Pattern Recognition 44 (2011) 159–171

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Regularizing multiple kernel learning using response surface methodology

Mehmet Gönen *, Ethem Alpaydın

Department of Computer Engineering, Boğaziçi University, TR-34342 Bebek, İstanbul, Turkey

ARTICLE INFO

Article history:
Received 26 June 2009
Received in revised form
12 May 2010
Accepted 2 July 2010

Keywords:
Support vector machine
Multiple kernel learning
Regularization
Response surface methodology

ABSTRACT

In recent years, several methods have been proposed to combine multiple kernels using a weighted linear sum of kernels. These different kernels may be using information coming from multiple sources or may correspond to using different notions of similarity on the same source. We note that such methods, in addition to the usual ones of the canonical support vector machine formulation, introduce new regularization parameters that affect the solution quality and, in this work, we propose to optimize them using response surface methodology on cross-validation data. On several bioinformatics and digit recognition benchmark data sets, we compare multiple kernel learning and our proposed regularized variant in terms of accuracy, support vector count, and the number of kernels selected. We see that our proposed variant achieves statistically similar or higher accuracy results by using fewer kernel functions and/or support vectors through suitable regularization; it also allows better knowledge extraction because unnecessary kernels are pruned and the favored kernels reflect the properties of the problem at hand.

1. Introduction

Support vector machine (SVM) is a discriminative classifier proposed for binary classification and is based on the theory of structural risk minimization [1–4]. Given a sample of N independent and identically distributed training instances, $\{\boldsymbol{x}_i,y_i\}_{i=1}^N$, where $\boldsymbol{x}_i\in\mathbb{R}^D$ and $y_i\in\{-1,+1\}$ is its class label, SVM basically finds the linear discriminant with the maximum margin in the feature space induced by the mapping function, $\Phi:\mathbb{R}^D\rightarrow\mathbb{R}^S$. The decision function is

$$f(\boldsymbol{x})=\mathrm{sgn}(\langle \boldsymbol{w},\Phi(\boldsymbol{x})\rangle+b)$$

whose parameters can be learned by solving the following quadratic optimization problem:

$$\begin{aligned}\text{min.}\quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2+C\sum_{i=1}^N\xi_i\\ \text{w.r.t.}\quad & \boldsymbol{w}\in\mathbb{R}^S,\quad \boldsymbol{\xi}\in\mathbb{R}_+^N,\quad b\in\mathbb{R}\\ \text{s.t.}\quad & y_i(\langle \boldsymbol{w},\Phi(\boldsymbol{x}_i)\rangle+b)\geq 1-\xi_i\quad \forall i\end{aligned}$$

where $\boldsymbol{w}$ is the vector of weight coefficients, C is a predefined positive trade-off parameter between model simplicity and classification error, $\boldsymbol{\xi}$ is the vector of slack variables, and b is the bias term of the separating hyperplane. Instead of solving this optimization problem directly, the Lagrangian dual function enables us to obtain the following dual formulation:

$$\begin{aligned}\text{max.}\quad & \sum_{i=1}^N\alpha_i-\frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N\alpha_i\alpha_j y_i y_j\overbrace{\langle \Phi(\boldsymbol{x}_i),\Phi(\boldsymbol{x}_j)\rangle}^{k(\boldsymbol{x}_i,\boldsymbol{x}_j)}\\ \text{w.r.t.}\quad & \boldsymbol{\alpha}\in\mathbb{R}_+^N\\ \text{s.t.}\quad & \sum_{i=1}^N\alpha_i y_i=0\\ & C\geq\alpha_i\geq 0\quad\forall i\end{aligned}\tag{1}$$

where $\boldsymbol{\alpha}$ is the vector of dual variables corresponding to each separation constraint and the obtained kernel matrix is positive semidefinite. Solving this, we get $\boldsymbol{w}=\sum_{i=1}^N\alpha_i y_i\Phi(\boldsymbol{x}_i)$ and the decision function can be written as

$$f(\boldsymbol{x})=\mathrm{sgn}\left(\sum_{i=1}^N\alpha_i y_i k(\boldsymbol{x}_i,\boldsymbol{x})+b\right).$$

There are several kernel functions successfully used in the literature such as the linear kernel (k_L), the polynomial kernel (k_P), and the Gaussian kernel (k_G):

$$k_L(\boldsymbol{x}_i,\boldsymbol{x}_j)=\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle$$

$$k_P(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle+1)^q,\quad q\in\mathbb{N}$$

$$k_G(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp(-\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2^2/s^2),\quad s\in\mathbb{R}_{++}.$$

There are also kernel functions proposed for particular applications, such as natural language processing [5] and bioinformatics [6].

* Corresponding author.
E-mail addresses: gonen@boun.edu.tr (M. Gönen), alpaydin@boun.edu.tr (E. Alpaydın).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.07.008

Selecting the kernel function and its parameters (e.g. q or s) is an important issue in SVM training. Kernel selection is generally done by selecting the best kernel function after applying a cross-validation procedure; that is, they are used singly and separately and the one that leads to highest accuracy on validation data is chosen. In recent years, kernel combination methods have been proposed where we use multiple kernels instead of selecting one specific kernel function and its corresponding parameters. Each kernel function captures a different measure of similarity and using several instead of one may enable a better solution. If we have input coming from multiple sources each with its own kernel (or kernels), combining kernels allows information from these sources to be integrated effectively. The reasoning is similar to combining different classifiers: Instead of choosing a single one and putting all our eggs in the same basket, it is better to have a set and let an algorithm do the picking or combination.

Multiple kernel learning (MKL) [7–9] optimizes the support vector parameters and the kernel combination weights jointly. MKL is proposed to obtain sparse kernel ensembles eliminating some of the kernels by assigning them zero weights during training.

The MKL formulation of Bach et al. [8] introduces new regularization parameters that are directly related to the sparsity of the solution obtained. The easiest approach of setting them all equal does not always work well because it implies giving equal a priori weight to all kernels. The approach in Bach et al. [10] is a heuristic to simply estimate the weights but it is clear that ideally, these weights should also be trained in a coupled manner with the kernel machines.

We propose using response surface methodology (RSM) on validation error to optimize these parameters using data and to obtain more regularized solutions. We show that optimizing the regularization parameters using an RSM-based approach leads to more sparse kernel ensembles where some kernels are given zero weight without diminishing accuracy, when compared to the canonical MKL on several benchmark data sets. Not using a kernel in the ensemble may be either because: (a) the notion of similarity used in the kernel function is not appropriate or (b) the data source used to calculate the kernel does not carry useful information. If a kernel function is assigned zero weight in the combination rule, it means that this specific kernel function or the data source used in this kernel function does not carry discriminative information for the problem and hence can be pruned.

The rest of this paper is organized as follows: Section 2 reviews different kernel combination methods. In Section 3, we explain how regularization can be achieved for the MKL formulation of Bach et al. [8] by extending it using RSM. Section 4 demonstrates the effect of regularization in kernel combination with experiments on several toy and real-world, bioinformatics data sets. A survey of related work is given in Section 5, and we conclude in Section 6.

2. Kernel combination methods

The kernel matrix in the objective function must be positive semidefinite to solve the optimization problem in (1) efficiently. In this case, the mathematical model becomes a convex optimization problem and there are several methods that can be used to obtain the global optimum. Simple rules, such as scaling with a positive number, summation, and multiplication, as summarized in (2), allow us to obtain new kernel functions from existing ones; they ensure that the combined kernel has also a positive semidefinite kernel matrix [11]:

$$k(\boldsymbol{x}_i,\boldsymbol{x}_j)=ak_1(\boldsymbol{x}_i,\boldsymbol{x}_j),\quad a\in\mathbb{R}_+$$

$$k(\boldsymbol{x}_i,\boldsymbol{x}_j)=k_1(\boldsymbol{x}_i,\boldsymbol{x}_j)+k_2(\boldsymbol{x}_i,\boldsymbol{x}_j)$$

$$k(\boldsymbol{x}_i,\boldsymbol{x}_j)=k_1(\boldsymbol{x}_i,\boldsymbol{x}_j)k_2(\boldsymbol{x}_i,\boldsymbol{x}_j). \tag{2}$$

Summation rule is applied successfully to computational biology [12] and optical digit recognition [13] where heterogeneous data sets exist by the nature of these problems. In both works, summation of two or more kernels obtained from different representations of the same data set is used.

Such simple rules make assumptions about the combination rule before training. For example, unweighted summation of kernel functions gives equal importance to each kernel. Lanckriet et al. [7,14] replace the kernel function in the objective and decision functions with a weighted linear combination of kernels where weights are also learned. The new objective function is

$$\min_{\boldsymbol{\eta}}\max_{\boldsymbol{\alpha}}\ \sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\sum_{m=1}^{P}\eta_m k_m(\boldsymbol{x}_i,\boldsymbol{x}_j) \tag{3}$$

where m indexes kernels, P is their number, each $k_m(\cdot,\cdot)$ corresponds to a different kernel function, and η_m is the combination weight of $k_m(\cdot,\cdot)$. The new decision function becomes

$$f(\boldsymbol{x})=\operatorname{sgn}\left(\sum_{i=1}^{N}\alpha_i y_i\sum_{m=1}^{P}\eta_m k_m(\boldsymbol{x}_i,\boldsymbol{x})+b\right). \tag{4}$$

Lanckriet et al. [7,14] represent this problem using a semidefinite programming (SDP) formulation by constraining the combined kernel matrix to be positive semidefinite. Lanckriet et al. [14] simplify further the model into a quadratically constrained quadratic programming (QCQP) problem by considering only nonnegative combination weights. The QCQP formulation can be solved for both the support vector coefficients ($\boldsymbol{\alpha}$) and the combination weights ($\boldsymbol{\eta}$), whereas the SDP formulation gives only the combination weights (without any restriction of nonnegativity) and requires another step of solving a canonical SVM problem to find the support vector coefficients. The QCQP formulation has been tested on genomic data fusion with different kernel functions evaluated on different data representations [15].

Bach et al. [8] propose to use an unweighted sum of discriminant values calculated over different feature spaces in the decision function:

$$f(\boldsymbol{x})=\sum_{m=1}^{P}\langle\boldsymbol{w}_m,\Phi_m(\boldsymbol{x})\rangle+b. \tag{5}$$

They rewrite the primal optimization problem that uses the weighted l_1-norm on feature spaces and the l_2-norm within each feature space:

$$\begin{aligned}
\text{min.}\quad & \frac{1}{2}\left(\sum_{m=1}^{P}d_m\|\boldsymbol{w}_m\|_2\right)^2+C\sum_{i=1}^{N}\xi_i\\
\text{w.r.t.}\quad & \boldsymbol{w}_m\in\mathbb{R}^{S_m},\quad \boldsymbol{\xi}\in\mathbb{R}_+^N,\quad b\in\mathbb{R}\\
\text{s.t.}\quad & y_i\left(\sum_{m=1}^{P}\langle\boldsymbol{w}_m,\Phi_m(\boldsymbol{x}_i)\rangle+b\right)\geq 1-\xi_i\quad\forall i
\end{aligned} \tag{6}$$

where d_m is the nonnegative weight assigned to the feature space induced by $\Phi_m(\cdot)$, $\boldsymbol{w}_m$ is the vector of weight coefficients, and S_m is the dimensionality of the corresponding feature space. This model can be treated as a second-order cone program and the resulting dual problem is

$$\begin{aligned}
\text{min.}\quad & \frac{1}{2}\gamma^2-\sum_{i=1}^{N}\alpha_i\\
\text{w.r.t.}\quad & \gamma\in\mathbb{R},\quad \boldsymbol{\alpha}\in\mathbb{R}_+^N\\
\text{s.t.}\quad & d_m\gamma\geq\left\|\sum_{i=1}^{N}\alpha_i y_i\Phi_m(\boldsymbol{x}_i)\right\|_2\quad\forall m\\
& \sum_{i=1}^{N}\alpha_i y_i=0\\
& C\geq\alpha_i\geq 0\quad\forall i.
\end{aligned} \tag{7}$$

This formulation has nonlinear constraints different from the original dual problem and is a case of conic programming. It is difficult to

solve this model because of the high space and time complexity. Squaring each side of the first set of constraints in (7) and replacing γ^2 with 2γ reveal the following QCQP problem:

$$\begin{aligned}
&\text{min.} \quad \gamma - \sum_{i=1}^{N} \alpha_i \\
&\text{w.r.t.} \quad \gamma \in \mathbb{R}, \quad \boldsymbol{\alpha} \in \mathbb{R}_+^N \\
&\text{s.t.} \quad d_m^2 \gamma \geq \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \overbrace{\langle \Phi_m(\boldsymbol{x}_i), \Phi_m(\boldsymbol{x}_j) \rangle}^{k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)} \quad \forall m \\
&\qquad \sum_{i=1}^{N} \alpha_i y_i = 0 \\
&\qquad C \geq \alpha_i \geq 0 \quad \forall i
\end{aligned}$$

where we again get the optimal kernel weights from the optimal dual variables ($\boldsymbol{\eta}$) corresponding to the first P constraints. By taking the derivative of the Lagrangian dual with respect to γ, we see that the kernel weights satisfy

$$\sum_{m=1}^{P} d_m^2 \eta_m = 1. \tag{8}$$

From the optimality conditions, we observe that $\boldsymbol{w}_m = \eta_m \sum_{i=1}^{N} \alpha_i y_i \Phi_m(\boldsymbol{x}_i)$ and by plugging this expression into (5), we obtain the same decision function as (4). Sonnenburg et al. [9] develop a semi-infinite linear programming equivalent to this model and a solution method to solve the model iteratively calling a canonical SVM solver, instead of solving it as a QCQP problem.

3. Regularizing multiple kernel learning via response surface methodology

We use the mathematical model in (6) developed by Bach et al. [8] as our base model, where the regularization term in the objective function consists of $\{\boldsymbol{w}_m\}_{m=1}^P$ and $\{d_m\}_{m=1}^P$. In general, $\{d_m\}_{m=1}^P$ are treated as scaling factors to balance the scale differences between kernel function outputs. It is a common procedure to normalize the kernel outputs before training:

$$\overline{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\sqrt{k(\boldsymbol{x}_i, \boldsymbol{x}_i) k(\boldsymbol{x}_j, \boldsymbol{x}_j)}}$$

and the trace of each normalized kernel becomes N. All $\{d_m\}_{m=1}^P$ values are taken as 1 after normalization; this strategy is used in Sonnenburg et al. [9].

The solution obtained from the optimization problem gives us a linear combination of kernel functions that satisfies (8) obtained from the Karush–Kuhn–Tucker optimality conditions [8]. This equation defines upper bounds for $\{\eta_m\}_{m=1}^P$ and we can control the feasible region for $\{\eta_m\}_{m=1}^P$ by changing $\{d_m\}_{m=1}^P$. For example, if the value assigned to d_m is high, the kernel corresponding weight η_m moves closer to zero and will be eliminated from the final kernel combination.

We consider $\{d_m\}_{m=1}^P$ as regularization parameters and from this perspective, the training process should also search for better $\{d_m\}_{m=1}^P$, instead of simply selecting them all equal to 1. In order to show the effect of $\{d_m\}_{m=1}^P$ on regularization, we create four toy data sets which are mixtures of Gaussians with different number of components (see Table 1) and we test the effect of different $\{d_m\}_{m=1}^P$ using MKL on them.

We combine three kernels (k_L, k_P, and k_G) on toy data sets by taking d_L as 1 and changing d_P and d_G on a grid in the log scale. We use the second-degree ($q=2$) polynomial kernel and estimate s in the Gaussian kernel as follows:

$$s = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{x}_{nn(i)}\|_2 \tag{9}$$

where $nn(i)$ is the index of the nearest neighbor of $\boldsymbol{x}_i$.

Fig. 1 shows the plot of the misclassification error on each data set for different values of d_P and d_G. The complexity of the discriminant increases as we increase the number of components, which increases the need for nonlinear kernels. We see that on Gauss2 and Gauss3 data sets, the best classification performance is

Table 1
Prior probabilities and Gaussian parameters used for toy data sets.

Data set	Positive class (*priors*, *means*, *covariances*)	Negative class (*priors*, *means*, *covariances*)
Gauss2	$p_1=0.50\ \mu_1=\begin{pmatrix}-1.0\\+1.0\end{pmatrix}\ \Sigma_1=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$	$p_2=0.50\ \mu_2=\begin{pmatrix}+1.0\\-2.2\end{pmatrix}\ \Sigma_2=\begin{pmatrix}0.8&0.0\\0.0&4.0\end{pmatrix}$
Gauss3	$p_1=0.25\ \mu_1=\begin{pmatrix}-2.0\\+1.0\end{pmatrix}\ \Sigma_1=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$ $p_2=0.25\ \mu_2=\begin{pmatrix}+2.0\\+1.0\end{pmatrix}\ \Sigma_2=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$	$p_3=0.50\ \mu_3=\begin{pmatrix}+0.0\\-2.2\end{pmatrix}\ \Sigma_3=\begin{pmatrix}0.8&0.0\\0.0&4.0\end{pmatrix}$
Gauss4	$p_1=0.25\ \mu_1=\begin{pmatrix}-3.0\\+1.0\end{pmatrix}\ \Sigma_1=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$ $p_2=0.25\ \mu_2=\begin{pmatrix}+1.0\\+1.0\end{pmatrix}\ \Sigma_2=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$	$p_3=0.25\ \mu_3=\begin{pmatrix}-1.0\\-2.2\end{pmatrix}\ \Sigma_3=\begin{pmatrix}0.8&0.0\\0.0&4.0\end{pmatrix}$ $p_4=0.25\ \mu_4=\begin{pmatrix}+3.0\\-2.2\end{pmatrix}\ \Sigma_4=\begin{pmatrix}0.8&0.0\\0.0&4.0\end{pmatrix}$
Gauss5	$p_1=0.16\ \mu_1=\begin{pmatrix}-4.0\\+1.0\end{pmatrix}\ \Sigma_1=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$ $p_2=0.18\ \mu_2=\begin{pmatrix}+0.0\\+1.0\end{pmatrix}\ \Sigma_2=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$ $p_3=0.16\ \mu_3=\begin{pmatrix}+4.0\\+1.0\end{pmatrix}\ \Sigma_3=\begin{pmatrix}0.8&0.0\\0.0&2.0\end{pmatrix}$	$p_4=0.25\ \mu_4=\begin{pmatrix}-2.0\\-2.2\end{pmatrix}\ \Sigma_4=\begin{pmatrix}0.8&0.0\\0.0&4.0\end{pmatrix}$ $p_5=0.25\ \mu_5=\begin{pmatrix}+2.0\\-2.2\end{pmatrix}\ \Sigma_5=\begin{pmatrix}0.8&0.0\\0.0&4.0\end{pmatrix}$

In the Gauss2 data set, each class has one Gaussian component. In the Gauss3 data set, the positive class has two and the negative class has one component. The Gauss4 data set has two components in each class and the Gauss5 data set has three components in the positive class and two components in the negative class.

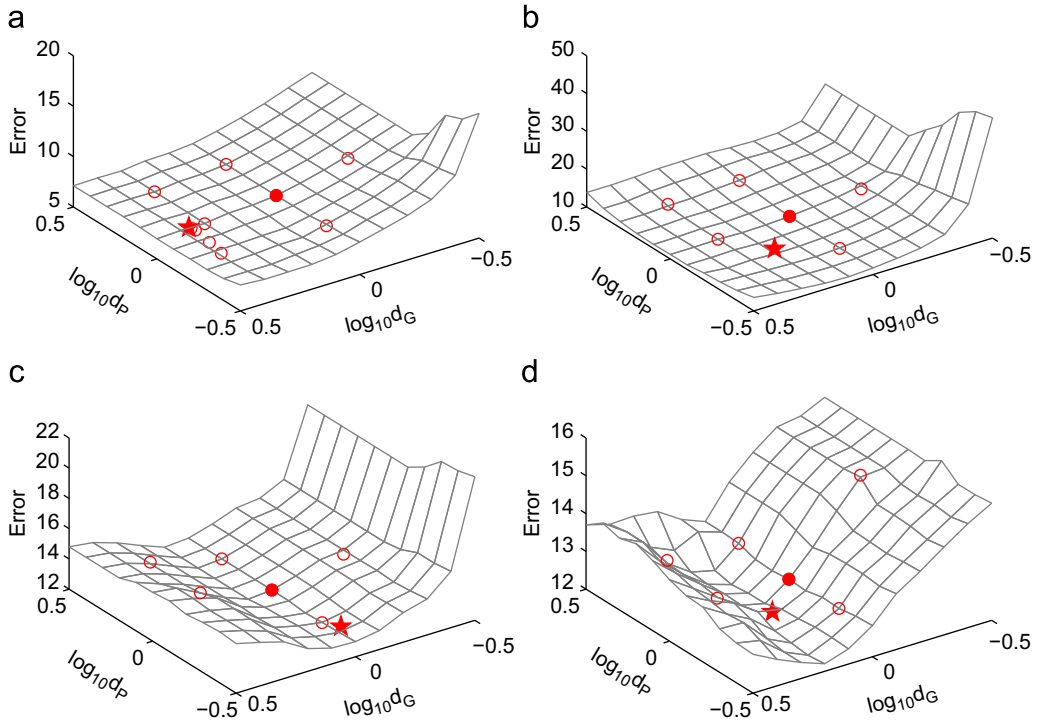

Fig. 1. The average validation error over $\{d_m\}_{m=1}^P$ grid on toy data sets. We see that taking $\{d_m\}_{m=1}^P = 1$ (0 in the log scale) may not always be optimal (shown by a filled circle). The circles show the sampled points and the star shows the solution found by our proposed response surface approach. (a) Gauss2. (b) Gauss3. (c) Gauss4. (d) Gauss5.

obtained when we choose d_P and d_G larger than or near 1, that is, when we force the model to use the more complex kernels k_P and k_G with smaller coefficients to avoid overfitting. On the other hand, d_P and d_G are chosen smaller than or near 1, resulting larger coefficients for k_P and k_G to avoid underfitting on Gauss4 and Gauss5 data sets. This indicates that each data set has its own smoothness constraint, and that $\{d_m\}_{m=1}^P$ should not just be taken equal but should be selected carefully to avoid overfitting or underfitting.

In this study, we propose to use RSM on validation error for selecting $\{d_m\}_{m=1}^P$ in an outer loop. RSM is a collection of statistical and mathematical techniques developed especially for process optimization [16]. It is assumed that the system response, r, is written as some unknown function of P factors, $\boldsymbol{d} = \{d_m\}_{m=1}^P$, as follows:

$$r = f(\{d_m\}_{m=1}^P) + \varepsilon$$

where ε is a random error component. We do not know $f(\cdot)$ and instead, we approximate it by $\hat{f}(\cdot)$ using a low-order polynomial function (e.g. quadratic). We start by taking a small sample $(\mathbf{D},\boldsymbol{r})$ of $\boldsymbol{d}$ and the corresponding r values, around some center in the $\boldsymbol{d}$ space. RSM consists of two basic steps: First, we fit the response surface $\hat{f}(\cdot)$ to $(\mathbf{D},\boldsymbol{r})$ and then, we optimize the response, that is, from $\hat{f}(\cdot)$ we sample a $\boldsymbol{d}^*$ value, which we believe will return a better $f(\cdot)$ value. In the case of a quadratic fit, this can be calculated analytically as the optimum point of $\hat{f}(\cdot)$. $\boldsymbol{d}^*$ and the corresponding actual r^* are added to $(\mathbf{D},\boldsymbol{r})$ and the procedure continues until there is no further improvement.

In our case, factors correspond to kernels and response corresponds to validation accuracy. We select a second-order model in the log scale (in order to ensure nonnegativity of $\{d_m\}_{m=1}^P$) for estimating the misclassification error. Without loss of generality, we can fix the regularization parameter for one of the kernels and the misclassification error can be expressed as

$$\beta_0 + \sum_{m=2}^{P} \beta_m \log_{10} d_m + \sum_{m=2}^{P} \sum_{h=m}^{P} \beta_{mh} \log_{10} d_m \log_{10} d_h$$

where β_0, $\{\beta_m\}_{m=1}^P$, and $\{\beta_{mh}\}_{m=1,h=m}^P$ are the model parameters.

At each iteration, because we are fitting a quadratic, at least $P(P+1)/2$ points are required to estimate the model parameters. To initialize, we use the second-order Koshal design [16] that uses three levels for each variable and with three factors (kernels), it is given as

$$\begin{array}{c} k_1 \\ k_2 \\ k_3 \end{array} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +\Delta & -\Delta & 0 & 0 & +\Delta \\ 0 & 0 & 0 & +\Delta & -\Delta & +\Delta \end{pmatrix}$$

where the rows correspond to kernels and the columns correspond to different $\{d_m\}_{m=1}^P$ values in the log scale assigned to kernels. For example, the first column is often referred as the "center point" in RSM (which in our case for the first iteration of RSM corresponds to having all $\{d_m\}_{m=1}^P$ equal to 1). Then, we construct "axial points" by moving towards negative and positive directions in each dimension by a small increment Δ (columns 2–5). The remaining runs required for estimating all model parameters are selected as "factorial points" by moving toward positive direction in each pair of dimensions (column 6); that is, we sample around the center point. For three kernels, the sample points of the initial grid are shown in Fig. 1 (Filled circle is the center point and empty circles are other points). We train MKLs at these points and check their misclassification error on the validation data. Then, we fit a quadratic to those set of errors and find analytically its optimum, which gives us the next sample point. We then sample there, add it to the list and make a new fit, until the improvement is negligible.

Using a second-order model in RSM corresponds to using Newton's method to find the minimum of the smoothed validation error curve. If the error curve obtained from sampled points is close to a quadratic surface, RSM converges very fast. If the error surface has an irregular shape with multiple local minima, RSM converges to one of the local minima like Newton's method.

Algorithm 1. Regularized multiple kernel learning (RMKL)

LD: the matrix containing weight vectors in log scale
VE: the matrix containing average cross-validation errors of each weight vector
$\boldsymbol{d}^{(t)}$: the weight vector at iteration t
$ve^{(t)}$: the average cross-validation error at iteration t
$\boldsymbol{ld}^{(t)}$: the weight vector at iteration t in log scale

1: $gridSize \Leftarrow P(P+1)/2$ — start of initialization
2: $\mathbf{LD} \Leftarrow \mathbf{0}$ — center point ($\mathbf{0}$ vector in the log scale)
3: **for** $m=2$ to P **do**
4: $\quad\mathbf{LD} \Leftarrow [\mathbf{LD} \;\; +\Delta\boldsymbol{e}_m]$ — high level for each dimension
5: $\quad\mathbf{LD} \Leftarrow [\mathbf{LD} \;\; -\Delta\boldsymbol{e}_m]$ — low level for each dimension
6: **end for**
7: **for** $m=2$ to P **do**
8: $\quad$**for** $h=m+1$ to P **do**
9: $\quad\quad\mathbf{LD} \Leftarrow [\mathbf{LD} \;\; +\Delta\boldsymbol{e}_m+\Delta\boldsymbol{e}_h]$ — high levels for each pair of dimensions
10: $\quad$**end for**
11: **end for**
12: **for** $t=1$ to $gridSize$ **do**
13: $\quad\boldsymbol{d}^{(t)} \Leftarrow \begin{pmatrix} 1 \\ 10^{\mathbf{LD}(:,t)} \end{pmatrix}$ — moves the point from the log scale to original space
14: $\quad ve^{(t)} \Leftarrow \text{TrainMKLSVM}(\boldsymbol{d}^{(t)})$ — returns the average error on validation sets
15: $\quad\mathbf{VE} \Leftarrow [\mathbf{VE} \quad ve^{(t)}]$
16: **end for** — end of initialization
17: **loop**
18: $\quad t \Leftarrow t+1$
19: $\quad\boldsymbol{ld}^{(t)} \Leftarrow \text{ResponseSurface}(\mathbf{LD},\mathbf{VE})$ — fits surface and finds the best point
20: $\quad\boldsymbol{d}^{(t)} \Leftarrow \begin{pmatrix} 1 \\ 10^{\boldsymbol{ld}^{(t-1)}} \end{pmatrix}$ — moves the point from the log scale to original space
21: $\quad$**if** $\|\boldsymbol{ld}^{(t)}-\boldsymbol{ld}^{(t-1)}\|_2 < \varepsilon$ **then** — checks convergence
22: $\quad\quad$**return** $\boldsymbol{d}^{(t-1)}$ — returns the last point
23: $\quad$**end if**
24: $\quad ve^{(t)} \Leftarrow \text{TrainMKLSVM}(\boldsymbol{d}^{(t)})$ — returns the average error on validation sets
25: $\quad\mathbf{LD} \Leftarrow [\mathbf{LD} \quad \boldsymbol{ld}^{(t)}]$
26: $\quad\mathbf{VE} \Leftarrow [\mathbf{VE} \quad \boldsymbol{ve}^{(t)}]$
27: **end loop**

quadratic response surface. The optimum operating point is calculated at its minimum. MKL method is trained at this point on the training set and its validation set (different from the training set) error is added to the sample points (Lines 17–27) and the next fit is done. RSM continues until convergence, which can be checked by $\|\boldsymbol{d}^{(t+1)}-\boldsymbol{d}^{(t)}\|_2 < \varepsilon$ where ε is a small threshold.

The usual approach for finetuning parameters in machine learning is exhaustive grid search. There are two main motivations for using RSM instead of exhaustive grid search: (a) RSM may require significantly fewer points: grid search requires $L^{(P-1)}$ points for the case with P kernels and L levels whereas RSM starts with $P(P+1)/2$ initial points and converges long before $L^{(P-1)}$ iterations, as our experiments show. (b) RSM can obtain the result in terms of arbitrary factor values other than predefined factor levels and does a finer search than the resolution of the grid.

Algorithm 1 illustrates the idea of our proposed regularized MKL (RMKL) in more detail. An initial search grid is constructed between Lines 2–11 using the second-order Koshal design. MKL is trained on validation sets with $\{d_m\}_{m=1}^{P}$ taken from the points of the initial search grid and validation errors are calculated for response surface calculations between Lines 12–16. The initial search grid points and their validation errors are used to start RSM and used to fit the

An example is given in Fig. 1. The stars show the points of convergence of RSM. We can see that it converges to points which are in deep valleys of the error function. The effect of regularization can also be clearly seen in Fig. 2. MKL method selects a combination with weights $(\eta_L-\eta_P-\eta_G)=(0.00-0.25-0.75)$ on the Gauss3 data set and it overfits. When we train with the proposed RSM-based method, it converges to the point $(d_L-d_P-d_G)=(1.00-0.85-1.46)$ selecting a different combination with weights $(\eta_L-\eta_P-\eta_G)=(0.00-0.48-0.31)$, shown by a star in Fig. 1(b). This particular run takes seven iterations where the first six iterations are used to initialize RSM. This solution implies that the second-order polynomial kernel is favored over the Gaussian kernel and this leads to a smaller combination weight for the Gaussian kernel and, as we see in Fig. 2(a) and (b), a smoother separating boundary using fewer support vectors.

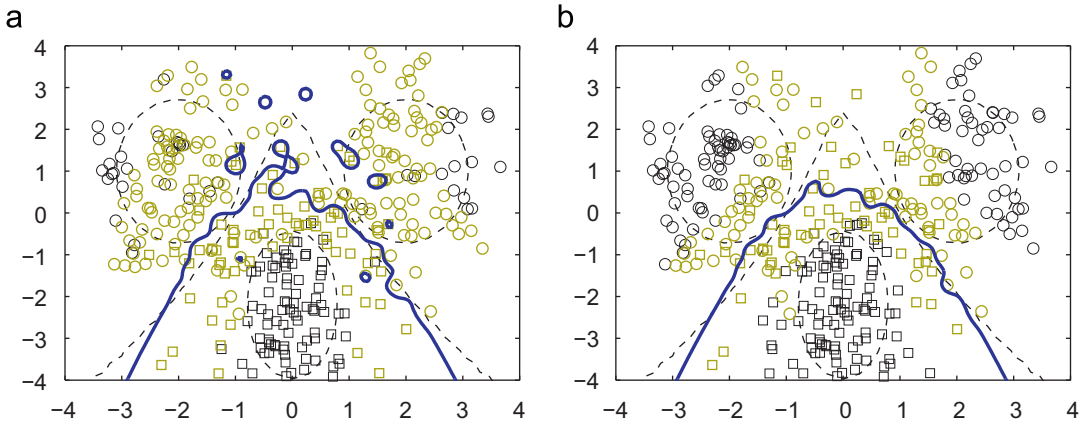

Fig. 2. Separating hyperplanes (solid lines) and support vectors (thick points) on the GAUSS3 data set with $(k_L–k_P–k_G)$ combination. Dashed lines show the Gaussians from which data are sampled and the optimal Bayes' discriminant. (a) MKL $(\eta_L–\eta_P–\eta_G)=(0.00–0.25–0.75)$. (b) RMKL $(\eta_L–\eta_P–\eta_G)=(0.00–0.48–0.31)$, shown by a star in Fig. 1(b). We see that the latter gives smaller weight to the Gaussian kernel favoring the second-order polynomial kernel and finds a smoother boundary using fewer support vectors.

Table 2
The average accuracies, support vector percentages, and combination weights with $(k_L–k_P–k_G)$ combination on bioinformatics data sets.

	MKL		RMKL		
	Test accuracy $\eta_L–\eta_P–\eta_G$	SV	Test accuracy $\eta_L–\eta_P–\eta_G$	SV	$d_L–d_P–d_G$
ACCEPTORS	90.45 ± 0.42 0.30–0.70–0.00	50.12 ± 1.11	**91.19 ± 0.38** 0.00–0.00–4.59	**44.90 ± 0.82**	1.00–1.55–0.47
DONORS	94.77 ± 0.28 0.11–0.03–0.86	27.78 ± 0.47	94.78 ± 0.21 0.17–0.00–0.73	**26.25 ± 0.45**	1.00–1.52–1.06
ARABIDOPSIS	85.29 ± 0.84 0.25–0.75–0.00	62.32 ± 1.10	**85.92 ± 0.85** 0.00–0.00–5.16	62.39 ± 0.98	1.00–1.00–0.44
VERTEBRATES	86.22 ± 0.22 0.20–0.80–0.00	53.84 ± 0.72	86.15 ± 0.30 0.70–0.15–0.00	**40.86 ± 0.44**	1.00–1.42–1.16
POLYADENYLATION	68.25 ± 1.24 0.01–0.16–0.84	72.14 ± 1.02	68.70 ± 1.34 0.00–0.00–1.09	72.81 ± 1.06	1.00–1.54–0.96

The regularization parameters found by RMKL are also reported.

4. Experiments

In our experiments, we use the MOSEK optimization package [17] to solve QCQP problems. MOSEK enables us to obtain the support vector coefficients ($\boldsymbol{\alpha}$) and the kernel combination weights ($\boldsymbol{\eta}$) from the primal–dual solution found.

Our methodology is as follows: Given a data set, if separate training and test splits are not supplied, a random one-third is reserved as the test set and then the remaining two-thirds is resampled using 5×2 cross-validation to generate 10 training and validation sets, with stratification. The validation sets of all folds are used to optimize C by trying values 0.01, 0.1, 1, 10, and 100. The best configuration is used to train the final learners on the training folds and their performance is measured on the test set. So, for each data set, we have 10 test set results.

In the result tables, we report the average test accuracy percentages and support vector percentages. These are made bold if the difference between MKL and the regularized variant is statistically significant using the 5×2 cv paired F test [18]. To check for significant difference on a number of data sets, we use Wilcoxon's signed-rank test [19].

Δ parameter for RSM method is selected as 0.3 in our experiments. This corresponds to selecting the low and high factor levels as 0.5 ($10^{-0.3}$) and 2 ($10^{+0.3}$).

4.1. Combining general purpose kernels

We perform experiments on five different bioinformatics data sets by combining the linear kernel, the second-degree polynomial kernel and the Gaussian kernel whose width parameter is estimated as in (9). ACCEPTORS and DONORS are human splice site detection data sets consisting of 3889 and 6246 data instances, respectively [20]. ARABIDOPSIS and VERTEBRATES are translation initiation site detection data sets containing 2048 and 13454 instances, respectively [21]. POLYADENYLATION is a polyadenylation signal prediction data set containing 9255 instances for human DNA and mRNA sequences [22]. We use the supplied training and test splits on ACCEPTORS, DONORS, and POLYADENYLATION data sets.

We see in Table 2 that RMKL uses statistically significantly fewer support vectors on ACCEPTORS and DONORS data sets and obtains statistically significantly higher accuracy on the ACCEPTORS data set. MKL uses k_L and k_P with nonzero weights whereas RMKL assigns zero weights to these kernels and uses only k_G on the ACCEPTORS data set. This selection improves the average testing accuracy statistically significantly and even though RMKL uses only the Gaussian kernel, it stores statistically significantly fewer support vectors. On the DONORS data set, RMKL assigns more weight to the linear kernel, removing the second-order polynomial kernel and using the Gaussian kernel less; the

accuracy does not change but the percentage of support vectors decreases statistically significantly. This shows that RMKL is able to choose between kernels depending on the complexity of the discriminant to be learned: If the boundary needs to be complex, the Gaussian kernel is favored; if it is simple, linear or polynomial kernels are given higher weights.

On the Arabidopsis data set, too, RMKL chooses to use only k_G instead of using both k_L and k_P. It obtains statistically significantly higher accuracy by storing a comparable number of support vectors. On the Vertebrates data set, RMKL gives higher weight to k_L than k_P and this causes a significant decrease in the support vector count. RMKL stores nearly 13 per cent fewer support vectors than MKL on the average.

On the Polyadenylation data set, RMKL obtains comparable accuracy and support vector count. k_L and k_P are replaced with k_G (one kernel is calculated instead of two) and RMKL increases the average testing accuracy but not statistically significantly.

We also perform experiments on the multiple features (MultiFeat) digit recognition data set from the UCI repository, composed of six different data representations for 2000 handwritten numerals. The properties of these different data representations are summarized in Table 3. Two binary classification problems are generated from the MultiFeat data set: In the EO data set, we separate even digits from odd digits; in the SL data set, we separate small ('0'–'4') digits from large ('5'–'9') digits.

In our experiments, we combine six linear kernels calculated from each of the representations by MKL and RMKL. We see in Table 4 that RMKL uses five data representations (eliminating Pix by assigning zero weight) on the EO data set whereas MKL uses all data representations with nonzero weights. Both methods obtain comparable average accuracy results on the test set. On the SL data set, RMKL eliminates two data representations (Pix and Zer) without a significant decrease in accuracy while storing statistically significantly fewer support vectors.

In order to see the differences between kernel combination rules, we apply kernel principal component analysis (KPCA) [3] to the kernels obtained using unweighted sum (Sum), MKL and RMKL. Fig. 3 gives the two-dimensional projections obtained on the EO data set. We see that using a weighted sum (learning the combination weights using MKL or RMKL) produces a better projection than using an unweighted sum; there does not seem to be a significant difference between the projections obtained by MKL and RMKL.

Table 3
Different data representations in the MultiFeat data set.

Name	Dimension	Data source
Fac	216	Profile correlations
Fou	76	Fourier coefficients of the character shapes
Kar	64	Karhunen–Loève coefficients
Mor	6	Morphological features
Pix	240	Pixel averages in 2×3 windows
Zer	47	Zernike moments

Table 5
Kernels for protein function prediction problem [14].

Kernel	Explanation	Data source
k_{Pfam}	Pfam kernel	Protein sequences
k_{PfamE}	Enriched Pfam kernel	Protein sequences
k_{TAP}	Diffusion kernel	Protein interactions
k_{Phys}	Diffusion kernel	Protein interactions
k_{Gen}	Diffusion kernel	Protein interactions
k_{Exp}	Correlation kernel	Gene expression profiles
k_{ExpG}	Gaussian kernel	Gene expression profiles
k_{SW}	Smith-Waterman kernel	Protein sequences

Table 4
The average accuracies, support vector percentages, and combination weights with (k_{Fac}–k_{Fou}–k_{Kar}–k_{Mor}–k_{Pix}–k_{Zer}) combination on the MultiFeat data set.

	MKL		RMKL		
	Test accuracy	SV	Test accuracy	SV	
	η_{Fac}–η_{Fou}–η_{Kar}–η_{Mor}–η_{Pix}–η_{Zer}		η_{Fac}–η_{Fou}–η_{Kar}–η_{Mor}–η_{Pix}–η_{Zer}		d_{Fac}–d_{Fou}–d_{Kar}–d_{Mor}–d_{Pix}–d_{Zer}
EO	98.31 ± 0.34	14.86 ± 0.79	98.18 ± 0.29	14.50 ± 0.96	
	0.30–0.29–0.11–0.01–0.28–0.02		0.40–0.24–0.22–0.05–0.00–0.27		1.00–1.07–1.02–0.58–1.63–0.53
SL	97.40 ± 0.37	32.59 ± 0.82	97.13 ± 0.31	$\mathbf{27.58 \pm 1.56}$	
	0.25–0.30–0.09–0.15–0.15–0.07		0.31–1.11–0.13–0.47–0.00–0.00		1.00–0.59–1.19–0.50–1.58–2.14

The regularization parameters found by RMKL are also reported.

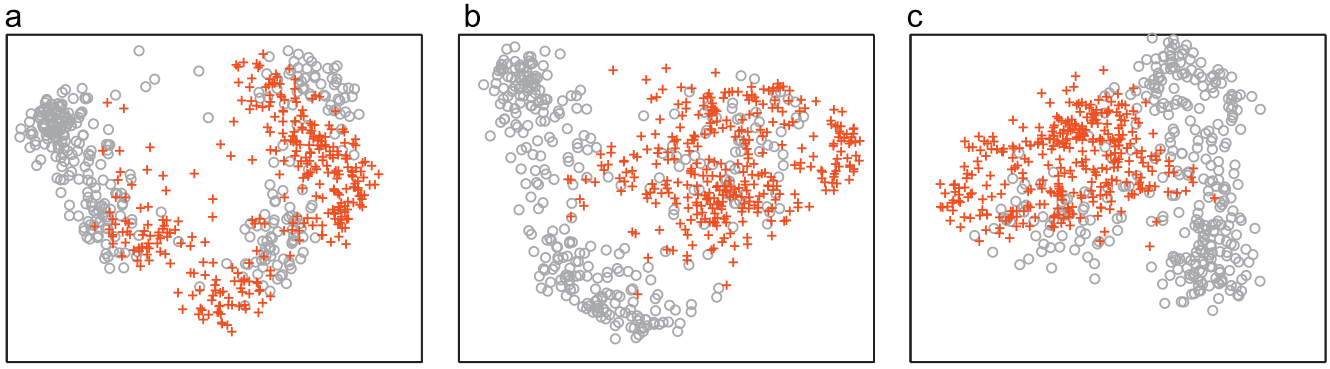

Fig. 3. Two-dimensional projections obtained on the EO data set by KPCA. (a) Sum: The first two eigenvectors explains 24.86% of the variance. (b) MKL: The first two eigenvectors explains 23.66% of the variance. (c) RMKL: The first two eigenvectors explains 16.32% of the variance.

Table 6
The average accuracies, support vector percentages, and combination weights with ($k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}$) combination on the protein function prediction experiments.

	MKL		RMKL		
	Test accuracy $\eta_{\text{Pfam}}-\eta_{\text{TAP}}-\eta_{\text{Phys}}-\eta_{\text{Gen}}-\eta_{\text{Exp}}$	SV	Test accuracy $\eta_{\text{Pfam}}-\eta_{\text{TAP}}-\eta_{\text{Phys}}-\eta_{\text{Gen}}-\eta_{\text{Exp}}$	SV $d_{\text{Pfam}}-d_{\text{TAP}}-d_{\text{Phys}}-d_{\text{Gen}}-d_{\text{Exp}}$	
Y1	78.59 ± 0.87	**91.50 ± 1.40**	79.21 ± 0.45	94.85 ± 1.05	
	0.70–0.04–0.08–0.14–0.05		0.46–0.06–0.05–0.12–0.09		1.00–1.23–1.32–1.30–1.33
Y2	93.23 ± 0.00	90.95 ± 3.15	93.56 ± 0.25	85.93 ± 2.51	
	0.37–0.03–0.15–0.22–0.22		0.34–0.25–0.41–0.00–0.00		1.00–1.00–1.00–2.06–2.06
Y3	87.17 ± 0.39	96.11 ± 0.40	87.47 ± 0.28	**93.14 ± 1.05**	
	0.12–0.02–0.18–0.32–0.37		0.23–0.07–0.25–0.45–0.00		1.00–1.00–1.00–1.00–2.08
Y4	85.62 ± 0.51	87.22 ± 1.43	87.07 ± 0.58	**82.81 ± 2.07**	
	0.22–0.03–0.24–0.30–0.21		0.34–0.22–0.33–0.04–0.00		1.00–0.94–1.10–1.33– 1.57
Y5	91.90 ± 0.26	91.24 ± 1.44	92.29 ± 0.21	**77.52 ± 4.48**	
	0.20–0.01–0.18–0.26–0.34		0.35–0.01–0.00–0.00–0.74		1.00–1.25–1.39–1.59–0.93
Y6	86.73 ± 0.36	93.58 ± 1.26	87.64 ± 0.43	92.78 ± 1.26	
	0.59–0.00–0.15–0.25–0.01		0.40–0.01–0.04–0.46–0.00		1.00–1.24–1.32–1.06–1.76
Y7	90.31 ± 0.30	**82.56 ± 0.88**	90.49 ± 0.36	86.45 ± 0.68	
	0.20–0.00–0.28–0.22–0.30		0.32–0.00–0.54–0.10–0.00		1.00–1.37–0.97–1.30–1.62
Y8	92.92 ± 0.20	91.49 ± 3.03	93.26 ± 0.26	95.91 ± 1.34	
	0.16–0.02–0.16–0.22–0.45		1.00–0.00–0.00–0.00–0.00		1.00–10.0–10.0–10.0–10.0
Y9	94.75 ± 0.10	82.28 ± 3.69	95.20 ± 0.40	84.18 ± 2.39	
	0.14–0.15–0.19–0.10–0.42		0.39–0.00–0.12–0.00–0.10		1.00–2.39–1.54–3.18–1.78
Y10	89.16 ± 0.36	93.39 ± 1.51	89.05 ± 0.32	**75.94 ± 2.87**	
	0.09–0.00–0.27–0.24–0.39		0.23–0.00–0.25–0.45–0.00		1.00–1.41–1.13–1.00–1.42
Y11	94.65 ± 0.00	96.24 ± 1.05	94.43 ± 0.13	**86.68 ± 1.79**	
	0.05–0.03–0.15–0.17–0.60		0.22–0.15–0.24–0.39–0.00		1.00–1.00–1.00–1.00–2.08
Y12	95.26 ± 0.33	76.32 ± 1.91	95.24 ± 0.45	79.63 ± 2.24	
	0.99–0.01–0.00–0.00–0.00		0.57–0.01–0.05–0.20–0.00		1.00–1.37–1.33–1.27–1.33
Y13	97.73 ± 0.04	77.14 ± 4.49	97.74 ± 0.00	**9.74 ± 0.83**	
	0.11–0.12–0.15–0.18–0.44		0.13–0.15–0.20–0.00–0.52		1.00–1.00–1.00–1.41–1.00
5×2 cv paired F test (W–T–L)			0–13–0	6–5–2	
Direct comparison (W–T–L)			10–0–3	8–0–5	
Wilcoxon's rank test (W/T/L)			W	T	

The regularization parameters found by RMKL are also reported.

4.2. Combining domain-specific kernels

We perform protein function prediction experiments on the MIPS Comprehensive Yeast Genome Database (CYGD) [23] which categorizes 3588 proteins into 13 top-level categories that can be interpreted as 13 binary classification tasks (Y1,Y2,…,Y13), one for each category. The reason for decomposing into binary classification problems instead of using a multiclass formulation is that some proteins belong to more than one category. We use the eight kernel functions shown in Table 5, also used in Lanckriet et al. [14].

Two different kernel subsets described in Lanckriet et al. [14] are also used in our experiments: (k_{Pfam}–k_{TAP}–k_{Phys}–k_{Gen}–k_{Exp}) and (k_{PfamE}–k_{TAP}–k_{Phys}–k_{Gen}–k_{ExpG}–k_{SW}). We also report the count of (W)ins–(T)ies–(L)osses on the 13 tasks with the regularized variant from direct comparison and the 5×2 cv paired F test. Wilcoxon's signed-rank test is used to compare MKL and RMKL in terms of average accuracies and support vector counts and the results are shown as (W)in, (T)ie, or (L)oss.

In Table 6, comparing MKL and our proposed regularized variant, RMKL, we see that the average accuracy percentage on the test set remains statistically similar on the 13 tasks. Direct comparison between average accuracies shows that the average accuracy increases on 10 out of 13 tasks, and Wilcoxon's signed-rank test finds a significant win of RMKL accuracy over MKL. Comparing support vector percentages, RMKL has six significant wins and eight direct comparison wins; using Wilcoxon's signed-rank test, however, there is no significant difference between the support vector percentages.

We can also compare the combination weights ($\{\eta_m\}_{m=1}^{P}$) for both methods and the regularization parameters ($\{d_m\}_{m=1}^{P}$) found by RMKL in Table 6, and we see that another difference between these two methods is the number of active kernels with nonzero weights. RMKL method, on 10 out of 13 tasks, assigns nonzero combination weights to fewer kernel functions, compared to MKL. This leads to a decrease in kernel calculations, and therefore testing time for new instances. As an extreme case, on the Y8 task, RMKL converges to a point where all $\{d_m\}_{m=1}^{P}$ coefficients except for the first kernel function are equal to 10 and are effectively pruned. For this task, we obtain combination rules that use only one kernel function (k_{Pfam}) with higher average accuracy. RMKL uses fewer kernels on Y2, Y3, Y4, Y7, Y10, and Y11 tasks by assigning a larger regularization parameter to k_{Exp}. This can be interpreted as an indication that gene expression profiles do not give useful information for these classification tasks and can safely be removed from the ensemble of kernels.

Table 7 lists the average accuracy and support vector percentages for the second subset of kernels. As on the first subset, the regularized variant achieves similar accuracy results and statistically significantly reduces the support vector percentage on six out of 13 tasks, increasing on four tasks. With direct comparison, the numbers of wins increase to six and eight for the average accuracy on the test set and support vectors stored, respectively. On this second subset, Wilcoxon's signed-rank test does not report a difference between MKL and RMKL in terms of average accuracies on the test set nor support vector counts.

Table 7 also gives the combination weights ($\{\eta_m\}_{m=1}^{P}$) for both methods and the regularization parameters ($\{d_m\}_{m=1}^{P}$) found by

Table 7
The average accuracies, support vector percentages, and combination weights with ($k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}$) combination on the protein function prediction experiments.

	MKL		RMKL		
	Test accuracy $\eta_{\text{PfamE}}-\eta_{\text{TAP}}-\eta_{\text{Phys}}-\eta_{\text{Gen}}-\eta_{\text{ExpG}}-\eta_{\text{SW}}$	SV	Test accuracy $\eta_{\text{PfamE}}-\eta_{\text{TAP}}-\eta_{\text{Phys}}-\eta_{\text{Gen}}-\eta_{\text{ExpG}}-\eta_{\text{SW}}$	SV	$d_{\text{PfamE}}-d_{\text{TAP}}-d_{\text{Phys}}-d_{\text{Gen}}-d_{\text{ExpG}}-d_{\text{SW}}$
Y1	80.72 ± 0.74 0.17–0.03–0.10–0.16–0.05–0.48	93.24 ± 0.78	80.68 ± 0.52 0.45–0.17–0.20–0.03–0.10–0.00	**89.29 ± 1.28**	1.00–1.02–1.06–1.18–1.02–2.52
Y2	94.32 ± 0.21 0.00–0.01–0.12–0.21–0.06–0.59	75.74 ± 2.90	94.48 ± 0.22 0.00–0.00–0.21–0.00–0.07–0.71	**72.15 ± 3.05**	1.00–2.06–1.00–2.06–1.00–1.00
Y3	87.51 ± 0.38 0.04–0.01–0.15–0.30–0.05–0.45	81.13 ± 1.69	87.53 ± 0.34 0.23–0.04–0.22–0.40–0.11–0.00	**73.17 ± 1.14**	1.00–1.00–1.00–1.00–1.00–2.08
Y4	85.63 ± 0.48 0.15–0.03–0.20–0.23–0.06–0.33	86.10 ± 1.11	85.53 ± 0.77 0.45–0.04–0.38–0.01–0.04–0.00	**76.23 ± 1.30**	1.00–1.12–1.05–1.37–1.25–1.50
Y5	93.68 ± 0.50 0.14–0.01–0.16–0.09–0.19–0.40	62.69 ± 3.23	94.59 ± 0.23 0.50–0.01–0.01–0.00–1.01–0.00	**34.16 ± 1.49**	1.00–1.21–1.32–1.46–0.68–1.48
Y6	87.44 ± 0.74 0.11–0.01–0.11–0.19–0.04–0.54	93.01 ± 0.96	87.90 ± 0.61 0.42–0.14–0.33–0.00–0.11–0.00	**84.12 ± 1.38**	1.00–1.00–1.00–2.06–1.00–2.06
Y7	91.05 ± 0.24 0.08–0.01–0.28–0.23–0.02–0.38	**75.72 ± 1.28**	90.70 ± 0.35 0.35–0.00–0.49–0.11–0.00–0.00	81.55 ± 1.09	1.00–1.57–1.00–1.22–1.46–1.50
Y8	93.78 ± 0.43 0.01–0.03–0.18–0.24–0.03–0.52	**57.60 ± 2.13**	93.77 ± 0.31 0.00–0.00–0.00–0.00–0.00–4.10	81.87 ± 1.66	1.00–1.34–1.36–1.17–1.29–0.49
Y9	95.00 ± 0.32 0.05–0.27–0.20–0.11–0.04–0.33	76.14 ± 2.69	94.99 ± 0.31 0.04–0.00–0.29–0.00–0.08–0.58	72.54 ± 3.77	1.00–2.06–1.00–2.06–1.00–1.00
Y10	89.56 ± 0.36 0.00–0.00–0.27–0.22–0.02–0.48	**70.95 ± 2.80**	89.58 ± 0.36 0.00–0.00–0.27–0.23–0.00–0.50	89.83 ± 1.49	1.00–1.00–1.00–1.00–2.08–1.00
Y11	94.43 ± 0.11 0.00–0.03–0.19–0.24–0.02–0.53	87.63 ± 1.79	94.43 ± 0.11 0.00–0.03–0.19–0.24–0.02–0.53	87.63 ± 1.79	1.00–1.00–1.00–1.00–1.00–1.00
Y12	96.41 ± 0.32 0.43–0.02–0.19–0.04–0.01–0.31	**36.42 ± 1.02**	96.22 ± 0.29 0.68–0.00–0.15–0.03–0.00–0.02	46.70 ± 1.53	1.00–1.38–1.25–1.25–1.38–1.25
Y13	97.80 ± 0.09 0.00–0.18–0.12–0.15–0.07–0.50	61.70 ± 9.56	97.83 ± 0.11 0.00–0.00–0.18–0.00–0.15–0.67	50.16 ± 9.26	1.00–2.06–1.00–2.06–1.00–1.00
5×2 cv paired F test (W–T–L)			0–13–0	6–3–4	
Direct comparison (W–T–L)			6–1–6	8–1–4	
Wilcoxon's rank test (W/T/L)			T	T	

The regularization parameters found by RMKL are also reported.

RMKL. On 12 out of 13 tasks, RMKL assigns nonzero weights to fewer kernel functions. As mentioned before, this leads to a significant reduction in the total time needed to calculate the output for a given test input. Note that here, we do not explicitly penalize nonzero weights; the number of kernels used decreases as a part of the regularization process. On the Y8 task, RMKL converges to a point where regularization coefficient for k_{SW} is smaller than 1. We can say that the Smith–Waterman kernel provides the most informative similarity measure for this classification task and its combination weight goes up to 4.10, removing all other kernels. A similar behavior is also observed for k_{PfamE} on Y1, Y4, Y5, Y6, and Y12 tasks. On all these tasks, RMKL assigns combination weights greater than 0.42 to k_{PfamE}. The reduction in the number of kernel functions used in the decision function after training is also observed in this set of experiments.

5. Related work and discussions

Chapelle et al. [24] propose a similar approach for choosing multiple parameters for SVMs. Their method tries to minimize the estimated test error bound and instead of explicitly fitting a response surface, updates parameters with a gradient-descent step calculated from the error bound. Our proposed method fits an approximate response surface for the test error using validation errors obtained over the sample points and finds the minimum point of the fitted response. Momma and Bennett [25] use a more similar approach to select support vector regression parameters. They do not fit a response surface either; instead, they perform a moving grid search strategy by changing the center point of the grid.

RSM is also used by Blum et al. [26] in protein structure prediction together with a Monte Carlo search procedure. Their model is not a kernel machine but they optimize the parameters of a specific function used in bioinformatics, called Rosetta energy function. They formulate the energy function in terms of input features and try to optimize it through optimizing the response surface. In order to get rid of irregularities emerging due to the high number of input features, they eliminate some of the features and calculate the response surface using the remaining features. In our case, the number of dimensions (factors) is limited by the number of kernel functions and this does not lead to any convergence problem in our experiments.

Regularization issues in multiple kernel learning have previously been studied. Bach et al. [10] try to learn the entire regularization path for multiple kernel learning. The regularization path is calculated for the parameter which corresponds to the C parameter in the objective function of (6). We do not consider the optimization of C in the regularization process, we simply use a cross-validation procedure for this purpose, but it can also be added to this process by appending it as another dimension (factor) to RSM. The effect of $\{d_m\}_{m=1}^{P}$ onto regularization is also mentioned, though their optimization is not discussed. Micchelli and Pontil [27,28] formulate the multiple kernel learning problem from a different perspective; they directly perform optimization for convex combination parameters using square loss regularization.

Bach et al. [10] and Bach [29] also state that selecting the regularization parameters ($\{d_m\}_{m=1}^{P}$) in a data-dependent manner may lead to better results. For example, Bach et al. [10] propose to

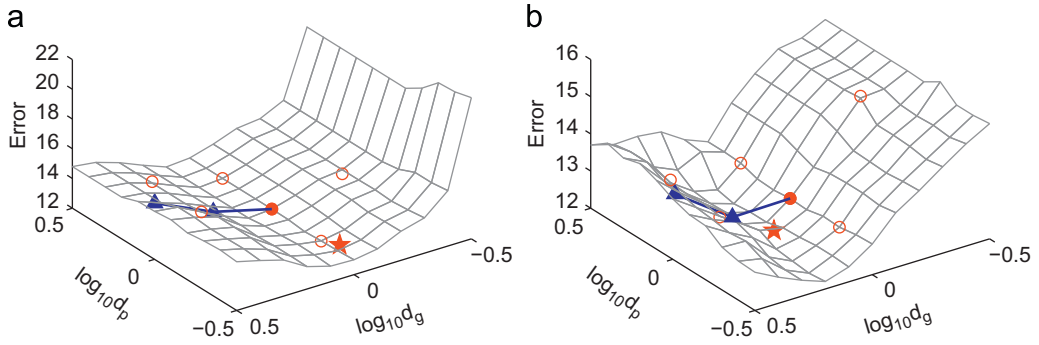

Fig. 4. Misclassification errors over $\{d_m\}_{m=1}^{P}$ grid on (a) Gauss4 and (b) Gauss5 data sets. The circles and star show the sampled points and the solution found by our proposed RSM approach. The triangles and line show the sampled points and search direction if we use the eigenvalues of combined kernel matrices, as used in Bach et al. [10]. (Only solutions for $\gamma=0$, 0.1, and 0.2 are shown; for other values of γ, the solutions fall outside.) We see that the best solution, shown by star, may not be on the path found using eigenvalues.

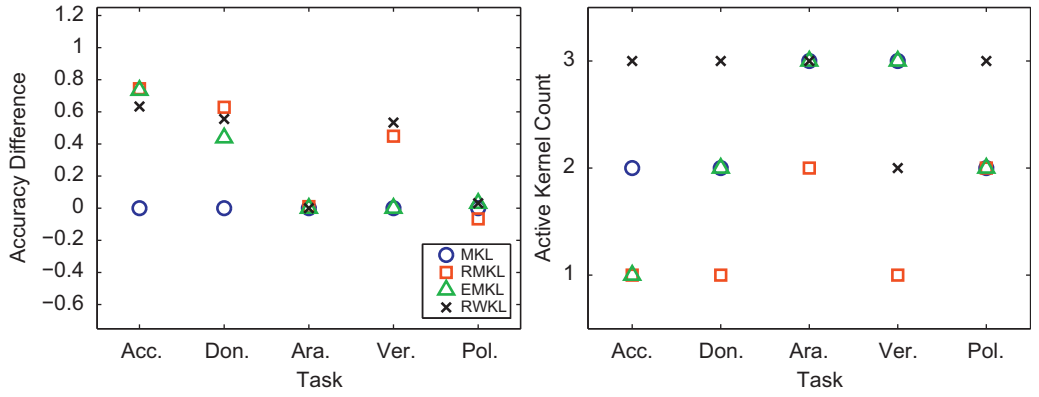

Fig. 5. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with (k_L–k_P–k_G) combination. In accuracy comparisons, the average accuracy of MKL is used as the baseline performance.

select these parameters by looking at the eigenvalues of combined kernel matrices and their methodology is as follows: Given P different kernel matrices, the numbers ($\{e_m\}_{m=1}^{P}$) of the eigenvalues greater than $1/2$ for each kernel matrix[1] are calculated and the regularization parameters are taken as $d_m=e_m^{\gamma}$ where γ is selected between 0 and 1 (we refer to this method as EMKL). In our experiments, we optimize γ by trying values 0,0.1,0.2,…,1 on the validation sets of all folds and choosing the best. Our proposed method using RSM selects the regularization parameters by looking at the performance measures obtained with different parameter selections and does not consider any prior information.

We also note that selecting the regularization parameters with the help of the eigenvalues integrates the kernel matrix complexity into the selection process before training. For example, the second-degree polynomial and the Gaussian kernel usually have higher $\{e_m\}_{m=1}^{P}$ values than the linear kernel and this leads to the penalization of these kernels if we choose γ larger than 0. Even if we use γ values smaller than 0, the regularization parameters are selected as a function of e_m values and this restricts us to use a predetermined region in the parameter space. Our method allows $\{d_m\}_{m=1}^{P}$ to converge to any point in the parameter space independently. The advantage of this difference can be clearly seen on Gauss4 and Gauss5 data sets in Fig. 4. The search direction obtained by using the eigenvalues (shown by the line) may not be a good direction for searching the optimum point of the response surface (shown by the star). EMKL can improve the performance of MKL but it performs parameter selection only on this search direction and the selected parameter set may be suboptimal. RMKL selects $\{d_m\}_{m=1}^{P}$ from the whole parameter space by starting from a grid of samples (circles) around the center point (corresponding to the original MKL selection).

On the Arabidopsis data set for example, as shown in Table 2, RMKL favors the Gaussian kernel instead of the linear and the second-degree polynomial kernels. On this data set, if we use the eigenvalues to decide on the regularization parameters, we penalize the Gaussian kernel due to the high number of large eigenvalues and fail to obtain the result obtained by RMKL.

It is also possible to perform RSM on combination weights ($\{\eta_m\}_{m=1}^{P}$) directly by solving the canonical SVM optimization problems without using the MKL formulation; we refer to this method as RWKL. This approach clearly speeds up the training phase but we cannot take advantage of the sparsity provided by the objective function of MKL formulation. The objective function of (6) forces the model to choose sparse kernel combinations whereas replacing the objective function of (1) with (3) does not favor sparsity.[2]

[1] In Bach et al. [10], it was $1/2N$ but we normalize kernel matrices to unit diagonal instead of unit trace. So, we count the eigenvalues greater than $1/2$.

[2] Actually, this is the approach we followed when we used the localized multiple kernel variant. There, we have $\eta_m(\boldsymbol{x}|\mathbf{V})$, an input-dependent kernel gating system whose parameters, $\mathbf{V}$, are learned by gradient-descent [30].

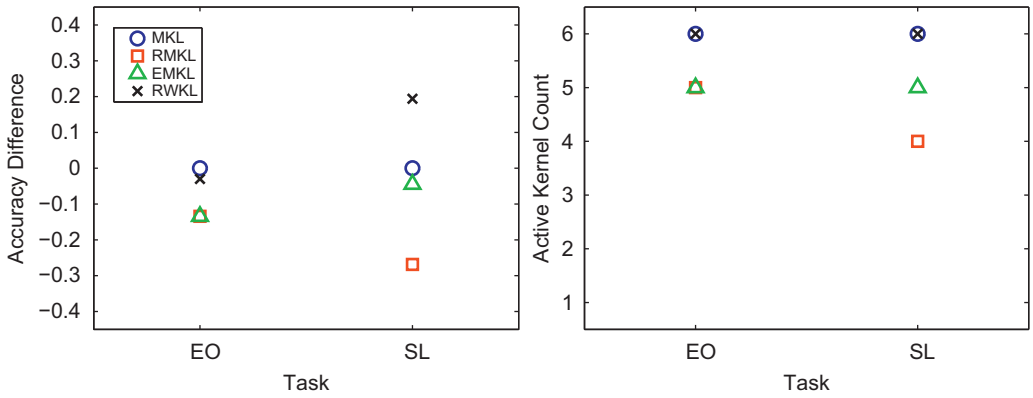

Fig. 6. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with (k_{FAC}–k_{FOU}–k_{KAR}–k_{MOR}–k_{PIX}–k_{ZER}) combination.

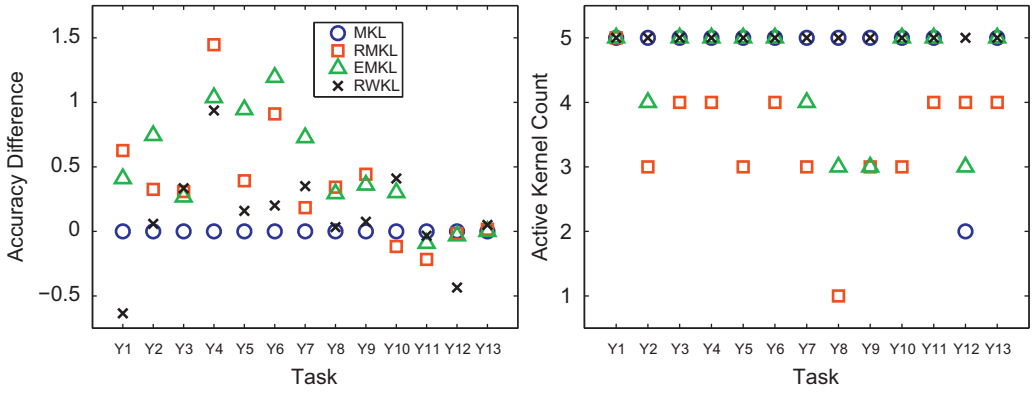

Fig. 7. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with ($k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}$) combination.

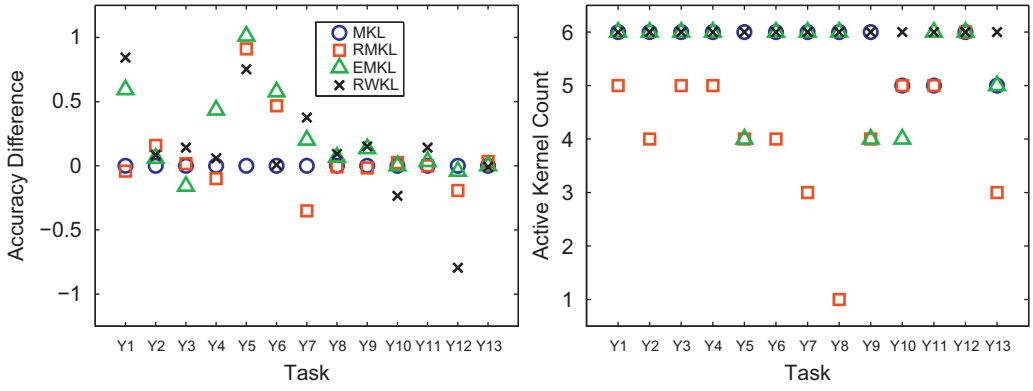

Fig. 8. Comparison of MKL, RMKL, EMKL, and RWKL methods in terms of the average test accuracy and number of kernels used with ($k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}$) combination.

Fig. 5 compares MKL, RMKL, EMKL, and RWKL methods with (k_L–k_P–k_G) combination on the bioinformatics data sets. We see that RMKL obtains better or similar accuracy results, compared to MKL on these data sets except POLYADENYLATION. RMKL, EMKL, and RWKL obtain similar results for all data sets. However, RMKL always uses fewer or as many kernels on all tasks.

Fig. 6 compares MKL, RMKL, EMKL, and RWKL methods with (k_{FAC}–k_{FOU}–k_{KAR}–k_{MOR}–k_{PIX}–k_{ZER}) combination on the MULTIFEAT data set. We see that RMKL obtains statistically similar accuracy results, compared to other methods on these two tasks, by using fewer or as many kernels.

When we compare MKL, RMKL, EMKL, and RWKL on the protein function prediction problem with ($k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}$) combination, we see in Fig. 7 that RMKL, EMKL, and RWKL generally obtain better accuracy results than MKL. However, there are significant differences between the number of used kernels by RMKL and other methods on almost all tasks. RMKL obtains smaller kernel ensembles due to the regularization and sparsity effects of MKL and eliminates the redundant kernels/data sources.

When we make the same comparison for ($k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}$) combination, Fig. 8 shows that all

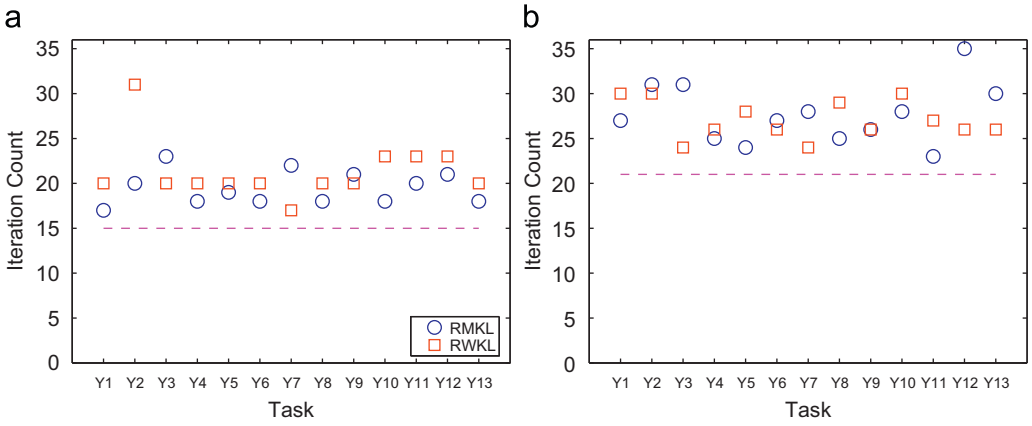

Fig. 9. The number of iterations performed by RMKL and RWKL with ($k_{Pfam}-k_{TAP}-k_{Phys}-k_{Gen}-k_{Exp}$) and ($k_{PfamE}-k_{TAP}-k_{Phys}-k_{Gen}-k_{ExpG}-k_{SW}$) combinations. Dashed lines show the number of iterations required to initialize RSM. (a) P=5. (b) P=6.

methods obtains comparable accuracy results on almost all tasks. RMKL again uses fewer kernels compared to the other three methods without diminishing accuracy.

The running time of the proposed method is directly related to the running time of the MKL solver and the convergence speed of RSM. The convergence time can be long especially with large number of kernels (in practice, we see convergence in terms of iterations) but it will always be faster and more detailed than exhaustive grid search. The running time can be decreased by using a subset of training points in the intermediate steps of RSM; once the best $\{d_m\}_{m=1}^{P}$ set is found, the whole training set can be used for final training before test. Fig. 9 shows the number of iterations performed by RMKL and RWKL for all tasks of protein function prediction problem. The running time of RMKL and EMKL is directly related to the number of times the MKL solver is called. RMKL calls the MKL solver 15–35 times for all tasks of protein function prediction problem, whereas EMKL calls the solver 11 times for the different γ values (0,0.1,0.2,…,1.0) we try. However, RWKL calls a canonical SVM solver at each iteration and 15–30 times in total. Clearly, the running time of RWKL is smaller than both RMKL and EMKL due to the complexity difference between the optimization problems of SVM and MKL.

RSM converges in 1–14 additional iterations for RMKL after the initialization phase and requires much fewer iterations than grid search, whereas grid search requires 81 (3^4) and 243 (3^5) iterations for the cases of P=5 and 6, respectively, if we use three levels for $\{d_m\}_{m=1}^{P}$ parameters and arbitrarily fix one of them. When P, the number of kernels, is high, RMKL has an advantage over grid search in terms of computational complexity. Fitting a quadratic approximation in RSM requires $\mathcal{O}(P^2)$ sample points which may be costly when P, the number of kernels, is large. One can fit a first-order RSM model using $\mathcal{O}(P)$ sample points to initialize and use gradient-descent to find the next point to be sampled. Still, the behavior and comparison of RSM and grid search, when P is on the order of tens or hundreds, remain a future study.

6. Conclusions

This work introduces a model selection procedure for selecting the regularization parameters in MKL by considering the cross-validation performance. We see that the newly introduced regularization parameters should not be set equal arbitrarily; our experimental results show that they have an effect on accuracy and we propose to use RSM to have a regularized variant of MKL to search for the best set using validation data. Our proposed method, RMKL, is tested on several bioinformatics and digit recognition data sets and we see that it eliminates some of the kernel functions or decreases the support vector count, sometimes also improves accuracy, but never sacrifices from accuracy.

Optimizing the regularization parameter of each kernel allows us to obtain more robust decision functions for the classification task at hand. Some of the kernels are eliminated as a side effect of regularization on validation data. Kernels which do not help increase the classification accuracy are pruned by selecting their regularization parameters accordingly, obtaining smoother discriminants and storing fewer support vectors. Eliminating some of the kernels directly or decreasing the number of stored support vectors reduces the testing time for new instances. In an application where there is a single source and multiple kernels, determining which kernels are favored and which are not needed gives us information, indicating which notions of similarity are valid. When different kernels use information from different sources, pruning kernels corresponds to pruning redundant information sources; after all, not all sources may be necessary.

Though our results are for binary classification, our approach is also applicable to combining multiple kernels in regression and outlier detection by suitably modifying the objective function and the constraints.

Acknowledgements

This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, the Boğaziçi University Scientific Research Project 07HA101, and the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant EEEAG 107E222. The work of M. Gönen was supported by the Ph.D. scholarship (2211) from TÜBİTAK.

References

[1] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, 1995.
[2] V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.
[3] B. Schölkopf, A.J. Smola, Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, 2002.
[4] J.M. Moguerza, A. Muñoz, Support vector machines with applications, Statistical Science 21 (3) (2006) 322–336.

[5] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, Text classification using string kernels, Journal of Machine Learning Research 2 (2002) 419–444.
[6] B. Schö lkopf, K.TsudaJ.P. Vert, Kernel Methods in Computational Biology, The MIT Press, 2004.
[7] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semi-definite programming, in: Proceedings of the 19th International Conference on Machine Learning, 2002.
[8] F.R. Bach, G.R.G. Lanckriet, M.I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, in: Proceedings of the 21st International Conference on Machine Learning, 2004.
[9] S. Sonnenburg, G. Rätsch, C. Schä fer, B. Schölkopf, Large scale multiple kernel learning, Journal of Machine Learning Research 7 (2006) 1531–1565.
[10] F.R. Bach, R. Thibaux, M.I. Jordan, Computing regularization paths for learning multiple kernels, in: Advances in Neural Information Processing Systems (NIPS), 2004.
[11] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000.
[12] P. Pavlidis, J. Weston, J. Cai, W.N. Grundy, Gene functional classification from heterogeneous data, in: Proceedings of the 5th Annual International Conference on Computational Molecular Biology, 2001.
[13] J.M. Moguerza, A. Muñoz, I.M. de Diego, Improving support vector classification via the combination of multiple sources of information, in: Proceedings of Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, 2004.
[14] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, Journal of Machine Learning Research 5 (2004) 27–72.
[15] G.R.G. Lanckriet, T.D. Bie, N. Cristianini, M.I. Jordan, W.S. Noble, A statistical framework for genomic data fusion, Bioinformatics 20 (16) (2004) 2626–2635.
[16] R.H. Myers, D.C. Montgomery, Response Surface Methodology: Process and Product Optimization using Designed Experiments, Wiley-Interscience, 2002.
[17] Mosek, 2010, The MOSEK Optimization Tools Manual Version 6.0 (Revision 66), MOSEK ApS, Denmark.
[18] E. Alpaydın, Combined 5×2 cv F test for comparing supervised classification learning algorithms, Neural Computation 11 (1999) 1885–1892.
[19] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–83.
[20] D. Kulp, D. Haussler, M.G. Reese, F.H. Eeckman, A generalized hidden Markov model for the recognition of human genes in DNA, in: Proceedings of 4th International Conference on Intelligent Systems for Molecular Biology, 1996.
[21] A.G. Pedersen, H. Nielsen, Neural network prediction of translation initiation sites in eukaryotes: perspectives for EST and genome analysis, in: Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology, 1997.
[22] H. Liu, H. Han, J. Li, L. Wong, An in-silico method for prediction of polyadenylation signals in human sequences, in: Proceedings of the 14th International Conference on Genome Informatics, 2003.
[23] H.W. Mewes, D. Frishman, C. Gruber, B. Geier, D. Haase, A. Kaps, K. Lemcke, G. Mannhaupt, F. Pfeiffer, C. Schüller, S. Stocker, B. Weil, MIPS: a database for genomes and protein sequences, Nucleic Acid Research 28 (2000) 37–40.
[24] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (2002) 131–159.
[25] M. Momma, K.P. Bennett, A pattern search method for model selection of support vector regression, in: Proceedings of the SIAM International Conference on Data Mining, 2002.
[26] B. Blum, M.I. Jordan, D. Kim, R. Das, P. Bradley, D. Baker, Feature selection methods for improving protein structure prediction with Rosetta, in: Advances in Neural Information Processing Systems (NIPS), 2007.
[27] C.A. Micchelli, M. Pontil, Learning the kernel function via regularization, Journal of Machine Learning Research 6 (2005) 1099–1125.
[28] C.A. Micchelli, M. Pontil, Feature space perspectives for learning the kernel, Machine Learning 66 (2007) 297–319.
[29] F. Bach, Consistency of the group Lasso and multiple kernel learning, Journal of Machine Learning Research 9 (2008) 1179–1225.
[30] M. Gönen, E. Alpaydın, Localized multiple kernel learning, in: Proceedings of the 25th International Conference on Machine Learning, 2008.

Mehmet Gönen received the B.Sc. degree in industrial engineering, the M.Sc. and the Ph.D. degrees in computer engineering from Boğaziçi University, İstanbul, Turkey, in 2003, 2005, and 2010, respectively. He is a Teaching Assistant at the Department of Computer Engineering, Boğaziçi University. His research interests include support vector machines, kernel methods, Bayesian methods, and real-time control and simulation of flexible manufacturing systems.

Ethem Alpaydın received his B.Sc. from the Department of Computer Engineering of Boğaziçi University in 1987 and the degree of Docteur es Sciences from Ecole Polytechnique Fédérale de Lausanne in 1990. He did his postdoctoral work at the International Computer Science Institute, Berkeley, in 1991 and afterwards was appointed as Assistant Professor at the Department of Computer Engineering of Boğaziçi University. He was promoted to Associate Professor in 1996 and Professor in 2002 in the same department. As visiting researcher, he worked at the Department of Brain and Cognitive Sciences of MIT in 1994, the International Computer Science Institute, Berkeley, in 1997 and IDIAP, Switzerland, in 1998. He was awarded a Fulbright Senior scholarship in 1997 and received the Research Excellence Award from the Boğaziçi University Foundation in 1998, the Young Scientist Award from the Turkish Academy of Sciences in 2001 and the Scientific Encouragement Award from the Scientific and Technological Research Council of Turkey in 2002. His book Introduction to Machine Learning was published by The MIT Press in October 2004. Its German edition was published in 2008, its Chinese edition in 2009, and its second edition in 2010. Its Turkish edition is in preparation. He is a senior member of the IEEE, an editorial board member of The Computer Journal (Oxford University Press) and an associate editor of Pattern Recognition (Elsevier).