Learning Kernels with Upper Bounds of Leave-One-Out Error

Yong Liu
yongliu@tju.edu.cn

Shizhong Liao
szliao@tju.edu.cn

Yuexian Hou
yxhou@tju.edu.cn

School of Computer Science and Technology
Tianjin University, Tianjin 300072, P. R. China

ABSTRACT

We propose a new leaning method for Multiple Kernel Learning (MKL) based on the upper bounds of the leave-one-out error that is an almost unbiased estimate of the expected generalization error. Specifically, we first present two new formulations for MKL by minimizing the upper bounds of the leave-one-out error. Then, we compute the derivatives of these bounds and design an efficient iterative algorithm for solving these formulations. Experimental results show that the proposed method gives better accuracy results than that of both SVM with the uniform combination of basis kernels and other state-of-art kernel learning approaches.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter Learning*; H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier Design and Evaluation*

General Terms

Algorithms, Theory, Experimentation

Keywords

Multiple Kernel Learning (MKL), Support Vector Machine, Leave-One-Out Error, Generalization Error Bound

1. INTRODUCTION

Kernel methods, such as support vector machines (SVM), have been widely used in pattern recognition and machine learning. A good kernel function, which implicitly characterizes a suitable transformation of input data, can greatly benefit the accuracy of the predictor. However, when there are many available kernels, it is difficult for the user to pick out a suitable one.

Instead of using a single kernel, recent applications have shown that using multiple kernels can enhance the interpretability of the decision function and improve performance. In such cases, a convenient approach is to consider the kernel as a convex combination of the basis kernels. Within this framework, the problem of learning the kernel is transformed to the problem of deciding the combination coefficients. Lanckriet et al. [7] formulate this problem as a semidefinite programming problem which allows to learn the combination coefficients and SVM classifier together, which is usually called multiple kernel learning (MKL). To enhance the computational efficiency, different approaches for solving this MKL problem have been proposed, semi-infinite linear program [9], second-order cone program [6], gradient-based methods [8], and second-order optimization [2].

For SVM, it is well known that the estimation error, which denotes the gap between the expected error and the empirical error, is bounded by $\sqrt{O(R^2\gamma^{-2})/\ell}$, where R is the radius of the minimum enclosing ball of training data in the feature space endowed with the kernel used, γ is the margin of SVM classifier, and ℓ is the number of training data. However, most existing MKL approaches decide the value of combination coefficients only by maximizing the margin γ^2. In this way, although γ^2 is the maximum, the R^2 may be very large too, so the estimation error bound may be too loose to guarantee good generalization performance of SVM.

To address the above problem, we directly minimize the upper bounds of expected error to decide the combination coefficients. Specifically, we first present two minimization formulations for MKL based on upper bounds of leave-one-out error. Using the derivatives of these bounds with respect to combination coefficients, we then propose a gradient descent algorithm to address these minimization formulations. Experiments show that our method gives significant performance improvements both over SVM with the uniform combination of basis kernels and over other state-of-art kernel learning methods.

2. MULTIPLE KERNEL LEARNING

We consider the classification learning problem from training data $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)\}$ where $\boldsymbol{x}_i$ belongs to some input space $\mathcal{X}$ and $y_i \in \{+1, -1\}$ denoting the class label of examples $\boldsymbol{x}_i$. In the Support Vector Machines (SVM) methodology, we map these input points to a feature space using a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that defines an inner product in this feature space, that is, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$, where $\Phi(\boldsymbol{x})$ is a mapping of $\boldsymbol{x}$ to a reproducing kernel Hilbert space $\mathcal{H}$ induced by K.

Here, we consider the kernel $K_{\boldsymbol{\theta}}$ depending on a set of parameters $\boldsymbol{\theta}$. The classifier given by the SVM is $f(\boldsymbol{x}) = \operatorname{sign}(\sum_{i=1}^{\ell} \alpha_i^0 y_i K_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}) + b)$, where the coefficients $\boldsymbol{\alpha}^0$ are the solution of the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j K_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & \sum_{i=1}^{\ell} \alpha_i y_i = 0, \ \alpha_i \geq 0, \ i = 1, \dots, \ell. \end{aligned} \tag{1}$$

This formulation of the SVM optimization problem is called the *hard margin* formulation since no training errors are allowed. For the non-separable case, one needs to allow training errors which results in the so called *soft margin* SVM algorithm. It can be shown that *soft margin* SVM with quadratic penalization of errors can be considered as a special case of the *hard margin* version with the modified kernel (see [5]). So in the rest paper, we only focus on the *hard margin* SVM.

The used kernel we consider is a linear convex combination of multiple basis kernels, as

$$K_{\boldsymbol{\theta}} = \sum_{i=1}^{m} \theta_i K_i, \quad \text{with } \theta_i \geq 0, \ \sum_{i=1}^{m} \theta_i = 1,$$

where $K_i, i = 1 \dots, m$ are the basic kernels and m is the total number of basic kernels. Within this framework, the problem of learning the kernel is transformed to the problem of deciding the combination coefficients $\boldsymbol{\theta} = (\theta_1, \dots, \theta_m)^{\mathrm{T}}$.

2.1 Estimating the Performance of SVM

Ideally we would like to choose the value of the combination coefficients $\boldsymbol{\theta}$ that minimize the true risk of the SVM classifier. Unfortunately, since this quantity is not accessible, one has to build estimates or bounds for it.

Leave-One-Out Error.

The leave-one-out procedure consists of removing one element from the training data, constructing the decision rule on the basis of the remaining training data and then testing on the removed element.

Let us denote the number of errors in the leave-one-out procedure by $\mathcal{L}((\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_\ell, y_\ell))$ and by f^i the classifier obtained with an SVM when the training data $(\boldsymbol{x}_i, y_i)$ removed, so we can write that $\mathcal{L}((\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_\ell, y_\ell)) = \sum_{p=1}^{\ell} \phi(-y_p f^p(\boldsymbol{x}_p))$, where ϕ is a step function: $\phi(t) = 1$ when $t > 0$ and $\phi(t) = 0$ otherwise. It is known that the leave-one-out procedure gives an almost unbiased estimate of the expected generalization error.

Although the leave-one-out estimator is a good choice when estimating the generalization error, it is very costly to actually compute since it requires running the training algorithm ℓ times. The strategy is thus to upper bound or approximate this estimator by an easy to compute quantity T having.

Radius-Margin Bound.

For *hard margin* SVM without threshold (b=0), Vapnik [10] proposes the following upper bound of leave-one-out procedure:

$$\mathcal{L}((\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_\ell, y_\ell)) \leq \frac{R^2_{K_{\boldsymbol{\theta}}}}{\gamma^2_{K_{\boldsymbol{\theta}}}} =: T_{\mathrm{RM}},$$

where $R_{K_{\boldsymbol{\theta}}}$ is the radius of the smallest sphere enclosing the training points in the feature space and $\gamma_{K_{\boldsymbol{\theta}}}$ is the margin of the SVM classifier with $K_{\boldsymbol{\theta}}$.

Span Bound.

Vapnik & Chapelle [3] derive an estimate using the concept of span of support vectors. Under the assumption that the set of support vectors remains the same during the leave-one-out procedure,

$$\mathcal{L}((\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_\ell, y_\ell)) \leq \sum_{p=1}^{\ell} \phi\left(\alpha_p^0 S_p^2 - 1\right) =: T_{\mathrm{Span}},$$

S_p is the distance between the point $\Phi_{K_{\boldsymbol{\theta}}}(\boldsymbol{x}_p)$ and the set Γ_p, where $\Gamma_p = \left\{ \sum_{i \neq p, \alpha_i^0 > 0} \lambda_i \Phi_{K_{\boldsymbol{\theta}}}(\boldsymbol{x}_i) \middle| \sum_{i \neq p} \lambda_i = 1 \right\}$.

2.2 Multiple Kernel Learning Formulations

To guarantee good generalization performance of SVM, we consider to minimize the upper bounds of leave-one-out error. Two minimization formulations for Multiple Kernel Learning are proposed:
With the radius-margin bound.

$$\min_{\boldsymbol{\theta}} \ T_{\mathrm{RM}} := \frac{R^2_{K_{\boldsymbol{\theta}}}}{\gamma^2_{K_{\boldsymbol{\theta}}}}, \quad \text{s.t. } \theta_i \geq 0, \ \sum_{i=1}^{m} \theta_i = 1. \tag{2}$$

With the span bound:

$$\min_{\boldsymbol{\theta}} T_{\mathrm{Span}} := \sum_{p=1}^{\ell} \phi\left(\alpha_p^0 S_p^2 - 1\right), \ \text{s.t.} \theta_i \geq 0, \ \sum_{i=1}^{m} \theta_i = 1. \tag{3}$$

3. SOLVING THE MKL PROBLEM

In order to solve the optimization problems (2) and (3), the projected gradient algorithm is adopted, which requires the computation of the derivatives of T_{RM} and T_{Span}.

3.1 Computing Derivative of Radius-Margin

According to [4], we know that the radius of the minimum enclosing ball $R^2_{K_{\boldsymbol{\theta}}}$ can be obtained by:

$$\begin{aligned} R^2_{K_{\boldsymbol{\theta}}} &= \max_{\boldsymbol{\beta}} \sum_{i=1}^{\ell} \beta_i K_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i,j=1}^{\ell} \beta_i \beta_j K_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & \text{s.t.} \sum_{i=1}^{\ell} \beta_i = 1, \beta_i \geq 0. \end{aligned} \tag{4}$$

The derivative of T_{RM} with respect to θ_k is given in the following theorem:

THEOREM 1. *Let the parameters $\boldsymbol{\alpha}^0 = (\alpha_1^0, \dots, \alpha_\ell^0)^{\mathrm{T}}$ and $\boldsymbol{\beta}^0 = (\beta_1^0, \dots, \beta_\ell^0)^{\mathrm{T}}$ are the solutions of optimization problem (1) and (4) respectively. Denote vectors $\boldsymbol{\mu} = (K_k(\boldsymbol{x}_1, \boldsymbol{x}_1), \dots, K_k(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell))^{\mathrm{T}}$ and $\boldsymbol{\nu} = (K_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \boldsymbol{x}_1), \dots, K_{\boldsymbol{\theta}}(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell))^{\mathrm{T}}$. Then, the derivative of T_{RM} with respect to θ_k can be written as*

$$\frac{\partial T_{\mathrm{RM}}}{\partial \theta_k} = -\mathbf{1}^{\mathrm{T}} \boldsymbol{\alpha}^0 (\boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{\beta}^0 - {\boldsymbol{\beta}^0}^{\mathrm{T}} \boldsymbol{K}_k \boldsymbol{\beta}^0) + Z({\boldsymbol{\alpha}^0}^{\mathrm{T}} \boldsymbol{K}_k \boldsymbol{\alpha}^0)/2,$$

where $Z = \boldsymbol{\nu}^{\mathrm{T}} \boldsymbol{\beta}^0 - {\boldsymbol{\beta}^0}^{\mathrm{T}} \boldsymbol{K}_{\boldsymbol{\theta}} \boldsymbol{\beta}^0$, $\boldsymbol{K}_k$ and $\boldsymbol{K}_{\boldsymbol{\theta}}$ are the kernel matrices corresponding to kernel K_k and $K_{\boldsymbol{\theta}}$ respectively.

PROOF. Note that $\frac{\partial T_{\mathrm{RM}}}{\partial \theta_k} = \| \frac{1}{\gamma^2_{K_{\boldsymbol{\theta}}}} \|_2^2 \frac{\partial R^2_{K_{\boldsymbol{\theta}}}}{\partial \theta_k} + R^2_{K_{\boldsymbol{\theta}}} \frac{\partial \| 1/\gamma^2_{K_{\boldsymbol{\theta}}} \|_2^2}{\partial \theta_k}$.
According to [4], $\frac{\partial \| 1/\gamma^2_{K_{\boldsymbol{\theta}}} \|_2^2}{\partial \theta_k} = -\frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i^0 \alpha_j^0 y_i y_j \frac{\partial K_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial \theta_k}$

and $\frac{\partial R^2_{K_{\boldsymbol\theta}}}{\partial\theta_k} = \sum_{i=1}^{\ell}\beta_i^0\frac{\partial K_{\boldsymbol\theta}(\boldsymbol x_i,\boldsymbol x_i)}{\partial\theta_k} - \sum_{i,j=1}^{\ell}\beta_i^0\beta_j^0\frac{\partial K_{\boldsymbol\theta}(\boldsymbol x_i,\boldsymbol x_j)}{\partial\theta_k}$. Since $K_{\boldsymbol\theta} = \sum_{i=1}^m\theta_iK_i$, thus $\frac{\partial K_{\boldsymbol\theta}(\boldsymbol x_i,\boldsymbol x_j)}{\partial\theta_k} = K_k(\boldsymbol x_i,\boldsymbol x_j)$. It is shown in [10] that $1/\gamma^2_{K_{\boldsymbol\theta}} = \sum_{i=1}^{\ell}\alpha_i^0$. Thus the theorem is proven. □

3.2 Computing the Derivative of Span Bound

The estimate of the performance of SVM through the span bound (3) requires the use of the step function ϕ which is not differentiable. However, we would like to use the projected gradient method to minimize this estimate of the test error. So we must smooth this step function. Similar to [4], we consider to use a contracting function : $\phi(\boldsymbol x) = (1+\exp(-c\boldsymbol x+d))^{-1}$, where c, d are non-negative constants. In our experiment, we took $c = 5$ and $d = 0$ which is the same as in [4].

Let sv denote the set of support vectors, sv $= \{\boldsymbol x_i|\ \alpha_i^0 > 0, i = 1,\dots,\ell\}$, $\boldsymbol K_{\boldsymbol\theta_{\rm sv}}$ are the kernel matrix corresponding to sv, $\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}} = \begin{pmatrix}\boldsymbol K_{\boldsymbol\theta_{\rm sv}} & \boldsymbol 1\\ \boldsymbol 1^{\rm T} & 0\end{pmatrix}$ and $\boldsymbol K^0_{k_{\rm sv}} = \begin{pmatrix}\boldsymbol K_{k_{\rm sv}} & \boldsymbol 0\\ \boldsymbol 0^{\rm T} & 0\end{pmatrix}$, where $\boldsymbol 1$ is all-ones vector and $\boldsymbol 0$ is all-zeros vector.

THEOREM 2. *The derivative of the span S_p^2 with respect to the parameter θ_k can be written as*

$$\frac{\partial S_p^2}{\partial\theta_k} = S_p^4\left(\tilde{\boldsymbol K}^{-1}_{\boldsymbol\theta_{\rm sv}}\boldsymbol K^0_{k_{\rm sv}}\tilde{\boldsymbol K}^{-1}_{\boldsymbol\theta_{\rm sv}}\right)_{pp}. \tag{5}$$

PROOF. Note that $\frac{\partial\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}}{\partial\theta_k} = \boldsymbol K^0_{k_{\rm sv}}$. According to [4], we know that $S_p^2 = 1/\left(\tilde{\boldsymbol K}^{-1}_{\boldsymbol\theta_{\rm sv}}\right)_{pp}$. Thus, it is easy to verify that $\frac{\partial S_p^2}{\partial\theta_k} = S_p^4\left(\tilde{\boldsymbol K}^{-1}_{\boldsymbol\theta_{\rm sv}}\frac{\partial\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}}{\partial\theta_k}\tilde{\boldsymbol K}^{-1}_{\boldsymbol\theta_{\rm sv}}\right)_{pp} = S_p^4\left(\tilde{\boldsymbol K}^{-1}_{\boldsymbol\theta_{\rm sv}}\boldsymbol K^0_{k_{\rm sv}}\tilde{\boldsymbol K}^{-1}_{\boldsymbol\theta_{\rm sv}}\right)_{pp}$, the theorem is proven. □

There is however a problem in this approach: the value given by the span is not continuous. Inspired by the Chapelle et al. [4], we replace the constraint by a regularization term in the computation of the span to smooth the span value,

$$\bar S_p^2 = \min_{\boldsymbol\lambda,\sum\lambda_i=1}\left\|\Phi_{K_{\boldsymbol\theta}}(\boldsymbol x_p) - \sum_{i\neq p}^{\ell}\lambda_i\Phi_{K_{\boldsymbol\theta}}(\boldsymbol x_i)\right\| + \eta\sum_{i\neq p}^{\ell}\frac{1}{\alpha_i^0}\lambda_i^2.$$

With this new definition of the span, the $\bar S_p^2$ can be written as $\bar S_p^2 = 1/(\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}+\boldsymbol Q)^{-1}_{pp} - \boldsymbol Q_{pp}$, where $\boldsymbol Q$ is a diagonal matrix with elements $[\boldsymbol Q]_{ii} = \eta/\alpha_i^0$, $[\boldsymbol Q]_{n_{\rm sv}+1,n_{\rm sv}+1} = 0$ and $n_{\rm sv}$ is the number of support vectors.

THEOREM 3. *Assuming that $\boldsymbol G$ is a diagonal matrix with the elements $[\boldsymbol G]_{ii} = -\eta/(\alpha^0_{{\rm sv}_i})^2$ and $[\boldsymbol G]_{n_{\rm sv}+1,n_{\rm sv}+1} = 0$. $\bar{\boldsymbol A}$ is the inverse of $\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}$ with the last row and last column removed. Then*

$$\frac{\partial\bar S_p^2}{\partial\theta_k} = \left(1/\boldsymbol B^{-1}_{pp}\right)^2\left(\boldsymbol B^{-1}(\boldsymbol K^0_{k_{\rm sv}}+\boldsymbol G\boldsymbol F)\boldsymbol B^{-1}\right)_{pp} - (\boldsymbol G\boldsymbol F)_{pp},$$

where $\boldsymbol B = \tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}+\boldsymbol Q$, $\boldsymbol Y_{\rm sv} = diag((y_{{\rm sv}_1},\dots,y_{{\rm sv}_{n_{\rm sv}}})^{\rm T})$ and $\boldsymbol F = diag(\boldsymbol Y_{\rm sv}\bar{\boldsymbol A}\boldsymbol K_{k_{\rm sv}}\boldsymbol Y_{\rm sv}\boldsymbol\alpha^0_{\rm sv}; 0)$.

PROOF. Recall that $\bar S_p^2 = 1/(\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}+\boldsymbol Q)^{-1}_{pp} - \boldsymbol Q_{pp}$, Thus $\frac{\partial\bar S_p^2}{\partial\theta_k} = \left(1/\boldsymbol B^{-1}_{pp}\right)^2\left(\boldsymbol B^{-1}\left(\frac{\partial\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}}{\partial\theta_k}+\frac{\partial\boldsymbol Q}{\partial\theta_k}\right)\boldsymbol B^{-1}\right)_{pp} - \left(\frac{\partial\boldsymbol Q}{\partial\theta_k}\right)_{pp}$. According to [2], we know that $\frac{\partial\boldsymbol\alpha^0_{\rm sv}}{\partial\theta_k} = -\boldsymbol Y_{\rm sv}\bar{\boldsymbol A}\boldsymbol K_{k_{\rm sv}}\boldsymbol Y_{\rm sv}\boldsymbol\alpha^0_{\rm sv}$. Thus it is easy to verify that $\frac{\partial\boldsymbol Q}{\partial\theta_k} = \boldsymbol G\boldsymbol F$. In addition, since $\frac{\partial\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}}{\partial\theta_k} = \boldsymbol K^0_{k_{\rm sv}}$, the theorem is proven. □

4. ALGORITHM

For convenience, we use $T_{\boldsymbol\theta}$ to denote $T_{\rm RM}$ or $T_{\rm Span}$. With the derivative of $T_{\boldsymbol\theta}$ with respect to the combination coefficients, we use the standard gradient projection approach with the Armijo rule [1] for selecting step sizes to address optimization problems (2) and (3). The overall algorithm is described in Algorithm 1 and Algorithm 2 which show that the above described steps are performed until a stopping criterion is met. This stopping criterion can be either based on a duality gap, or more simply, on a maximal number of iterations, or on the variation of $\boldsymbol\theta$ between two consecutive steps.

Algorithm 1: Adaptive Radius-Margin Multiple Kernel Learning Algorithm (RMMKL)

```
 1: set $\theta_i^1 = \frac{1}{m}$ for $i = 1,\dots,m$
 2: for $t = 1, 2, \dots$ do
 3:    solve the classical SVM problem with
       $K_{\boldsymbol\theta} = \sum_{i=1}^m\theta_i^tK_i$;
 4:    solve the optimization problem (4);
 5:    compute $\frac{\partial T_{\rm RM}}{\partial\theta_k}$ for $k = 1, 2, \dots, m$;
 6:    compute descent direction $D_t$ and optimal step $\gamma_t$;
 7:    $\boldsymbol\theta_k^{t+1} \leftarrow \boldsymbol\theta_k^t + \gamma_tD_{t,k}$;
 8:    if stopping criterion then
 9:       break
10:    end if
11: end for
```

Algorithm 2: Adaptive Span Multiple Kernel Learning Algorithm (SPMKL)

```
 1: set $\theta_i^1 = \frac{1}{m}$ for $i = 1,\dots,m$
 2: for $t = 1, 2, \dots$ do
 3:    solve the classical SVM problem with
       $K_{\boldsymbol\theta} = \sum_{i=1}^m\theta_i^tK_i$;
 4:    compute the inverse of the matrices $\tilde{\boldsymbol K}_{\boldsymbol\theta_{\rm sv}}$ and $\boldsymbol B$;
 5:    compute $\frac{\partial T_{\rm Span}}{\partial\theta_k}$ for $k = 1, 2, \dots, m$;
 6:    compute descent direction $D_t$ and optimal step $\gamma_t$;
 7:    $\boldsymbol\theta_k^{t+1} \leftarrow \boldsymbol\theta_k^t + \gamma_tD_{t,k}$;
 8:    if stopping criterion then
 9:       break
10:    end if
11: end for
```

5. EXPERIMENTS

In this section, we illustrate the performances of our presented RMMKL and SPMKL approachs, in comparison with SVM with the uniform combination of basis kernels (Unif) where $K_{\boldsymbol\theta} = \sum_{i=1}^m\frac{1}{m}K_i$, the kernel learning approach (KL) [4] which only learns a single kernel, and the margin-based MKL method (MKL) [8].

The evaluation is made on eleven public available data sets from UCI repository [1] and LIBSVM Data[2] (see Table 1). For a fair comparison, we have selected the same termination criterion for the iterative algorithms (KL, MKL, RMMKL and SPMKL): iteration terminates when $\|\boldsymbol\theta^{t+1} - \boldsymbol\theta^t\|_2/\|\boldsymbol\theta^t\|_2 \le 0.01$ or the maximal number of iteration (500)

[1]http://www.ics.uci.edu/∼mlearn/MLRepository.html.
[2]http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Table 1: The testing accuracies (Acc) with standard deviations, and the average numbers of selected basis kernels (Nk). We set the numbers of our method to be bold if our method outperforms both Unif and other two kernel learning approaches.

Index	1		2		3		4		5	
Method	Unif		KL		MKL		RMMKL		SPMKL	
Data sets	Acc	Nk	Acc	Nk	Acc	Nk	Acc	Nk	Acc	Nk
Ionosphere	94.1 ± 1.2	20	84.5 ± 1.8	1	92.9 ± 1.6	3.8	**95.7** ± 1.0	2.8	**96.9** ± 0.8	2.6
Splice	52.7 ± 0.1	20	74.2 ± 2.5	1	79.8 ± 1.6	1.0	**88.0** ± 2.4	3.2	**88.2** ± 2.4	2.2
Live	58.0 ± 0.0	20	64.2 ± 3.9	1	59.1 ± 1.4	4.2	64.1 ± 4.2	3.6	**64.4** ± 4.2	4.0
Fourclass	81.2 ± 1.9	20	94.4 ± 1.4	1	97.8 ± 1.4	7.8	**100** ± 0.0	1.0	**100** ± 0.0	1.6
Heart	83.9 ± 2.1	20	83.4 ± 5.2	1	84.1 ± 5.7	7.4	**84.2** ± 5.4	5.2	84.0 ± 5.9	5.8
Germannum	70.0 ± 0.0	20	71.6 ± 1.8	1	70.0 ± 0.0	6.2	**73.7** ± 1.6	5.4	**74.1** ± 1.2	5.0
Musk1	62.4 ± 2.4	20	62.9 ± 3.1	1	85.5 ± 3.1	2.0	**93.3** ± 2.3	3.0	**93.5** ± 2.2	3.8
Mdbc	94.6 ± 1.7	20	97.5 ± 1.8	1	97.0 ± 1.8	1.2	97.4 ± 1.6	5.2	**98.4** ± 2.9	6.0
Mpbc	76.3 ± 2.7	20	58.0 ± 6.5	1	76.4 ± 2.9	7.2	**76.5** ± 1.6	2.8	**77.5** ± 2.9	3.2
Sonar	77.5 ± 1.8	20	80.2 ± 5.7	1	81.0 ± 5.2	2.6	**86.0** ± 2.6	2.8	**86.6** ± 2.6	2.6
Coloncancer	67.2 ± 11	20	78.4 ± 3.6	1	82.6 ± 8.5	13	**85.2** ± 4.2	7.2	**86.8** ± 5.6	5.6

has been reached. We use gaussian kernels $K_{\text{Gauss}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 / 2\sigma^2\right)$ and polynomial kernels $K_{\text{Ploy}}(\boldsymbol{x}, \boldsymbol{x}') = \left(1 + \boldsymbol{x} \cdot \boldsymbol{x}'\right)^d$ as our basis kernels: 10 gaussian kernels with bandwidths $\sigma \in \{0.5, 1, 2, 5, 7, 10, 12, 15, 17, 20\}$ and 10 polynomial kernels of degree d from 1 to 10.

The experimental setting of the KL is the same as the one used in [4]. The code of MKL is from SimpleMKL [8]. The initial $\boldsymbol{\theta}$ is set to be $\frac{1}{20}\mathbf{1}$. The trade-off coefficients C in SVM, KL, MKL, RMMKL and SPMKL are automatically determined by 4-fold cross-validations on training sets. In all methods, C is selected from the set $\{0.01, 0.1, 1, 10, 100\}$. For each data set, we have run all the algorithms 50 times with different training and test splits (50% of all the examples for training and testing).

The average accuracies with standard deviations and average numbers of selected basis kernels are reported in Table 1. The results in Table 1 can be summarized as follows: Our methods (Index 4,5) give the best results on most data sets. RMMKL outperforms all other methods (Index 1, 2, 3) on 9 out of 11 sets, and also give results closing to the best ones of other methods on the remaining 2 sets. In particular, RMMKL gains 5 or more percents of accuracies on Splice, Liver, Musk1 and Sonar over MKL, and gains more than 10 percents on Ionosphere, Splice, Musk1, Mpbc over KL. On the set Musk1, RMMKL even gains 20.4 percents over KL. For SPMKL (Index 5) the results are similar to RMMKL: SPMKL outperforms other methods (Index 1, 2, 3) on 10 out of 11 sets, with only 1 inverse result. So it implicates that learning based on the kernel expect error bound can guarantee good generalization performance of the SVM.

6. CONCLUSIONS

Based on the upper bounds of the leave-one-out error, we present an accurate and efficient MKL method. we first establish two optimization formulations, and then propose the efficient gradient-based algorithms, called RMMKL and SPMKL, for computing the formulations. Experimental results validate that our approach outperforms both SVM with the uniform combination of basic kernels and other state-of-art kernel learning methods.

Future work aims at improving both speed and sparsity in kernels of the algorithm and extending this method to other SVM algorithm such as the one-class SVM or the SVR. We also plan to explore a new criterion based on the spectral of the integral operators to measure the kernel or kernel matrix for MKL.

Acknowledgments

The work is supported in part by the FP7 Marie Curie International Research Staff Exchange Scheme (IRSES) under grant No. 247590, the National Natural Science Foundation of China under grant No. 61070044, and the Natural Science Foundation of Tianjin under grant No. 11JCYBJC00700.

7. REFERENCES

[1] D. P. Bertsekas. *Nonlinear programming.* Athena Scientific Belmont, MA, 1999.

[2] O. Chapelle and A. Rakotomamonjy. Second order optimization of kernel parameters. In *Proceedings of the NIPS Workshop on Kernel Learning: Automatic Selection of Optimal Kernels.* Cambridge, MA: MIT Press, 2008.

[3] O. Chapelle and V. Vapnik. Model selection for support vector machines. In *Advances in Neural Information Processing Systems 12.* Cambridge, MA: MIT Press, 1999.

[4] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning,* 46(1):131–159, 2002.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning,* 20(3):273–297, 1995.

[6] L. Jia, S. Liao, and L. Ding. Learning with uncertain kernel matrix set. *Journal of Computer Science and Technology,* 25(4):709–727, 2010.

[7] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research,* 5:27–72, 2004.

[8] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. Simplemkl. *Journal of Machine Learning Research,* 9:2491–2521, 2008.

[9] S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient multiple kernel learning algorithm. In *Advances in Neural Information Processing Systems 18.* Cambridge, MA: MIT Press, 2006.

[10] V. Vapnik. *Statistical learning theory.* Wiley-Interscience, New York, 1998.