Building Sparse Multiple-Kernel SVM Classifiers

Mingqing Hu, Yiqiang Chen, and James Tin-Yau Kwok

***Abstract*—The support vector machines (SVMs) have been very successful in many machine learning problems. However, they can be slow during testing because of the possibly large number of support vectors obtained. Recently, Wu *et al.* (2005) proposed a sparse formulation that restricts the SVM to use a small number of expansion vectors. In this paper, we further extend this idea by integrating with techniques from multiple-kernel learning (MKL). The kernel function in this sparse SVM formulation no longer needs to be fixed but can be automatically learned as a linear combination of kernels. Two formulations of such sparse multiple-kernel classifiers are proposed. The first one is based on a convex combination of the given base kernels, while the second one uses a convex combination of the so-called "equivalent" kernels. Empirically, the second formulation is particularly competitive. Experiments on a large number of toy and real-world data sets show that the resultant classifier is compact and accurate, and can also be easily trained by simply alternating linear program and standard SVM solver.**

***Index Terms*—Gradient projection, kernel methods, multiple-kernel learning (MKL), sparsity, support vector machine (SVM).**

I. Introduction

THE support vector machine (SVM), which is solidly based on the theory of structural risk minimization in statistical learning [23], has outperformed many traditional learning algorithms. It is now generally recognized as a powerful tool for various machine learning problems [8], [20], [19]. As is well known, the SVM first maps the inputs to a high-dimensional (possibly infinite-dimensional) feature space and then finds a large margin hyperplane between the two classes. Computationally, this leads to a quadratic programming (QP) problem, which can be easily solved even with off-the-shelf optimization software. Moreover, the SVM relies only on the dot product in the feature space, which can be computed efficiently via the so-called *kernel* trick.

Manuscript received March 17, 2008; revised January 04, 2009; accepted January 19, 2009. First published March 31, 2009; current version published May 01, 2009. This work was supported in part by the National High Technology Research and Development Program of China (863 Program) under Grant 2007AA01Z305, by the National Natural Science Foundation of China under Grant 60775027, and by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Competitive Earmarked Research Grant (CERG) 614907.

M. Hu and Y. Chen are with the Institute of Computing Technology (ICT), Chinese Academic of Sciences (CAS), Beijing 100080, China (e-mail: humingqing@ict.ac.cn; yqchen@ict.ac.cn).

J. T.-Y. Kwok is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: jamesk@cs.ust.hk).

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.2009.2014229

An essential ingredient of the SVM and other kernel methods is the kernel. In principle, the kernel can be chosen by standard model selection methods such as cross validation. However, recent research has focused on developing more efficient kernel optimization algorithms [1]–[3], [10]–[12], [14]–[17], [21], [22], [26]. Often, the key idea is to learn the kernel from a prescribed family of kernels $\mathcal{K}$, which is often required to be convex. An objective function defined on the kernel is then specified and optimized over $\mathcal{K}$. As a result, the obtained classifier may include multiple homogeneous or even heterogeneous kernels, and this framework is usually called multiple-kernel learning (MKL) [3], [16], [26].

Besides having to choose the kernel, another potential disadvantage of the SVM is that it is often not as compact as the other classifiers such as neural networks. This in turn results in slower prediction on testing. Recently, this problem has been studied extensively for single-kernel SVM classifiers. Various remedies are introduced, such as the reduced set method [6], [19], bottom-up method [13], building of a sparse large margin classifier [24], [25], and the incremental building of a reduced-complexity classifier [9]. In particular, the reduced set and bottom-up methods can be viewed as postprocessing techniques, where an SVM has to be first trained. On the other hand, approaches in [9], [24], and [25] attempt to obtain a sparse SVM directly by modifying the SVM's optimization problem.

In this paper, we will focus on the sparse SVM formulation proposed in [24] and [25]. In order to obtain both a good kernel and a compact representation of the SVM, we integrate MKL methods with the sparse single-kernel SVM in [24] and [25]. We propose two methods to construct a sparse multiple-kernel SVM, where the kernel can be automatically selected during the optimization process.

The rest of this paper is organized as follows. Introductions to MKL and the sparse single-kernel SVM are provided in Sections II and III, respectively. Section IV then describes the proposed sparse multiple-kernel SVM formulations. Experimental results are presented in Section V, and the last section gives some concluding remarks.

In the sequel, vectors and matrices are denoted in boldface. Moreover, the transpose of a vector/matrix (in both the input and feature spaces) will be denoted by the superscript ^{T}.

II. Multiple-Kernel Learning

In a binary classification problem, we are given ℓ pairs of training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{\ell}$, where $\boldsymbol{x_i} \in \mathcal{X}$ (the input space) and $y_i \in \{\pm 1\}$. Each $\boldsymbol{x}$ in $\mathcal{X}$ is then mapped to a $\Phi(\boldsymbol{x})$ in the kernel-induced feature space, which is related to the kernel function K by $\Phi(\boldsymbol{x})^T\Phi(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}')$ for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. The standard SVM tries to find a hyperplane $\boldsymbol{\omega}^T\Phi(\boldsymbol{x}) + b$ that has large margin and small training error. Mathematically, this leads to the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{\omega}^T\Phi(\boldsymbol{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0, \qquad i=1,2,\ldots,\ell. \end{aligned} \tag{1}$$

Here, $\boldsymbol{\xi} = [\xi_1,\ldots,\xi_\ell]^T$ is the vector of slack variables for the errors, and C is a user-defined regularization parameter that trades off the margin with error.

Instead of having a single kernel K, suppose that we have a set of M base kernels $K_1,\ldots,K_M$, with the corresponding kernel-induced feature maps $\Phi_1,\ldots,\Phi_M$. The following MKL formulation is first introduced in [3] and then further developed in [16], [17], and [21]:

$$\begin{aligned}\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\left(\sum_{k=1}^{M}\|\boldsymbol{\omega}_k\|_2\right)^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{s.t.} \quad & y_i\left(\sum_{k=1}^{M}\boldsymbol{\omega}_k^T\Phi_k(\boldsymbol{x}_i)+b\right) \geq 1-\xi_i, \\ & \xi_i \geq 0, \qquad i=1,2,\ldots,\ell \end{aligned} \tag{2}$$

where $\boldsymbol{\omega} = \{\boldsymbol{\omega}_1,\ldots,\boldsymbol{\omega}_M\}$ and $\boldsymbol{\omega}_k$ is the weight for component Φ_k. As shown in [3], the optimality conditions require these $\boldsymbol{\omega}_k$'s to be written as

$$\boldsymbol{\omega}_k = \mu_k\sum_{i=1}^{\ell}\alpha_i y_i \Phi_k(\boldsymbol{x}_i)$$

where $0 \leq \alpha_i \leq C$, and

$$\mu_k \geq 0, \qquad k=1,2,\ldots,M \quad \text{and} \quad \sum_{k=1}^{M}\mu_k = 1. \tag{3}$$

Note that $\boldsymbol{\omega}$ in (2) is regularized with a mixed (ℓ_2,ℓ_1)-norm instead of the standard ℓ_2-norm in (1). As discussed in [21], the ℓ_1 part of this mixed norm encourages the sparsity of $\boldsymbol{\omega}$ at the kernel level, i.e., sparsity in the $\boldsymbol{\mu} = [\mu_1,\ldots,\mu_M]^T$ obtained. Moreover, the resultant kernel K is a convex combination of the base kernels K_i's

$$K = \sum_{k=1}^{M}\mu_k K_k. \tag{4}$$

From the learned $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$, the resultant decision function can be obtained as

$$f(\boldsymbol{x}) = \sum_{k:\mu_k>0}\mu_k\sum_{i:\alpha_i>0}\alpha_i y_i K_k(\boldsymbol{x},\boldsymbol{x}_i) + b.$$

The number of numerical operations required in predicting $\boldsymbol{x}$ is thus proportional to the product of the number of support vectors and the number of active base kernels selected.

Note that the mixing coefficients μ_k's are a product of the optimality conditions of (2) and do not appear explicitly in the optimization problem. To make these coefficients of the convex combination explicit, we follow another MKL formulation proposed recently in [26]

$$\begin{aligned}\min_{\boldsymbol{\mu}}\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\sum_{k=1}^{M}\mu_k\|\boldsymbol{\omega}_k\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{s.t.} \quad & y_i\left(\sum_{k=1}^{M}\mu_k\boldsymbol{\omega}_k^T\Phi_k(\boldsymbol{x}_i)+b\right) \geq 1-\xi_i, \\ & \xi_i \geq 0, \qquad i=1,2,\ldots,\ell \\ & \sum_{k=1}^{M}\mu_k = 1 \\ & \mu_k \geq 0, \qquad k=1,2,\ldots,M. \end{aligned} \tag{5}$$

Similar to the ℓ_1 part of the mixed norm in (2), the ℓ_1 constraint on $\boldsymbol{\mu}$ encourages a sparse $\boldsymbol{\mu}$ solution, and thus the resultant classifier selects a sparse combination of base kernels. Interestingly, for the binary classification scenario considered in this paper, it can be shown that the dual of (5) is identical to the dual of (2) [26].

On the computational aspect, a number of techniques have been proposed to solve the apparently more challenging optimization problem. Examples include sequential minimization optimization (SMO) [3], convex quadratically constrained quadratic programming (QCQP) [26], semi-infinite linear programming (SILP) [26], projected gradient [16], and reduced gradient methods [17].

III. Sparse Single-Kernel SVM

To build a sparse SVM classifier, Wu *et al.* [24], [25] modified the primal problem in (1) to:

$$\begin{aligned}\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{\omega}^T\Phi(\boldsymbol{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0, \qquad i=1,\ldots,\ell \\ & \boldsymbol{\omega} = \sum_{j=1}^{N_{xv}}\beta_j\Phi(\boldsymbol{z}_j) \end{aligned} \tag{6}$$

where $\boldsymbol{z}_i$'s are called the *expansion vectors* (XVs). The constraint in (6) is used to explicitly control the sparsity of SVM. Substituting (6) into the objective and constraints, we obtain

$$\begin{aligned}\min_{\boldsymbol{\beta},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{\beta}^T\mathbf{K}_z\boldsymbol{\beta} + C\sum_{i=1}^{\ell}\xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{\beta}^T\Phi^z(\boldsymbol{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0, \qquad i=1,\ldots,\ell \end{aligned} \tag{7}$$

where $\boldsymbol{\beta} = [\beta_1,\ldots,\beta_{N_{xv}}]^T$

$$\Phi^z(\boldsymbol{x}_i) = [\Phi(\boldsymbol{z}_1)^T\Phi(\boldsymbol{x}_i),\ldots,\Phi(\boldsymbol{z}_{N_{xv}})^T\Phi(\boldsymbol{x}_i)]^T \tag{8}$$

and $\mathbf{K}_z$ is the $N_{xv}\times N_{xv}$ kernel matrix defined on the $\boldsymbol{z}_i$'s

$$[\mathbf{K}_z]_{ij} = K(\boldsymbol{z}_i,\boldsymbol{z}_j). \tag{9}$$

In the sequel, we assume that $\mathbf{K}_z$ is positive definite (pd) and thus invertible. This can be easily enforced by adding a small ridge to its diagonal.

The dual of (7) can be easily shown to be

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j K^z(\boldsymbol{x}_i,\boldsymbol{x}_j)\\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \qquad i=1,\ldots,\ell\\ & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \end{aligned} \tag{10}$$

where

$$K^z(\boldsymbol{x}_i,\boldsymbol{x}_j) = \Phi^z(\boldsymbol{x}_i)^T\mathbf{K}_z^{-1}\Phi^z(\boldsymbol{x}_j) \tag{11}$$

is often called the "equivalent" kernel. It is easy to see that the matrix $[K^z(\boldsymbol{x}_i,\boldsymbol{x}_j)]$ is pd. Thus, for a fixed set of expansion vectors $\boldsymbol{z}_i,\ldots,\boldsymbol{z}_{N_{xv}}$, problem (10) can be simply solved by training a standard SVM with kernel K^z. Moreover, $\boldsymbol{\beta}$ can be recovered from the learned $\boldsymbol{\alpha}$ as

$$\boldsymbol{\beta} = \mathbf{K}_z^{-1}\sum_{i=1}^{\ell}\alpha_i y_i \Phi^z(\boldsymbol{x}_i). \tag{12}$$

To learn the expansion vectors z_i's, they are also treated as variables to be optimized in (10), leading to

$$\begin{aligned}\min_{\boldsymbol{Z}}\max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i\alpha_j y_i y_j K^z(\boldsymbol{x}_i,\boldsymbol{x}_j)\\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \qquad i=1,\ldots,\ell\\ & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \end{aligned} \tag{13}$$

where $\boldsymbol{Z} = \{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_{N_{xv}}\}$. Consider the inner optimization subproblem over $\boldsymbol{\alpha}$ [i.e., problem (10)], and define its optimal objective value as a function $W(\boldsymbol{Z})$ of $\boldsymbol{Z}$. Because of the constraint (6), $W(\boldsymbol{Z})$ is nonconvex and so we perform gradient descent on $W(\boldsymbol{Z})$. Since $\mathbf{K}_z$ is pd, the kernel matrix $[K^z(\boldsymbol{x}_i,\boldsymbol{x}_j)]$ is also pd. By using a theorem[1] due to Bonnans and Shapiro [4], the relevant gradients of $W(\boldsymbol{Z})$ can be obtained by the following lemma [24], [25].

Lemma III.1: Let z_{uv} be the vth component of $\boldsymbol{z}_u$ $(1 \le u \le N_{xv}, 1 \le v \le d)$. The derivative of $W(\boldsymbol{Z})$ with respect to (w.r.t.) z_{uv} is

$$\frac{\partial W(\boldsymbol{Z})}{\partial z_{uv}} = -\frac{1}{2}\sum_{i,j=1}^{\ell}\alpha_i^z\alpha_j^z y_i y_j \frac{\partial K^z(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\partial z_{uv}} \tag{14}$$

where α_i^z maximizes (10) for the given $\boldsymbol{Z}$.

In other words, the partial derivative of $W(\boldsymbol{Z})$ is computed as if $\boldsymbol{\alpha}^z = [\alpha_i^z]$ is not dependent on $\boldsymbol{Z}$. The $(\partial K^z(\boldsymbol{x}_i,\boldsymbol{x}_j))/(\partial z_{uv})$ term in (14) can in turn be obtained from (11), as

$$\begin{aligned}\frac{\partial K^z(\boldsymbol{x}_i,\boldsymbol{x}_j)}{\partial z_{uv}} = & \frac{\partial \Phi^z(\boldsymbol{x}_i)}{\partial z_{uv}}^T\mathbf{K}_z^{-1}\Phi^z(\boldsymbol{x}_j) + \Phi^z(\boldsymbol{x}_i)^T\\ & \times \frac{\partial \mathbf{K}_z^{-1}}{\partial z_{uv}}\Phi^z(\boldsymbol{x}_j) + \Phi^z(\boldsymbol{x}_i)^T\mathbf{K}_z^{-1}\frac{\partial \Phi^z(\boldsymbol{x}_j)}{\partial z_{uv}}\end{aligned}$$

and

$$\frac{\partial \mathbf{K}_z^{-1}}{\partial z_{uv}} = -\mathbf{K}_z^{-1}\frac{\partial \mathbf{K}_z}{\partial z_{uv}}\mathbf{K}_z^{-1}.$$

[1]For completeness, this theorem is reproduced in the Appendix.

The optimization algorithm then alternates between standard SVM training [to obtain $\boldsymbol{\alpha}$ from (10) for a fixed $\boldsymbol{Z}$] and gradient descent on $W(\boldsymbol{Z})$ (to obtain $\boldsymbol{Z}$).

Finally, on using (6) and (12), the decision function can be obtained as

$$f(\boldsymbol{x}) = \sum_{i:\alpha_i>0}\alpha_i y_i \Phi^z(\boldsymbol{x}_i)^T\mathbf{K}_z^{-1}\Phi^z(\boldsymbol{x}) + b.$$

Since $\sum_{i=1}^{\ell}\alpha_i y_i\Phi^z(\boldsymbol{x}_i)^T\mathbf{K}_z^{-1}$ can be precomputed and $\Phi^z(\boldsymbol{x}) \in \mathbb{R}^{N_{xv}}$, prediction on $\boldsymbol{x}$ involves only N_{xv} multiplications and additions.

IV. Sparse Multiple-Kernel SVM

As mentioned in Section I, because of the central role of the kernel function, a good choice of the kernel is imperative to the success of any kernel method. Naturally, this also holds for the sparse single-kernel SVM in Section III. Note that while learning of the expansion vectors will adapt the corresponding "equivalent" kernel K^z in (11) and so can be viewed as some form of kernel learning, one cannot expect K^z to be a good kernel unless the original kernel K is good. Indeed, recall that the main motivation for having a sparse single-kernel SVM is to obtain a compact, but not necessarily more accurate, classifier. Hence, if the original SVM classifier (with an inappropriately chosen kernel) does not perform well, one should not expect that its sparsified version will be an accurate classifier. So, instead of using a single, carefully selected kernel, we will borrow MKL techniques in Section II to the construction of sparse SVM classifiers. Consequently, not only can we obtain a compact classifier, but also an appropriate kernel (in the form of a convex combination of some given kernels) can be automatically determined during the optimization process.

In Section IV-A, we proceed by adding an explicit sparsity constraint to the multiple-kernel SVM, in a manner analogous to that discussed in Section III. In other words, the desired kernel is simply a convex combination of the given base kernels. However, a problem with this formulation is that the resultant optimization problem is quite complicated. In Section IV-B, we provide another formulation by utilizing the "equivalent" kernels introduced in Section III. Similar to the sparse single-kernel SVM, both extensions require the learning of expansion vectors, and this will be discussed in Section IV-C.

A. Convex Combination of Base Kernels

In this section, we consider the use of a convex combination of M kernels $K_1,\ldots,K_M$, with the corresponding kernel-induced feature maps $\Phi_1,\ldots,\Phi_M$. Using the MKL formulation in (5), for a fixed $\boldsymbol{\mu} = [\mu_i]$, the primal in (6) is extended to

$$\begin{aligned}\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\sum_{k=1}^{M}\mu_k\|\boldsymbol{\omega}_k\|^2 + C\sum_{i=1}^{\ell}\xi_i\\ \text{s.t.} \quad & y_i\left(\sum_{k=1}^{M}\mu_k\boldsymbol{\omega}_k^T\Phi_k(\boldsymbol{x}_i) + b\right) \ge 1-\xi_i,\\ & \xi_i \ge 0, \qquad i=1,\ldots,\ell\\ & \boldsymbol{\omega}_k = \sum_{j=1}^{N_{xv}}\beta_j\Phi_k(\boldsymbol{z}_j)\end{aligned} \tag{15}$$

where $\boldsymbol{\omega} = [\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_M]^T$. Obviously, when $M = 1$, this reduces to that of the sparse single-kernel SVM.

Proceeding as in Section III, we put $\boldsymbol{\omega}_k = \sum_{j=1}^{N_{xv}} \beta_j \Phi_k(\boldsymbol{z}_j)$ back into the objective and constraints, and obtain

$$\begin{aligned} \min_{\boldsymbol{\beta}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{\beta}^T \tilde{\mathbf{K}}_z \boldsymbol{\beta} + C\sum_{i=1}^{\ell} \xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{\beta}^T \tilde{\Phi}^z(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \qquad i = 1, \ldots, \ell \end{aligned}$$

where the kernel matrix $\tilde{\mathbf{K}}_z \in \mathbb{R}^{N_{xv} \times N_{xv}}$ is now

$$[\tilde{\mathbf{K}}_z]_{ij} = \sum_{k=1}^{M} \mu_k K_k(\boldsymbol{z}_i, \boldsymbol{z}_j)$$

and

$$\begin{aligned} \tilde{\Phi}^z(\boldsymbol{x}_i) &= \sum_{k=1}^{M} \mu_k [\Phi_k(\boldsymbol{z}_1)^T \Phi_k(\boldsymbol{x}_i), \ldots, \Phi_k(\boldsymbol{z}_{N_{xv}})^T \Phi_k(\boldsymbol{x}_i)]^T \\ &= \sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_i) \end{aligned}$$

with $\Phi_k^z(\boldsymbol{x}_i) = [K_k(\boldsymbol{z}_1, \boldsymbol{x}_i), \ldots, K_k(\boldsymbol{z}_{N_{xv}}, \boldsymbol{x}_i)]^T$. Its dual can be easily shown to be

$$\begin{aligned} W_z(\boldsymbol{\mu}) = \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, \ell \\ & \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \end{aligned} \tag{16}$$

where

$$\tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_i)^T\right) \tilde{\mathbf{K}}_z^{-1} \left(\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_j)\right) \tag{17}$$

and we have denoted the optimal objective value of this optimization problem as a function of $\boldsymbol{\mu}$.

Finally, to learn $\boldsymbol{\mu}$, we also minimize w.r.t. the mixing coefficients $\boldsymbol{\mu}$ of the convex combination, leading to

$$\begin{aligned} \min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \qquad i = 1, \ldots, \ell \\ & \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \\ & \mu_k \geq 0, \qquad k = 1, \ldots, M \\ & \sum_{k=1}^{M} \mu_k = 1. \end{aligned} \tag{18}$$

Note that because of the presence of the ℓ_1 constraint $\sum_{k=1}^{M} \mu_k = 1$, we will have a sparse $\boldsymbol{\mu}$ solution.

However, $\tilde{K}^z$ in (17) is neither linear nor convex in $\mu_1, \ldots, \mu_M$. Hence, even for fixed $\boldsymbol{\alpha}$ and $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N_{xv}}]^T$, one cannot reduce (18) to a convex QP and solve for the μ_i's. Instead, we will employ the method of gradient projection [18]. First, by again using Theorem 1 (in the Appendix) as in Section III, the gradient of $W_z(\boldsymbol{\mu})$ in (16) w.r.t. $\boldsymbol{\mu}$ can be computed in an analogous manner as follows.

Lemma IV.1: The derivative of $W_z(\boldsymbol{\mu})$ in (16) w.r.t. $\boldsymbol{\mu}$ is

$$\frac{\partial W_z(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = \left[\frac{\partial W_z(\boldsymbol{\mu})}{\partial \mu_1}, \frac{\partial W_z(\boldsymbol{\mu})}{\partial \mu_2}, \ldots, \frac{\partial W_z(\boldsymbol{\mu})}{\partial \mu_M}\right]^T \tag{19}$$

where

$$\frac{\partial W_z(\boldsymbol{\mu})}{\partial \mu_m} = -\frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i^{\mu} \alpha_j^{\mu} y_i y_j \frac{\partial \tilde{\mathbf{K}}^z(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \mu_m} \tag{20}$$

and α_i^{μ} maximizes (16) for the given $\boldsymbol{\mu}$.

From (17), the partial derivative of $\tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j)$ w.r.t. $\mu_m (m = 1, 2, \ldots, M)$ in (20) can be computed as

$$\begin{aligned} \frac{\partial \tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \mu_m} = & \left(\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_i)^T\right) \frac{\partial \tilde{\mathbf{K}}_z^{-1}}{\partial \mu_m} \left(\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_j)\right) \\ & + \Phi_m^z(\boldsymbol{x}_i)^T \tilde{\mathbf{K}}_z^{-1} \left(\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_j)\right) \\ & + \left(\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_i)^T\right) \tilde{\mathbf{K}}_z^{-1} \Phi_m^z(\boldsymbol{x}_j) \end{aligned}$$

where

$$\begin{aligned} \frac{\partial \tilde{\mathbf{K}}_z^{-1}}{\partial \mu_m} &= -\tilde{\mathbf{K}}_z^{-1} \frac{\partial \tilde{\mathbf{K}}_z}{\partial \mu_m} \tilde{\mathbf{K}}_z^{-1} \\ &= -\tilde{\mathbf{K}}_z^{-1} \tilde{\mathbf{K}}_z^{m} \tilde{\mathbf{K}}_z^{-1} \end{aligned}$$

and

$$[\tilde{\mathbf{K}}_z^{m}]_{ij} = K_m(\boldsymbol{z}_i, \boldsymbol{z}_j), \qquad m = 1, 2, \ldots, M.$$

If the constraints on $\boldsymbol{\mu}$ in (18) (i.e., $\sum_{k=1}^{M} \mu_k = 1$ and $\mu_k \geq 0$) do not exist, the search direction will simply be the negative of the gradient in (19). However, to cater for these constraints, we use the gradient projection method [18] that projects the negative gradient onto the subspace orthogonal to the subspace defined by the active inequality constraints and equality constraint. Moving in this direction not only guarantees a decrease of the objective value, but also ensures that both the inequality and equality constraints are satisfied. We then alternate between gradient projection and standard SVM training.

Finally, from the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$, the decision function $f(\boldsymbol{x})$ can be obtained as

$$\sum_{i:\alpha_i > 0} \alpha_i y_i \left(\sum_{k:\mu_k > 0} \mu_k \Phi_k^z(\boldsymbol{x}_i)\right)^T \tilde{\mathbf{K}}_z^{-1} \left(\sum_{k:\mu_k > 0} \mu_k \Phi_k^z(\boldsymbol{x})\right) + b. \tag{21}$$

Recall that $\Phi_k^z(\boldsymbol{x}) \in \mathbb{R}^{N_{xv}}$, then the number of numerical operations required in computing $f(\boldsymbol{x})$ is thus proportional to the product of the number of expansion vectors N_{xv} and the number of active base kernels.

B. Convex Combination of "Equivalent" Kernels

As can be seen in the previous section, the kernel function in (17) leads to a complicated expression for the gradient. In this section, we provide another formulation of the sparse multiple-kernel SVM, which is based on the "equivalent" kernels introduced in Section III.

Recall that the "equivalent" kernel K^z in (11) plays a key role in deriving the sparse single-kernel classifier in Section III. Given M base kernels $K_1, \dots, K_M$, we have M corresponding equivalent kernels $K_1^z, \dots, K_M^z$. As information about the expansion vectors has already been absorbed into these equivalent kernels, we now consider constructing a new kernel based on a convex combination of them. In other words, we assume that

$$\tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{k=1}^{M} \mu_k K_k^z(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{22}$$

where

$$\begin{aligned} K_k^z(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \Phi_k^z(\boldsymbol{x}_i)^T \left(\mathbf{K}_z^k\right)^{-1} \Phi_k^z(\boldsymbol{x}_j) \\ \left[\mathbf{K}_z^k\right]_{ij} &= K_k(\boldsymbol{z}_i, \boldsymbol{z}_j) \\ \Phi_k^z(\boldsymbol{x}_i) &= \left[\Phi_k(\boldsymbol{z}_1)^T \Phi_k(\boldsymbol{x}_i), \dots, \Phi_k\left(\boldsymbol{z}_{N_{xv}}\right)^T \Phi_k(\boldsymbol{x}_i)\right]^T \\ &= \left[K_k(\boldsymbol{z}_1, \boldsymbol{x}_i), \dots, K_k(\boldsymbol{z}_{N_{xv}}, \boldsymbol{x}_i)\right]^T. \end{aligned} \tag{23}$$

Note that, unlike the kernel K^z in (17), this new $\tilde{K}^z$ is now linear in μ_k's. This will be very useful later in the optimization.

Putting the new multiple kernel (22) into problem (10), we obtain

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \sum_{k=1}^{M} \mu_k K_k^z(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \qquad i = 1, \dots, \ell \\ & \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \end{aligned} \tag{24}$$

and we denote its optimal objective value as a function $W_z(\boldsymbol{\mu})$ of $\boldsymbol{\mu}$. One can also derive the primal of (24) as

$$\begin{aligned} \min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C \sum_{i=1}^{\ell} \xi_i \\ \text{s.t.} \quad & y_i(\boldsymbol{\omega}^T \Phi^z(\boldsymbol{x}_i) + b) \ge 1 - \xi_i, \\ & \xi_i \ge 0, \qquad i = 1, \dots, \ell. \end{aligned} \tag{25}$$

This is almost identical to (1) except that

$$\Phi^z(\boldsymbol{x}_i)^T \Phi^z(\boldsymbol{x}_j) = \sum_{k=1}^{M} \mu_k K_k^z(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

On also optimizing $\boldsymbol{\mu}$ as in Section IV-A, we have

$$\begin{aligned} \min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \sum_{k=1}^{M} \mu_k K_k^z(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & 0 \le \alpha_i \le C, \qquad i = 1, \dots, \ell \\ & \boldsymbol{y}^T \boldsymbol{\alpha} = 0, \\ & \mu_k \ge 0, \qquad k = 1, \dots, M \\ & \sum_{k=1}^{M} \mu_k = 1. \end{aligned} \tag{26}$$

Note that for a fixed $\boldsymbol{\mu}$, (26) reduces to a standard SVM optimization problem (with kernel $\tilde{K}^z$) for the solving of $\boldsymbol{\alpha}$. For a fixed $\boldsymbol{\alpha}$, since $\tilde{K}^z$ is now linear in μ_k's, both the objective and constraints in (26) are linear in $\mu_1, \dots, \mu_M$ and so (26) reduces to a standard linear programming (LP) for $\boldsymbol{\mu}$. Hence, unlike the formulation considered in Section IV-A, here we can learn a sparse multiple-kernel classifier by simply alternating between LP and standard SVM training.

As an aside, instead of using an LP solver to solve for $\boldsymbol{\mu}$ (with a fixed $\boldsymbol{\alpha}$), one can also use gradient projection as in Section IV-A. In this case, the mth component of the gradient of $W_z(\boldsymbol{\mu})$ in (24) is

$$\frac{\partial W_z(\boldsymbol{\mu})}{\partial \mu_m} = -\frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i^{\mu} \alpha_j^{\mu} y_i y_j K_m^z(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{27}$$

where $\alpha_i^{\mu}, i = 1, 2, \dots, M$, maximizes (24) for the given $\boldsymbol{\mu}$.

From the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$, the decision function $f(\boldsymbol{x})$ can be obtained as

$$\sum_{i:\alpha_i > 0} \alpha_i y_i \sum_{k:\mu_k > 0} \mu_k \Phi_k^z(\boldsymbol{x}_i)^T (\mathbf{K}_z^k)^{-1} \Phi_k^z(\boldsymbol{x}) + b. \tag{28}$$

Recall that $\mathbf{K}_z^k \in \mathbb{R}^{N_{xv} \times N_{xv}}$ and $\Phi_k^z(\boldsymbol{x}_i) \in \mathbb{R}^{N_{xv}}$. Hence, again as in Section IV-A, the number of numerical operations required in computing $f(\boldsymbol{x})$ is proportional to the product of the number of expansion vectors N_{xv} and the number of active base kernels.

As a short summary, we have presented two different approaches to design a sparse multiple-kernel SVM. The previous section first constructs a linear combination of kernels and then derives the equivalent (sparse) multiple kernel in (17), while this section first constructs the individual equivalent kernels and then combines them in a convex manner to form (22). From the computational perspective, the kernel in (17) and the subsequent optimization are more complicated as the mixing coefficients $\mu_1, \dots, \mu_M$ appear nonlinearly in the kernel. On the other hand, the kernel in (22) is linear in $\boldsymbol{\mu}$ and the resultant classifier can be more easily trained by alternating linear program and standard SVM solver.

C. Learning the Expansion Vectors

In this section, we address the issue of learning the expansion vectors $\boldsymbol{Z} = \{z_i\}$. The general approach applies to both formulations in Sections IV-A and IV-B. Let $W(\boldsymbol{Z})$ denote the optimal objective value function [(18) in Section IV-A or (26) in Section IV-B] as a function of $\boldsymbol{Z}$. As in Section III, we will use gradient descent to solve this nonconvex optimization problem and thus require the computation of the gradient $\nabla W(\boldsymbol{Z})$. However, in this multiple-kernel case, the joint optimization problem over $(\boldsymbol{\mu}, \boldsymbol{\alpha})$ is not convex and so Theorem 1 cannot be applied directly for computing the gradient. Recall that, as mentioned in Section III, the motivation of employing the gradient to update $\boldsymbol{Z}$ is to minimize the objective function $W(\boldsymbol{Z})$. Hence, updating of $\boldsymbol{Z}$ in the sparse MKL problems should follow the same principle. Thus, we design an updating procedure (shown in Algorithm 1) to minimize the objective function $W(\boldsymbol{Z})$. The main idea is to assume that $\boldsymbol{\mu}$ is constant when updating $\boldsymbol{Z}$. Then, Theorem 1 can be used to calculate the gradient $\nabla W(\boldsymbol{Z})$. Moreover,

TABLE I
DATA SETS USED IN THE EXPERIMENTS

data set	dimensionality	#training patterns	#test patterns
Banana	2	400	4,900
Breast Cancer	9	200	77
Titanic	3	150	2,051
Waveform	21	400	4,600
German	20	700	300
Image	18	1,300	1,010
Heart	13	170	100
Diabetis	8	468	300
Ringnorm	20	400	7,000
Thyroid	5	140	75
Twonorm	20	400	7,000
Flare Solar	9	666	400
Splice	60	1,000	2,175

TABLE II
AVERAGE NUMBER OF ACTIVE KERNELS USED IN THE SMKC1 AND SMKC2 SOLUTIONS. THE METHOD THAT OBTAINS A SPARSER SOLUTION IS MARKED WITH $\oplus$ WHEN THE DIFFERENCE IS STATISTICALLY SIGNIFICANT (AT A LEVEL OF 5%)

	$N_{xv}/N_{sv} = 2\%$		$N_{xv}/N_{sv} = 4\%$	
data set	SMKC1	SMKC2	SMKC1	SMKC2
Banana	$3.3\pm1.8^{\oplus}$	4.2 ± 0.8	$1.2\pm0.9^{\oplus}$	4.4 ± 0.7
Breast Cancer	4.2 ± 3.2	$2.1\pm1.4^{\oplus}$	3.6 ± 2.7	$2.5\pm0.6^{\oplus}$
Titanic	1.4 ± 1.7	1.9 ± 0.4	$1.3\pm0.5^{\oplus}$	1.7 ± 0.7
Waveform	1.7 ± 1.1	1.8 ± 0.4	$1.4\pm0.5^{\oplus}$	1.9 ± 0.3
German	4.7 ± 3.7	3.7 ± 3.0	2.4 ± 1.8	2.0 ± 0.0
Image	$1.6\pm0.5^{\oplus}$	2.4 ± 0.7	$2.6\pm1.8^{\oplus}$	4.2 ± 0.7
Heart	$1.2\pm0.4^{\oplus}$	1.7 ± 0.7	$1.0\pm0.0^{\oplus}$	1.9 ± 0.3
Diabetis	5.3 ± 2.0	$3.1\pm0.5^{\oplus}$	1.5 ± 0.8	$1.0\pm0.0^{\oplus}$
Ringnorm	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.2	1.0 ± 0.0
Thyroid	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
Twonorm	1.1 ± 0.3	1.1 ± 0.3	$1.0\pm0.0^{\oplus}$	1.3 ± 0.5
Flare Solar	$1.2\pm0.8^{\oplus}$	2.9 ± 0.6	$2.0\pm0.8^{\oplus}$	2.7 ± 1.6
Splice	3.2 ± 2.2	2.3 ± 0.5	$1.4\pm0.5^{\oplus}$	4.6 ± 2.3

on searching for the step size, the evaluation of $W(\boldsymbol{Z})$ only involves optimizing over the variable $\boldsymbol{\alpha}$.

Algorithm 1 Learning the expansion vectors.

```
1: Perform $k$-means clustering (where $k = N_{xv}$) and choose
   the $N_{xv}$ cluster centers as initial expansion vectors.
2: **repeat**
      3: Solve problem (18) [or (26)] at given $\boldsymbol{Z}$ and obtain
         the solution $\boldsymbol{\mu}^z$ and $\boldsymbol{\alpha}^z$. Set $\boldsymbol{\mu} = \boldsymbol{\mu}^z$.
      4: Use (29) [or (30)] to compute the gradient $\nabla W(\boldsymbol{Z})$.
      5: Perform line search to find the optimal step
         size with objective value of problem (18) [or (26)]
         evaluated at constant $\boldsymbol{\mu}$, and return the updated $\boldsymbol{Z}$.
6: **until** the minimum of objective function value $W(Z)$ is
   achieved or the maximum number of iterations is reached.
```

The partial derivative of $W(\boldsymbol{Z})$ in (16) w.r.t. z_{uv} can be written as

$$\frac{\partial W(\boldsymbol{Z})}{z_{uv}} = -\frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i^z \alpha_j^z y_i y_j \frac{\partial \tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial z_{uv}} \tag{29}$$

where

$$\begin{aligned}\frac{\partial \tilde{K}^z(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial z_{uv}} &= \left[\sum_{k=1}^{M} \mu_k \frac{\partial \Phi_k^z(\boldsymbol{x}_i)}{\partial z_{uv}}\right]^T \tilde{\mathbf{K}}_z^{-1} \left[\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_j)\right] \\ &\quad + \left[\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_i)\right]^T \frac{\partial \tilde{\mathbf{K}}_z^{-1}}{\partial z_{uv}} \left[\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_j)\right] \\ &\quad + \left[\sum_{k=1}^{M} \mu_k \Phi_k^z(\boldsymbol{x}_i)\right]^T \tilde{\mathbf{K}}_z^{-1} \left[\sum_{k=1}^{M} \mu_k \frac{\partial \Phi_k^z(\boldsymbol{x}_j)}{\partial z_{uv}}\right] \\ \frac{\partial \tilde{\mathbf{K}}_z^{-1}}{\partial z_{uv}} &= -\tilde{\mathbf{K}}_z^{-1} \frac{\partial \tilde{\mathbf{K}}_z}{\partial z_{uv}} \tilde{\mathbf{K}}_z^{-1}.\end{aligned}$$

Similarly, for (24), we have

$$\frac{\partial W(\boldsymbol{Z})}{z_{uv}} = -\frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i^z \alpha_j^z y_i y_j \sum_{k=1}^{M} \mu_k \frac{\partial K_k^z(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial z_{uv}} \tag{30}$$

where

$$\begin{aligned}\frac{\partial K_k^z(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial z_{uv}} &= \frac{\partial \Phi_k^z(\boldsymbol{x}_i)}{\partial z_{uv}}^T \left(\mathbf{K}_z^k\right)^{-1} \Phi_k^z(\boldsymbol{x}_j) \\ &\quad + \Phi_k^z(\boldsymbol{x}_i)^T \frac{\partial \left(\mathbf{K}_z^k\right)^{-1}}{\partial z_{uv}} \Phi_k^z(\boldsymbol{x}_j) \\ &\quad + \Phi_k^z(\boldsymbol{x}_i)^T \left(\mathbf{K}_z^k\right)^{-1} \frac{\partial \Phi_k^z(\boldsymbol{x}_j)}{\partial z_{uv}} \\ \frac{\partial (\mathbf{K}_z^k)^{-1}}{\partial z_{uv}} &= -\left(\mathbf{K}_z^k\right)^{-1} \frac{\partial \mathbf{K}_z^k}{\partial z_{uv}} \left(\mathbf{K}_z^k\right)^{-1}.\end{aligned}$$

Since it is essentially performing coordinate descent, the procedure converges to a locally optimal solution as in [25]. Note, on the other hand, that the standard (nonsparse) MKL formulations in Section II are convex and hence can have globally optimal solutions. Hence, it is a price that one has to pay for having a sparse classifier.

V. EXPERIMENTS

In this section, we perform classification experiments on a number of toy and real-world data sets[2]: *Banana, Breast Cancer, Titanic, Waveform, German, Image, Heart, Diabetes, Ringnorm, Thyroid, Twonorm, Flare Solar*, and *Splice* (Table I). The first six have also been used in [24] and [25] for the sparse single-kernel SVM. Each data set has 100 training/test splits. Following the scheme in [24] and [25], we will report results averaged over the first 30 splits.

In the sequel, the two sparse multiple-kernel SVM formulations proposed in Sections IV-A and IV-B will be denoted as SMKC1 and SMKC2, respectively. In the experiments, they will be compared with the following:

1) SK-SVM: standard (single-kernel) SVM;
2) SSKC: the sparse version of SK-SVM discussed in Section III;
3) MK-SVM: the multiple-kernel SVM in Section II.

The MK-SVM and its sparse formulations (SMKC1/SMKC2) are implemented in Matlab and C++, and the LIBSVM [7] package is used to solve the inner SVM optimization problem.

[2]http://ida.first.fraunhofer.de/projects/bench/

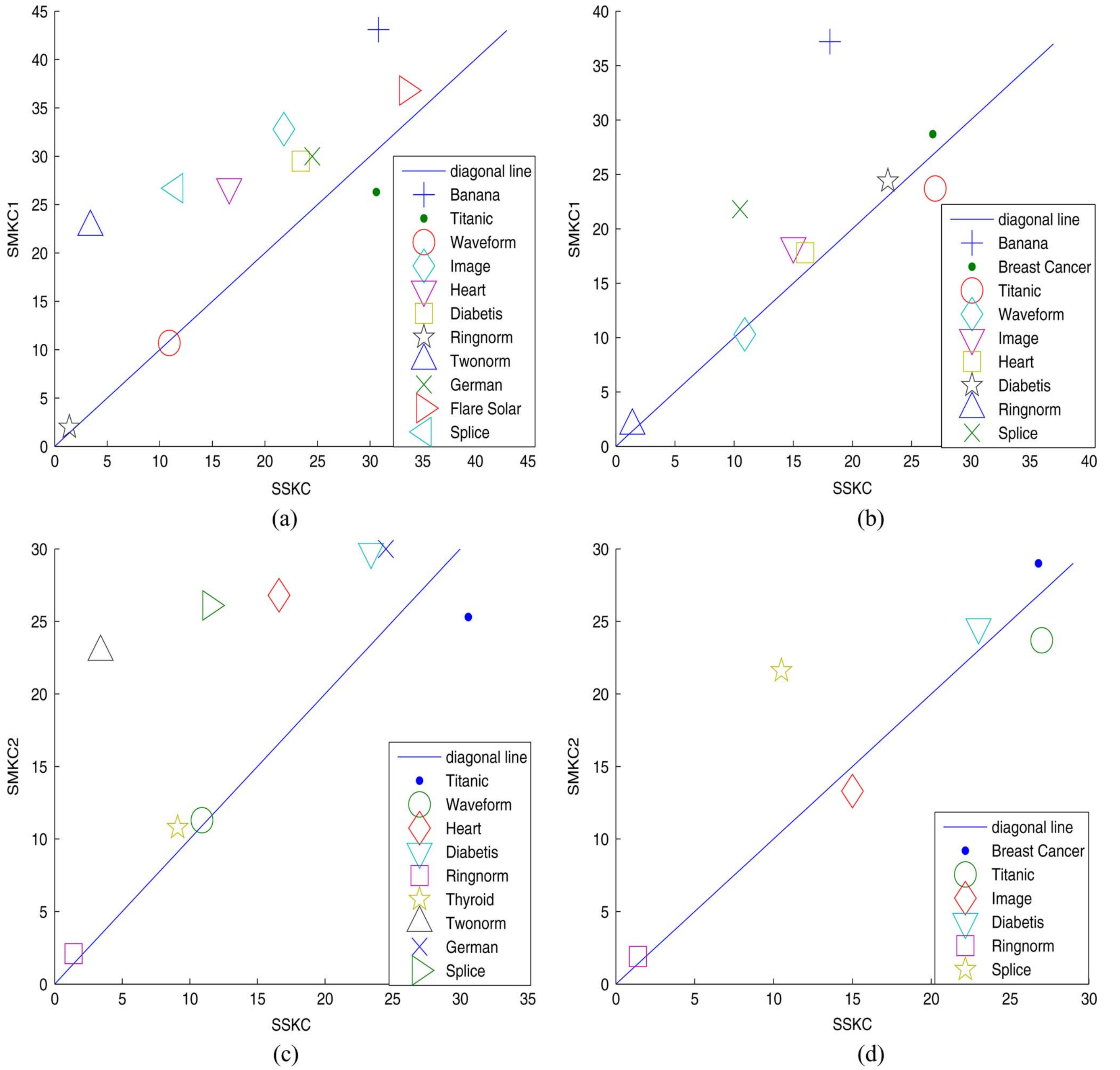

Fig. 1. Test error rates (in percent) for SSKC (abscissa) and SMKC1/SMKC2 (ordinate), with the expansion vectors fixed. The standard deviation is shown by the vertical/horizontal lines, and the diagonal line indicates where the two error rates are equal. (a) SMKC1, $N_{xv}/N_{sv} = 2\%$. (b) SMKC1, $N_{xv}/N_{sv} = 4\%$. (c) SMKC2, $N_{xv}/N_{sv} = 2\%$. (d) SMKC2, $N_{xv}/N_{sv} = 4\%$.

LIBSVM is also used for the implementation of SK-SVM. The SSKC implementation[3] is obtained from [24] and [25].

We will use the Gaussian kernel

$$K(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|^2).$$

By taking γ over the values of $\{2^{-8}, 2^{-6}, \ldots, 2^{6}, 2^{8}\}$, we form nine base kernels for MK-SVM, SMKC1, and SMKC2. The initial value of $\boldsymbol{\mu}$ is set to $[(1/M), (1/M), \ldots, (1/M)]$, so that all base kernels have the same initial weight. For SK-SVM, there is only one Gaussian kernel and its optimum γ is provided with the data sets, whereas for SSKC, its γ is selected by cross validation. As for the C parameter, it is individually tuned over the range $\{2^{-8}, 2^{-7}, \ldots, 2^{7}, 2^{8}\}$ by cross validation for each of SSKC, MK-SVM, SMKC1, and SMKC2, while that for SK-SVM is again provided with the data sets. Moreover, the k-means clustering scheme is used to initialize the expansion vectors of SSKC, SMKC1, and SMKC2.

In our implementation, step 5 in Algorithm 1 is implemented via the *minimizer* routine.[4] Line search using the quadratic and cubic polynomial approximations and the Wolfe–Powell stopping criteria [5] are used together with the slope ratio method [18] for guessing the initial step sizes. Experiments are performed on a 1.8-GHz AMD machine with 2-GB RAM.

[3]http://www.kyb.tuebingen.mpg.de/bs/people/mingrui/SKM.zip

[4]Available online at http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/

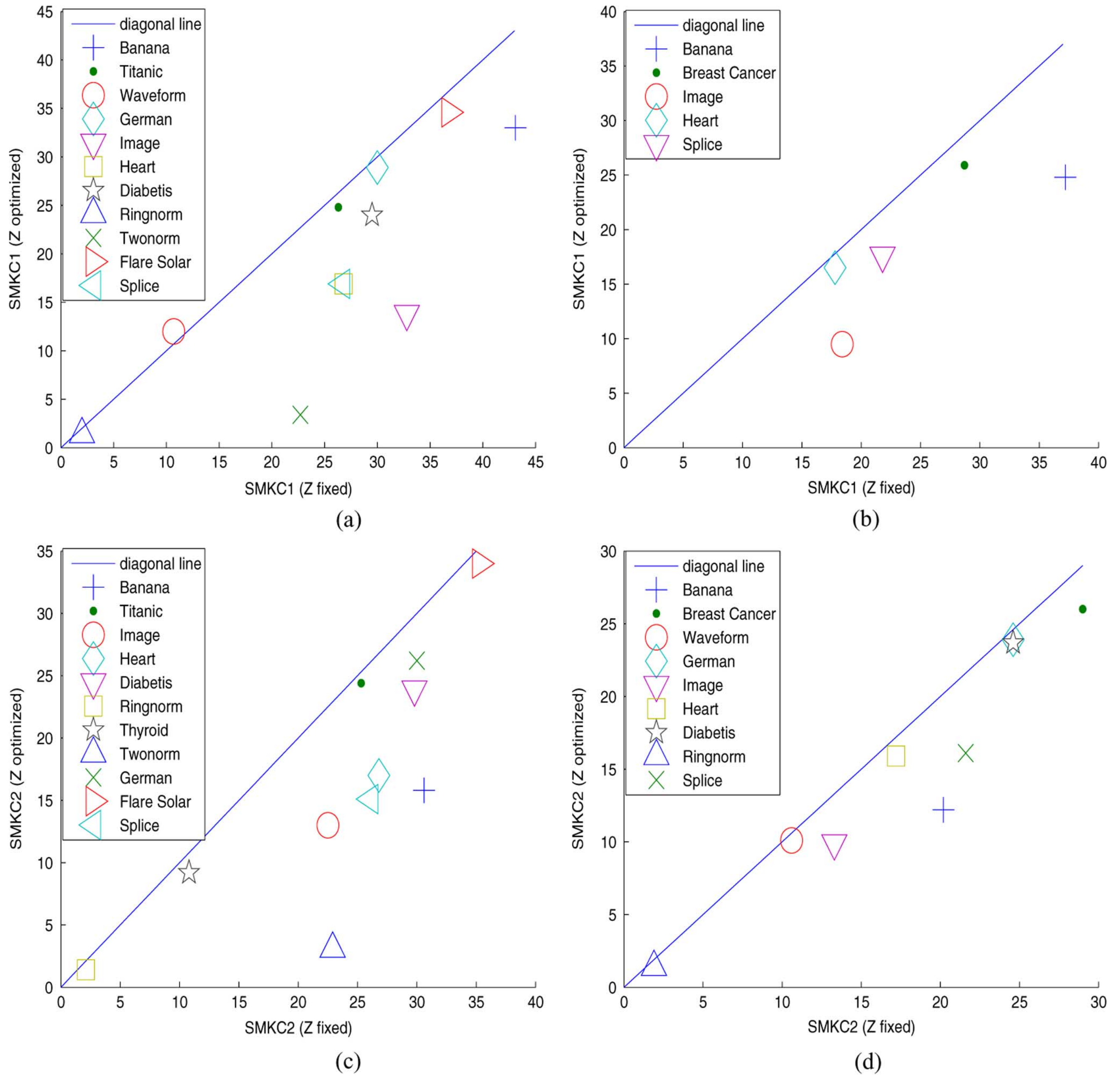

Fig. 2. Test error rates (in percent) for SMKC1 and SMKC2 with the expansion vectors fixed (abscissa) versus optimized (ordinate). (a) SMKC1, $N_{xv}/N_{sv} = 2\%$. (b) SMKC1, $N_{xv}/N_{sv} = 4\%$. (c) SMKC2, $N_{xv}/N_{sv} = 2\%$. (d) SMKC2, $N_{xv}/N_{sv} = 4\%$.

A. Importance of Optimizing the Expansion Vectors

In this section, we demonstrate the importance of optimizing the expansion vectors $\boldsymbol{z}_i$'s. We first study the performance that can be achieved by only optimizing $\boldsymbol{\mu}$. In other words, we do not update the $\boldsymbol{z}_i$'s, which are simply chosen as the cluster centers obtained by the k-means algorithm on the training data.

We experiment with two sparsity settings: $N_{xv}/N_{sv} = 2\%$ and $N_{xv}/N_{sv} = 4\%$, where N_{sv} is the number of support vectors obtained by SK-SVM, and N_{xv} is the number of expansion vectors that can be used by SSKC/SMKC1/SMKC2. Detailed results are in Table V of the Appendix. Here, we only show the results that are statistically significant (based on the paired t-test at the significance level of 5%). As can be seen from Fig. 1, it is not sufficient to only optimize $\boldsymbol{\mu}$. In particular, from the detailed results in Table V, SSKC outperforms SMKC1 on nine data sets at $N_{xv}/N_{sv} = 2\%$ and on seven data sets at $N_{xv}/N_{sv} = 4\%$, and outperforms SMKC2 on eight data sets at $N_{xv}/N_{sv} = 2\%$ and on four data sets at $N_{xv}/N_{sv} = 4\%$.

Next, we demonstrate that the performance of SMKC can be significantly improved when the expansion vectors are optimized. We follow the same setup above, but with the expansion vectors in SMKC1 and SMKC2 optimized with the gradient method in Section IV-C.

Detailed results can be found in Table VI of the Appendix. Here, we again only show the results that are statistically significant. First, as expected, Fig. 2 shows that optimizing the expansion vectors greatly improves the accuracies of both SMKC1 and SMKC2. Indeed, as can be seen from Fig. 3, SMKC1 now becomes comparable with SSKC and SMKC2 is even better

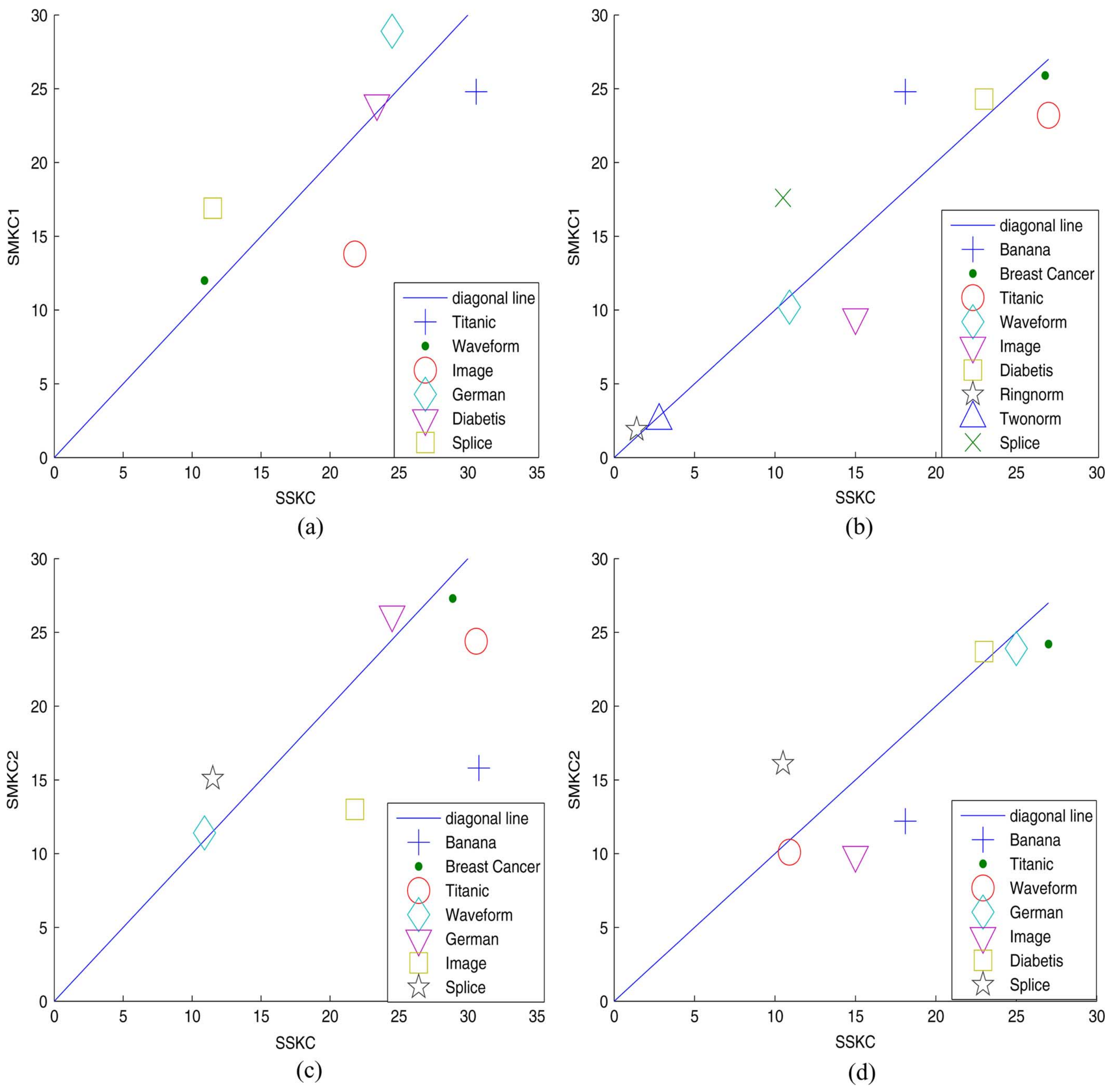

Fig. 3. Test error rates (in percent) for SSKC (abscissa) and SMKC1/SMKC2 (ordinate). (a) SMKC1, $N_{xv}/N_{sv} = 2\%$. (b) SMKC1, $N_{xv}/N_{sv} = 4\%$. (c) SMKC2, $N_{xv}/N_{sv} = 2\%$. (d) SMKC2, $N_{xv}/N_{sv} = 4\%$.

than SSKC. To be more specific, SMKC2 outperforms SSKC on four data sets at $N_{xv}/N_{sv} = 2\%$ and on five data sets at $N_{xv}/N_{sv} = 4\%$. Recall that the kernel parameter in SSKC is obtained by careful cross validation. Hence, similar to the use of MKL in training nonsparse classifiers, SMKC2 has the strong advantage over SSKC that it can automatically choose a good combination of kernels while attaining comparable or even better generalization performance.

B. Varying the Sparsity Requirement

In the last section, we only experimented with $N_{xv}/N_{sv} = 2\%$ and $N_{xv}/N_{sv} = 4\%$. Here, we vary N_{xv}/N_{sv} from 1% to 10% to better demonstrate the effects of N_{xv}/N_{sv} on both the test error (Fig. 4) and the training time (Fig. 5). Here, we only show the results on the *Banana* data set. Similar trends can be observed on the other data sets.

As can be seen, there is a tradeoff between accuracy and sparsity. Nevertheless, the generalization performance of SMKC2 is relatively stable over a large range of N_{xv}/N_{sv}, and is even slightly better than the nonsparse classifiers when the sparsity requirement is relaxed to $N_{xv}/N_{sv} = 10\%$. As for the training time, in general, it increases as more expansion vectors are used, though the difference is still within the range of statistical fluctuations.

TABLE III
TRAINING TIME (IN SECONDS) OF SMKC1/SMKC2. THE FASTER METHOD THAT IS STATISTICALLY SIGNIFICANT AT A LEVEL OF 5% IS MARKED WITH $\oplus$

	$N_{xv}/N_{sv} = 2\%$		$N_{xv}/N_{sv} = 4\%$	
data set	SMKC1	SMKC2	SMKC1	SMKC2
Banana	531.6±215.4	308.6±64.3 $\oplus$	514.8±112.0	509.0±121.3
Breast Cancer	100.0±55.6	24.1±14.2 $\oplus$	111.2±52.4	93.3±24.5
Titanic	21.3±7.5	17.2±5.8 $\oplus$	37.8±20.2	23.8±10.9 $\oplus$
Waveform	267.5±57.8	194.2±37.9$\oplus$	621.4±167.8	449.3±82.7$\oplus$
German	7522.6±702.0	2370.6±701.0$\oplus$	7787.2±1895.6$\oplus$	9502.0±1201.6
Image	9204.9±2018.8$\oplus$	12595.6±1975.8	11234.5±1567.3$\oplus$	10468.3±453.6
Heart	37.9±12.3	11.2±3.9$\oplus$	93.5±18.0	28.1±5.7$\oplus$
Diabetis	1166.9±163.8	590.3±134.1$\oplus$	1227.1±255.8	729.1±103.8$\oplus$
Ringnorm	355.9±44.5	112.9±16.5$\oplus$	193.8±27.4 $\oplus$	355.9±44.5
Thyroid	12.8±0.5	1.9±0.1$\oplus$	14.5±1.4	1.9±0.1$\oplus$
Twonorm	122.2±48.4	79.1±39.1$\oplus$	240.0±18.3	123.8±32.8$\oplus$
Flare Solar	3904.5±1948	700.9±1265.3$\oplus$	5231.4±234.1	3245.7±1234.5$\oplus$
Splice	7271.2±2391.8	1582.1±273.2$\oplus$	9289.5±3424.5	2003.5±456.6$\oplus$

TABLE IV
AVERAGE NUMBER OF SUPPORT VECTORS (#SV) OBTAINED BY MK-SVM AND THE AVERAGE NUMBER OF EXPANSION VECTORS (#XV) IN SMKC1/SMKC2, TOGETHER WITH THE AVERAGE NUMBER OF ACTIVE KERNELS USED

	#SV	#XV		#active kernels			(#SV or #XV) ×#active kernels		
		SMKC1/SMKC2			SMKC/SMKC2			SMKC1/SMKC2	
data set	MK-SVM	2%	4%	MK-SVM	2%	4%	MK-SVM	2%	4%
Banana	280.1	2.0	3.5	1.9	3.3/4.2	1.2/4.4	532.1	6.6/8.4	4.2/15.4
Breast Cancer	196.1	2.1	4.6	2.7	4.2/2.1	3.6/2.5	529.5	8.8/4.4	16.6/11.5
Titanic	80.8	1.2	2.7	2.1	1.4/1.9	1.3/1.7	169.7	1.7/2.3	3.5/4.6
Waveform	192.4	3.0	6.4	1.0	1.7/1.8	1.4/1.9	192.4	5.1/5.4	9.0/12.2
German	690.6	8.1	16.3	2.9	4.7/3.7	2.4/2.0	2002.7	38.1/30.0	39.1/32.6
Image	792.4	9.0	17.9	4.8	1.6/2.4	2.6/4.2	3803.5	14.4/21.6	46.5/75.2
Heart	108.1	1.9	3.2	1.9	1.2/1.7	1.0/1.9	205.4	2.3/3.2	3.2/6.1
Diabetis	456.7	4.7	9.3	3.3	5.3/3.1	1.5/1.0	1507.1	24.9/14.6	14.0/9.3
Ringnorm	214.3	3.0	6.0	1.0	1.0/1.0	1.0/1.0	214.3	3.0/3.0	6.0/6.0
Thyroid	82.1	1.0	1.0	3.6	1.0/1.0	1.0/1.0	295.6	1.0/1.0	1.0/1.0
Twonorm	154.4	1.6	3.0	2.0	1.1/1.1	1.0/1.3	308.8	1.8/1.8	3.0/3.9
Flare Solar	500.6	11.0	22.2	1.7	1.2/2.9	2.0/2.7	851.0	13.2/31.9	44.4/59.4
Splice	831.7	13.1	26.3	2.7	3.2/2.3	1.4/4.6	2245.6	41.9/30.1	36.9/121.0

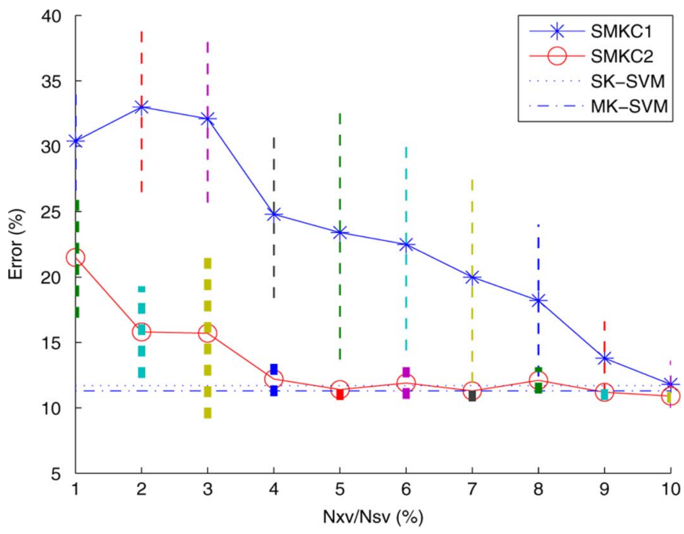

Fig. 4. Test error rates of SMKC1/SMKC2 on the *Banana* data set, when N_{xv}/N_{sv} varies from 1% to 10%.

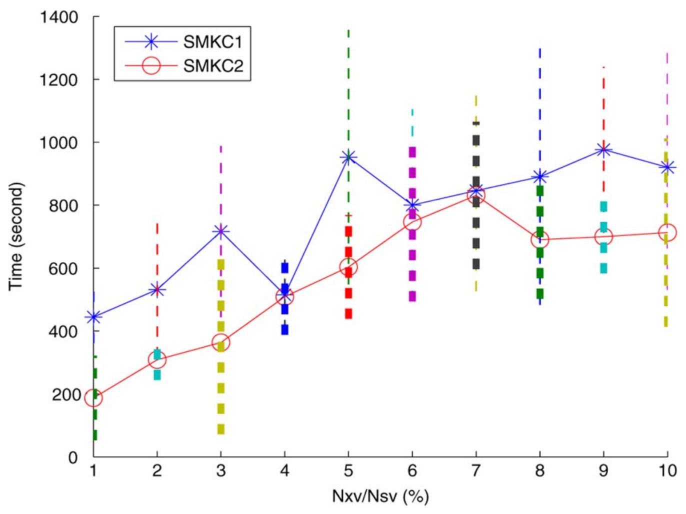

Fig. 5. Training time (in seconds) of SMKC1/SMKC2 on the *Banana* data set, when N_{xv}/N_{sv} varies from 1% to 10%.

C. SMKC1 Versus SMKC2

Results in the previous sections show that SMKC2 is more accurate than SMKC1. Here, we look at another attribute, namely, sparsity. Recall that nine base kernels are provided for SMKC1 and SMKC2. Table II compares the number of active kernels used in the final solutions. As can be seen, the models obtained by both SMKC1 and SMKC2 are sparse, with the SMKC1 solution sometimes slightly sparser.

The last desideratum is training time. As shown in Algorithm 1, the differences between SMKC1 and SMKC2 lie in steps 3 and 4. For step 4, SMKC1 and SMKC2 turn out to have the same computational complexity as both require computing the inverse

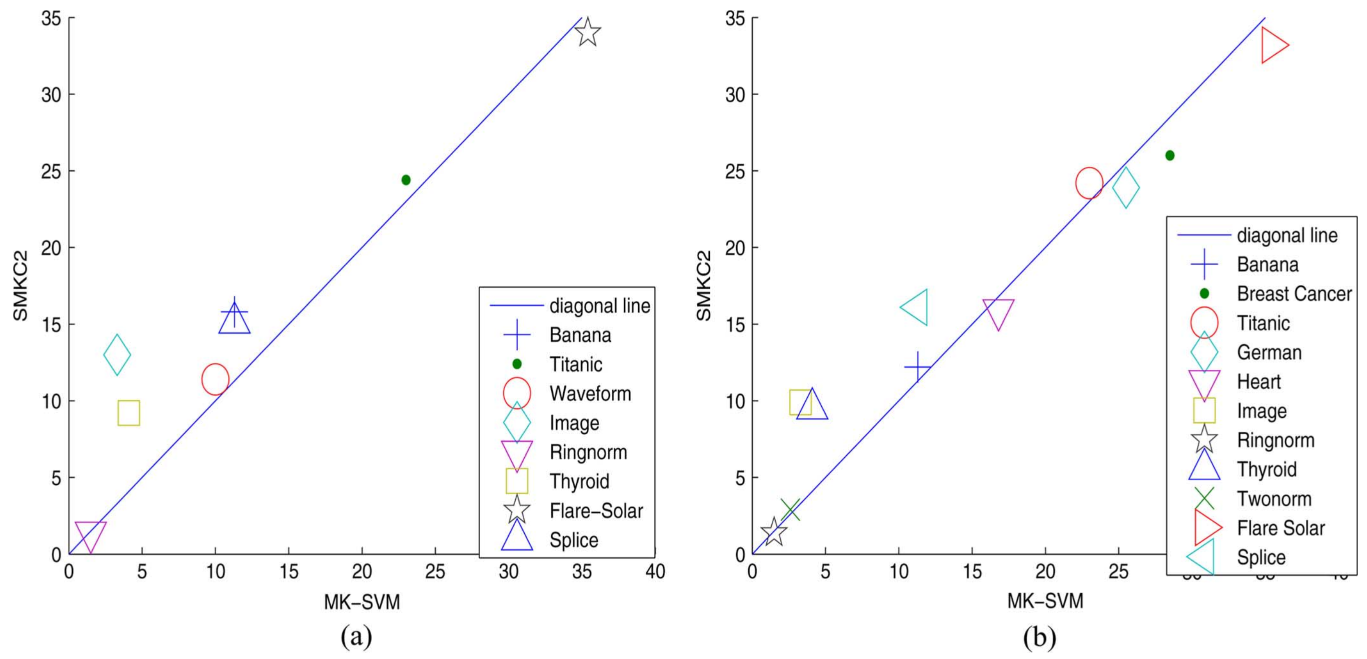

Fig. 6. Test error rates (in percent) for MK-SVM (abscissa) and SMKC2 (ordinate). (a) $N_{xv}/N_{sv} = 2\%$. (b) $N_{xv}/N_{sv} = 4\%$.

TABLE V
TEST ERROR RATES (IN PERCENT) FOR THE VARIOUS METHODS (WITH z_i'S FIXED FOR SMKC1 AND SMKC2), WHEN $N_{xv}/N_{sv} = 2\%$ AND 4%. THE SYMBOL $\oplus$ (RESPECTIVELY, $\bigtriangledown$) INDICATES THAT THE DIFFERENCE BETWEEN SSKC AND SMKC1 (RESPECTIVELY, SMKC2) IS STATISTICALLY SIGNIFICANT AT A LEVEL OF 5%

data set	$N_{xv}/N_{sv} = 2\%$			$N_{xv}/N_{sv} = 4\%$		
	SSKC	SMKC1	SMKC2	SSKC	SMKC1	SMKC2
Banana	$30.8 \pm 5.5^{\oplus}$	43.1 ± 3.2	30.6 ± 3.7	$18.1 \pm 4.8^{\oplus}$	37.2 ± 4.4	20.2 ± 3.8
Breast Cancer	28.9 ± 5.1	28.5 ± 4.6	28.5 ± 4.5	$26.8 \pm 4.9^{\oplus \bigtriangledown}$	28.7 ± 5.1	29.0 ± 4.8
Titanic	30.6 ± 4.8	$26.3 \pm 3.0^{\oplus}$	$25.3 \pm 1.0^{\bigtriangledown}$	27.0 ± 5.2	$23.7 \pm 1.8^{\oplus}$	$23.7 \pm 1.3^{\bigtriangledown}$
Waveform	$10.9 \pm 0.6^{\bigtriangledown}$	$10.7 \pm 0.3^{\oplus}$	11.3 ± 0.6	10.9 ± 0.5	$10.3 \pm 0.6^{\oplus}$	10.6 ± 0.8
German	$24.5 \pm 2.9^{\oplus \bigtriangledown}$	30.0 ± 2.0	30.0 ± 1.9	25.0 ± 2.9	25.3 ± 3.0	24.6 ± 2.5
Image	$21.8 \pm 6.2^{\oplus}$	32.8 ± 3.3	22.5 ± 2.9	$15.0 \pm 2.3^{\oplus}$	18.4 ± 3.9	$13.3 \pm 1.3^{\bigtriangledown}$
Heart	$16.6 \pm 3.2^{\oplus \bigtriangledown}$	26.8 ± 13.7	26.8 ± 13.5	$16.0 \pm 3.0^{\oplus}$	17.8 ± 3.3	17.2 ± 3.4
Diabetis	$23.4 \pm 1.6^{\oplus \bigtriangledown}$	29.5 ± 2.8	29.8 ± 3.2	$23.0 \pm 1.9^{\oplus \bigtriangledown}$	24.4 ± 2.0	24.6 ± 2.1
Ringnorm	$1.4 \pm 0.01^{\oplus \bigtriangledown}$	2.0 ± 0.2	2.1 ± 0.2	$1.4 \pm 0.01^{\oplus \bigtriangledown}$	1.9 ± 0.2	1.9 ± 0.2
Thyroid	$9.1 \pm 2.7^{\bigtriangledown}$	9.5 ± 2.8	10.8 ± 3.1	8.9 ± 2.7	9.5 ± 2.8	9.5 ± 2.8
Twonorm	$3.4 \pm 0.1^{\oplus \bigtriangledown}$	22.7 ± 22.9	22.9 ± 23.0	2.8 ± 0.3	3.9 ± 3.9	3.9 ± 3.9
Flare Solar	$33.5 \pm 6.1^{\oplus}$	36.8 ± 3.7	35.3 ± 2.0	33.3 ± 6.2	34.4 ± 2.2	34.5 ± 1.9
Splice	$11.5 \pm 2.4^{\oplus \bigtriangledown}$	26.7 ± 1.9	26.1 ± 1.8	$10.5 \pm 0.6^{\oplus \bigtriangledown}$	21.8 ± 2.0	21.6 ± 2.0

of an $N_{xv} \times N_{xv}$ matrix. However, for step 3, the gradient computation of SMKC1 in (20) requires computing the inverse of $\mathbf{K}_z$, which incurs an additional $\mathcal{O}(N_{xv}^3)$ time over SMKC2. Hence, overall, SMKC1 training is slower than SMKC2. This is also experimentally confirmed by the timing results in Table III and Fig. 5.

Overall, SMKC2 is more accurate, faster on training, and its kernel combination obtained is almost as sparse as that of SMKC1. Hence, we recommend the use of SMKC2 instead of SMKC1.

D. Comparison With MK-SVM

In this section, we compare with the multiple-kernel SVM (MK-SVM) in Section II. Recall that because of the ℓ_1 constraint on $\boldsymbol{\mu}$, a sparse $\boldsymbol{\mu}$ solution is encouraged and the resultant MK-SVM classifier can select a sparse combination of base kernels. This is also experimentally verified in Table IV. However, unlike SMKC1 and SMKC2, its number of support vectors is not explicitly controlled and so MK-SVM can require a much larger number of support vectors. Recall that the main motivation of building a sparse classifier is to reduce its prediction time. As discussed in Sections III, IV-A, and IV-B, this is proportional to the product of the number of support vectors for MK-SVM (or the number of expansion vectors for SMKC1/SMKC2) and the number of active kernels selected. As can be seen from Table IV, the product values for SMKC1/SMKC2 are smaller than those of MK-SVM by two to three orders of magnitude. Hence, their prediction speeds are also much faster.

However, there is a price to pay for faster prediction. As can be seen from Fig. 6, the test error rate for SMKC2 is slightly higher than that for MK-SVM at $N_{xv}/N_{sv} = 4\%$. Nevertheless, the difference is usually small and SMKC2, despite having a sparse solution, is still very competitive. Such a sparsity–accuracy tradeoff can also be observed in the single-kernel case between the (nonsparse) SK-SVM and (sparse) SSKC (Table VI).

TABLE VI
TEST ERROR RATES (IN PERCENT) FOR THE VARIOUS METHODS. THE SYMBOL $\oplus$ (RESPECTIVELY, $\triangledown$) INDICATES THAT THE DIFFERENCE BETWEEN SSKC AND SMKC1 (RESPECTIVELY, SMKC2) IS STATISTICALLY SIGNIFICANT AT A LEVEL OF 5%. SIMILARLY, THE SYMBOL ® INDICATES THE DIFFERENCE BETWEEN MK-SVM AND SMKC2 AT $N_{xv}/N_{sv} = 4\%$

			$N_{xv}/N_{sv} = 2\%$			$N_{xv}/N_{sv} = 4\%$		
data set	SK-SVM	MK-SVM	SSKC	SMKC1	SMKC2	SSKC	SMKC1	SMKC2
Banana	11.7 ± 0.7	$11.3\pm0.9^{\circledR}$	30.8 ± 5.5	33.0 ± 6.5	$15.8\pm3.5^{\triangledown}$	$18.1\pm4.8^{\oplus}$	24.8 ± 6.4	$12.2\pm1.3^{\triangledown}$
Breast Cancer	26.5 ± 4.9	28.5 ± 5.2	28.9 ± 5.1	28.6 ± 5.6	$27.3\pm4.5^{\triangledown}$	26.8 ± 4.9	$25.9\pm4.4^{\oplus}$	$26.0\pm4.1^{\circledR}$
Titanic	22.2 ± 0.6	$23.0\pm0.8^{\circledR}$	30.6 ± 4.8	$24.8\pm2.6^{\oplus}$	$24.4\pm1.4^{\triangledown}$	27.0 ± 5.2	$23.2\pm1.0^{\oplus}$	$24.2\pm2.1^{\triangledown}$
Waveform	10.0 ± 0.5	10.0 ± 0.5	$10.9\pm0.6^{\oplus\triangledown}$	12.0 ± 0.5	11.4 ± 0.9	10.9 ± 0.5	$10.2\pm0.7^{\oplus}$	$10.1\pm0.5^{\triangledown}$
German	22.9 ± 1.7	25.5 ± 2.6	$24.5\pm2.9^{\oplus\triangledown}$	28.9 ± 3.4	26.2 ± 3.9	25.0 ± 2.9	25.7 ± 3.3	$23.9\pm2.0^{\triangledown\circledR}$
Image	3.0 ± 0.6	$3.3\pm0.7^{\circledR}$	21.8 ± 6.2	$13.8\pm1.7^{\oplus}$	$13.0\pm1.3^{\triangledown}$	15.0 ± 2.3	$9.5\pm1.6^{\oplus}$	$9.9\pm1.1^{\triangledown}$
Heart	16.2 ± 3.0	16.8 ± 2.9	16.6 ± 3.2	16.9 ± 3.5	17.0 ± 3.5	16.0 ± 3.0	16.5 ± 3.4	$15.9\pm2.8^{\circledR}$
Diabetis	27.7 ± 1.9	23.9 ± 1.8	$23.4\pm1.6^{\oplus}$	24.0 ± 1.5	23.9 ± 2.0	$23.0\pm1.9^{\oplus\triangledown}$	24.3 ± 2.4	23.7 ± 2.0
Ringnorm	1.6 ± 0.1	1.5 ± 0.1	1.4 ± 0.01	1.4 ± 0.01	1.4 ± 0.01	$1.4\pm0.1^{\oplus}$	1.9 ± 0.2	$1.4\pm0.1^{\circledR}$
Thyroid	5.0 ± 2.3	$4.1\pm2.4^{\circledR}$	9.1 ± 2.7	9.0 ± 2.7	9.2 ± 2.6	8.9 ± 2.7	9.0 ± 2.7	9.5 ± 2.5
Twonorm	3.0 ± 0.2	$2.6\pm0.1^{\circledR}$	3.4 ± 0.1	3.4 ± 0.6	3.1 ± 0.5	2.8 ± 0.3	$2.5\pm0.1^{\oplus}$	2.9 ± 0.3
Flare Solar	32.4 ± 1.8	35.4 ± 2.0	33.5 ± 6.1	34.6 ± 1.7	34.0 ± 2.4	33.3 ± 6.2	34.5 ± 2.1	$33.2\pm2.8^{\circledR}$
Splice	10.9 ± 0.9	$11.3\pm0.8^{\circledR}$	$11.5\pm2.4^{\oplus\triangledown}$	16.9 ± 2.9	15.1 ± 1.6	$10.5\pm0.6^{\oplus\triangledown}$	17.6 ± 2.8	16.1 ± 2.6

VI. CONCLUSION

In this paper, we incorporated MKL into the construction of sparse SVM classifiers. We proposed two formulations that allow a convex combination of kernels to be automatically learned from the data. In particular, the one that is based on a convex combination of equivalent kernels (SMKC2) is very competitive. Computationally, the SMKC2 classifier can also be easily trained by alternating linear program and standard SVM solver. Consequently, not only can we obtain a compact classifier that is useful for fast prediction, an appropriate kernel can also be easily determined, which then ensures good generalization. Experimental results on a number of toy and real-world data sets demonstrate that the resultant classifier is both compact and accurate.

APPENDIX
A THEOREM DUE TO BONNANS AND SHAPIRO [4]

Theorem 1: Let X be a metric space and U be a normed space. Suppose that for all $x \in X$, the function $f(x, \cdot)$ is differentiable, that $f(x, u)$ and $D_u f(x, u)$ [the derivative of $f(x, \cdot)$] are continuous on $X \times U$, and let Φ be a compact subset of X. Let define the optimal value function as $v(u) = \inf_{x\in\Phi} f(x, u)$. The optimal value function is directionally differentiable. Furthermore, if for $u^0 \in U, f(\cdot, u^0)$ has a unique minimizer x^0 over Φ, then $v(u)$ is differentiable at u^0 and $Dv(u^0) = D_u f(x^0, u^0)$.

SOME DETAILED EXPERIMENTAL RESULTS

In this section, we report the detailed statistical results of SK-SVM, MK-SVM, SSKC, SMKC1, and SMKC2 for the cases when the expansion vectors in SMKC1/SMKC2 are fixed (Table V) and when they are optimized (Table VI).

REFERENCES

[1] A. Argyriou, R. Hauser, C. A. Micchelli, and M. Pontil, "A DC algorithm for kernel selection," in *Proc. 23rd Int. Conf. Mach.*, Pittsburgh, PA, Jun. 2006, pp. 41–49.

[2] A. Argyriou, C. A. Micchelli, and M. Pontil, "Learning convex combinations of continuously parameterized basic kernels," in *Proc. 18th Annu. Conf. Learn. Theory*, Bertinoro, Italy, 2005, pp. 338–352.

[3] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the SMO algorithm," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 6–14.

[4] J. F. Bonnans and A. Shapiro, "Optimization problems with perturbations: A guided tour," *SIAM Rev.*, vol. 40, no. 2, pp. 228–264, Jun. 1998.

[5] R. L. Burden and J. Douglas Faires, *Numerical Analysis*, 7th ed. Pacific Grove, CA: Brooks/Cole, 2001.

[6] C. J. C. Burges, "Simplified support vector decision rules," in *Proc. 13th Int. Conf. Mach. Learn.*, San Mateo, CA, 1996, pp. 71–77.

[7] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," 2001 [Online]. Available: http://www.csie.nte.edu.tw/~cjlin/libsvm/

[8] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[9] S. Keerthi, O. Chapelle, and D. Decoste, "Building support vector machines with reduced classifier complexity," *J. Mach. Learn. Res.*, vol. 7, pp. 1493–1515, 2006.

[10] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[11] Y. Lin and H. H. Zhang, "Component selection and smoothing in smoothing spline analysis of variance models," Dept. Statist., Univ. Wisconsin, Madison, WI, Tech. Rep. 1072r, 2003.

[12] C. A. Micchelli and M. Pontil, "Learning the kernel function via regularization," *J. Mach. Learn. Res.*, vol. 6, pp. 1099–1125, Dec. 2005.

[13] D. Nguyen and T. Ho, "An efficient method for simplifying support vector machines," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, 2005, pp. 617–624.

[14] C. S. Ong, A. J. Smola, and R. C. Williamson, "Hyperkernels," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003.

[15] C. S. Ong, A. J. Smola, and R. C. Williamson, "Learning the kernel with hyperkernels," *J. Mach. Learn. Res.*, vol. 6, pp. 1043–1071, Dec. 2005.

[16] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, Jun. 2007, pp. 775–782.

[17] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, 2008.

[18] S. S. Rao, *Engineering Optimization: Theory and Practice*, 3rd ed. New York: Wiley, 1996.

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

[20] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[21] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, Jul. 2006.

[22] I. W. Tsang and J. T. Kwok, "Efficient hyperkernel learning using second-order cone programming," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 48–58, Jan. 2006.

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

[24] M. Wu, B. Schölkopf, and B. Bakir, "A direct method for building sparse kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 603–624, 2006.

[25] M. Wu, B. Schölkopf, and G. Bakir, "Building sparse large margin classifiers," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, 2005, pp. 996–1003.

[26] A. Zien and C. S. Ong, "Multiclass multiple kernel learning," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, Jun. 2007, pp. 1191–1198.

Mingqing Hu received the B.Sc. and M.Sc. degrees from Xidian University, Xian, China, in 1999 and 2002, respectively, and the Ph.D. degree from the University of Trento, Trento, Italy, in February 2007.

For a year and a half before his doctoral studies, he worked in the industry. Currently, he is an Assistant Researcher at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China. His research interests include machine learning (in particular kernel methods) and computer vision.

Yiqiang Chen received the B.A.Sc. and M.A. degrees from the University of Xiangtan, Xiangtan City, China, in 1996 and 1999, respectively, and the Ph.D. degree from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2002.

In 2004, he was a Visiting Scholar Researcher at the Department of Computer Science, Hong Kong University of Science and Technology (HKUST), Hong Kong. Currently, he is an Associate Professor and Vice Director of the pervasive computing research center at ICT, CAS. His research interests include artificial intelligence, pervasive computing, and human computer interface.

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

He then joined the Department of Computer Science, Hong Kong Baptist University, as an Assistant Professor. He returned to the Hong Kong University of Science and Technology in 2000 and is now an Associate Professor in 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 for the IEEE TRANSACTIONS ON NEURAL NETWORKS and *Neurocomputing*. He also received the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding 2004 Paper Award in 2006.