MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

SpicyMKL

Taiji SUZUKI and Ryota TOMIOKA

METR 2009–45 September 2009

SpicyMKL

Taiji SUZUKI and Ryota TOMIOKA

Department of Mathematical Informatics
Graduate School of Information Science and Technology
The University of Tokyo
{t-suzuki,tomioka}@mist.i.u-tokyo.ac.jp

September 2009

Abstract

We propose a new optimization algorithm for Multiple Kernel Learning (MKL) with general convex loss functions. The proposed algorithm is a proximal minimization method that utilizes the "smoothed" dual objective function and converges super-linearly. The sparsity of the intermediate solution plays a crucial role for the efficiency of the proposed algorithm. Consequently our algorithm scales well with increasing number of kernels. Experimental results show that our algorithm is favorable against existing methods especially when the number of kernels is large (> 1000).

1 Introduction

Kernel methods are powerful nonparametric methods in machine learning and data analysis. Typically a kernel method fits a decision function that lies in some Reproducing Kernel Hilbert Space (RKHS). In such a learning framework, the choice of a kernel has a strong impact on the performance of a method. Instead of using a single kernel, *Multiple Kernel Learning (MKL)* aims to find an optimal combination of multiple kernels. In fact, it has been reported [11] that using multiple kernel improves performances in learning tasks that involves multiple and heterogeneous data sources. More specifically MKL fits a decision function of the form of $f(x) = \sum_{m=1}^{M} f_m(x) + b$ where each f_m belongs to different RKHSs $\mathcal{H}_m$ ($m = 1, \dots, M$) corresponding to different basis kernels k_m. Each basis kernel k_m may be constructed on different feature subsets of input x, or different kernel types (e.g., Gaussian, polynomial) with different parameter values (e.g., Gaussian width, polynomial order), or even may rely on different heterogeneous data sources associated with the same learning problem. This provides considerable flexibility to fit various types of problems. According to recent formulations [1, 17, 13], MKL selects the decision function as the minimizer of the following optimization problem:

$$\underset{f_m \in \mathcal{H}_m, b \in \mathbb{R}}{\text{minimize}} \quad \sum_{i=1}^{N} \ell\left(y_i, \sum_{m=1}^{M} f_m(x_i) + b\right) + C \sum_{m=1}^{M} \|f_m\|_{\mathcal{H}_m} \tag{1}$$

where $\{x_i, y_i\}_{i=1}^{N}$ are labeled training examples, $\ell(\cdot, \cdot)$ is a convex loss function (e.g., hinge, logistic, squared loss) and $\| \cdot \|_{\mathcal{H}_m}$ is the norm in the RKHS $\mathcal{H}_m$[1]. A nice property of the above formulation is that the solution becomes *sparse* due to the mixed norm penalization

[1]Note that in the literature [1, 17, 13], a different but an equivalent regularization term has been considered. See Sec. 4 for details.

$\sum_m \|f_m\|_{\mathcal{H}_m}$ similarly to *group Lasso* [22, 2]. Thus we can select kernels in a convex optimization problem. The resulting decision function is nicely interpretable because only a small number of f_ms are used. However, solving MKL is challenging because the objective function is non-differentiable due to the non-smoothness of the regularization term. To overcome this difficulty, several methods have been proposed.

Roughly speaking, two types of methods have been proposed so far. The first are constraints-based methods [11, 1] that cast the problem as constrained convex optimization problems. [11] formulated MKL as semi-definite programming (SDP) problem. [1] casted the problem as a second order conic programming (SOCP) problem and proposed an SMO-like algorithm to deal with medium-scale problems. The second are upper-bound-based methods [17, 13, 5]. These methods upper-bound the objective function by a smooth function with some auxiliary variables; they iteratively (a) solves a single kernel learning problem, such as SVM, and (b) updates the auxiliary variable. A nice property of this type of methods is that it can make use of existing well-tuned solvers for the single kernel problem. Semi-Infinite Linear Program (SILP) approach proposed by [17] utilizes a cutting plane method for the update of the auxiliary variable. SimpleMKL proposed by [13] performs a gradient descent on the auxiliary variables. It was reported that SimpleMKL converges faster than former methods. [20] proposed a novel Level Method as an improvement of SILP and SimpleMKL. HessianMKL proposed by [5] replaced the gradient descent update of SimpleMKL with a Newton update. At each iteration, HessianMKL solves a Quadratic Programming (QP) problem with the size of the number of kernels to obtain the Newton update direction. Therefore it is efficient when the number of kernels is small.

In this article, we propose a new efficient MKL algorithm, which we call SpicyMKL. The proposed method computes descent steps through the optimization of a smoothed dual objective function, which arises from a *proximal minimization* [15] in the primal. From the general theory of proximal minimization method, the proposed method converges *super-linearly*. The primal variable is *sparse* at each iteration due to the so-called *soft threshold* operation [8, 7, 6, 21]; this sparsity is effectively exploited in the proposed algorithm. Therefore SpicyMKL scales well with increasing number of kernels. Numerical experiments show that we are able to train a classifier with 3000 kernels in less than 10 seconds.

2 Framework of MKL

In the MKL problem, we assume that we are given n samples $(x_i, y_i)_{i=1}^N$ where x_i belongs to an input space $\mathcal{X}$ and y_i belongs to an output space $\mathcal{Y}$ (usual settings are $\mathcal{Y} = \{\pm 1\}$ for classifications and $\mathcal{Y} = \mathbb{R}$ for regressions). We define the gram matrix with respect to the kernel function k_m as $K_m = (k_m(x_i, x_j))_{i,j}$. We assume the gram matrix K_m is positive definite[2]. The inner product induced by a positive definite matrix $K \in \mathbb{R}^{N\times N}$ is written as $\langle \alpha, \beta\rangle_K := \alpha^\top K \beta$ for $\alpha, \beta \in \mathbb{R}^n$, and the norm induced by this inner product is written by $\|\alpha\|_K := \sqrt{\langle \alpha, \alpha\rangle_K}$.

MKL fits the decision function of the form $f(x) + b = \sum_{m=1}^M f_m(x) + b$ as the minimizer of Eq. (1) where each f_m is an element of a different RKHS $\mathcal{H}_m$. By the representer theorem [10], the optimal solution of Eq. (1) is attained in the form of $f_m(x) = \sum_i k_m(x, x_i)\alpha_{m,i}$. If we write $\alpha_m = (\alpha_{m,1}, \ldots, \alpha_{m,N})^\top$, $\alpha = (\alpha_1^\top, \ldots, \alpha_M^\top)^\top \in \mathbb{R}^{MN}$ and $\bar{K} = (K_1, \ldots, K_M) \in \mathbb{R}^{N\times NM}$, then the optimization problem Eq. (1) is reduced to the following finite dimensional optimization problem:

$$\underset{\alpha\in\mathbb{R}^{MN}, b\in\mathbb{R}}{\text{minimize}} \quad \sum_{i=1}^N \ell(y_i, (\bar{K}\alpha)_i + b) + C \sum_{m=1}^M \|\alpha_m\|_{K_m}$$

[2]To avoid numerical instability, we added 10^{-8} to diagonal elements of K_m in the numerical experiments.

where $(\cdot)_i$ represents the i-th element of a vector. Here the loss function $\ell(y, f)$ may be taken as a *hinge loss* $\max(1 - yf, 0)$ or a *logistic loss* $\log(1 + \exp(-yf))$ for a classification problem, or a *squared loss* $(y - f)^2$ or a *SVR loss* $\max(|y - f| - \epsilon, 0)$ for a regression problem. For simplicity we rewrite the above problem as

$$\text{(P)} \qquad \underset{\alpha\in\mathbb{R}^{MN}, b\in\mathbb{R}}{\text{minimize}} \qquad f_\ell(\bar{K}\alpha + b\mathbf{1}) + \phi_{CK}(\alpha), \tag{2}$$

where $\mathbf{1} = (1, \ldots, 1)^\top$ and

$$f_\ell(z) = \sum_{i=1}^{N} \ell(y_i, z_i), \qquad \phi_{CK}(\alpha) = \sum_{m=1}^{M} \phi_{CK_m}(\alpha_m) = C \sum_{m=1}^{M} \|\alpha_m\|_{K_m}.$$

According to [1, 17], it can be shown that the optimal solution of (P) has a form of $f^\star(x) + b^\star = \sum_{i=1}^N \hat{\alpha}_i^\star(\sum_{m=1}^M d_m^\star k_m(x, x_i)) + b^\star$ where $0 \le d_m^\star \le 1$ and $\sum_{m=1}^M d_m^\star = 1$.

3 An augmented Lagrangian method for MKL

In this section, we first introduce our method as a proximal minimization method. Second, we assume that the loss function is twice differentiable and derive a Newton method for the inner minimization. Finally, the method is extended to the situation that the loss function is non-differentiable.

3.1 Dual augmented Lagrangian method as a proximal minimization method:

The minimization problem (P) is a convex but a non-differentiable problem. We apply the proximal minimization method [15] to our problem (P) to obtain a new variant of the dual augmented Lagrangian method proposed in [18] (see also [9, 12, 3]). The proximal minimization method converts the problem (P) into a sequence of "smoothed" minimization problems as follows:

$$(\alpha^{(t+1)}, b^{(t+1)}) = \underset{\alpha\in\mathbb{R}^{MN}, b\in\mathbb{R}}{\operatorname{argmin}} \left(f_\ell(\bar{K}\alpha + b\mathbf{1}) + \phi_{CK}(\alpha) + \sum_{m=1}^{M} \frac{\|\alpha_m - \alpha_m^{(t)}\|_{K_m}^2}{2\gamma_m^{(t)}} + \frac{(b - b^{(t)})^2}{2\gamma_b^{(t)}} \right), \tag{3}$$

where $0 < \gamma_m^{(1)} \le \gamma_m^{(2)} \le \ldots$ and $0 < \gamma_b^{(1)} \le \gamma_b^{(2)} \le \ldots$ are nondecreasing sequences of *penalty parameters* and $(\alpha^{(t)}, b^{(t)})$ is an approximate minimizer at the t-th iteration. Starting from some initial solution $(\alpha^{(0)}, b^{(0)})$, it is known that the sequence $\alpha^{(t)}$ $(t = 0, 1, 2, \ldots)$ converges to the minimum of (P) at a rate roughly proportional to $1/\min(\gamma_m^{(t)}, \gamma_b)$; thus when the sequences of penalty parameters go to infinity the proposed algorithm converges super-linearly [15]. In order to carry out the minimization in Eq. (3) in practice, we use the Lagrangian duality [4] and rewrite Eq. (3) as a min-max problem (see also [15]).

$$\begin{aligned} \min_{\substack{\alpha\in\mathbb{R}^{MN}\\ b\in\mathbb{R}}} \Bigg(\max_{\substack{\rho\in\mathbb{R}^N\\ u\in\mathbb{R}^{MN}}} & \left(-f_\ell^*(-\rho) - \sum_{m=1}^M \phi_{CK_m}^*(K_m u_m) - b\sum_{i=1}^N \rho_i - \sum_{m=1}^M \alpha_m^\top K_m(\rho - u_m) \right) \\ & + \sum_{m=1}^M \tfrac{1}{2\gamma_m^{(t)}} \|\alpha_m - \alpha_m^{(t)}\|_{K_m}^2 + \tfrac{1}{2\gamma_b^{(t)}}(b - b^{(t)})^2 \Bigg), \end{aligned} \tag{4}$$

where f_ℓ^* and $\phi_{CK_m}^*$ are the convex conjugate functions of f_ℓ and ϕ_{CK_m} (see [4]) and we define $u = (u_1^\top, \ldots, u_M^\top) \in \mathbb{R}^{MN}$ and $u_m \in \mathbb{R}^N$ $(m = 1, \ldots, M)$. It is easy to verify that the inner

Table 1: Algorithm of SpicyMKL

```
1. Choose a sequence $\gamma_m^{(t)} \to \infty$ $(m = 1, \dots, M)$, $\gamma_b^{(t)} \to \infty$ as $t \to \infty$.
2. Minimize the augmented Lagrangian with respect to $\rho$:
   $\rho^{(t)} = \mathrm{argmin}_\rho \left( f_\ell^*(-\rho) + \sum_m \frac{1}{2\gamma_m^{(t)}} \|\mathrm{ST}^m_{\gamma_m^{(t)} C}(\alpha_m^{(t)} + \gamma_m^{(t)} \rho^{(t)})\|_{K_m}^2 + \frac{1}{2\gamma_b^{(t)}} (b^{(t)} + \gamma_b^{(t)} \sum_i \rho_i)^2 \right)$.
3. Update $\alpha_m^{(t+1)} \leftarrow \mathrm{ST}^m_{\gamma_m^{(t)} C} \left( \alpha_m^{(t)} + \gamma_m^{(t)} \rho^{(t)} \right)$, $b^{(t+1)} \leftarrow b^{(t)} + \gamma_b \sum_i \rho_i^{(t)}$.
4. Repeat 2. and 3. until the stopping criterion is satisfied.
```

maximization yields the first two terms in Eq. (3). Now we can exchange the order of minimization and maximization because the function to be min-maxed in the above equation is a convex function for (α, b) and a concave function for (ρ, u) (see Chapter 36 of [14]). By minimizing Eq. (4) with respect to (α, b) and maximizing it with respect to u_m we obtain the following update equations (see Appendix A for the derivation):

$$\alpha_m^{(t+1)} = \mathrm{ST}^m_{\gamma_m^{(t)} C}(\alpha_m^{(t)} + \gamma_m^{(t)} \rho^{(t)}) \quad (m = 1, \dots, M), \tag{5}$$

$$b^{(t+1)} = b^{(t)} + \gamma_b^{(t)} \textstyle\sum_{i=1}^N \rho_i^{(t)}. \tag{6}$$

where the well known soft thresholding function (see [8, 7, 6, 21]) ST^m_C is defined for the MKL problem as follows:

$$\mathrm{ST}^m_C(v) = v \frac{\max(\|v\|_{K_m} - C, 0)}{\|v\|_{K_m}},$$

and $\rho^{(t)} \in \mathbb{R}^n$ is the minimizer of the function $\varphi_{\gamma^{(t)}}(\rho; \alpha^{(t)}, b^{(t)})$ defined as follows:

$$\varphi_\gamma(\rho; \alpha^{(t)}, b^{(t)}) = f_\ell^*(-\rho) + \sum_{m=1}^M \frac{1}{2\gamma_m} \|\mathrm{ST}^m_{\gamma_m C}(\alpha_m^{(t)} + \gamma_m \rho)\|_{K_m}^2 + \frac{1}{2\gamma_b} (b^{(t)} + \gamma_b \sum_i \rho_i)^2, \tag{7}$$

where $\gamma = (\gamma_1, \dots, \gamma_M, \gamma_b)^\top \in \mathbb{R}^{M+1}$. At every iteration we minimize $\varphi_{\gamma^{(t)}}(\rho; \alpha^{(t)}, b^{(t)})$ with respect to ρ and use the minimizer $\rho^{(t)}$ in the update rules (Eqs. (5) and (6)). The overall algorithm is shown in Table 1. We call the proposed algorithm *Sparse Iterative MKL (SpicyMKL)*. The above update equations (5)-(6) exactly correspond the augmented Lagrangian method for the dual of (P) (see [18]) but derived in a simpler way using the techniques from [15].

3.2 Minimizing the augmented Lagrangian function:

Note that the augmented Lagrangian (AL) function $\varphi_\gamma(\rho; \alpha, b)$ (Eq. (7)) that we need to minimize at every iteration is convex and *differentiable*. This minimization can be carried out using standard techniques such as the Newton method or the quasi-Newton method. We use the Newton method because we can exploit the sparsity in the intermediate solution in the computation of the gradient and the Hessian of the objective function. The case where $\ell^*(y, \cdot)$ is non-differentiable is discussed in the next subsection. Let $v_m = \alpha_m + \gamma_m \rho$. If the conjugate loss function f_ℓ^* is twice differentiable (more specifically if $\ell^*(y, \cdot)$ is so), the gradient and the Hessian of $\varphi_\gamma(\rho; \alpha, b)$ can be written as follows:

$$\nabla_\rho \varphi_\gamma(\rho; \alpha, b) = \nabla_\rho f_\ell^*(-\rho) + \sum_{m \in M_+} K_m \mathrm{ST}^m_{\gamma_m C}(\alpha_m + \gamma_m \rho) + (b + \gamma_b \sum_i \rho_i)\mathbf{1}, \tag{8}$$

$$\nabla^2_\rho \varphi_\gamma(\rho;\alpha,b) = \nabla^2_\rho f^*_\ell(-\rho) + \sum_{m\in M_+} \gamma_m \Big((1-q_m)K_m + q_m K_m \tilde{v}_m \tilde{v}_m^\top K_m\Big) + \gamma_b \mathbf{1}\mathbf{1}^\top, \tag{9}$$

where M_+ is the set of indices such that $\|v_m\|_{K_m} > \gamma_m C$, $q_m = \frac{\gamma_m C}{\|v_m\|_{K_m}}$, and $\tilde{v}_m = v_m/\|v_m\|_{K_m}$ $(m \in M_+)$.

Remark 1. The computation of the gradient and the Hessian of $\varphi_\gamma(\rho;\alpha,b)$ is efficient because they require only the terms corresponding to the *active kernels*, i.e. the set of m such that $\|\alpha_m + \gamma_m \rho\|_{K_m} > \gamma_m C$.

Note that the domain of $\ell^*(y,\cdot)$ may be some closed interval in $\mathbb{R}$ as long as we know that the minimum of the AL function $\varphi_\gamma(\rho;\alpha,b)$ is not attained at the boundary of the domain. This is for example the case for the logistic loss function whose conjugate is the negative entropy function $\ell^*(y_i,-\rho_i) = (y_i\rho_i)\log(y_i\rho_i) + (1-y_i\rho_i)\log(1-y_i\rho_i)$. In this case, the violation of the constraint can be easily prevented by the line search performed at each Newton iteration. The case the minimum is attained typically at the boundary (e.g., the hinge loss) is handled in the next subsection.

3.3 Explicitly handling boundary constraints:

The Newton method with line search described in the last section is unsuitable when the conjugate loss function $\ell^*(y,\cdot)$ has a non-differentiable point in the interior of its domain or it has finite gradient at the boundary of its domain. We use the same augmented Lagrangian technique for these cases. More specifically we introduce additional primal variables so that the AL function $\varphi_\gamma(\cdot;\alpha,b)$ becomes differentiable. We explain this in the case of hinge loss for classification. Generalization to other cases is straightforward, but we omit the details due to the lack of spaces. To this end, we introduce two sets of slack variables $\xi = (\xi_1,\ldots,\xi_N)^\top \geq 0, \zeta = (\zeta_1,\ldots,\zeta_N)^\top \geq 0$ as in standard SVM literatures (see e.g., [16]). The basic update equation (Eq. (3)) is rewritten as follows[3]:

$$(\alpha^{(t+1)}, b^{(t+1)}, \xi^{(t+1)}, \zeta^{(t+1)}) =$$
$$\operatorname*{argmin}_{\substack{\alpha\in\mathbb{R}^{(MN)},b\in\mathbb{R}\\ \xi\in\mathbb{R}^N_+,\zeta\in\mathbb{R}^N_+}} \left\{ f(\alpha,b,\xi,\zeta) + \sum_{m=1}^M \frac{\|\alpha_m - \alpha_m^{(t)}\|^2_{K_m}}{2\gamma_m^{(t)}} + \frac{(b-b^{(t)})^2}{2\gamma_b^{(t)}} + \frac{\|\xi-\xi^{(t)}\|^2}{2\gamma_\xi^{(t)}} + \frac{\|\zeta-\zeta^{(t)}\|^2}{2\gamma_\zeta^{(t)}} \right\},$$

where

$$f(\alpha,b,\xi,\zeta) = \begin{cases} \sum_{i=1}^N \xi_i + \phi_{CK}(\alpha) & (\text{if } y_i((\sum_{m=1}^M K_m\alpha_m)_i + b) = 1 - \xi_i + \zeta_i, \forall i), \\ +\infty & (\text{otherwise}). \end{cases}$$

This function f can again be expressed in terms of maximum over $\rho \in \mathbb{R}^N$, $u \in \mathbb{R}^{MN}$ as follows:

$$f(\alpha,b,\xi,\zeta) = \max_{\rho\in\mathbb{R}^N, u\in\mathbb{R}^{MN}} \Bigg\{ -\sum_{i=1}^N(-y_i\rho_i) - \sum_{m=1}^M \phi^*_{CK_m}(K_m u_m) - b\sum_{i=1}^N \rho_i - \sum_{m=1}^M \alpha_m^\top K_m(\rho - u_m)$$
$$+ \sum_{i=1}^N \xi_i(1-y_i\rho_i) + \sum_{i=1}^N \zeta_i(y_i\rho_i) \Bigg\}.$$

[3] $\mathbb{R}_+$ is the set of non-negative real numbers

We exchange the order of minimization and maximization as before and remove α, b, ξ, ζ, and u_m by explicitly minimizing or maximizing over them (see also Appendix A). Finally we obtain the following update equations.

$$\alpha_m^{(t+1)} = \mathrm{ST}^m_{\gamma_m^{(t)}C}(\alpha_m^{(t)} + \gamma_m^{(t)}\rho^{(t)}), \qquad b^{(t+1)} = b^{(t)} + \gamma_b^{(t)} \textstyle\sum_{i=1}^N \rho_i^{(t)} \tag{10}$$

$$\xi_i^{(t+1)} = \max(0, \xi_i^{(t)} - \gamma_\xi^{(t)}(1 - y_i\rho_i^{(t)})), \qquad \zeta_i^{(t+1)} = \max(0, \zeta_i^{(t)} - \gamma_\zeta^{(t)} y_i\rho_i^{(t)}) \tag{11}$$

where $\rho^{(t)} \in \mathbb{R}^N$ is the minimizer of the function $\varphi_{\gamma^{(t)}}(\rho; \alpha^{(t)}, b^{(t)}, \xi^{(t)}, \zeta^{(t)})$ defined as follows:

$$\begin{aligned}\varphi_\gamma(\rho; \alpha, b, \xi, \zeta) = -\sum_{i=1}^N y_i\rho_i + \sum_{m=1}^M \tfrac{1}{2\gamma_m}\|\mathrm{ST}^m_{\gamma_m C}(\alpha_m + \gamma_m\rho)\|_{K_m}^2 + \tfrac{1}{2\gamma_b}(b + \gamma_b \sum_{i=1}^N \rho_i)^2 \\ + \tfrac{1}{2\gamma_\xi}\sum_{i=1}^N \max(0, \xi_i - \gamma_\xi(1 - y_i\rho_i))^2 + \tfrac{1}{2\gamma_\zeta}\sum_{i=1}^N \max(0, \zeta_i - \gamma_\zeta y_i\rho_i)^2,\end{aligned} \tag{12}$$

and $\gamma = (\{\gamma_m\}_{m=1}^M, \gamma_b, \gamma_\xi, \gamma_\zeta)^\top \in \mathbb{R}_+^{M+3}$. The gradient and the Hessian of φ_γ with respect to ρ can be obtained in a similar way to Eqs. (8) and (9). Thus we apply the Newton method. The overall algorithm is analogous to Table 1 with update equations (10)-(12).

3.4 Technical details of computations:

We used *Armijo's rule* to find a step size of the Newton method. During the back tracking to find the step size, the computational bottle-neck is the computation of $\|\rho + c\Delta\rho\|_{K_m}$ $(m = 1, \ldots, M)$ where $\Delta\rho$ is the Newton update direction and $0 < c \leq 1$ is a step size. However, this computation is needed only on the active kernels $\{m \mid \|\rho + \frac{\alpha_m}{\gamma_m}\|_{K_m} > C \text{ or } \|\rho + \Delta\rho + \frac{\alpha_m}{\gamma_m}\|_{K_m} > C\}$ because of the convexity of $\|\cdot\|_{K_m}$. This reduces considerable amount of computation.

4 Relations to existing methods

4.1 Iterative Shrinkage/Thresholding:

Another approach to minimize Eq. (3) is to linearly approximate the loss term $f_\ell(\bar{K}\alpha + b\mathbf{1})$ as follows:

$$f_\ell(\bar{K}\alpha + b\mathbf{1}) \simeq f_\ell(z^{(t)}) + \nabla_z f_\ell(z^{(t)})(\bar{K}(\alpha - \alpha^{(t)}) + (b - b^{(t)})\mathbf{1})$$

where $z^{(t)} = \bar{K}\alpha^{(t)} + b^{(t)}\mathbf{1}$. Minimization over α and b yields the following update equations:

$$\alpha_m^{(t+1)} = \mathrm{ST}^m_{\gamma_m^{(t)}C}\left(\alpha_m^{(t)} - \gamma_m^{(t)}\nabla_z f_\ell(z^{(t)})\right), \quad b^{(t+1)} = b^{(t)} - \gamma_b^{(t)}\sum_{i=1}^N (\nabla_z f_\ell(z^{(t)}))_i$$

This is equivalent to the popular iterative shrinkage/thresholding (IST) algorithm ([8, 7, 6, 19] see also [21]) generalized to the MKL setting. Thus the proposed SpicyMKL can be considered as the *exact* version of the proximal minimization method (Eq. (3)) whereas the IST approach approximately minimizes Eq. (3).

4.2 Correspondence of regularization terms:

The regularization term in SILP, SimpleMKL and HessianMKL is defined by $\frac{C'}{2}(\sum_m \|f_m\|_{\mathcal{H}_m})^2$ instead of $C(\sum_m \|f_m\|_{\mathcal{H}_m})$ in our formulation (see Eq. (1)). However, two formulations are equivalent because the minimizer $\{f_m^\star\}_{m=1}^M$ of our formulation with the regularization parameter

C also minimizes the objective function of SimpleMKL with the regularization parameter C' that is defined by

$$C' = C(\textstyle\sum_{m=1}^{M} \|f_m^\star\|_{\mathcal{H}_m}). \tag{13}$$

This is given by the relation $\nabla_{f_m} \frac{1}{2}(\sum_m \|f_m\|_{\mathcal{H}_m})^2 = (\sum_m \|f_m\|_{\mathcal{H}_m}) \nabla_{f_m} \|f_m\|_{\mathcal{H}_m}$ where ∇_{f_m} is a subdifferential with respect to f_m.

5 Numerical experiments

In this section, we experimentally investigate the performance of the proposed method and existing MKL methods using several datasets of binary classification tasks[4]. We compared our algorithm SpicyMKL to SimpleMKL [13] and HessianMKL [5].

5.1 Performances on UCI benchmark datasets

The experimental settings were borrowed from the paper [13] of SimpleMKL, but we used larger number of kernels. We used 5 datasets from the UCI repository: 'Liver', 'Pima', 'Ionospher', 'Wpbc', 'Sonar'. The candidate kernels were Gaussian kernels with 24 different bandwidths (0.1 0.25 0.5 0.75 1 2 3 4 $\cdots$ 19 20). and polynomial kernels of degree 1 to 3. All of 27 different kernel funcionts (Gaussian kernels with different bandwidths and polynomial kernels of degrees 1 to 3) were applied to individual variables as well as jointly over all the variables, i.e., in total we have $27 \times (n + 1)$ candidate kernels, where n is the number of variables. All kernel matrices were normalized to unit trace, and were precomputed prior to running the algorithms.

For SpicyMKL, we report the result from two loss functions, the hinge loss and the logistic loss. For SimpleMKL and HessianMKL, we used the hinge loss. All methods were implemented in Matlab®. For SimpleMKL and HessianMKL, we used Matlab codes available from `http://asi.insa-rouen.fr/enseignants/~arakotom/code/mklindex.html` and `http://olivier.chapelle.cc/ams/` respectively.

For each dataset, we randomly chose 80 % of all sample points for training samples and the remaining 20 % were used for test samples. This procedures were repeated 10 times. Experiments were run on 3 different regularization parameters $C = 0.005$, 0.05 and 0.5. We converted the regularization parameter C of our formulation (2) to that for SimpleMKL and HessianMKL by Eq. (13). We employed a stopping criterion utilizing the *relative duality gap*, (primal obj $-$ dual obj)/primal obj, for both algorithms: with tolerance 0.01. The primal objective for SpicyMKL can be computed by using $\alpha^{(t)}$ and $b^{(t)}$. In order to compute the dual objective, we first project ρ to the l_∞ ball by $\tilde{\rho}' = \rho / \max\{\max_m\{\|\rho\|_{K_m}/C\}, 1\}$ and next project to the equality constraint $\tilde{\rho} = \tilde{\rho}' - \mathbf{1}(\sum \tilde{\rho}_i')/N$. Then we compute the dual objective function of SpicyMKL as $-f_\ell^*(-\tilde{\rho})$. The same technique can be found in [19].

The performance of each method is summarized in Figure 5.1. From top to bottom, are shown means of CPU time, test accuracy, and the number of kernels finally selected by the algorithms, with standard deviations over 10 trials. We can see that SpicyMKL tends to be faster than SimpleMKL (factor of $5 \sim 80$), and faster than HessianMKL when the number of kernels M is large. In all datasets, SpicyMKL becomes faster as C increases. This is because as the regularization becomes stronger the number of active kernels during the optimization decreases at a faster rate. Accuracies of all methods are nearly identical. This indicates that SpicyMKL properly converges to the optimal one. SpicyMKL using the logistic loss tends to show faster CPU time than that using the hinge loss. This is because, by the strict convexity of the conjugate function of the logistic loss, the Newton method in the inner loop (minimization of φ_γ with respect to ρ) converges faster than the hinge loss. An interesting point is that although

[4]All the experiments were executed on Intel Core i7 2.93GHz with 6GB RAM.

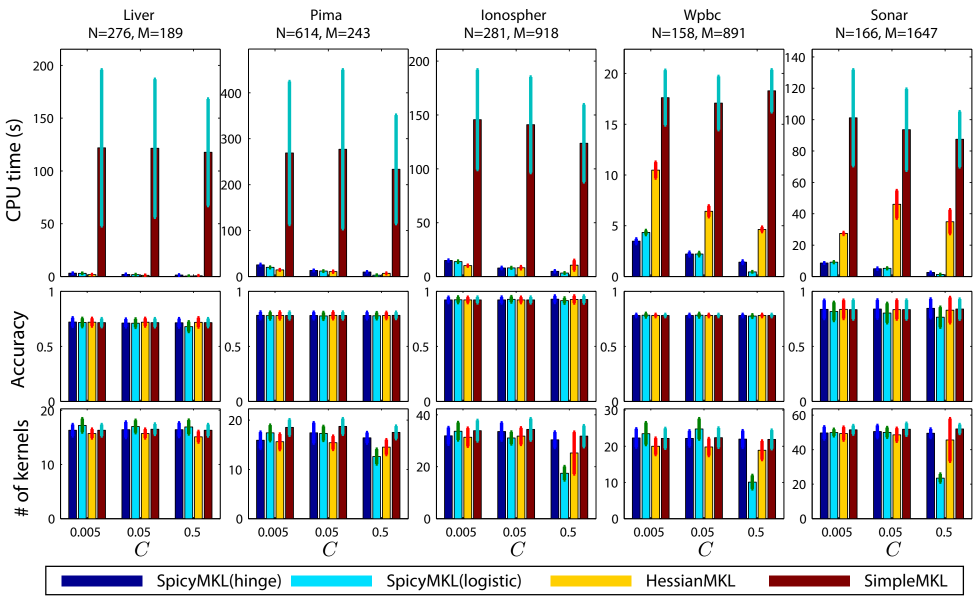


Figure 1: Mean and standard deviation of performance measures for each MKL method for UCI datasets.

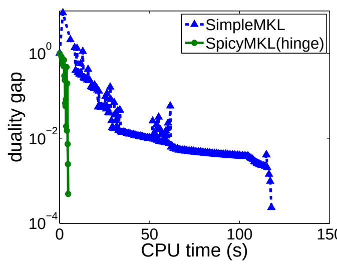
the logistic loss is often faster to train, yet the accuracy is nearly identical to that of the hinge loss. Moreover, when C is large (strong regularization), the solution under the logistic loss tends to be sparser than that under the hinge loss. For the hinge loss, the number of kernels selected by SpicyMKL is almost the same as that selected by SimpleMKL.

Figure 2(a) contains plots of the relative duality gaps of SpicyMKL (with hinge loss) and SimpleMKL against CPU time, both on the 'Ionospher' dataset. We can see that the duality gap of SpicyMKL rapidly drops. Figure 2(b) shows the number of kernels as a function of the CPU time spent by the algorithm. Here we again observe rapid decrease in the number of kernels in SpicyMKL. This reduces huge amount of computation per iteration.

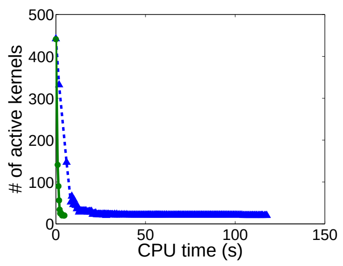
5.2 Scaling against the sample size and the number of kernels

Here we investigate the scaling of CPU time against the number of kernels and the sample size. We used 2 datasets from IDA benchmark repository[5]: 'Ringnorm' and 'Splice'. The same relative duality gap criterion with tolerance 0.01 is used. We generated the basis kernels by randomly selecting subsets of features and applying a Gaussian kernels with random width $\sigma = 5\chi^2 + 0.1$, where χ^2 is a chi-squared random variable. In Figure 3 the number of kernels is increased from 50 to 6000. The vertical axis shows the CPU time averaged over 10 random train-test splitting where the size of training set was fixed to 200. We observe that the CPU time of HessianMKL is the smallest for small number of kernels, but it grows rapidly as the number of kernels increases. On the other hand, CPU time of SpicyMKL has a milder dependency to the number of kernels. In particular, SpicyMKL is tens times faster than SimpleMKL and HessianMKL when the number of kernels is 6000. In Figure 4 the number of training samples is increased. The number of kernels is fixed to 20. The scaling behaviour of the CPU time of SpicyMKL is comparable to other methods.

[5]`http://ida.first.fhg.de/projects/bench/benchmarks.htm`

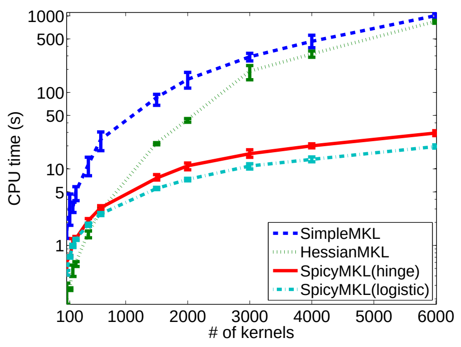


(a) Evolution of Duality Gap

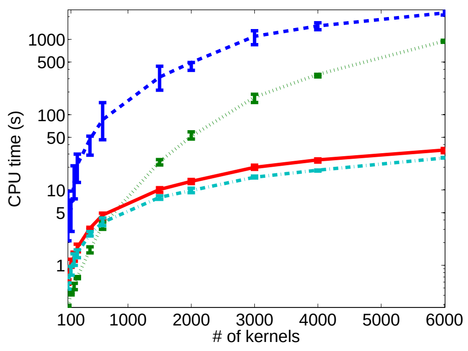


(b) Evolution of # of Active Kernels

Figure 2: Duality gap and # of active kernels against CPU time



(a) Ringnorm

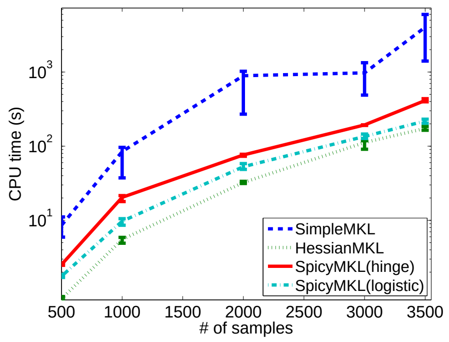


(b) Splice

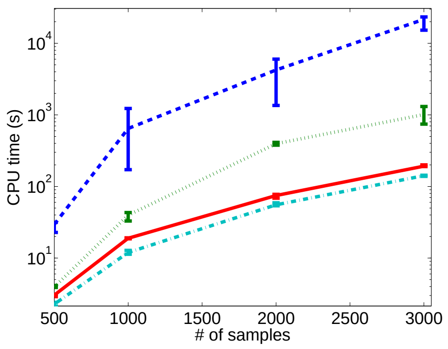
Figure 3: CPU time as a function of the number of kernels

6 Conclusion and future direction

In this article, we have proposed a new efficient training algorithm for MKL with general convex loss functions. The proposed SpicyMKL algorithm generates a sequence of primal variables by iteratively optimizing a smoothed version of the dual MKL problem. The outer loop of SpicyMKL is a proximal minimization method and it converges super-linearly. The inner minimization is efficiently carried out by the Newton method. The numerical experiments show SpicyMKL scales well with increasing number of kernels and it has similar scaling behaviour against the number of samples to conventional methods. The logistic-loss SpicyMKL has shown the best computational efficiency and improved sparsity at a comparable test accuracy. Future



(a) Ringnorm



(b) Splice

Figure 4: CPU time as a function of the sample size

work includes a second order modification of the update rule of the primal variables and applying the techniques used SpicyMKL for upper-bound based methods.

A Derivation of the update equations

We start from the min-max equation (Eq. (4)) and derive the SpicyMKL update equations (Eqs. (5)-(7)). First by exchanging the order of minimization and maximization and completing the squares, we obtain,

$$\begin{aligned}\text{Eq. (4)} = \max_{\substack{\rho\in\mathbb{R}^N\\ u\in\mathbb{R}^{MN}}} \Bigg\{ & -f_\ell^*(-\rho) - \sum_{m=1}^{M}\left(\phi_{CK_m}^*(K_m u_m) + {\alpha_m^{(t)}}^\top K_m(\rho-u_m) + \tfrac{\gamma_m^{(t)}}{2}\|\rho-u_m\|_{K_m}^2\right) \\ & + \min_{\substack{\alpha\in\mathbb{R}^{MN}\\ b\in\mathbb{R}}}\left(\sum_{m=1}^{M}\frac{\|\alpha_m-\alpha_m^{(t)}-\gamma_m^{(t)}(\rho-u_m)\|_{K_m}^2}{2\gamma_m^{(t)}} + \frac{\left(b-b^{(t)}-\gamma_b^{(t)}\sum_{i=1}^{N}\rho_i\right)^2}{2\gamma_b^{(t)}}\right) - b^{(t)}\sum_{i=1}^{N}\rho_i - \tfrac{\gamma_b^{(t)}}{2}\Big(\sum_{i=1}^{N}\rho_i\Big)^2\Bigg\}. \qquad (14)\end{aligned}$$

Furthermore by turning the maximization into a minimization and moving the minimization with respect to u_m inside, we have,

$$\text{Eq. (4)} = -\min_{\rho\in\mathbb{R}^N}\left\{ f_\ell^*(-\rho) + \textstyle\sum_{m=1}^{M}\Phi_m(\rho) + b^{(t)}\sum_{i=1}^{N}\rho_i + \frac{\gamma_b^{(t)}}{2}\left(\sum_{i=1}^{N}\rho_i\right)^2\right\}, \qquad (15)$$

where

$$\Phi_m(\rho) = \min_{u'_m\in\mathbb{R}^N}\left(\phi_{CK_m}^*(K_m u'_m/\gamma_m^{(t)}) + \frac{1}{2\gamma_m^{(t)}}\|u'_m - (\alpha_m^{(t)}+\gamma_m^{(t)}\rho)\|_{K_m}^2\right) + \text{const},$$

(we redefined $\gamma_m^{(t)}u_m$ as u'_m for notational convenience) and const is a term that only depends on $\gamma^{(t)}$ and $\alpha^{(t)}$. Now since $\phi_{CK_m}(v) = C\|v\|_{K_m}$, we have

$$\phi_{CK_m}^*(K_m u) = \begin{cases} 0 & (\|u\|_{K_m}\le C), \\ +\infty & (\text{otherwise}). \end{cases}$$

Let $v_m^{(t)} = \alpha_m^{(t)} + \gamma_m^{(t)}\rho$. From a simple geometric consideration, we have,

$$u_m'^{(t)} = v_m^{(t)}\frac{\min(\|v_m^{(t)}\|_{K_m}, \gamma_m^{(t)}C)}{\|v_m^{(t)}\|_{K_m}} = v_m^{(t)} - \mathrm{ST}^m_{\gamma_m^{(t)}C}(v_m).$$

Thus we obtain,

$$\Phi_m(\rho) = \frac{1}{2\gamma_m^{(t)}}\|\mathrm{ST}^m_{\gamma_m^{(t)}C}(\alpha_m^{(t)}+\gamma_m^{(t)}\rho)\|_{K_m}^2. \qquad (16)$$

We obtain Eq. (7) by substituting Eq. (16) into Eq. (15) and rearranging terms. Furthermore, from the minimization in Eq. (14) we have:

$$\alpha_m^{(t+1)} = \alpha_m^{(t)} + \gamma_m^{(t)}\rho^{(t)} - u_m'^{(t)} = \mathrm{ST}^m_{\gamma_m^{(t)}C}(\alpha_m^{(t)}+\gamma_m^{(t)}\rho^{(t)}), \quad b^{(t+1)} = b^{(t)} + \gamma_b\textstyle\sum_{i=1}^{N}\rho_i^{(t)}$$

where $\rho^{(t)}$ is the minimizer in Eq. (15) or Eq. (7).

References

[1] F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm, 2004.

[2] F. R. Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

[3] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press, 1982.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[5] O. Chapelle and A. Rakotomamonjy. Second order optimization of kernel parameters. In *NIPS 2008 Workshop on Kernel Learning: Automatic Selection of Optimal Kernels*, Whistler, 2008.

[6] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005.

[7] I. Daubechies, M. Defrise, and C. D. Mol. An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint. *Communications on Pure and Applied Mathematics*, LVII:1413–1457, 2004.

[8] M. Figueiredo and R. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Trans. Image Process.*, 12:906–916, 2003.

[9] M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory & Applications*, 4:303–320, 1969.

[10] G. S. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

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

[12] M. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, London, New York, 1969.

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

[14] G. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, 1970.

[15] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Math. of Oper. Res.*, 1:97–116, 1976.

[16] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond.* MIT Press, Cambridge, MA, 2002.

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

[18] R. Tomioka and M. Sugiyama. Dual Augmented Lagrangian Method for Efficient Sparse Reconstruction, 2009.

[19] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse Reconstruction by Separable Approximation. *IEEE Trans. Signal Process.*, 2009.

[20] Z. Xu, R. Jin, I. King, and M. R. Lyu. An extended level method for efficient multiple kernel learning. pages 1825–1832, 2009.

[21] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman Iterative Algorithms for L1-Minimization with Applications to Compressed Sensing. *SIAM J. Imaging Sciences*, 1(1):143–168, 2008.

[22] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of The Royal Statistical Society Series B*, 68(1):49–67, 2006.