Smooth Optimization for Effective Multiple Kernel Learning

Zenglin Xu
Cluster of Excellence
Saarland Univ. & MPI Inf
zlxu@mpi-inf.mpg.de

Rong Jin
Computer Sci. & Eng.
Michigan State University
rongjin@cse.msu.edu

Shenghuo Zhu
Information Analysis
NEC Labs America
zsh@sv.nec-labs.com

Michael R. Lyu
Computer Sci. & Eng.
Chinese Univ. of Hong Kong
{lyu,king}@cse.cuhk.edu.hk

Irwin King

Abstract

Multiple Kernel Learning (MKL) can be formulated as a convex-concave minmax optimization problem, whose saddle point corresponds to the optimal solution to MKL. Most MKL methods employ the L_1-norm simplex constraints on the combination weights of kernels, which therefore involves optimization of a non-smooth function of the kernel weights. These methods usually divide the optimization into two cycles: one cycle deals with the optimization on the kernel combination weights, and the other cycle updates the parameters of SVM. Despite the success of their efficiency, they tend to discard informative complementary kernels. To improve accuracy, we introduce smoothness to the optimization procedure. Furthermore, we transform the optimization into a single smooth convex optimization problem and employ the Nesterov's method to efficiently solve the optimization problem. Experiments on benchmark data sets demonstrate that the proposed algorithm clearly improves current MKL methods in a number scenarios.

Introduction

Multiple kernel learning (MKL) has been an attractive topic in machine learning (Lanckriet et al. 2004). Multiple kernel learning searches for the positive linear combination of base kernel functions/matrices that maximizes a generalized performance measure. Typical measures studied for multiple kernel learning include maximum margin classification errors (Lanckriet et al. 2004; Bach, Lanckriet, and Jordan 2004; Zien and Ong 2007), kernel-target alignment (Cristianini et al. 2001), and Fisher discriminative analysis (Ye, Chen, and Ji 2007).

Due to the success of multiple kernel learning, substantial work has devoted to derive effective and efficient optimization methods for MKL. Since the seminal work of (Lanckriet et al. 2004), which searches for the linear combination of kernel weights from a simplex using semi-definite programming (SDP), there have been two main trends in recent advances of MKL.

The first trend is to reduce computational time. In (Bach, Lanckriet, and Jordan 2004), a block-norm regularization method based on Second Order Cone Programming (SOCP) was proposed in order to solve medium-scale problems. Due to the high computation cost of SDP and SOCP, these methods cannot process more kernels and more training data. Recent advances suggest that alternating approaches (Sonnenburg et al. 2006; Rakotomamonjy et al. 2008; Xu et al. 2009) have more advantages in improving the efficiency of MKL. In these approaches, the iterative algorithm alternates between the optimization of kernel weights and the optimization of the SVM classifier. In each step, given the current solution of kernel weights, it solves a classical SVM with the combined kernel; then a specific procedure is used to update the kernel weights. In the above, most MKL methods search for the kernel parameters from a unit simplex.

The second trend is to search for the kernel combination parameters from a non-simplex space. The subspace of the simplex, which is also known as L_1-norm of the combination weights, leads to a sparse solution. This property is helpful in the case of a large number of base kernels and has good interpretability (Sonnenburg et al. 2006). However, as argued in (Kloft et al. 2008), the simplex constraint may discard complementary information when base kernels encode orthogonal information. This may therefore degrade the prediction accuracy. To improve the accuracy in this scenario, an L_2-norm of weights, which is known as a ball constraint, is introduced in their work. A general extension is the L_q-norm MKL (Kloft et al. 2009) ($q \geq 1$). Other researchers also explore the possibility of non-linear combination of kernels (Cortes, Mohri, and Rostamizadeh 2009). Although the solution space has been enlarged, it usually leads to non-convex optimization problem and incurs high computational cost. Moreover, the solution is difficult to interpret. In this paper, we focus our paper on the linear combination of kernels due to the computational issue.

From the above analysis, it is not hard to see that the simplex constraint corresponds to a non-smooth function. The non-smooth optimization may result in a non-stable solution. This is because the optimal solution of a linear program is the extreme points of the simplex. Moreover, when kernels encode orthogonal characterizations of a problem, the extreme solution may discard useful complementary information and thus result in poor generalization performance. Although replacing the simplex constraint of the kernel weights with an L_2-norm constraint may lead to a non-sparse solution (Kloft et al. 2008), it has high computa-

tional cost.

To avoid the dispute between L_1-norm and L_2-norm, we first introduce a more natural concept, the entropy of the kernel weights, as a regularizer for the kernel combination weights. This will transform the non-smooth function induced by the simplex constraint into a smooth one. We will further show that the approximated MKL problem has a Lipschitz-continuous gradient. We then introduce an efficient gradient method, which is known as Nesterov's method, for the smooth problem of MKL based on (Nesterov 2005). The resulted method has an efficiency estimate of $\mathcal{O}(\sqrt{\frac{L}{\varepsilon}})$, where L is the Lipschitz constant of the gradient of the objective function. Moreover, we compute the convergence rate of the Nesterov's method for MKL. Finally, experiments on the benchmark data sets demonstrate that the proposed algorithm clearly improves current MKL methods in a number of scenarios.

Related Work

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{n \times d}$ denote the collection of n training samples that are in a d-dimensional space. We further denote by $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \{-1, +1\}^n$ the binary class labels for the data points in $\mathbf{X}$. We employ the maximum margin classification error, an objective used in SVM, as the generalized performance measure.

We first introduce the functional framework of multiple kernel learning, which is defined from the perspective of function regularization in a Reproducing Kernel Hilbert Space (RKHS). Each kernel function h_i is associated with a unique RKHS $\mathcal{H}_i$. Consider a subspace $\mathcal{H}_i'$, such that

$$\mathcal{H}_i' = \{h | h \in \mathcal{H}_i : \frac{\|h\|_{\mathcal{H}_i}}{p_i} \leq \infty\},$$

and the inner product is defined as $\langle f, g \rangle_{\mathcal{H}_i'} = \frac{1}{p_i}\langle f, g \rangle_i$.

By defining the optimal RKHS $\mathcal{H}$ is the direct sum of the spaces $\mathcal{H}_i'$, i.e., $H = \bigoplus_{i=1}^m \mathcal{H}_i'$, then the primal MKL problem can be formulated as follows (Rakotomamonjy et al. 2008):

$$\min_{\mathbf{p} \in \mathcal{P}} \quad \varphi(\mathbf{p}), \tag{1}$$

where $\mathcal{P}$ is the domain of $\mathbf{p}$. More specifically, when the L_1-norm is employed, $\mathcal{P}$ is defined as:

$$\mathcal{P} = \{\mathbf{p} \in \mathbb{R}^m : \mathbf{p}^\top \mathbf{e} = 1,\ 0 \leq \mathbf{p} \leq 1\}. \tag{2}$$

When the L_2-norm is employed, $\mathcal{P}$ can be defined as:

$$\mathcal{P} = \{\mathbf{p} \in \mathbb{R}^m : (\mathbf{p} - \mathbf{p}_0)^\top \Sigma^{-1} (\mathbf{p} - \mathbf{p}_0) = 1,\ 0 \leq \mathbf{p} \leq 1\},$$

where $\mathbf{p}_0$ and Σ^{-1} are the already known structure parameters.

When SVM is used as the performance criterion, the MKL objective $\varphi(\mathbf{p})$ is a function defined as

$$\begin{aligned} \min_{\{h_i\}_{i=1}^m, b, \xi} \quad & \frac{1}{2}\sum_{i=1}^m \frac{1}{p_i}\|h_i\|_{\mathcal{H}_i}^2 + C\sum_{j=1}^n \xi_j \\ \text{s. t.} \quad & y_j \sum_{i=1}^m h_i(\mathbf{x}_j) + b y_j \geq 1 - \xi_j,\ \xi_j \geq 0,\ \forall j, \end{aligned} \tag{3}$$

where C is the trade-off parameter in SVM.

It is not hard to show that the function $\varphi(\mathbf{p})$ can also be calculated as:

$$\varphi(\mathbf{p}) = \max_{\alpha \in \mathcal{Q}}\ \alpha^\top \mathbf{e} - \frac{1}{2}(\alpha \circ \mathbf{y})^\top \left(\sum_{i=1}^m p_i \mathbf{K}_i\right)(\alpha \circ \mathbf{y}), \tag{4}$$

where $\mathcal{Q}$ is the domain of α:

$$\mathcal{Q} = \{\alpha \in \mathbb{R}^n : \alpha^\top \mathbf{y} = 0,\ 0 \leq \alpha \leq C\}. \tag{5}$$

Here, $\mathbf{e}$ is a vector of all ones, $\{\mathbf{K}_i\}_{i=1}^m$ is a group of base kernel matrices associated with $\mathcal{H}_i'$, and $\circ$ defines the element-wise product between two vectors. It is easy to verify that the overall optimization problem $\min_{\mathbf{p}} \max_{\alpha} \varphi(\mathbf{p}, \alpha)$ is convex on $\mathbf{p}$ and concave on α. Thus the above optimization problem can be regarded as a convex-concave problem.

Recently, advanced optimization methods, such as Semi-infinite Linear Programming (SILP) (Sonnenburg et al. 2006), Subgradient Descent (SD) (Rakotomamonjy et al. 2008), and Level method (Xu et al. 2009), have been applied to handle large-scale MKL problems. All of the above methods solve MKL via a way of alternating optimization. We summarize them into a unified framework in Algorithm 1. Note that a superscript is used to indicate the index of iteration, a convention that is used throughout this paper. We use $[x]^t$ to denote x to the power of t in the case of ambiguity.

Algorithm 1 Alternating approach for solving MKL

1: Initialize a valid solution $\mathbf{p}^0 \in \mathcal{P}$ and $i = 0$
2: **repeat**
3: Solve the dual of SVM with kernel $\mathbf{K} = \sum_{j=1}^m p_j^i \mathbf{K}_j$ and obtain optimal solution α^i
4: Update kernel weights by minimizing $\varphi(\mathbf{p})$
5: Update $i = i + 1$ and calculate the stopping criterion Δ^i
6: **until** $\Delta^i \leq \varepsilon$

As indicated in Algorithm 1, these methods divide the MKL problem into two cycles: the inner cycle solves a standard SVM problem to update α, and the outer cycle updates the kernel weight vector $\mathbf{p}$. Another commonality is that they all use the gradient to update the new solution. We denote $\nabla\varphi(\mathbf{p}^i)$ as the subgradient of $\varphi(\mathbf{p})$ with respect to $\mathbf{p}$ at $(\mathbf{p}^i, \alpha^i)$, which is calculated as:

$$\nabla\varphi(\mathbf{p}^i) = (t_1, \ldots, t_m), \tag{6}$$

where $t_j = -\frac{1}{2}(\alpha^i \circ \mathbf{y})^\top \mathbf{K}_j (\alpha^i \circ \mathbf{y})$, for $j = 1, \ldots, m$.

On the other hand, these methods differ in the 4-th step in Algorithm 1: the SILP method updates $\mathbf{p}$ by solving a cutting plane model, the SD method updates $\mathbf{p}$ using the subgradient of the current solution, while the level method updates $\mathbf{p}$ by projecting the solution of the cutting plane model to a level set. More specifically, in the i-th iteration, the optimum values of $\varphi^i(\mathbf{p})$ for SILP, SD and Level methods are calculated as follows:

$$\begin{aligned} \varphi_{SILP}^{i*} &= \min_{\nu, \mathbf{p} \in \mathcal{P}} \{\nu : \nu \geq g^i(\mathbf{p})\}, \\ \varphi_{SD}^{i*} &= \min_{\mathbf{p} \in \mathcal{P}} \frac{1}{2}\|\mathbf{p} - \mathbf{p}^i\|_2^2 + \gamma_i (\mathbf{p} - \mathbf{p}^i)^\top \nabla_{\mathbf{p}} \varphi(\mathbf{p}^i), \\ \varphi_{Level}^{i*} &= \min_{\mathbf{p} \in \mathcal{L}^i} \frac{1}{2}\|\mathbf{p} - \mathbf{p}^i\|_2^2, \end{aligned}$$

where γ_i in the SD method is the step size that needs to be decided dynamically (e.g., by a line search). The cutting plane model used in SILP is defined as

$$g^i(\mathbf{p}) = \max_{1\le j\le i} f(\mathbf{p}^j, \alpha^j) + (\mathbf{p}-\mathbf{p}^j)^\top \nabla_{\mathbf{p}}\varphi(\mathbf{p}^j).$$

In the level method, $\mathcal{L}^i$ denotes the level set and is computed as

$$\mathcal{L}^i \quad = \quad \{\mathbf{p}\in\mathcal{P} : g^i(\mathbf{p}) \le \ell^i = \lambda\overline{\varphi}^i + (1-\lambda)\underline{\varphi}^i\},$$

where the lower bound $\underline{\varphi}^i$, and the upper bound $\overline{\varphi}^i$ are computed as follows:

$$\underline{\varphi}^i = \min_{\mathbf{p}\in\mathcal{P}} g^i(\mathbf{p}),\ \overline{\varphi}^i = \min_{1\le j\le i} \varphi(\mathbf{p}^j).$$

These alternating methods depend greatly on the procedures of updating the kernel combination weights. Due to the simplex constraint of the kernel weights, these methods may discard kernels with complementary information. In order to overcome the ineffectiveness of L_1-MKL when dealing with complementary base kernels, and to overcome the inefficiency of L_2-MKL when dealing with a large number of base kernels, in this paper, we propose to convert the non-smooth optimization problem hidden in L_1-MKL into a smooth one and then employ Nesterov's method to efficiently solve it.

Smooth Optimization for Multiple Kernel Learning

In this section, we discuss how to derive a smooth optimization problem and how to solve it. For the limit of space, we omit all the proofs to theorems.

Smooth Multiple Kernel Learning

For convenience, we first reformulate the optimization problem $\min_{\mathbf{p}\in\mathcal{P}}\varphi(\mathbf{p})$ described in (4) as the following:

$$\min_{\mathbf{p}\in\mathcal{P}}\ \max_{\alpha\in\mathcal{Q}'}\quad \alpha^\top e - \frac{1}{2}\sum_{i=1}^m p_i\alpha^\top\mathbf{G}_i\alpha \tag{7}$$

where

$$\begin{aligned}\mathcal{Q}' &= \{\alpha\in\mathbb{R}^n : 0\le\alpha\le C\}.\\ \mathbf{G}_i &= \mathrm{diag}(\mathbf{y})\mathbf{K}_i\mathrm{diag}(\mathbf{y}),\ i=1,\ldots,m.\end{aligned}$$

For the domain of α, it is interesting to note that we eliminate the bias term and therefore do not have the constraint $\alpha^\top y = 0$. We can introduce an additional constant attribute into each example to account for the bias term. In particular, we can set the additional constant attribute to be a very large value so that the corresponding weight is small and therefore can be ignored.

To transform the minmax problem in (7) into a smooth function, which is a prerequisite of Nesterov's method, we rewrite the problem by switching min and max operators and define $f(\alpha)$ as:

$$f(\alpha)\ \equiv \max_{\mathbf{p}\in\mathcal{P}}\ -\alpha^\top e + \frac{1}{2}\sum_{i=1}^m p_i\alpha^\top\mathbf{G}_i\alpha, \tag{8}$$

where $\alpha\in\mathcal{Q}'$.

One of the main problems with solving the above problem directly is that the function $f(\alpha)$ in (8) is a not smooth. Therefore, the best convergence rate is $O(1/\varepsilon^2)$ (Nesterov 2005) in the worst case analysis, where ε is the error rate. Thus, the key to improve the efficiency of the optimization problem in (8) is to approximate $f(\alpha)$ by a smooth function. To this end, we introduce the entropy $H(\mathbf{p})$ as the regularizer, i.e.,

$$H(\mathbf{p}) = -\sum_{i=1}^m p_i\ln p_i.$$

In optimization literature, $H(\mathbf{p})$ is also called entropy prox-function. $H(p_i) = 0$ if and only if $p_i = 0$ or $p_i = 1$. Therefore, by introducing the entropy regularizer, the extreme solutions will be penalized.

We further introduce an approximation function $f_\lambda(\alpha)$, which is defined as follows:

$$\max_{p\in\mathcal{P}'} -\alpha^\top e + \frac{1}{2}\sum_{i=1}^m p_i\alpha^\top\mathbf{G}_i\alpha - \frac{\lambda}{2}\sum_{i=1}^m p_i\ln p_i, \tag{9}$$

where λ is a smoothing constant whose value will be decided later. We define $\mathcal{P}' = \{\mathbf{p}\in\mathbb{R}^m : 0\le\mathbf{p}\le 1\}$.

In the following, we will prove that the regularizer $H(p)$ transforms the original piecewise linear function maximizing $\mathbf{p}$ as described in (8) to be a smooth $C^{1,1}$ function.

To do this, we first derive the dual form of (9), which is stated in Lemma 1.

Lemma 1. *The optimization problem in (9) achieves the same optimal value as that of the following optimization problem:*

$$f_\lambda(\alpha) = -\alpha^\top e + \frac{\lambda}{2}\ln\left(\sum_{i=1}^m \exp(\frac{1}{\lambda}\alpha^\top\mathbf{G}_i\alpha - 1)\right). \tag{10}$$

The following theorem shows that $f_\lambda(\alpha)$ universally upper bounds $f(\alpha)$.

Theorem 1. *$f_\lambda(\alpha)$ universally upper bounds $f(\alpha)$ by a factor of $\frac{\lambda}{2}(\ln m - 1)$, i.e.,*

$$f_\lambda(\alpha) - f(\alpha) \le \frac{\lambda}{2}(\ln m - 1). \tag{11}$$

In the following, we will show that the above optimization problem is a convex and smooth optimization problem. The results are proved in Corollary 2 and Corollary 3, respectively.

Corollary 2. *The problem $\min_{\alpha\in\mathcal{Q}'} f_\lambda(\alpha)$ is a convex optimization problem.*

In order to verify whether $f_\lambda(\alpha)$ is a smooth function, we first calculate the gradient of $f_\lambda(\alpha)$ as follows:

$$\nabla f_\lambda(\alpha) \quad = \quad -\mathbf{e} + \frac{1}{2}\sum_{i=1}^m \theta_i\mathbf{G}_i\alpha, \tag{12}$$

where

$$\theta_i = \frac{\exp(\frac{1}{\lambda}\alpha^\top\mathbf{G}_i\alpha - 1)}{\sum_{j=1}^m \exp(\frac{1}{\lambda}\alpha^\top\mathbf{G}_i\alpha - 1)}. \tag{13}$$

Then the key to prove the smoothness is to show that $\nabla f_\lambda(\alpha)$ is Lipschitz continuous. Corollary 3 validates that $f_\lambda(\alpha)$ is a smooth function.

Corollary 3. *The optimization function of $f_\lambda(\alpha)$ is a smooth $C_L^{1,1}$ function, i.e., the gradient function $\nabla f_\lambda(\alpha)$ is Lipschitz continuous. The Lipschitz constant is bounded by*

$$L = a_g + \frac{nC^2 b_g}{\lambda},$$

where

$$a_g = \frac{1}{2} \max_{1\le i\le m, 1\le j\le n} [\mathbf{G}_i]_{j,j}, \quad b_g = \frac{1}{2} \max_{1\le i\le m} [\Lambda_{\max}(\mathbf{G}_i)]^2.$$

In the above, $\Lambda_{\max}(\mathbf{M})$ denotes the maximum eigenvalue of matrix $\mathbf{M}$.

According to (Nesterov 2005), a continuous function with Lipschitz-continuous gradient can have an efficiency estimate of the order $\mathcal{O}(\sqrt{\frac{L}{\varepsilon}})$, which is more efficient than methods based on black-box-oracle models. In this paper, we employ the Nesterov's method to solve the approximated smooth function of multiple kernel learning.

Nesterov's Method for Smooth MKL

The Nesterov's method is used to approximate the original non-smooth function with a smooth function with a Lipschitz-continuous gradient (Nesterov 2005), and then employ gradient-based methods to iteratively update the solution. In this section, we present how to employ the Nesterov's method to solve the smooth problem of multiple kernel learning.

First, given the current solution α^k to (10), we first calculate the gradient based on (12). Then we calculate an auxiliary solution β^k by solving the following optimization problem

$$\beta^k = \arg\min_{\beta\in\mathcal{Q}'} (\beta - \alpha^k)^\top \nabla f_\lambda(\alpha^k) + \frac{L}{2}\|\beta - \alpha^k\|_2^2, \quad (14)$$

where the first term is to decrease the objective value of $f_\lambda(\alpha)$ along the gradient and the second term is a regularization term.

Minimizing (14), we can obtain an analytical solution as follows

$$\beta_i^k = T\left(\alpha_i^k - \frac{[\nabla f_\lambda(\alpha^k)]_i}{L}\right), \quad (15)$$

where the operator $T(u)$ is defined as

$$T(u) = \begin{cases} 0 & u < 0 \\ C & u > C \\ u & \text{other wise} \end{cases}.$$

In (14), we can obtain a series of approximated solution, denoted by $\{\beta^k\}_{k=0}^\infty$, for MKL using gradient descent.

It is required that $\{\beta^k\}_{k=0}^\infty$ satisfy the following condition:

$$A_k f(\beta_k) \le \min_{\alpha\in\mathcal{Q}'} \sum_{k=0}^{\infty} \eta_i (f_\lambda(\alpha^i) + \langle \nabla f_\lambda(\alpha^i), \alpha - \alpha^i\rangle) + \frac{L}{\sigma} d(\alpha), \quad (16)$$

where $d(\cdot)$ is the distance function in $\mathcal{Q}'$ and σ is a constant. $A_k = \sum_{i=0}^k \eta_i$ and $\{\eta_i\}_{k=0}^\infty$ are step-size parameters. We denote the solution of Eq. (16) as γ^k and obtain $\{\gamma^k\}_{k=0}^\infty$.

By setting $\eta_i = \frac{i+1}{2}$, we can obtain

$$\gamma^k = \arg\min_{\alpha\in\mathcal{Q}'} \sum_{j=0}^{k} \frac{j+1}{2} [f_\lambda(\alpha^j) + (\alpha - \alpha^j)^\top \nabla f_\lambda(\alpha^j)] + \frac{L}{\sigma} d(\alpha). \quad (17)$$

In the above, the prox-function $d(\alpha)$ is defined as $d(\alpha) = \frac{1}{2}\|\alpha\|_2^2$. Hence, to ensure the strong convexity of $d(\alpha)$, we set $\sigma = 1$.

The analytical solution of (17) can be described as follows:

$$\gamma_i^k = T\left(-\frac{1}{L}\sum_{j=0}^{k} \frac{j+1}{2} [\nabla f_\lambda(\alpha^j)]_i\right), \ i = 1, \ldots, n. \quad (18)$$

Then the solution of iteration k will be computed as

$$\alpha^{k+1} = \tau_k \gamma^k + (1 - \tau_k)\beta^k, \quad (19)$$

where $\tau_k = \frac{2}{k+3}$.

Finally, we describe Nesterov's method for MKL in Algorithm 2.

Algorithm 2 Nesterov's Method for Smooth MKL

1: Initialize the number of iterations N
2: Initialize α^0 to be the optimal solution to a SVM problem with the kernel matrix $\mathbf{K} = \frac{1}{m}\sum_{i=1}^m \mathbf{K}_i$
3: Compute λ and L
4: **for** $k = 0, \ldots, N$ **do**
5: Update β^k according to (15)
6: Update γ^k according to (18)
7: Update α^k according to (19)
8: **end for**

We then discuss the convergence rate of Algorithm 2.

Theorem 4. *The convergence rate of the above algorithm for the optimization of $f_\lambda(\alpha)$ is in the order of $\mathcal{O}(1/N^2)$ where N is the number of steps. In particular,*

$$f_\lambda(\beta^N) - f_\lambda(\alpha_\lambda^*) \le 2\frac{L\|\alpha_\lambda^*\|_2^2}{(N+1)^2}$$

where α_λ^* is the optimal solution of the problem (10), i.e., $\alpha_\lambda^* = \arg\min_\alpha f_\lambda(\alpha)$.

Corollary 5 describes the convergence rate of the algorithm for the minimization of $f(\alpha)$.

Corollary 5. *The convergence rate for the minimization of the original function $f(\alpha)$ is bounded by*

$$f(\alpha^N) - f(\alpha^*) \le \frac{na_g}{2(N+1)^2} + \frac{nC^2\sqrt{b_g(\ln m - 1)}}{N+1}. \quad (20)$$

Finally, we recover the kernel weights according to (13). Then based on the new kernel weights, we could train an SVM classifier for predicting.

Experiment

We conduct experiments to evaluate the effectiveness and efficiency of the proposed algorithm in contrast with two state-of-the-art MKL algorithms, the L_1-MKL (Rakotomamonjy et al. 2008) and L_2-MKL (Kloft et al. 2008). We denote the proposed algorithm by SMKL.

To better understand the properties of the proposed algorithm, we first design a toy data set, where no features are redundant. Then we evaluate the algorithms on seven UCI data sets and five text data sets. We summarize the data sets used in this paper in Table 1. Among the text data, *text1*, *text2*, and *text3* are three text data sets are extracted from the *Ohsumed*[1], representing three challenging binary classification problems, i.e., *C1 vs C2*, *C5 vs C6*, and *C12 vs C13*, respectively. *text4* and and *text5* denote the classification of two news groups of the 20-*Newsgroups*[2] repository, *auto vs motor* and *baseball vs hockey*, respectively.

Table 1: Data sets used in the experiments, where d and n represents data dimensionality and cardinality, respectively.

Data set	d	n	**Data set**	d	n
iono	33	351	*breast*	9	300
sonar	60	208	*pima*	8	768
heart	13	270	*wdbc*	13	569
wpbc	33	198	*text1*	1686	581
text2	2302	871	*text3*	2059	772
text4	5341	2000	*text5*	6311	2000

All experiments are conducted on a PC with 3.2GHz CPU and 2GB memory. For each selected data set, we repeat all the algorithms 20 times. We normalize the training data with zero mean and unit variance, and normalize the test data according to the mean and variance of the training data. The regularization parameter C in SVM is selected using cross validation. We stop the algorithms when the duality gap is less than 0.01 or the number of iterations larger than 500. For the comparison algorithms, we employ the SimpleMKL (Rakotomamonjy et al. 2008) toolbox to implement L_1-MKL and L_2-MKL, respectively.

Experiment on Toy Data

We produce a toy data set with 500 data points and 40 dimensions. We set the label according to the mean of the first 34 dimensions and set the left 6 dimensions as irrelevant. We randomly select 10% from the data to form the training data. Then we employ the following rule to form the base kernels: for each base kernel $\mathbf{K}_i = \mathbf{v}_i\mathbf{v}_i^\top$ for $i = 1, \ldots, d$, where $\mathbf{v}_i$ is the i-th feature of data $\mathbf{X} \in \mathbb{R}^{n\times d}$. Thus the final kernel matrix is formulated as $\mathbf{K} = \sum_{i=1}^{d} p_i\mathbf{v}_i\mathbf{v}_i^\top$. It is believed that the base kernels are complementary to each other. This setting can be used to learn a relational kernel for text mining (Cortes, Haffner, and Mohri 2004) and weighted spectrum kernel for splice site detection in bioinformatics (Sonnenburg et al. 2006).

[1] `ftp://medir.ohsu.edu/pub/ohsumed`

[2] `http://people.csail.mit.edu/jrennie/20Newsgroups/`

We then remove one irrelevant feature from the training data one by one until no irrelevant features are left. We call the ratio of the number of relevant features to the total number as the complementary information ratio ρ. Then for each ρ_i ($i = 1, \ldots, 7$), we show the corresponding prediction results of three MKL algorithms in Fig. 1. It is observed that with a high ratio, the Nesterov's method could significantly improve the prediction accuracy. This clearly indicates that smooth optimization of MKL will benefit the learning problem when most base kernels are complementary.

It is also interesting to discuss the running time of these algorithms. Among these three algorithms, the L_2-MKL requires more running time than the other two algorithms. Its time cost is almost hundred times of the other algorithms. This therefore limits its application in large scale multiple kernel learning algorithms. When comparing SMKL with L_1 MKL, SMKL costs a little bit more time than the L_1-MKL but still in a similar scale.

Experiment on Real-world Datasets

We then evaluate the algorithms on the UCI data sets and the text data sets. According to our design of base kernels which will be described below, we will separate the experiment into two parts, one on UCI datasets and the other on text datasets.

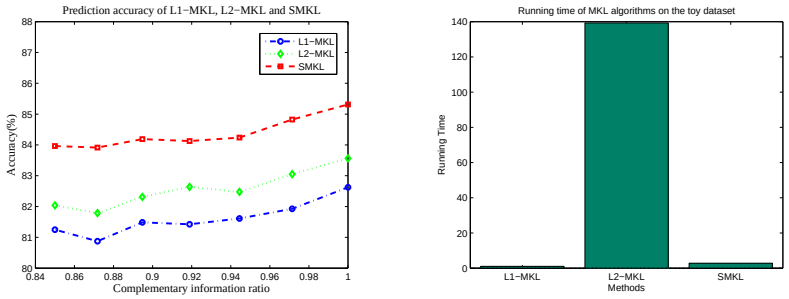
For the experiment on the UCI data sets, we adopt the similar experimental setting as the toy data. We randomly select 20% from the data to form the training data. We adopt the following setting to design the base kernels:

- Gaussian kernels with 10 different widths ($\{2^{-3}, 2^{-2}, \ldots, 2^{6}\}$) on each single feature
- Polynomial kernels of degree 1 to 3 on each single feature.

In this way, most base kernels are complementary to each other.

We show the classification results in Table 2. It is observed that SMKL has obtained significantly better classification accuracy than the other two methods. This is because that the base kernels composed by single features may be complementary to each other. And the entropy regularizer in SMKL better captures this complementary information. Moreover, L_1-MKL achieves the worst performance due to the simplex constraint in L_1-MKL, where only a few dominant features are selected and some useful features are discarded during the optimization. For example, on average, only 26 over 429 kernels are selected for the *iono* data. Although L_2-MKL can also obtain non-sparse solution, its result is worse than SMKL. This shows that the ball constraint in L_2-MKL is not an appropriate regularizer compared with the entropy regularizer in SMKL. Moreover, L_2-MKL is very time consuming for all the datasets. In summary, when base kernels are complementary, the smooth optimization of multiple kernel learning can improve the predication accuracy over other MKL algorithms.

For the experiment of text datasets, we employ MKL algorithms to optimize the weights of relational kernel (Cortes, Haffner, and Mohri 2004). For the selected data sets, 10% of the data are used for training and the rest for testing. Other settings are similar to the above experiments.



(a) *Accuracy* (b) *Running time*

Figure 1: Comparison of three MKL algorithms on the toy data. (a) The prediction accuracy comparison when varying the number of relevant features. (b) The running time comparison for three algorithms when using all relevant features.

Table 2: The classification accuracy (%) and std values of MKL algorithms when the base kernels are constructed on single features. The base kernels can be regarded as complementary kernels.

Data Set	L_1-MKL	L_2-MKL	SMKL
iono	87.1±2.0	89.0±2.1	**89.3**±2.6
breast	95.4±0.6	95.8±0.7	**96.3**±0.6
sonar	73.6±4.2	75.6±4.6	**77.2**± 4.3
pima	69.0±2.2	70.0±1.6	**71.6**±1.2
wdbc	93.4±1.2	94.1±1.2	**94.5**±1.1
heart	77.3±3.5	78.7±2.6	**78.9**± 2.5
wpbc	72.6±3.2	75.0±2.9	**75.1**± 2.7

We show the results in Table 3. It is observed that the advantage of the proposed algorithm is much more clear for text categorization. For example, the improvement of SMKL over L_1-MKL is over 7% on *text1*. Its success can be explained by the fact that the features/terms in text are complementary to each other in semantics.

Table 3: The classification accuracy (%) and std values of MKL algorithms on text data.

Data Set	L_1-MKL	SMKL
text1	72.7±2.0	**77.3**±1.6
text2	79.5±1.6	**81.0**±1.8
text3	78.1±2.0	**81.5**±2.2
text4	85.1±2.7	**87.6**±2.3
text5	98.0±1.2	**98.8**±0.3

Conclusion

In this paper, we have presented a smooth optimization framework for multiple kernel learning. The original non-smooth function of MKL based on the simplex constraint is first approximated by a function with an entropy regularizer, which is Lipschitz-continuous and forms a universal upper bound to the original function. We then employ a gradient-based method, known as Nesterov's method, to efficiently solve the smooth problem of MKL. Moreover, we also prove the convergence rate of the derived algorithm. The experiments on several benchmark data sets have shown the effectiveness of the proposed smooth optimization scheme for multiple kernel learning compared with both L_1-MKL and L_2-MKL .

Acknowledgment

The work was supported by the US National Science Foundation (IIS-0643494), US National Institute of Health (1R01GM079688-01), Research Grants Council of Hong Kong (CUHK4158/08E and CUHK4128/08E), and MSRA (FY09-RES-OPP-103).

References

Bach, F. R.; Lanckriet, G. R. G.; and Jordan, M. I. 2004. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proc. of Int. conf. on Mach. Learn.*, 41–48.

Cortes, C.; Haffner, P.; and Mohri, M. 2004. Rational kernels: Theory and algorithms. *J. Mach. Learn. Res.* 5:1035–1062.

Cortes, C.; Mohri, M.; and Rostamizadeh, A. 2009. Learning non-linear combinations of kernels. In *Advances in Neural Information Processing Systems 22*, 396–404.

Cristianini, N.; Shawe-Taylor, J.; Elisseeff, A.; and Kandola, J. S. 2001. On kernel-target alignment. In *Advances in Neural Information Processing Systems 13*, 367–373.

Kloft, M.; Brefeld, U.; Laskov, P.; and Sonnenburg, S. 2008. Non-sparse multiple kernel learningg. In *NIPS workshop on Kernel Learning: Automatic Selection of Optimal Kernels*.

Kloft, M.; Brefeld, U.; Sonnenburg, S.; Laskov, P.; Müller, K.-R.; and Zien, A. 2009. Efficient and accurate lp-norm multiple kernel learning. In *Advances in Neural Information Processing Systems 22 (NIPS)*.

Lanckriet, G. R. G.; Cristianini, N.; Bartlett, P.; Ghaoui, L. E.; and Jordan, M. I. 2004. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.* 5:27–72.

Nesterov, Y. 2005. Smooth minimization of non-smooth functions. *Math. Program.* 103(1):127–152.

Rakotomamonjy, A.; Bach, F. R.; Canu, S.; and Grandvalet, Y. 2008. SimpleMKL. *J. Mach. Learn. Res.* 9:1179–1225.

Sonnenburg, S.; Rätsch, G.; Schäfer, C.; and Schölkopf, B. 2006. Large scale multiple kernel learning. *J. Mach. Learn. Res.* 7:1531–1565.

Xu, Z.; Jin, R.; King, I.; and Lyu, M. 2009. An extended level method for efficient multiple kernel learning. In *Advances in Neural Information Processing Systems 21*, 1825–1832.

Ye, J.; Chen, J.; and Ji, S. 2007. Discriminant kernel and regularization parameter learning via semidefinite programming. In *Proc. of Int. conf. on Mach. Learn.*, 1095–1102.

Zien, A., and Ong, C. S. 2007. Multiclass multiple kernel learning. In *Proc. of Int. conf. on Mach. Learn.*, 1191–1198.