Sample-adaptive Multiple Kernel Learning

Xinwang Liu, Lei Wang, Jian Zhang, Jianping Yin

Science and Technology on Parallel and
Distributed Processing Laboratory
National University of Defense Technology
Changsha, China, 410073

School of Computer Science and Software Engineering
University of Wollongong, NSW, Australia, 2522

Faculty of Engineering and Information Technology
University of Technology Sydney, NSW, Australia, 2007

School of Computer, National University of Defense Technology
Changsha, China, 4100735

Email: {1022xinwang.liu@gmail.com, leiw@uow.edu.au,
Jian.Zhang@uts.edu.au and jpyin@nudt.edu.cn} [*]

Abstract

Existing multiple kernel learning (MKL) algorithms *indiscriminately* apply a same set of kernel combination weights to all samples. However, the utility of base kernels could vary across samples and a base kernel useful for one sample could become noisy for another. In this case, rigidly applying a same set of kernel combination weights could adversely affect the learning performance. To improve this situation, we propose a sample-adaptive MKL algorithm, in which base kernels are allowed to be adaptively switched on/off with respect to each sample. We achieve this goal by assigning a latent binary variable to each base kernel when it is applied to a sample. The kernel combination weights and the latent variables are jointly optimized via margin maximization principle. As demonstrated on five benchmark data sets, the proposed algorithm consistently outperforms the comparable ones in the literature.

[*]This paper will be published at the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14) held July 27–31, 2014 in Qubec City, Qubec, Canada.

1 Introduction

Kernel methods such as support vector machines (SVMs) have been an active research topic in the past decade [17, 18, 5]. As well known, effectively learning an optimal kernel is of great importance to the success of kernel methods. Along this line of research, many pioneering kernel learning algorithms have been proposed [3, 13, 2]. In particular, multiple kernel learning (MKL) has attracted much attention [19, 16, 20, 8, 7, 11, 5, 21, 22, 10, 12, 6]. It not only provides an efficient way to learn an optimal kernel, but also builds an elegant framework to integrate multiple heterogeneous data sources. The existing research work on MKL has made significant contributions in two aspects: speeding up computation [19, 16, 20, 15, 1] and improving classification performance [7, 5, 12, 8, 6]. The first contribution makes MKL applicable to large scale learning tasks, while the second one helps MKL attain superior classification performance.

By pre-specifying a group of base kernels, existing MKL algorithms learn the kernel combination weights based on a given training sample set. Usually, a same set of combination weights is *indiscriminately* applied to all samples. Since the weight can be viewed as an indicator of the utility of a base kernel for classification, existing algorithms implicitly assume that this utility remains unchanged in classifying all the samples. Nevertheless, the utility of a base kernel could change with samples because i) In MKL, each base kernel represents an evaluation of the pair-wise sample similarity. This evaluation is not necessarily equally effective across all samples; ii) Some input features of a sample could be contaminated by noise in practical applications. This can make a base kernel computed with these input features unreliable for this specific sample; and iii) In addition, when the kernel value (sample similarity) of a base kernel is collected from human evaluation or laboratory test, unexpected errors in this process could significantly corrupt the similarity of a sample to the others. All these cases can turn a base kernel useful for one sample out to a noise for another. Nevertheless, this issue has not been well addressed in the current literature of MKL.

A straightforward solution to the above issue may be to learn an individual set of kernel combination weights for every sample. However, this will lead to an optimization problem with many (number of base kernels $\times$ number of samples) *real-valued* variables. Also, this will result in an over-flexible kernel learning model. A sufficiently strong regularization has to be imposed upon these combination weights, which could further complicate the optimization process (generally leading to a large scale quadratic-constrained quadratic programming for SVMs-based MKL). A possible remedy to reduce the number of variables and the model flexibility is to use a parametric model to predict the kernel combination weights for a sample, as developed in the work of localized MKL (LMKL) algorithm in [9], where a gating model is used to predict the kernel combination weights locally. The framework of LMKL is elegant and it is able to improve the performance of MKL. However, how to define an appropriate parametric model remains an issue. For example, the linear gating model in [9] assumes that kernel combination weights change smoothly in any local region.

Note that the assumption may not be true when base kernels are irregularly corrupted across different samples. In addition, when the input features of a sample are contaminated by noise, the kernel combination weights predicted via a parametric model will not be accurate any more.

To handle the variation of the utility of base kernels across samples, we hope that there is a latent mechanism in MKL that can dynamically choose an appropriate subset of base kernels for each sample. Following this idea, *we define a latent binary vector to each individual sample to adaptively switch each base kernel on/off*. More specifically, by switching a base kernel off for a sample we mean that this kernel will not take part in evaluating the similarity of this sample to any other samples. This equals to filling a whole row and column of the corresponding base kernel matrix with zeros. With this latent mechanism, we can maintain to learn a same set of kernel combination weights for all samples as before. This avoids an over-flexible learning model and the smoothness assumption required by a parametric model. Also, as will be seen, allocating a latent variable to switch the base kernels allows the optimization of the latent variables and the kernel combination weights to be well separated. This makes the proposed algorithm still largely follow existing MKL framework and take advantage of existing efficient optimization algorithms. Although the number of latent variables is still as large as the number of base kernels multiplied by the number of samples, their optimization can be decomposed in a sample-wise way. Each sub-problem in this decomposition is a 0/1 linear integer programming which can be solved via off-the-shelf packages. In addition, the proposed algorithm completely works at the kernel level and thus can effectively handle the corruption and noise at both the base kernel level and the input sample level (since noise at the input sample level will be finally reflected at the base kernel level). Experimental study is conducted on multiple MKL benchmark data sets. Compared with existing MKL algorithms, the proposed algorithm can consistently achieve higher classification performance.

2 Background and notations

In existing MKL literature, each sample is mapped onto a multiple-kernel-induced feature space and a linear classifier is learned in this space. The feature mapping used in MKL takes the form of $\phi(\cdot) = [\phi_1^\top(\cdot), \phi_2^\top(\cdot), \cdots, \phi_m^\top(\cdot)]^\top$, which are induced by m pre-defined base kernels $\{\kappa_p(\cdot,\cdot)\}_{p=1}^m$. MKL learns the classifier by maximizing the margin between classes via solving the following problem [2, 19]

$$\begin{aligned} &\min_{\{\boldsymbol{\omega}_p\}_{p=1}^m, b, \boldsymbol{\xi}} \frac{1}{2}\left(\sum\nolimits_{p=1}^m \|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}\right)^2 + C\sum\nolimits_{i=1}^n \xi_i \\ &s.t.\ y_i\left(\sum\nolimits_{p=1}^m \boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i) + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0,\ \forall i, \end{aligned} \tag{1}$$

where $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denotes a collection of n training samples and their labels, $\boldsymbol{\omega}_p$ and $\mathcal{H}_p$ represent the normal and the feature space corresponding to the

p-th base kernel, $\boldsymbol{\xi}$ and b are the slack variables and bias term, and C is a regularization parameter.

According to [14], the problem in Eq.(1) is proven to be equivalent to the one in Eq.(2)

$$\begin{aligned} \min_{\{\boldsymbol{\omega}_p\}_{p=1}^m, b, \boldsymbol{\xi}, \boldsymbol{\gamma}\in\Delta} \; & \frac{1}{2}\sum\nolimits_{p=1}^{m}\frac{\|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}^2}{\gamma_p} + C\sum\nolimits_{i=1}^{n}\xi_i \\ & s.t.\; y_i\left(\sum\nolimits_{p=1}^{m}\boldsymbol{\omega}_p^\top\phi_p(\mathbf{x}_i)+b\right) \geq 1-\xi_i,\; \xi_i \geq 0,\; \forall i, \end{aligned} \tag{2}$$

where γ_p is the combination weight of the p-th base kernel and $\Delta = \{\boldsymbol{\gamma} : \sum_{p=1}^m \gamma_p = 1,\, \gamma_p \geq 0,\, \forall p\}$. Usually, $\{\boldsymbol{\omega}_p\}_{p=1}^m$, b and $\boldsymbol{\xi}$ are obtained by solving the dual problem of the inner minimization problem. This leads to a min-max optimization problem in Eq.(3)

$$\begin{aligned} \min_{\boldsymbol{\gamma}\in\Delta}\max_{\boldsymbol{\alpha}} \; & -\frac{1}{2}(\boldsymbol{\alpha}\circ\mathbf{y})^\top\left(\sum_{p=1}^{m}\gamma_p\mathbf{K}_p\right)(\boldsymbol{\alpha}\circ\mathbf{y}) + \boldsymbol{\alpha}^\top\mathbf{e} \\ & s.t.\; \boldsymbol{\alpha}^\top\mathbf{y} = 0,\; 0 \leq \alpha_i \leq C, \forall i, \end{aligned} \tag{3}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_n]^\top$ are the Lagrange multipliers, $\mathbf{y} = [y_1, y_2, \cdots, y_n]^\top$, $(\boldsymbol{\alpha}\circ\mathbf{y})$ denotes the component-wise multiplication between $\boldsymbol{\alpha}$ and $\mathbf{y}$, $\mathbf{K}_p$ is the p-th base kernel matrix and $\mathbf{e}$ is a column vector with all elements being one. After obtaining the optimal $\boldsymbol{\alpha}$, b and $\boldsymbol{\gamma}$, the decision score of a test sample $\mathbf{x}$ is calculated via

$$f(\mathbf{x}) = \sum\nolimits_{i=1}^{n}\alpha_i y_i\left(\sum\nolimits_{p=1}^{m}\gamma_p K_p(\mathbf{x}_i, \mathbf{x})\right) + b. \tag{4}$$

As can been seen in Eq.(3) and Eq.(4), a same set of kernel combination weights $\boldsymbol{\gamma}$ is applied to all samples in existing MKL algorithms. Following our previous analysis, we improve this situation by allowing different samples to use different base kernel subsets.

3 Sample-adaptive MKL (SAMKL)

3.1 Problem formulation

Noting that a base kernel induces a feature mapping, switching off the base kernel can be conceptually realized by switching off the mapping. Therefore, we define the mapping of a sample $\mathbf{x}_i$ by introducing a latent variable for each base-kernel-induced feature mapping. Let $\mathbf{h}_i = [h_{i1}, h_{i2}, \cdots, h_{im}]^\top \in \{1, 0\}^m$ be the latent binary vector with respect to $\mathbf{x}_i$. Specifically, $h_{ip} = 1\,(p = 1, \cdots, m)$ means that the corresponding mapping is useful for the classification of $\mathbf{x}_i$, while $h_{ip} = 0$ indicates it is not. Let $1 \leq r_1 < r_2 < \cdots < r_l \leq m$ be the indexes of the l non-zero components of $\mathbf{h}_i$. The mapping of a sample $\mathbf{x}_i$ is defined as

$$\phi(\mathbf{x}_i; \mathbf{h}_i) = [\phi_{r_1}^\top(\mathbf{x}_i), \phi_{r_2}^\top(\mathbf{x}_i), \cdots, \phi_{r_l}^\top(\mathbf{x}_i)]^\top. \tag{5}$$

With the above definition, different samples will be mapped onto different sub-spaces of a common feature space, denoted by $[\phi_1^\top(\mathbf{x}), \phi_2^\top(\mathbf{x}), \cdots, \phi_m^\top(\mathbf{x})]^\top$. This is different from existing MKL where all samples are mapped onto the common feature space. We use the sample-based margin proposed in [4] to measure the margin of each sample in the respective sub-space.

Let $\boldsymbol{\omega} = [\boldsymbol{\omega}_1^\top, \cdots, \boldsymbol{\omega}_m^\top]^\top$ denote the normal in a common feature space, where $\boldsymbol{\omega}_p$ is the component of $\boldsymbol{\omega}$ corresponding to the mapping $\phi_p(\cdot)$ induced by the p-th base kernel. The sample-based margin of $\mathbf{x}_i$ in multi-kernel-induced feature space is defined as

$$\rho(\phi(\mathbf{x}_i; \mathbf{h}_i)) \triangleq \frac{y_i \left(\sum_{p=1}^m h_{ip} \boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i)\right)}{\sum_{p=1}^m h_{ip} \|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}}, \tag{6}$$

where only part of the components of $\boldsymbol{\omega}$, namely those for which $h_{ip} = 1$, are involved in calculating the margin of $\mathbf{x}_i$. When all base kernel mappings are useful, this definition reduces to the one used by the existing MKL algorithms.

Following the principle of margin maximization, we maximize the minimum of all sample-based margins to seek the optimal normal $\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m$ and all the latent variables $\mathbf{h}_1, \cdots, \mathbf{h}_n$. Mathematically, the objective of our proposed MKL is expressed as in Eq.(7),

$$\max_{\{\{\boldsymbol{\omega}_p\}_{p=1}^m, \{\mathbf{h}_i\}_{i=1}^n\}} \left(\min_{i=1,\cdots,n} \frac{y_i \left(\sum_{p=1}^m h_{ip} \boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i)\right)}{\sum_{p=1}^m h_{ip} \|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}} \right). \tag{7}$$

The optimization problem in Eq.(7) is more difficult than the traditional margin-maximization problem in SVMs due to the facts that: (1) Both the numerator and denominator in the inner minimization vary across samples, while the denominator in the traditional SVMs problem is shared by all samples; and (2) Compared with the traditional problem, a group of additional latent binary variables are required to be optimized in Eq.(7). In the following part, we propose an efficient alternate optimization algorithm to solve this problem. It is worth pointing out that Eq.(7) is a prototype used to show the essential idea of our approach. We will further refine this model step by step by incorporating the slack variables, bias term and the prior on latent variables in the following part.

3.2 Optimization

We define an auxiliary variable $\tau_i = \frac{\sum_{p=1}^m h_{ip} \|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}}{\sum_{p=1}^m \|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}}$ $(1 \le i \le n)$. By substituting $\boldsymbol{\tau} = [\tau_1, \tau_2, \cdots, \tau_n]^\top$ into Eq.(7), we obtain

$$\max_{\{\{\boldsymbol{\omega}_p\}_{p=1}^m, \{\mathbf{h}_i\}_{i=1}^n\}} \min_{i=1,\cdots,n} \frac{\frac{y_i}{\tau_i} \left(\sum_{p=1}^m h_{ip} \boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i)\right)}{\sum_{p=1}^m \|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}}. \tag{8}$$

Following the way in deriving the SVMs objective in [4], Eq.(8) can be further rewritten as a constrained optimization problem in Eq.(9)

$$\max_{\{\{\boldsymbol{\omega}_p\}_{p=1}^m, \{\mathbf{h}_i\}_{i=1}^n\}} \frac{1}{\sum\limits_{p=1}^m \|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}}; \ s.t. \ \frac{y_i}{\tau_i}\left(\sum_{p=1}^m h_{ip}\boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i)\right) \geq 1. \tag{9}$$

Similar to the derivation in SVMs and according to [14], the problem in Eq.(9) is equivalent to the one in Eq.(10),

$$\min_{\substack{\{\{\boldsymbol{\omega}_p\}_{p=1}^m, \{\mathbf{h}_i\}_{i=1}^n\} \\ \boldsymbol{\gamma}\in\Delta}} \frac{1}{2}\sum_{p=1}^m \frac{\|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}^2}{\gamma_p}; s.t. \ \frac{y_i}{\tau_i}\left(\sum_{p=1}^m h_{ip}\boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i)\right) \geq 1. \tag{10}$$

After adding the slack variables $\boldsymbol{\xi} = [\xi_1, \xi_2, \cdots, \xi_n]^\top$ and the bias term b, we arrive at the following problem

$$\begin{aligned} &\min_{\{\{\boldsymbol{\omega}_p\}_{p=1}^m, \{\mathbf{h}_i\}_{i=1}^n\}, \boldsymbol{\xi}, b, \boldsymbol{\gamma}\in\Delta} \left(\frac{1}{2}\sum\nolimits_{p=1}^m \frac{\|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}^2}{\gamma_p} + C\sum\nolimits_{i=1}^n \xi_i\right), \\ &s.t. \ \frac{y_i}{\tau_i}\left(\sum\nolimits_{p=1}^m h_{ip}\boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \ \xi_i \geq 0. \end{aligned} \tag{11}$$

Besides maximizing the margin and minimizing the training errors, an ℓ_1-norm regularization term should be imposed on $\mathbf{h}_i \, (1 \leq i \leq n)$, making it effectively eliminate the base kernels that are not helpful for the classification of $\mathbf{x}_i$. Moreover, in order to avoid over-fitting, a prior has to be imposed on $\mathbf{h}_i$. In this work we enforce $\mathbf{h}_i$ not to be far from a pre-specified $\mathbf{h}_0$. In doing so, we obtain the optimization problem of the proposed SAMKL as follows,

$$\begin{aligned} &\min_{\{\boldsymbol{\omega}_p, \mathbf{h}_i\}, \boldsymbol{\xi}, b, \boldsymbol{\gamma}\in\Delta} \left(\frac{1}{2}\sum_{p=1}^m \frac{\|\boldsymbol{\omega}_p\|_{\mathcal{H}_p}^2}{\gamma_p} + C\sum_{i=1}^n \xi_i + C'\sum_{i=1}^n \|\mathbf{h}_i\|_1\right) \\ &s.t. \ \frac{y_i}{\tau_i}\left(\sum_{p=1}^m h_{ip}\boldsymbol{\omega}_p^\top \phi_p(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \ \xi_i \geq 0, \ \forall i \\ &\qquad \|\mathbf{h}_i - \mathbf{h}_0\|_1 \leq m_0, \ \mathbf{h}_i \in \{0, 1\}^m, \ \forall i, \end{aligned} \tag{12}$$

where $\mathbf{h}_0$ is an initial estimate, which is also a binary vector. $\mathbf{h}_0$ can be either empirically set or obtained by solving an ℓ_1-norm MKL (for example, SimpleMKL [16]) and then setting $\mathbf{h}_0$ according to the non-zero kernel weights. m_0 is a pre-defined parameter controlling the deviation of each $\mathbf{h}_i$ from $\mathbf{h}_0$, and it is the number of bits that they can differ from each other.

As can be seen, jointly optimizing $\{\mathbf{h}_i\}_{i=1}^n$ and $\{\boldsymbol{\omega}_p\}_{p=1}^m$, $\boldsymbol{\xi}$, b, $\boldsymbol{\gamma}$ via Eq.(12) is difficult since it is a mixed integer optimization problem. Instead, we propose an alternate optimization procedure to solve it. Specifically, at the t-th iteration, we first optimize $\{\boldsymbol{\omega}_p\}_{p=1}^m, \boldsymbol{\xi}, b, \boldsymbol{\gamma}$ with fixed $\{\mathbf{h}_i^{t-1}\}_{i=1}^n$ and $\boldsymbol{\tau}^{t-1}$ by solving Eq.(11), whose dual problem is

$$\begin{aligned}
&\min_{\boldsymbol{\gamma}\in\Delta}\max_{\boldsymbol{\alpha}} \left(-\frac{1}{2}\sum_{p=1}^{m}\gamma_p\sum_{i,j=1}^{n}\frac{\alpha_i\alpha_j y_i y_j h_{ip}^{t-1}h_{jp}^{t-1}}{\tau_i^{t-1}\tau_j^{t-1}}K_p(\mathbf{x}_i,\mathbf{x}_j)+\sum_{i=1}^{n}\alpha_i\right) \\
&\qquad s.t.\ \sum_{i=1}^{n}\alpha_i\left(\frac{y_i}{\tau_i^{t-1}}\right)=0,\ 0\le\alpha_i\le C,\ \forall i.
\end{aligned} \tag{13}$$

This optimization problem can be efficiently solved via existing MKL packages. Let $(\boldsymbol{\alpha}^t,\boldsymbol{\gamma}^t,b^t)$ denote the solution at the t-th iteration, which is obtained based on the value of $\mathbf{h}$ and $\boldsymbol{\tau}$ at the $(t-1)$-th iteration. After that, we update $\mathbf{h}$ at the t-th iteration by solving the following integer linear programming problem,

$$\begin{aligned}
&\min_{\{\mathbf{h}_i\}_{i=1}^n}\sum_{i=1}^{n}\|\mathbf{h}_i\|_1 \\
&\qquad s.t. \frac{y_i}{\tau_i}(\mathbf{h}_i^\top\mathbf{g}_i^t+b^t)\ge 1-\xi_i, \mathbf{h}_i^\top(\mathbf{e}-2\mathbf{h}_0)+\mathbf{h}_0^\top\mathbf{e}\le m_0,
\end{aligned} \tag{14}$$

where $\mathbf{g}_i^t=[\gamma_1^t\sum_{j=1}^n\alpha_j^t\frac{y_j}{\tau_j^{t-1}}h_{j1}^{t-1}K_1(\mathbf{x}_j,\mathbf{x}_i),\cdots,\gamma_m^t\sum_{j=1}^n \alpha_j^t\frac{y_j}{\tau_j^{t-1}}h_{jm}^{t-1}K_m(\mathbf{x}_j,\mathbf{x}_i)]^\top$, $\mathbf{e}$ is a column vector with all elements being one, and $\{\mathbf{h}_i^{t-1}\}_{i=1}^n$ are the latent variables obtained at the $(t-1)$-th iteration. Note that the second linear constraint on $\mathbf{h}$ is due to the identity that $\|\mathbf{h}_i-\mathbf{h}_0\|_1=\mathbf{h}_i^\top(\mathbf{e}-2\mathbf{h}_0)+\mathbf{h}_0^\top\mathbf{e}$ for binary variables. Directly solving the optimization problem in Eq.(14) appears to be computationally intractable because its complexity is $\mathcal{O}(2^{nm})$. However, since the constraints are separately defined on each $\mathbf{h}_i$ and the objective function is a sum over each $\mathbf{h}_i$, the problem in Eq.(14) can be equivalently solved via solving n independent sub-problems, as stated in Eq.(15),

$$\begin{aligned}
&\min_{\mathbf{h}_i}\|\mathbf{h}_i\|_1 \\
&\qquad s.t. \frac{y_i}{\tau_i}(\mathbf{h}_i^\top\mathbf{g}_i^t+b^t)\ge 1-\xi_i, \mathbf{h}_i^\top(\mathbf{e}-2\mathbf{h}_0)+\mathbf{h}_0^\top\mathbf{e}\le m_0.
\end{aligned} \tag{15}$$

This reduces the total computational complexity to $\mathcal{O}(n\cdot 2^m)$. The optimization problem in Eq.(15) is a linear integer programming, which can be solved via off-the-shelf packages such as MOSEK[1]. After obtaining $\boldsymbol{\alpha}^t$, $\boldsymbol{\gamma}^t$ and $\mathbf{h}^t$, the value of $\boldsymbol{\tau}^t$ is obtained as

$$\tau_i^t=\frac{\sum_{p=1}^m h_{ip}^t\gamma_p^t\sqrt{\sum_{i,j=1}^n\alpha_i^t\alpha_j^t\frac{y_iy_j}{\tau_i^{t-1}\tau_j^{t-1}}h_{ip}^{t-1}h_{jp}^{t-1}K_p(\mathbf{x}_i,\mathbf{x}_j)}}{\sum_{p=1}^m \gamma_p^t\sqrt{\sum_{i,j=1}^n\alpha_i^t\alpha_j^t\frac{y_iy_j}{\tau_i^{t-1}\tau_j^{t-1}}h_{ip}^{t-1}h_{jp}^{t-1}K_p(\mathbf{x}_i,\mathbf{x}_j)}}, \tag{16}$$

where $\tau_i^{t-1}\,(1\le i\le n)$ is optimized in the last iteration. Our algorithm for solving SAMKL is presented in Algorithm 1, where obj^{t-1} and obj^t denote the objective values at the $(t-1)$-th and t-th iterations, respectively.

[1]http://www.mosek.com/

Algorithm 1 Proposed Sample-adaptive MKL Algorithm

```
1: Input: $\{\mathbf{K}_p\}_{p=1}^m$, $\mathbf{y}$, $C$ and $m_0$.
2: Output: $\boldsymbol{\alpha}$, $b$, $\boldsymbol{\gamma}$ and $\{\mathbf{h}_i\}_{i=1}^n$.
3: Initialize $\mathbf{h}_0$ and set $t = 1$ and $\boldsymbol{\tau}^0 = \mathbf{e}$.
4: repeat
5:     Update $(\boldsymbol{\alpha}^t, \boldsymbol{\gamma}^t, b^t, \boldsymbol{\xi}^t)$ by Eq.(13) with $(\mathbf{h}^{t-1}, \boldsymbol{\tau}^{t-1})$.
6:     for $i = 1$ to $n$ do
7:         Update $\mathbf{h}_i^t$ with $\left(\boldsymbol{\alpha}^t, \boldsymbol{\gamma}^t, b^t, \xi_i^t, \tau_i^{t-1}, \mathbf{h}_i^{t-1}\right)$ by Eq.(15).
8:     end for
9:     Update $\boldsymbol{\tau}^t$ with $\left(\boldsymbol{\alpha}^t, \boldsymbol{\gamma}^t, \mathbf{h}^t, \boldsymbol{\tau}^{t-1}, \mathbf{h}^{t-1}\right)$ by Eq.(16).
10:    $t = t + 1$.
11: until $\left(\mathrm{obj}^{t-1} - \mathrm{obj}^t\right)/\mathrm{obj}^t \le 1e-4$
```

4 Discussion

4.1 Number of switching patterns h

We analyze the number of possible patterns of $\mathbf{h}$ on a given training data set. It appears that this number could reach the value as high as $\sum_{i=0}^{m_0} C_m^i$. However, considering that each training sample can only contribute one unique pattern, the number of possible patterns will be actually capped by n, the number of training samples. Moreover, in practice this number is usually even less than n because different samples often share a same pattern $\mathbf{h}$, as can be observed from Figures 1(a), 1(b) and 1(c) in the experiments.

4.2 Inference

The classification procedure with the proposed SAMKL is slightly different from existing MKL algorithms, since the subset of base kernels that is useful for classification has to be inferred for a given test sample. By following the strategy in structure SVMs with latent variables [23], we enumerate all possible configurations (label and switching pattern) and select the pair maximizing the margin. Specifically, the class label of a sample in a binary classification case is inferred by

$$\begin{aligned}(y^\star, \mathbf{h}^\star) &= \arg\max_{\substack{y\in\{\pm 1\}\\ \mathbf{h}_i\in\mathcal{H}}} y\cdot\left(\boldsymbol{\omega}^\top\phi(\mathbf{x};\mathbf{h}_i)+b\right)\\ &= \arg\max_{\substack{y\in\{\pm 1\}\\ \mathbf{h}_i\in\mathcal{H}}} y\left(\sum_{p=1}^m \gamma_p \sum_{j=1}^n \frac{\alpha_j y_j h_{ip} h_{jp}}{\tau_j} K_p(\mathbf{x}_j, \mathbf{x}) + b\right),\end{aligned} \tag{17}$$

where $\mathcal{H} = \bigcup_{i=1}^n \{\mathbf{h}_i\}$ is the set of all the unique $\mathbf{h}$ patterns learned from the training stage. In the case of multi-class classification, a set of normal vectors $\hat{\boldsymbol{\omega}}_1, \cdots, \hat{\boldsymbol{\omega}}_L$ can be obtained when the one-versus-rest strategy is used, where L is the number of classes. We can work out the decision score for the l-th class as

$$\begin{aligned} f_l(\mathbf{x}) &= \max_{\mathbf{h}_i \in \mathcal{H}} \left(\hat{\boldsymbol{\omega}}_l^\top \phi(\mathbf{x}; \mathbf{h}_i) + \hat{b}_l \right) \\ &= \max_{\mathbf{h}_i \in \mathcal{H}} \left(\sum_{p=1}^{m} h_{ip} \gamma_p \sum_{j=1}^{n} \alpha_j \frac{y_j}{\tau_j} h_{jp} K_p(\mathbf{x}_j, \mathbf{x}) + \hat{b}_l \right). \end{aligned} \tag{18}$$

The class label is predicted as l which corresponds to the largest $f_l(\mathbf{x})$.

4.3 Computational efficiency

At each iteration, SAMKL alternately solves an MKL problem (line 5 in Algorithm 1) and n 0/1 programming problem (line $6-8$ in Algorithm 1). The size of each 0/1 programming problems is m, the number of base kernels. Note that m is usually not large in practical applications, for example, the largest m is 69 among all MKL benchmark data sets, which allows each 0/1 programming problem to be efficiently solved. In our experiments, Algorithm 1 converges in a small (usually less than ten) number of iterations. Also, the computation in the test stage is proportional to the size of $\mathcal{H}$, which is smaller than the number of training samples as analyzed previously. We will report the training and test time in the experiments.

5 Experimental result

Our experiment consists of two parts. The first part compares the proposed SAMKL with localized MKL (LMKL) in [9]. Both approaches assume that the base kernel weights vary across samples. Note that LMKL requires to access the original input data and only focuses on binary classification tasks. In the second part, the proposed SAMKL is compared with SimpleMKL [16], non-sparse MKL algorithm [20] and the unweighted MKL algorithm. All MKL algorithms are compared in terms of both classification performance and computational cost. Also, the adaptivity of the learned latent variables with respect to each training sample will be shown.

5.1 Results on protein fold prediction data set

We compare the proposed SAMKL with LMKL [9] on the protein fold prediction data set `http://mkl.ucsd.edu/dataset/`. SimpleMKL is also included as a reference. This data set has 27 classes and 12 base kernels, and its original input data are available. Besides, the training/test partition of this data set has been pre-specified. The code of LMKL is downloaded from the authors' website[2], which focuses on binary classification. To construct binary classification tasks, we select five classes from 27 classes with the largest number of training samples. Specifically, they are the 1st, 7th, 9th, 12th and 16th classes. Every two classes are selected to construct a binary classification task. By this way,

[2]http://users.ics.aalto.fi/gonen/icml08.php/

we generate ten binary classification tasks in total. For the proposed SAMKL, we empirically set $\mathbf{h}_0 = (1, 1, \cdots, 1)^\top$ and $m_0 = 3$. To ensure fair comparison, the regularization parameter C for all the three MKL algorithms is chosen from $[10^{-1}, 10^{0}, \cdots, 10^{4}]$ by five-fold cross-validation on training data sets. Each base kernel matrix is normalized to have a unit trace.

Table 1: Classification accuracy and traing/test time comparison among the proposed SAMKL, LMKL [9] and SimpleMKL [16] on the protein fold prediction data sets.

Task	SAMKL	LMKL	SimpleMKL
c7 vs. c16	**88.0** 32.7/0.04	80.4 12.1/0.01	86.9 2.9/0.01
c7 vs. c9	**91.2** 33.7/0.02	89.5 4.4/0.02	87.7 5.9/0.00
c7 vs. c1	**98.0** 21.1/0.02	96.0 4.3/0.02	**98.0** 2.7/0.00
c7 vs. c12	**85.7** 103.1/0.05	77.8 4.9/0.01	79.4 9.1/0.00
c16 vs. c9	**95.1** 89.5/0.02	83.6 5.3/0.01	**95.1** 13.9/0.00
c16 vs. c1	**98.2** 11.1/0.02	96.3 9.7/0.01	**98.2** 1.3/0.00
c16 vs. c12	**86.6** 39.5/0.02	80.6 17.5/0.01	**86.6** 3.1/0.00
c9 vs. c1	**94.7** 24.4/0.01	89.5 12.9/0.01	**94.7** 2.6/0.00
c9 vs. c12	**84.4** 25.3/0.01	68.8 4.2/0.01	**84.4** 5.8/0.00
c1 vs. c12	**92.0** 7.8/0.01	72.0 3.9/0.01	88.0 1.3/0.00

The classification accuracy on the ten tasks is reported in Table 1, where the highest values are shown in bold. As can be seen, the proposed SAMKL obtains better overall classification performance than LMKL and SimpleMKL, and its improvement over LMKL is significant. We attribute the superiority of SAMKL to its latent mechanism designed to adaptively switch off less useful base kernels for each individual sample during MKL. At the same time, the less satisfying performance of LMKL indicates that more appropriate parametric models for kernel weight prediction need to be sought and the appropriateness of the smoothness assumption may need to be reviewed.

The training and test time are reported in the second row of each cell in Table 1. Compared with SimpleMKL and LMKL, SAMKL leads to a mildly increased computational time due to the use of the proposed latent mechanism.

The learned latent variable $\mathbf{h}$ is shown in Figure 1(a), where one binary classification task (Class 7 vs. Class 9) is selected for demonstration. The $\mathbf{h}$ on each classification task is shown as an $n \times m$ matrix, where n and m are the number of training samples and base kernels, respectively. The red color indicates "1" while the blue and green colors indicate "0". The difference between blue and green color is clarified as follows. The blue color indicates those latent variables which switch off the base kernels whose weights are non-zeros. We call them "active" latent variables. The green color indicates those latent variables which switch off the base kernels whose weights are zeros. We call them "inactive" latent variables. As shown, $\mathbf{h}$ switches on/off the base kernels differently across training samples. Looking into these matrices shows that each row has exactly three "0"s. This is due to the constraint $\|\mathbf{h}_i - \mathbf{h}_0\| \leq m_0$, $\forall i$. This experiment preliminarily demonstrates the effectiveness and the properties of the proposed SAMKL.

5.2 Results on benchmark data sets

Table 2: Experimental comprison of the proposed SAMKL, SimpleMKL [16], ℓ_p-MKL [20] and UMKL on three protein and Caltech-101 data sets. The two rows of each cell represent mean accuracy ± standard deviation and training/test time (in seconds). Boldface means the best one.

DATASET	SAMKL	SIMPLEMKL	ℓ_p-MKL			UMKL
	PROPOSED		$p = 2$	$p = 4$	$p = 8$	
	90.6 ± 1.5	87.7 ± 2.3	86.4 ± 1.8	84.6 ± 2.0	83.8 ± 2.2	84.3 ± 1.9
PSORTPOS	343.0/5.8	157.0/0.8	9.2/0.1	5.1/0.1	3.5/0.1	0.6/0.1
	92.5 ± 1.0	89.9 ± 1.1	88.9 ± 1.1	87.7 ± 1.3	86.9 ± 1.2	84.0 ± 1.5
PSORTNEG	4666.8/54.1	1184.3/2.0	121.1/0.8	60.7/0.8	43.0/0.7	5.5/0.1
	89.5 ± 1.7	88.0 ± 1.70	86.4 ± 1.5	85.0 ± 1.9	84.1 ± 2.1	83.1 ± 2.2
PLANT	864.7/18.4	515.1/0.4	34.7/0.3	17.7/0.3	13.7/0.3	2.0/0.1
	67.2 ± 1.0	63.7 ± 1.3	65.3 ± 1.5	65.1 ± 1.5	65.1 ± 1.5	65.0 ± 1.8
CALTECH-101	157480/1591.7	30079/15.1	2906.8/1.5	1505.1/1.5	1007.3/1.5	165.7/0.2

Four benchmark data sets are used, including psortPos, psortNeg, plant and Caltech-101 data sets. All of them can be downloaded from `http://mkl.ucsd.edu/dataset/`. The first three are for protein subcellular localization and have been widely used by MKL algorithms [24]. Their class numbers are four, five and four, respectively. The 69 base kernel matrices have been pre-computed and provided on the above website. Caltech-101 contains 25 base kernel matrices based on a set of visual features extracted from the Caltech-101 object recognition data set. 15 training and 15 test examples are used for each class. The base kernel matrices of five random splits of training and test sets are pre-computed and provided. We compare the proposed SAMKL with the state-of-the-art MKL algorithms,

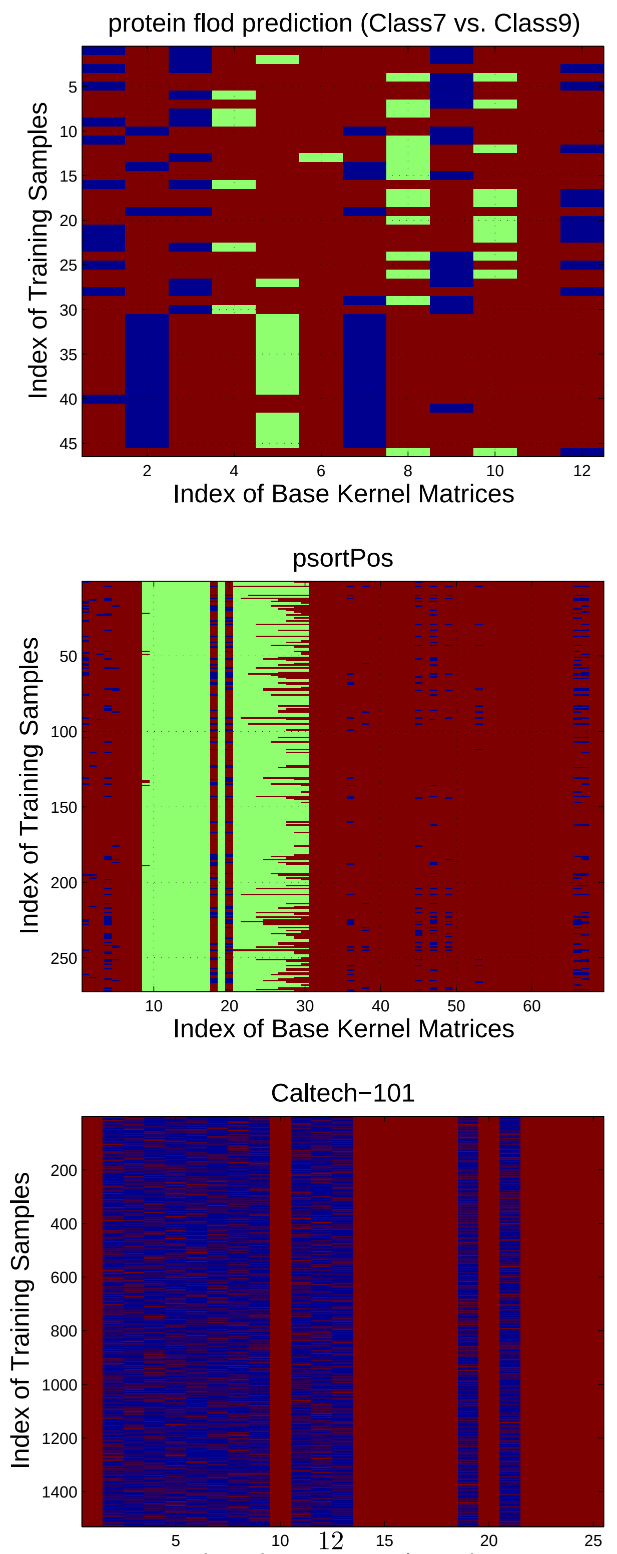

Figure 1: The latent variable **h** learned for each sample of a training group (red–"1", blue–"0 (active latent variables)", green–"0 (inactive latent variables)") on different data sets. (a): Class7 vs. Class9 (protein fold prediction); (b): psortPos; (c): Caltech-101.

including SimpleMKL [16] and ℓ_p-norm MKL [20] with $p = 2$, 4, 8, respectively. The uniformly weighted MKL (UMKL) is also included as a baseline. Note that LMKL in [9] is not included in comparison because it needs to access original input data, which are not available for all the above benchmark data sets. Furthermore, it focuses on binary classification tasks only.

Following [24], F1-score is used to measure classification performance on the psortPos and psortNeg data sets, while matthew correlation coefficient (MCC) is used for the plant data set. For Caltech-101, classification accuracy is used as in the literature. For the psortPos, psortNeg and plant data sets, we randomly split the data into 20 groups, with 50% : 50% for training and test. For Caltech-101, we use the five pre-defined training and test partitions. To conduct a rigorous comparison, the *paired Student's t-test* is performed. For the three protein data sets, C for each MKL algorithm is chosen from $[10^{-1}, 10^{0}, \cdots, 10^{4}]$ by five-fold cross-validation. For the Caltech-101, C is set to 10^4 experimentally. C' is not needed in our alternate optimization method. Each base kernel matrix is normalized to have a unit trace. For our proposed SAMKL, $\mathbf{h}_0$ is again set as $(1, 1, \cdots, 1)^\top$ and m_0 is empirically set as 20 and 10 on three protein data sets and Caltech-101, respectively.

5.2.1 Results on three protein and Caltech-101 data sets

As seen in Table 2, SAMKL consistently achieves superior performance to SimpleMKL [16], ℓ_p-norm MKL [20] and UMKL on all the three protein data sets. In the literature, SimpleMKL [16] is the one that achieves the best performance on these data sets [11]. However, SAMKL further improves its performance by 3.0%, 2.5% and 1.5%, respectively and these improvements are tested to be statistically significant. The above results indicate that introducing the latent variables allows each sample to effectively focus on more useful base kernels and avoid being affected by less useful ones, leading to better performance.

The latent variable $\mathbf{h}$ learned on the psortPos is plotted in Figure 1(b). As seen, $\mathbf{h}$ dynamically switches off many base kernels such as the 1st $\sim$ 5th, 18th and 20th, to name just a few. Cross-referencing the learned kernel combination weights, we can find that many base kernels switched off by $\mathbf{h}$ have non-zero combination weights (in blue color). Switching them allows different samples to effectively utilize an appropriate subset of base kernels. This confirms the sample-based adaptivity of the SAMKL.

The averaged result on Caltech101 is reported in the last row of Table 2. Again, we observe that SAMKL achieves the highest classification accuracy. Specifically, SAMKL gains 1.9% improvement over the second best one, i.e., ℓ_2-norm MKL. Also, the learned latent variables on a training group are plotted in Figure 1(c). In this figure, it is worth noting that all latent variables are "active" because we find that the optimal combination weights of all base kernels are non-zeros. All base kernels across samples are actively switched off by the latent variables. Together with the results on the protein subcellular localization, the result on Caltech-101 data set validates the effectiveness of the proposed SAMKL algorithm.

Note that the above training/test timing results are based on a relatively straightforward implementation of the Algorithm 1, where no specific tuning is conducted to speed up the training/test process. However, the n 0/1 programming problems at each iteration can be solved in a parallel way. Also, the test of each sample against different switching patterns can be performed parallelly. These properties will be exploited in future work to improve the computational efficiency of our algorithm.

6 Conclusion

This work proposes the SAMKL—a novel MKL algorithm which jointly performs MKL and infers the base kernel subsets that are useful for the classification of each sample. By allowing each sample to adaptively switch on/off each base kernel, SAMKL achieves clear improvement over the comparable MKL algorithms in the recent literature. Further improving the computational efficiency of the proposed SAMKL is another piece of our future work.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 60970034, 61170287, 61232016 and 61125201), the Australian Research Council Linkage Grant LP0991757 and UOW URC Near Miss Grant (DP120102145 and DP130100926). We thank Dr. Zenglin Xu for helpful discussions.

References

[1] Arash Afkanpour, András György, Csaba Szepesvári, and Michael Bowling. A randomized mirror descent algorithm for large scale multiple kernel learning. In *ICML*, pages 374–382, 2013.

[2] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004.

[3] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

[4] Gal Chechik, Geremy Heitz, Gal Elidan, Pieter Abbeel, and Daphne Koller. Max-margin classification of data with absent features. *Journal of Machine Learning Research*, 9:1–21, 2008.

[5] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13:795–828, 2012.

[6] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Multi-class classification with maximum margin multiple kernel. In *ICML*, pages 46–54, 2013.

[7] Kun Gai, Guangyun Chen, and Changshui Zhang. Learning kernels with radiuses of minimum enclosing balls. In *NIPS*, pages 649–657, 2010.

[8] Mehmet Gönen. Bayesian efficient multiple kernel learning. In *ICML*, 2012.

[9] Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *ICML*, pages 352–359, 2008.

[10] Chris Hinrichs, Vikas Singh, Jiming Peng, and Sterling C. Johnson. Q-mkl: Matrix-induced regularization in multi-kernel learning with applications to neuroimaging. In *NIPS*, pages 1430–1438, 2012.

[11] Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. l_p-norm multiple kernel learning. *Journal of Machine Learning Research*, 12:953–997, 2011.

[12] Abhishek Kumar, Alexandru Niculescu-Mizil, Koray Kavukcuoglu, and Hal Daumé III. A binary classification framework for two-stage multiple kernel learning. In *ICML*, pages 1295–1302, 2012.

[13] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[14] Charles A. Micchelli and Massimiliano Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

[15] Francesco Orabona and Jie Luo. Ultra-fast optimization algorithm for sparse multi kernel learning. In *ICML*, pages 249–256, 2011.

[16] Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

[17] Bernhard Schölkopf and Alex J. Smola. A short introduction to learning with kernels. In *Machine Learning Summer School*, pages 41–64, 2002.

[18] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[19] Sören Sonnenburg, Gunnar Rätsch, Christin Schäfer, and Bernhard Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[20] Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, pages 1175–1182, 2010.

[21] Fei Yan, Josef Kittler, Krystian Mikolajczyk, and Muhammad Atif Tahir. Non-sparse multiple kernel fisher discriminant analysis. *Journal of Machine Learning Research*, 13:607–642, 2012.

[22] Weilong Yang, Yang Wang, Arash Vahdat, and Greg Mori. Kernel latent svm for visual recognition. In *NIPS*, pages 818–826, 2012.

[23] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *ICML*, pages 1169–1176, 2009.

[24] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *ICML*, pages 1191–1198, 2007.