	KPCA	KFD	MMDA (SVM)	MMDA (CVM)
optdigits	82.80	81.40	94.40	94.90	**95.40**
satimage	86.30	87.10	88.60	**87.90**	87.80
pendigits	89.70	89.30	95.90	**97.20**	97.10
letters	81.00	58.90	79.30	85.30	**90.30**
mnist	36.20	10.10	10.30	–	**93.30**
usps	98.00	97.00	92.50	–	**99.30**
face	96.90	80.70	65.70	–	**98.40**

Table 3: Testing accuracies (in %) obtained by the decision tree using the extracted features.

network (with the same ANN setting as in Section 5.1). Recall that the first MMDA feature is obtained by using the standard SVM. Hence, to demonstrate the usefulness of the extra MMDA features, we also compare with the standard SVM as a baseline. Moreover, as mentioned in Section 5.1, we only experiment with $N_c, 3N_c$ and $5N_c$ MMDA features.

Table 4 shows the classification accuracies. The following general observations can be made: 1) Feature extraction can improve classification accuracy. In particular, the use of MMDA features can outperform a SVM. 2) MMDA, using either the original or new implementation, leads to better classification accuracies than the other feature extraction methods. 3) For MMDA, extracting several $(3N_c - 5N_c)$ features is often beneficial. Moreover, note that KPCA and KFD sometimes perform miserably on the large data sets because of the random sampling problem mentioned in Section 5.2.

5.4 Computational Advantages

Table 5 shows the CPU time needed 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. Besides, on the three largest data sets, the original MMDA implementation cannot even converge in 24 hours, while MMDA(CVM) successfully extracts good features in only several hundred/thousand seconds. MMDA(CVM) is also always 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 that explains why MMDA appears slower. However, one should also be reminded that such a random sampling scheme also leads to poor generalization performance of KPCA/KFD in our experiments.

As mentioned in Section 4.1, each MMDA feature, in the same manner as KPCA features and KFD features, can be expressed as a linear combination of kernel evaluations. Table 6 compares the average numbers of kernel evaluations involved in the different types of features extracted. As can be seen, the MMDA(CVM) features are much sparser than the others, including the original MMDA features. As kernel evaluations often dominant the computational cost in testing, MMDA(CVM) is thus much faster.

6. CONCLUSIONS

In this paper, we investigated the problem of feature extraction in large classification tasks. Ideally, a good feature extractor should 1) produce features that can lead to a high classification accuracy; and 2) be computationally efficient during both training and testing. While the original MMDA can extract features useful for classification, it is computationally inefficient on large data sets. Here, we extended the CVM algorithm and proposed an $(1+\epsilon)^2$-approximation algorithm for extracting kernel-based MMDA features. We examined some of its theoretical aspects, and demonstrated its efficiency through various experiments. The training time complexity only depends on ϵ and, 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.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grant 615005.

7. REFERENCES

[1] M. Bădoiu and K. L. Clarkson. Optimal core-sets for balls. In *DIMACS Workshop on Computational Geometry*, 2002.

[2] T. Friess, N. Cristianini, and C. Campbell. The kernel-adatron: a fast and simple learning procedure for support vector machines. In *Proceeding of the Fifteenth International Conference on Machine Learning*, pages 188–196, 1998.

[3] W. Kienzle and B. Schölkopf. Training support vector machines with multiple equality constraints. In *Proceedings of the European Conference on Machine Learning*, 2005.

[4] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S. Bang. Constructing support vector machine ensemble. *Pattern Recognition*, 36(12):2757–2767, 2003.

[5] A. Kocsor, K. Kovács, and C. Szepesvári. Margin maximizing discriminant analysis. In *Proceedings of the 15th European Conference on Machine Learning*, pages 227–238, Pisa, Italy, Sept. 2004.

[6] O. Mangasarian and E. Wild. Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):69– 74, 2006.

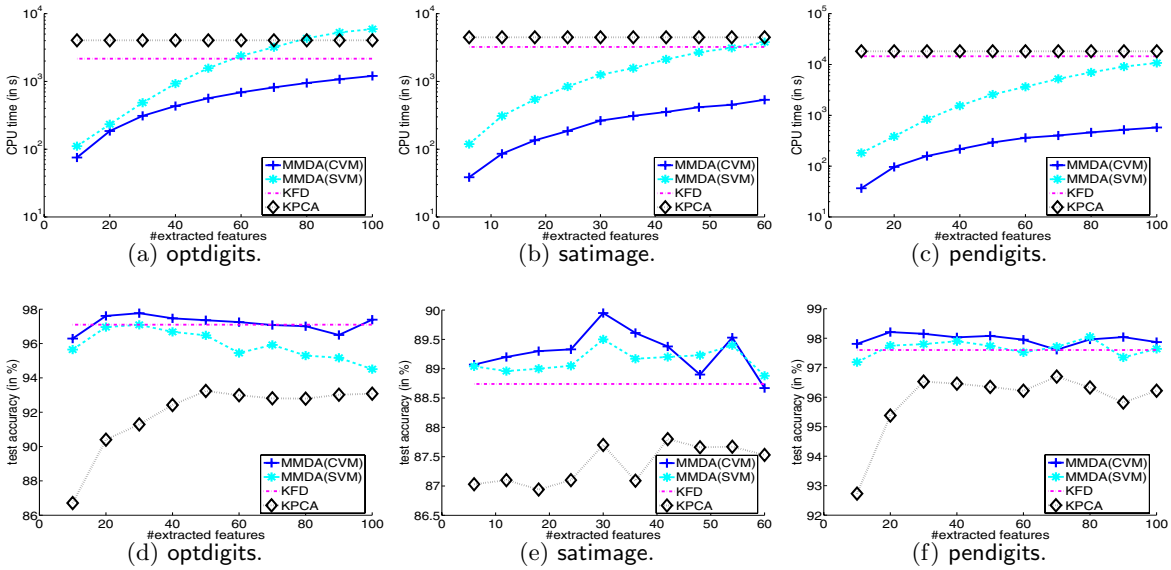
[7] S. Mika, G. Rätsch, J. Weston, B. Schoölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48, 1999.

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

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

[10] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[11] I. W. Tsang, J. T. Kwok, and K. T. Lai. Core vector regression for very large regression problems. In *Proceedings of the Twentieth-Second International Conference on Machine Learning*, pages 913–920, Bonn, Germany, Aug. 2005.



(a) optdigits. (b) satimage. (c) pendigits.

(d) optdigits. (e) satimage. (f) pendigits.

Figure 2: Performance at different numbers of extracted features. Top: CPU time; Bottom: Testing accuracy.

feature extractor		classifier	optdigits	satimage	pendigits	letters	mnist	usps	face
w/o feature extraction		SVM	96.66	89.60	96.74	90.95	–	–	–
		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
KPCA		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
KFD		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
MMDA(SVM)	#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	–	–	–
MMDA(CVM)	#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

Table 4: Testing accuracies on the various data sets.

feature extractor		optdigits	satimage	pendigits	letters	mnist	usps	face
KPCA		2,632	1,804	1,680	2,029	2,639	4,479	2,216
KFD		1,340	1,339	1,335	1,340	1,297	2,038	1,674
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	–	–	–
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

Table 5: CPU time (in seconds) required in the feature extraction process.

feature extractor		optdigits	satimage	pendigits	letters	mnist	usps	face
KPCA		3,823	4,435	7,494	3,500	3,500	3,500	3,500
KFD		3,823	4,435	7,494	3,500	3,500	3,500	3,500
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	–	–	–
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

Table 6: Average number of kernel evaluations involved in each extracted feature.