Two-Layer Multiple Kernel Learning

Jinfeng Zhuang
School of Computer Engineering
Nanyang Technological University
Singapore
zhua0016@ntu.edu.sg

Ivor W. Tsang
School of Computer Engineering
Nanyang Technological University
Singapore
IvorTsang@ntu.edu.sg

Steven C.H. Hoi
School of Computer Engineering
Nanyang Technological University
Singapore
chhoi@ntu.edu.sg

Abstract

Multiple Kernel Learning (MKL) aims to learn kernel machines for solving a real machine learning problem (e.g. classification) by exploring the combinations of multiple kernels. The traditional MKL approach is in general "shallow" in the sense that the target kernel is simply a linear (or convex) combination of some base kernels. In this paper, we investigate a framework of Multi-Layer Multiple Kernel Learning (MLMKL) that aims to learn "deep" kernel machines by exploring the combinations of multiple kernels in a multi-layer structure, which goes beyond the conventional MKL approach. Through a multiple layer mapping, the proposed MLMKL framework offers higher flexibility than the regular MKL for finding the optimal kernel for applications. As the first attempt to this new MKL framework, we present a Two-Layer Multiple Kernel Learning (2LMKL) method together with two efficient algorithms for classification tasks. We analyze their generalization performances and have conducted an extensive set of experiments over 16 benchmark datasets, in which encouraging results showed that our method performed better than the conventional MKL methods.

1 Introduction

Kernel learning is one of the active research topics in machine learning community. The family of kernel based machine learning algorithms have been extensively studied over the past decade [23]. Some well-known examples include Support Vector Machines (SVM) [9, 27], Kernel Logistic Regression [32], and Kernel PCA for denoising [20, 17], etc. These kernel methods have been successfully applied to a variety of real applications and often observed promising performance.

The most crucial element of a kernel method is *kernel*, which is in general a function that defines an inner product between any two examples in some induced Hilbert space [10, 23, 15]. By mapping data from an input space to the reproducing kernel Hilbert space (RKHS) [10], which could be potentially high-dimensional, traditional linear methods can be extended with reasonable effort to yield considerably better performance. Many empirical studies have shown that the choice of kernel often affects the resulting performance of kernel methods significantly. In fact, inappropriate kernels can result in sub-optimal or very poor performance.

For many real-world situations, it is often not an easy task to choose an appropriate kernel function, which usually may require some domain knowledge that would be difficult for non-expert users. To address such limitation, recent years have witnessed the active research of learning effective kernels automatically from data [4]. One popular example technique for kernel learning is *Multiple Kernel Learning* (MKL) [4, 24], which aims at learning a linear (or convex) combination of a set of predefined kernels in order to identify a good target kernel for the applications. Comparing with traditional kernel methods using a single fixed kernel, MKL does exhibit its strength of automated kernel parameter tuning and capability of concatenating heterogeneous data. Over the past few years, MKL has been actively investigated, in which a number of algorithms have been proposed to resolve the efficiency of MKL [4, 25, 21, 29], and a number of extended MKL techniques have been proposed to improve the regular

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

linear MKL method [11, 12, 3, 28, 16, 8].

Despite being studied actively, unfortunately existing MKL methods do not always produce considerably better empirical performance when comparing with a single kernel whose parameters were tuned by cross validation [11]. There are several possible reasons to account for such failure. One conjecture is that the target kernel domain $\mathcal{K}$ of MKL using a linear combination may not be rich enough to contain the optimal kernel. Therefore, some emerging study has attempted to improve the regular MKL by explore more general kernel domain $\mathcal{K}$ using some nonlinear combination [28]. Following the similar motivation, we speculate that the insignificant performance gain is probably due to the *shallow* learning nature of regular MKL that simply adopts a flat (linear/nonlinear) combination of multiple kernels.

To this end, this paper presents a novel framework of Multi-Layer Multiple Kernel Learning (MLMKL), which applies the idea of deep learning to improve the MKL task. Deep architecture has been being actively studied in machine learning community and has shown promising performance for some applications [13, 14, 19, 5]. Our study was partially inspired by the recent work [6] that first explored kernel methods with the idea of deep learning. However, unlike the previous work, our study in this paper mainly aims to address the challenge of improving the existing MKL techniques with deep learning. Specifically, we introduce a multilayer architecture for MLMKL, in which all base kernels in antecedent-layers are combined to form some inputs to other kernels in subsequent layers. We also provide an efficient alternating optimization algorithm to learn the decision function and the weight of base kernels simultaneously. To further minimize the requirement of domain knowledge to design the choices and the numbers of base kernels in each antecedent-layers, we also present an infinite base kernel learning algorithm for our proposed MLMKL framework.

The rest of this paper is organized as follows. Section 2 gives some preliminaries of multiple kernel learning and deep learning. Section 3 first presents the framework of MLMKL and then proposes a Two-Layer MKL method, followed by the development of two efficient algorithms and the analysis of their generalization performance. Section 4 discusses an extensive set of experiments for performance evaluation over a testbed with 16 publicly available benchmark data sets. Section 5 concludes this paper.

2 Preliminaries

In this Section, we introduce some preliminaries of MKL and some emerging studies on deep learning for kernel methods.

2.1 Multiple Kernel Learning

Consider a collection of n training samples $\mathbf{X}_1^n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input feature vector and y_i is the class label of $\mathbf{x}_i$. In general, the problem of conventional multiple kernel learning (MKL) can be formulated into the following optimization scheme [4]:

$$\min_{k\in\mathcal{K}} \min_{f\in\mathcal{H}_k} \quad \lambda\|f\|_{\mathcal{H}_k} + \sum_{i=1}^{n} \ell(y_i f(\mathbf{x}_i)), \quad (1)$$

where $\ell(\cdot)$ denotes some loss function, e.g. the hinge loss $\ell(t) = \max(0, 1-t)$ used for SVM, $\mathcal{H}_k$ is the reproducing kernel Hilbert space associated with kernel k, $\mathcal{K}$ denotes the optimization domain of the candidate kernels, and λ is a regularization parameter. The above optimization aims to simultaneously identify both the optimal kernel k from domain $\mathcal{K}$ and the optimal prediction function f from the reproducing kernel Hilbert space $\mathcal{H}_k$ induced by the optimal kernel k. If the optimal kernel k is given a prior, the above formulation is essentially reduced to the kernel SVM.

By the representer theorem [22], the decision function $f(\mathbf{x})$ for the above formulation is in form of a linear expansion of kernel evaluation on the training samples $\mathbf{x}_i$'s,

$$f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i k(\mathbf{x}_i, \mathbf{x}), \quad (2)$$

where β_i's are the coefficients.

In traditional MKL [18], $\mathcal{K}$ is chosen to be a set of a convex combination of predefined base kernels:

$$\mathcal{K}_{conv} = \Big\{ k(\cdot,\cdot) = \sum_{t=1}^{m} \mu_t k_t(\cdot,\cdot) : \sum_{t=1}^{m} \mu_t = 1, \mu_t \geq 0, t = 1, \ldots, m \Big\}, \quad (3)$$

where each candidate kernel k is some combination of the m base kernels $\{k_1, \ldots, k_m\}$, and μ_i is the coefficient of the ith base kernel. From (3), we can expand the decision function in (2) with the multiple kernels:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i \sum_{t=1}^{m} \mu_t k_t(\mathbf{x}_i, \mathbf{x}) = \sum_{i=1}^{n} \sum_{t=1}^{m} \beta_i \mu_t k_t(\mathbf{x}_i, \mathbf{x}), \quad (4)$$

where the final kernel is a linear combination of m base kernels.

Although it has been studied extensively, similar to SVM [27], the traditional MKL approach fall shorts in that the resulting kernel machine is "shallow" since it often adopts a simple conic combination of multiple kernels and train the classifier with the combined kernel in a "flat" architecture, which may not be powerful enough to fit diverse patterns in some real-world complicated tasks.

2.2 Deep Learning and Multilayer Kernels

Recently, a lot of machine learning studies have addressed one limitation of conventional learning techniques (such as SVM) regarding their shallow learning architectures. It has been shown that the deep architecture, such as multilayer neural nets, is often more preferable over the shallow ones. Very recently, Cho and Saul [6, 7] first introduced the idea of deep learning to kernel methods, which can be applied either in deep architectures or in shallow structures, like SVM. An l-layer kernel is the inner product after multiple feature mapping of inputs:

$$k^{(l)}(\mathbf{x}_i, \mathbf{x}_j) = \big\langle \underbrace{\Phi(\Phi(\ldots(\Phi(\mathbf{x}_i))))}_{l \text{ times}}, \underbrace{\Phi(\Phi(\ldots(\Phi(\mathbf{x}_i))))}_{l \text{ times}} \big\rangle,$$

here Φ is the underlying feature mapping function of k, $\langle \cdot, \cdot \rangle$ computes the inner product.

Specifically, we consider an example of two-layer RBF kernel. An RBF kernel is typically defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2},$$

where $\gamma > 0$ is the kernel parameter. By applying the idea of two-layer kernel with the RBF kernel, the composition yields

$$\begin{aligned} &\big\langle \Phi(\Phi(\mathbf{x}_i)), \Phi(\Phi(\mathbf{x}_j)) \big\rangle \\ &= e^{-\gamma \|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2} = e^{-2\gamma(1 - k(\mathbf{x}_i, \mathbf{x}_j))} = \kappa e^{2\gamma k(\mathbf{x}_i, \mathbf{x}_j)}, \end{aligned} \quad (5)$$

where κ is a constant that can be omitted. The similar idea can be applied for other types of kernels. In [6, 7], the authors provided a multiple layer composition approach with respect to a special family of arc-cosine kernel functions.

Remark. The work studied in [6] has some limitations. First, the proposed multi-layer kernel was applied to only a single type of kernel, typically some special kernel function, such as the arc-cosine kernel [6]. In a real application, a more desirable solution is to allow a combination of a variety of different kernels when designing the deep kernel. Second, the multi-layer kernel proposed in [6] is often "static", i.e., some fixed kernel (where degree n and level l parameters were chosen manually). No solution has been provided to optimize the kernel by learning the optimal parameters automatically. Our work was partially motivated by this work to address the above limitations.

3 Multi-Layer Multiple Kernel Learning

In this Section, we first introduce a general framework of Multi-Layer Multiple Kernel Learning (MLMKL), and then present an MLMKL paradigm, i.e., Two-Layer Multiple Kernel Learning.

3.1 Framework

Following the optimization framework of MKL, the basic idea of MLMKL is to relax the optimization domain $\mathcal{K}$ in traditional MKL optimization by adopting a family of deep kernels. Specifically, we first define a domain of l-level multi-layer kernels as follows:

$$\mathcal{K}^{(l)} = \Big\{ k^{(l)}(\cdot, \cdot) = g^{(l)}\big([k_1^{(l-1)}(\cdot, \cdot), \ldots, k_m^{(l-1)}(\cdot, \cdot)]\big) \Big\},$$

where $g^{(l)}$ is some function to combine multiple $(l-1)$-level kernels, which must ensure the resulting combination is a valid kernel. With this domain, in a way similar to regular MKL, we can formulate the optimization problem of l-level MLMKL into:

$$\min_{k \in \mathcal{K}^{(l)}} \min_{f \in \mathcal{H}_k} \quad \lambda \|f\|_{\mathcal{H}_k} + \sum_{i=1}^{n} \ell(y_i f(\mathbf{x}_i)).$$

To explain it intuitively, Figure 1 illustrates the architecture of an example three-layer MKL paradigm.

Despite sharing the similar optimization form, MLMKL is much more challenging than the conventional shallow MKL. This is because there are many unknown structures and variables, including the initialization of base kernels, the unknown combination functions $g^{(l)}$ at each level, and the final prediction model f. Apparently, it is not possible to fully optimize every aspect. In the following, we attempt to attack this challenge by considering a simplified paradigm, i.e., Two-Layer Multiple Kernel Learning (2LMKL).

3.2 Two-Layer Multiple Kernel Learning

To simplify the notations, we restrict our discussion on a Two-Layer Multiple Kernel Learning task in this Section. Our algorithm can also be extended to general multiple-layer MKL. Further, we employ an RBF kernel for the combination function $g^{(2)}$ and define the

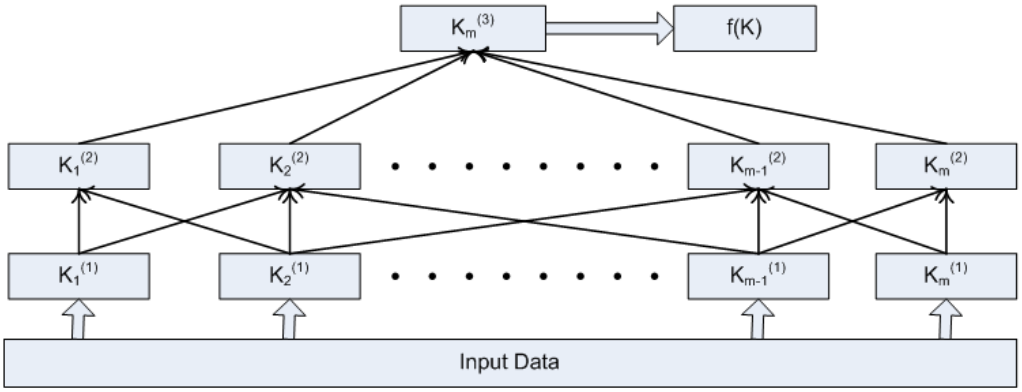

Figure 1: The architecture of the proposed deep multiple kernel learning framework. Here shows an example of three-layer MKL, and some connections are not displayed to simplify the figure.

two-layer multiple kernel domain as follows:

$$\mathcal{K}^{(2)} = \Big\{k^{(2)}(\mathbf{x}_i,\mathbf{x}_j;\boldsymbol{\mu}) = \exp\Big(\sum_{t=1}^{m}\mu_t k_t^{(1)}(\mathbf{x}_i,\mathbf{x}_j)\Big) : \boldsymbol{\mu}\in\mathbb{R}_+^m\Big\}, \quad (6)$$

where μ_t denotes the weight of t-th antecedent-layer kernel. Thus we can formulate the two-layer MKL with the kernel $\mathcal{K}^{(2)}$:

$$\min_{k\in\mathcal{K}^{(2)}}\min_{f\in\mathcal{H}_k}\frac{1}{2}\|f\|_{\mathcal{H}_k}^2+C\sum_{i=1}^{n}\max(0,1-y_i f(\mathbf{x}_i))+\sum_{t=1}^{m}\mu_t\ .$$

Note the last term is introduced as a regularization to prevent the coefficients being too large. We can further turn the above formulation into the following equivalent min-max optimization:

$$\min_{\boldsymbol{\mu}}\max_{\boldsymbol{\alpha}}\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k^{(2)}(\mathbf{x}_i,\mathbf{x}_j;\boldsymbol{\mu})+\sum_{t=1}^{m}\mu_t$$
$$\text{s.t. } 0\le\alpha_i\le C,\sum_{i=1}^{n}\alpha_i y_i=0,\mu_t\ge 0,t=1,\ldots,m, \quad (7)$$

where $\boldsymbol{\alpha}=[\alpha_1,\ldots,\alpha_n]^\top$ is the vector of dual variables and $\boldsymbol{\mu}=[\mu_1,\ldots,\mu_m]^\top$. Once solving the above optimization to find the solutions for $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$, it is straightforward to obtain the final decision function of the Two-Layer MKL machine:

$$f(\mathbf{x};\alpha,\boldsymbol{\mu})=\sum_{i=1}^{n}\alpha_i y_i k^{(2)}(\mathbf{x}_i,\mathbf{x};\boldsymbol{\mu})+b, \quad (8)$$

where the bias term b can be easily determined from KKT conditions. Due to the nonlinearity of $\exp(\cdot)$ function, we expect that the decision function in (8) can represent richer prediction tasks than that in (4).

The next challenge is how to resolve the above optimization. We consider an alternative optimization scheme. That is, (1) fix $\boldsymbol{\alpha}$ and solve $\boldsymbol{\mu}$; and (2) fix $\boldsymbol{\mu}$ and solve $\boldsymbol{\alpha}$. Specifically, let us denote by $J(\boldsymbol{\alpha},\boldsymbol{\mu})$ the following function:

$$J(\boldsymbol{\alpha},\boldsymbol{\mu})=\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k^{(2)}(\mathbf{x}_i,\mathbf{x}_j;\boldsymbol{\mu})-\sum_{i=1}^{n}\alpha_i-\sum_{i=1}^{m}\mu_i.$$

Since $k^{(2)}$ is positive semi-definite, the objective is convex over $\boldsymbol{\alpha}$. Thus it can be solved by standard QP solvers for a fixed $\boldsymbol{\mu}$. However, for any pair (i,j) of $y_i y_j=-1$, $\alpha_i\alpha_j y_i y_j k^{(2)}(\mathbf{x}_i,\mathbf{x}_j;\boldsymbol{\mu})\le 0$. Thus, J is non-convex over $\boldsymbol{\mu}$. We simply compute the d-th component of the gradient w.r.t. $\boldsymbol{\mu}$:

$$\big[\nabla J_{\boldsymbol{\mu}^{t-1}}\big]_d := \big[\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\alpha}^t,\boldsymbol{\mu}^{t-1})\big]_d$$
$$=\frac{1}{2}\sum_{ij}\alpha_i\alpha_j y_i y_j k^{(2)}(\mathbf{x}_i,\mathbf{x}_j;\boldsymbol{\mu}^{t-1})k_d^{(1)}(\mathbf{x}_i,\mathbf{x}_j)-1. \quad (9)$$

Then we update $\boldsymbol{\mu}$ by gradient ascent:

$$\boldsymbol{\mu}^t=\max(\boldsymbol{\mu}^{t-1}+\eta\nabla J_{\boldsymbol{\mu}^{t-1}},\mathbf{0}).$$

The step size η can be set by Armijo's rule such that the convergence is guaranteed. Let $\Theta=\{\theta_1,\ldots,\theta_m\}$ denote the set of hyper-parameters corresponding to base kernels $k_t^{(1)}$'s inside $k^{(2)}$ in (6). Algorithm 1 shows the detailed optimization steps of the proposed two-layer MKL algorithm with the given Θ.

3.3 Improved Two-Layer MKL Algorithm with Infinite Base Kernel Learning

Notice that, similar to traditional MKL algorithms, our proposed MLMKL algorithm also must assume a

Algorithm 1 Two-Layer MKL (2LMKL): $(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \text{TwoLayerMKLwithTheta}(\Theta_0; \mathbf{X}_1^n)$

Input: Training sample $\mathbf{X}_1^n$, initial set of base kernel parameters $\Theta_0 = \{\theta_1, \ldots, \theta_m\}$;
Output: weight vector $\boldsymbol{\mu}$ of base kernels, dual variables $\boldsymbol{\alpha}$ of SVM.

1: Randomly initialize $\boldsymbol{\mu}^0$, compute initial base kernels with Θ;
2: **repeat**
3: Compute the current kernel matrix with $\boldsymbol{\mu}^{t-1}$;
4: $\boldsymbol{\alpha}^t = \arg\min_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}, \boldsymbol{\mu}^{t-1})$ by SVM solver;
5: Compute $\nabla_{\boldsymbol{\mu}} J(\boldsymbol{\alpha}^t, \boldsymbol{\mu}^{t-1})$ by (9) as descent direction;
6: Determine the step size η^t by Armijo's rule, update $\boldsymbol{\mu}^t = \max(\boldsymbol{\mu}^{t-1} + \eta^t \nabla J_{\boldsymbol{\mu}^{t-1}}, \mathbf{0})$;
7: **until** convergence

set of predefined base kernels inside $k^{(2)}$ as in $\mathcal{K}^{(2)}$ provided beforehand. If the number of base kernels is too small, $k^{(2)}$ may not flexible enough to fit complicated patterns in a real problem. On the other hand, both the time cost and space cost of MKL/MLMKL increase with the cardinality of $\mathcal{K}_{conv}/\mathcal{K}^{(2)}$. This may be computationally inefficient when using too many base kernels. Moreover, though the proposed deep MKL architecture provides a flexibility for the design of multilayer kernels, determining an appropriate set of base kernels usually require some domain knowledge, which may be difficult for some non-expert users.

To partially address some of the above challenges, here we propose to generate the base kernels inside $k^{(2)}$ iteratively. This can be done by selecting a base kernel that optimizes the objective function in (7), which is similar to the idea of infinite kernel learning [2, 11] except that our base kernels are in the antecedent layer. Assume the inner base kernel is continuously parameterized by θ, for example, the bandwidth parameter of Gaussian kernel, or the degree of polynomial kernel. To expand the base kernel set $\mathcal{K}$, we choose a θ such that the resultant single kernel maximizes J with the current solution $\boldsymbol{\alpha}$:

$$\max_{\theta \in \mathbb{R}_+} J(\boldsymbol{\alpha}, \theta) = \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \exp(k(\mathbf{x}_i, \mathbf{x}_j; \theta)). \tag{10}$$

Again, this problem is non-convex over θ. Similar to solving $\boldsymbol{\mu}$, we compute the gradient of (10) w.r.t. θ and do gradient ascent. For example, if the inner base kernel is a Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j; \theta) = \exp(-\theta \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, the gradient can be then computed as follows:

$$\nabla J_\theta = \frac{1}{2} \sum_{i,j \in \mathbb{N}_n} \alpha_i \alpha_j y_i y_j e^{(k(\mathbf{x}_i, \mathbf{x}_j; \theta))} k(\mathbf{x}_i, \mathbf{x}_j; \theta) \|\mathbf{x}_i - \mathbf{x}_j\|^2. \tag{11}$$

After that, we employ a line search approach to determining the step size for gradient ascent. The proposed improved two-layer MKL algorithm iterates between the following two steps: (1) iteratively solve the dual variables $\boldsymbol{\alpha}$ and the kernel weight $\boldsymbol{\mu}$ as similar to the previous algorithm; (2) add a new θ to Θ by the base kernel generation method. Finally, Algorithm 2 summarizes the details of the improved two-layer multiple kernel learning algorithm, which is denoted as $\text{2LMKL}^{\text{Inf}}$ for short.

Algorithm 2 Infinite Two-Layer MKL ($\text{2LMKL}^{\text{Inf}}$): $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \Theta) = \text{TwoLayerMKL}(\Theta_0; \mathbf{X}_1^n)$

Input: Initial set of base kernel parameters Θ_0, training sample $\mathbf{X}_1^n$;
Output: Final set of base kernel parameters Θ, weight vector $\boldsymbol{\mu}$ of base kernels, dual variables $\boldsymbol{\alpha}$.

1: Initialize $\Theta = \Theta_0$;
2: **while true do**
3: $(\boldsymbol{\alpha}, \boldsymbol{\mu}) = \text{TwoLayerMKLwithTheta}(\Theta; \mathbf{X}_1^n)$;
4: $\theta = \text{NewKernel}(\boldsymbol{\alpha}; \mathbf{X}_1^n)$;
5: **if** $J(\boldsymbol{\alpha}, \theta) \leq J(\boldsymbol{\alpha}, \boldsymbol{\mu})$ **then**
6: **break**;
7: **end if**
8: $\Theta = \Theta \cup \theta$;
9: **end while**
10:
11: **Function** $\theta = \text{NewKernel}(\boldsymbol{\alpha}; \mathbf{X}_1^n)$;
12: Randomly initialize θ^0;
13: **while** $J(\boldsymbol{\alpha}, \theta^{t-1})$ is improving **do**
14: Compute the gradient ∇J_θ by the similar approach in Eqn. (11);
15: Determine a step size η^t, update $\theta^t = \theta^{t-1} + \eta^t \nabla J_\theta$;
16: **end while**

3.4 Analysis of Generalization Performance

We are aware of the trend of seeking new kernel combination methods beyond the traditional MKL. However, the kernel is the prior knowledge of the data. The construction of kernel cannot be fully "automated". When we add more flexibility to kernel learning, we are also potentially increasing the difficulty of finding the optimal kernel. It calls for theoretical analysis on these generalized kernel combination methods. We base our analysis of two-layer MKL mainly on the notion of *pseudo-dimension* of the kernel optimization domain $\mathcal{K}^{(2)}$ [26].

Theorem 1. *[26] Let* $L(f) = \mathbb{P}(f(\mathbf{x}_i) y_i \leq 0)$ *be the generalization risk of some prediction function* f *learned by solving (1), and* $L_n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(f(\mathbf{x}_i) y_i < \gamma)$ *be the empirical error. For a kernel family* $\mathcal{K}$ *with pseudo-dimension* $d_{\mathcal{K}}$*, the gen-*

Table 1: The statistics of the 16 binary-class data sets used in our experiments.

Data Set	Breast	Ionosphere	Diabetes	Waveform	Sonar	Adult	Liver	German
# instances	683	351	768	400	208	1,605	345	1,000
# dimensions	10	33	8	21	60	123	6	24
Data Set	Splice	Australian	Thyroid	Ringnorm	Heart	Banana	Titanic	FlareSolar
# instances	1,000	690	140	400	270	400	150	666
# dimensions	60	14	5	20	13	2	3	9

eralization risk of f is bounded as:

$$L(f) \leq L_n(f) + \sqrt{\tilde{O}(d_{\mathcal{K}} + 1/\gamma^2)/n},$$

where γ is the margin in loss function $l(t) = \max(0, \gamma - t)$, the $\tilde{O}$ notation hides logarithmic factors in its argument, the sample size and the allowed failure probability.

Here the pseudo-dimension $d_{\mathcal{K}}$ measures the richness / complexity of a kernel domain $\mathcal{K}$:

Definition 1. *Let $\mathcal{K}$ be a set of kernel functions mapping from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}$. We say that a set of paired examples $\mathbf{S}_n = \{(\mathbf{x}_i, \mathbf{x}'_i) \in \mathcal{X} \times \mathcal{X}, i = 1, \ldots, n\}$ is pseudo-shattered by $\mathcal{K}$ if there are real numbers $\mathbf{r} \in \mathbb{R}^n\}$ such that for any $\mathbf{b} \in \{-1, 1\}^n$ there is a function $k \in \mathcal{K}$ with property $sgn(k(\mathbf{x}_i, \mathbf{x}'_i) - r_i) = b_i$ for any $(\mathbf{x}_i, \mathbf{x}'_i) \in \mathbf{S}_n$. Then, we define a* **pseudo-dimension** *$d_{\mathcal{K}}$ of $\mathcal{K}$ to be the largest n such that there exist a $\mathbf{S}_n$ that can be pseudo-shattered by $\mathcal{K}$.*

Theorem 2. *Let $\mathcal{K}^{(1)} = \{k_1^{(1)}, \ldots, k_m^{(1)}\}$ be the inner base kernel family for the 2-layer deep kernel $\mathcal{K}^{(2)}$ defined in (6), where m is the number of base kernels. Assuming the evaluation of $k^{(1)}$ always positive, then the pseudo-dimension $d_{\mathcal{K}^{(2)}}$ is bounded by*

$$d_{\mathcal{K}^{(2)}} \leq m.$$

Proof. First, we re-write $\mathcal{K}^{(2)}$ as follows:

$$\mathcal{K}^{(2)} = \left\{ \prod \exp\left(\mu_t k_t^{(1)}\right) : k_t^{(1)} \in \mathcal{K}^{(1)} \right\}.$$

Thus each $k^{(2)} \in \mathcal{K}^{(2)}$ is the product of base kernels in form of $\exp(\mu k^{(1)})$. Consider the logarithmic operation $\ln \circ \mathcal{K}$, where for each kernel $k \in \mathcal{K}$ and any pair $(\mathbf{x}_i, \mathbf{x}_j)$, we have $(\ln \circ k)(\mathbf{x}_i, \mathbf{x}_j) = \ln k(\mathbf{x}_i, \mathbf{x}_j)$. Therefore, $\ln \circ \mathcal{K}^{(2)} = \left\{ \sum \mu_t k_t^{(1)} : k_t^{(1)} \in \mathcal{K}^{(1)} \right\}$. This is a linear space of dimension at most m (with basis $\mathcal{K}^{(1)}$ when all $k^{(1)}$ are linearly independent). According to Theorem 11.4 of [1], the Pseudo-dimension [26] of $\ln \circ \mathcal{K}^{(2)}$ is bounded by $d_{\ln \circ \mathcal{K}^{(2)}} \leq m$. We can recover $\mathcal{K}^{(2)} = \exp \circ \ln \circ \mathcal{K}^{(2)}$. Since the exponential function is monotonic, by applying Theorem 11.3 of [1], we arrive at

$$d_{\mathcal{K}^{(2)}} = d_{\exp \circ \ln \circ \mathcal{K}^{(2)}} \leq d_{\ln \circ \mathcal{K}^{(2)}} \leq m. \quad (12)$$

□

Remark: Despite the simplicity of its proof, Theorem 2 implies that the outer exponential computation of our new kernel does not increase the complexity of the kernel domain in terms of pseudo-dimension. Comparing with MKL, our MLMKL method, with more flexible or richer optimization domain, would thus have a better chance in finding the best prediction function without increasing the generalization risk explicitly.

Recently, Ying and Campbell [30] employed the *Rademacher chaos complexity* $\widehat{\mathcal{U}}_n(\mathcal{K})$ to measure the richness of $\mathcal{K}$ through its ability of fitting noisy similarity value:

$$\widehat{\mathcal{U}}_n(\mathcal{K}; \mathbf{X}_1^n) = \mathbb{E}_{\boldsymbol{\varepsilon}} \sup_{k \in \mathcal{K}} \left| \frac{1}{n} \sum_{i<j} \varepsilon_i \varepsilon_j k(\mathbf{x}_i, \mathbf{x}_j) \right| : k \in \mathcal{K}, \mathbf{x}_i \in \mathbf{X}_1^n,$$

where ε_i is a Rademacher random variable taking value ± 1 with uniform probability. We have

Corollary 1. *The Rademacher chaos complexity of $\mathcal{K}^{(2)}$ is bounded by*

$$\widehat{\mathcal{U}}_n(\mathcal{K}^{(2)}) \leq (192e + 1)\kappa^2 m, \quad (13)$$

here $\kappa = \max_{\mathbf{x}} k^{(2)}(\mathbf{x}, \mathbf{x})$, n is the size of training sample, e is natural constant.

The above corollary can be followed directly by combining Theorem 2 and the Rademacher chaos complexity result in [31, Theorem 3]. Finally, the generalization bound based on $\widehat{\mathcal{U}}_n(\mathcal{K}^{(2)})$ can be obtained immediately by combining the above Corollary with Lemma 9 of [30].

4 Experiments

4.1 Experimental Testbed and Setup

We evaluate the performance of the proposed Two-Layer MKL algorithms for binary classification tasks

Table 2: The evaluation of classification performance by comparing with a number of different algorithms. Each element in the table shows the mean and standard deviation of classification accuracy (%). The relative ranking of different MKL algorithms on each data is shown in (). The last row shows the average rank score over all data sets achieved by each algorithm.

Data Set	SVM	MKL$^{\text{level}}$	LpMKL	GMKL	IKL	MKM	2LMKL	2LMKL$^{\text{Inf}}$
Breast	96.8±1.0	96.5±0.8 (5)	96.2±0.7 (7)	**97.0±1.0** (2)	96.5±0.7 (5)	**97.1±1.0** (1)	**97.0±1.0** (2)	**96.9±0.7** (4)
Diabetes	76.7±1.8	75.8±2.5 (4)	72.6±2.5 (6)	66.4±2.5 (7)	76.0±3.0 (3)	75.8±2.5 (4)	**76.6±1.6** (1)	**76.6±1.9** (1)
Australian	84.6±1.4	85.0±1.5 (5)	84.5±1.6 (6)	80.0±2.3 (7)	85.4±1.2 (3)	85.3±0.9 (4)	**85.5±1.6** (2)	**85.7±1.6** (1)
Splice	85.0±1.4	88.4±2.4 (3)	87.1±1.6 (4)	92.4±1.4 (2)	80.0±1.5 (7)	84.6±1.5 (6)	**92.9±1.1** (1)	84.7±1.2 (5)
FlareSolar	67.5±2.0	67.6±2.0 (2)	64.8±1.8 (5)	65.3±1.8 (3)	64.8±1.8 (5)	64.4±1.7 (7)	**68.1±1.8** (1)	65.3±1.8 (3)
Titanic	78.0±3.2	77.1±2.9 (2)	77.0±3.0 (5)	76.7±3.1 (7)	76.8±2.8 (6)	77.1±3.0 (2)	77.1±3.0 (2)	**77.8±2.6** (1)
Iono	92.8±2.0	91.7±1.9 (7)	92.6±1.4 (4)	92.7±1.8 (3)	93.7±1.0 (2)	91.7±2.7 (6)	92.3±1.5 (5)	**94.4±0.9** (1)
Banana	89.7±1.5	**90.2±2.0** (1)	87.5±2.6 (4)	83.4±2.7 (6)	**90.2±1.8** (1)	80.5±5.3 (7)	86.8±2.1 (5)	**90.2±1.6** (1)
Ringnorm	98.5±0.7	98.1±0.8 (3)	96.7±1.0 (7)	97.5±1.0 (6)	**98.5±0.7** (1)	97.7±1.0 (5)	97.9±0.8 (4)	**98.5±0.8** (1)
Waveform	89.0±1.8	88.2±1.6 (6)	88.9±2.0 (4)	88.2±1.8 (6)	89.7±2.3 (3)	**90.0±1.6** (2)	88.7±1.9 (5)	**90.4±1.6** (1)
Heart	82.1±3.0	83.0±2.9 (4)	76.7±3.8 (7)	77.0±3.6 (6)	83.3±2.1 (2)	82.4±2.5 (5)	83.1±2.5 (3)	**83.6±2.1** (1)
Sonar	83.8±3.4	78.3±3.5 (7)	**84.8±3.2** (1)	78.8±4.6 (6)	81.0±5.0 (4)	83.1±3.8 (3)	79.0±4.3 (5)	**84.6±2.4** (2)
Thyroid	93.9±2.9	92.9±2.9 (6)	93.1±2.2 (5)	**94.6±2.1** (3)	**94.8±2.0** (1)	92.6±3.0 (7)	93.4±3.1 (4)	**94.8±2.2** (1)
Liver	70.5±4.1	62.3±4.5 (6)	69.4±2.9 (2)	63.6±2.6 (4)	60.0±2.9 (7)	**70.1±3.6** (1)	66.0±3.4 (3)	62.7±3.1 (5)
Adult	82.0±0.7	81.5±1.0 (4)	**82.1±0.6** (1)	75.5±1.1 (6)	75.1±1.1 (7)	81.7±0.9 (3)	79.1±2.4 (5)	81.8±1.0 (2)
German	75.2±1.9	71.4±2.8 (5)	74.3±1.4 (3)	70.4±1.6 (6)	70.0±1.5 (7)	**75.7±2.3** (1)	74.8±1.8 (2)	74.2±2.0 (4)
Rank	N/A	4.38	4.44	5.00	4.00	4.00	3.13	2.13

over a testbed of 16 publicly available data sets as shown in Table 1[1] [2].

Following the settings of previous MKL studies [29], for each data set, we create the set of base kernels $\mathcal{K}$ as follows: (1) Gaussian kernels with 10 different widths ($\{2^{-3}, 2^{-2}, \ldots, 2^{6}\}$) on all features and on each single feature; (2) polynomial kernels of degree 1 to 3 on all features and on each single feature. Each base kernel matrix is normalized to unit trace. For each data set, we randomly sample 50% of all instances as training data, and use the rest as test data. The training instances are normalized to be of zero mean and unit variance, and the test instances are also normalized using the same mean and variance of the training data. To get stable results, for each data set, we repeat each algorithm 20 times and compute the average results of the 20 runs.

For comparison, we have tried our best to compare as many state-of-the-art MKL methods as possible, which were proposed under different contexts for various applications. The goal of our experiment is mainly to examine if deep MKL is effective for improving the performance of the shallow MKL techniques. Specifically, we have compared the following algorithms:

SVM: The Support Vector Machine algorithm with a single Gaussian kernel. The band-width parameter is selected via 5-fold cross validation on the training data;

MKL$^{\text{Level}}$: The convex multiple kernel learning algorithm, that is, the target kernel class is $\mathcal{K}_{conv}$ defined in (3). We use the extended level method [29] to learn the kernel;

LpMKL: The MKL algorithm with L_p norm regularization over the kernel weight [16]. We adopt their cutting plane algorithm with second order Taylor approximation of L_p;

GMKL: The Generalized MKL algorithm in [28]. The target kernel class is the Hadamard product of single Gaussian kernel defined on each dimension;

IKL: The Infinite Kernel Learning algorithm proposed by [11]. We use LevelMKL as the embedded algorithm to solve the kernel weight $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$;

MKM: The Multilayer Kernel Machine with deep learning [6], which essentially trained SVM with multilayer arc-cosine kernel functions;

2LMKL: The proposed Two-Layer MKL algorithm described in Algorithm 1;

2LMKL$^{\text{Inf}}$: The proposed Infinite Two-Layer MKL algorithm described in Algorithm 2.

For parameter settings, the regularization parameter C in MKL or our 2LMKL algorithms is determined by 5-fold cross validation on the training data over the range of $\{10^{-2}, 10^{-1}, \ldots, 10^{2}\}$. For a fair comparison, the same set of base kernels was adopted by

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

[2] http://www.fml.tuebingen.mpg.de/Members/raetsch/benchmark

MKL^{Level}, LpMKL, and 2LMKL. For LpMKL, we examine $p = 2, 3, 4$ and report the best result. For fair comparison, in MKM we chose the layer $l = 2$, and found the best degree parameter ($\{0,1,2\}$) by cross validation. For $\text{2LMKL}^{\text{Inf}}$, the initial base kernel is a Gaussian kernel with 10 parameters and polynomial kernel with 3 degrees calculated on all the features. During the iterative process, we add one Gaussian kernel at each iteration.

4.2 Performance Analysis

Table 2 shows the detailed results of average classification accuracy and standard deviation values. To compare the overall performance, we count the ranks of the algorithms according to their performance on each data set. The average rank is included in the last row. From the results, we can draw several observations as follows.

First of all, by comparing the results between SVM and the four existing MKL methods, we found that the existing MKL algorithms do not always outperform SVM with an RBF kernel. For example, for the $\text{MKL}^{\text{Level}}$ algorithm, it outperformed SVM only over five data sets (*Australian, Splice, FlareSolar, Banana, and Heart*), and was surpassed significantly by SVM over several data sets (*German, Sonar, and Liver*, etc). Although it seems a little bit surprising, the similar observation was reported in some previous empirical study [11], which also found that regular MKL does not always outperform SVM with an RBF kernel who kernel parameter was tuned by cross validation. This observation validates our motivation for overcoming the *shallow* limitation of the regular MKL methods.

Second, by comparing the four existing MKL methods, we observed that IKL overall achieved the best performance among them, and GMKL tended to perform slightly worse than the other methods. Specifically, among all the MKL methods, IKL won three best cases, LpMKL won two best cases, while either $\text{MKL}^{\text{Level}}$ or GMKL won only one out of 16 cases. We conjecture the reason that IKL performed better is probably because IKL has the possibility of using a largely increased kernel set, which may be flexible for the classification task. Similarly, the reason that GMKL performed worse may be due to the relatively smaller kernel set, in which the base kernel set consists of only d kernels each of them was defined on a single dimension.

Third, by examining the results achieved the two proposed algorithms, 2LMKL and $\text{2LMKL}^{\text{Inf}}$, we found that both of them achieved rather impressive performance. Both of them considerably outperformed the other methods, including the existing MKL methods and the MKM method, over quite a number of data sets. Specifically, among all the MKL methods, 2LMKL won five best cases while $\text{2LMKL}^{\text{Inf}}$ won 11 best cases out of 16 cases. The encouraging performance showed our 2LMKL method is more effective than the regular MKL methods through the exploration of deep kernel learning capability, and is also more effective than the previous MKM method with deep learning.

Finally, comparing the two proposed two-layer MKL algorithms themselves, we observed that $\text{2LMKL}^{\text{Inf}}$ performed better than 2LMKL. This validates the efficacy of the proposed improvement by exploiting the idea of indefinite kernel learning.

5 Conclusion

This paper presented a general framework of multi-layer multiple kernel learning (MLMKL) to overcome the shallow learning nature of regular MKL. Under the framework, we propose a Two-Layer Multiple Kernel Learning (2LMKL) method, and developed two effective algorithms to solve it. We analyzed the generalization risk of the proposed two-layer MKL algorithms and conducted an extensive set of experiments. Our empirical results showed that the proposed 2LMKL algorithms usually perform better than the existing shallow MKL methods, demonstrating the efficacy of the 2LMKL approach.

Despite the promising results, MLMKL remains a rather new area for future research. In our future work, we plan to extend the current two-layer MKL scheme to higher-layer MKL solutions to further enhance the efficacy. Akin to the training scheme of deep learning[13], one can learn $\boldsymbol{\mu}^{(l)}$ in a bottom-up and layer-wise manner. In other words, we can embed the learned kernel $k_s^{(l)}$ (summation of base kernels at the current layer) into the base kernel functions to generate the candidate kernels $k_1^{(l+1)}, \ldots, k_n^{(l+1)}$ for the next layer. Then conventional MKL algorithms are adopted to solve the weight $\boldsymbol{\mu}^{(l+1)}$. This process can be repeated to construct deep kernels. We will also analyze the overfitting issue for MLMKL and investigate more theoretical insights about the power of multi-layer multiple kernel learning.

Acknowledgments

This research was in part supported by Singapore A* SERC Grant (102 158 0034), MOE Tier-1 Grant (RG15/08), Tier-1 Grant (RG67/07) and Tier-2 Grant (T208B2203).

References

[1] M. Anthony and P. Bartlett. *Neural Networks Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[2] A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *COLT*, pages 338–352, 2005.

[3] F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS*, pages 105–112, 2008.

[4] F. Bach, G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004.

[5] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[6] Y. Cho and L. K. Saul. Kernel methods for deep learning. In *NIPS*, pages 342–350, 2009.

[7] Y. Cho and L. K. Saul. Large-margin classification in infinite neural networks. *Neural Computation*, 22(10):2678–2697, 2010.

[8] C. Cortes, M. Mohri, and A. Rostamizadeh. Learning non-linear combinations of kernels. In *NIPS*, 2009.

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

[10] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[11] P. V. Gehler and S. Nowozin. Infinite kernel learning. In *TECHNICAL REPORT NO. TR-178,Max Planck Institute for Biological Cybernetics*, 2008.

[12] M. Gönen and E. Alpaydin. Localized multiple kernel learning. In *ICML*, pages 352–359, 2008.

[13] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[14] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[15] T. Hofmann, B. Scholkopf, and A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.

[16] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Muller, and A. Zien. Efficient and accurate l_p-norm multiple kernel learning. In *NIPS*, 2009.

[17] J. T. Kwok and I. W. Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15(6):1517–1525, 2004.

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

[19] H. Larochelle, D. Erhan, A. C. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, 2007.

[20] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel pca and denoising in feature spaces. In *NIPS*, pages 536–542, 1998.

[21] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *ICML*, pages 775–782, 2007.

[22] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *COLT/EuroCOLT*, pages 416–426, 2001.

[23] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[24] S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient multiple kernel learning algorithm. In *NIPS*, 2005.

[25] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[26] N. Srebro and S. Ben-David. Learning bounds for support vector machines with learned kernels. In *COLT*, pages 169–183, 2006.

[27] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[28] M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML*, page 134, 2009.

[29] Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. In *NIPS*, pages 1825–1832, 2008.

[30] Y. Ying and C. Campbell. Generalization bounds for learning the kernel. In *COLT*, 2009.

[31] Y. Ying and C. Campbell. Rademacher chaos complexity for learning the kernel problem. In *TECHNICAL REPORT, http://secamlocal.ex.ac.uk/people/staff/yy267/KLbound-version3.pdf*, 2010.

[32] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *NIPS*, pages 1081–1088, 2001.