Mach Learn (2011) 85:77–108
DOI 10.1007/s10994-011-5252-9

SpicyMKL: a fast algorithm for Multiple Kernel Learning with thousands of kernels

Taiji Suzuki · Ryota Tomioka

Received: 28 February 2010 / Accepted: 12 May 2011 / Published online: 3 June 2011

Abstract We propose a new optimization algorithm for Multiple Kernel Learning (MKL) called SpicyMKL, which is applicable to general convex loss functions and general types of regularization. The proposed SpicyMKL iteratively solves smooth minimization problems. Thus, there is no need of solving SVM, LP, or QP internally. SpicyMKL can be viewed as a proximal minimization method and converges super-linearly. The cost of inner minimization is roughly proportional to the number of active kernels. Therefore, when we aim for a sparse kernel combination, our algorithm scales well against increasing number of kernels. Moreover, we give a general block-norm formulation of MKL that includes non-sparse regularizations, such as elastic-net and ℓ_p-norm regularizations. Extending SpicyMKL, we propose an efficient optimization method for the general regularization framework. Experimental results show that our algorithm is faster than existing methods especially when the number of kernels is large (>1000).

Keywords Multiple kernel learning · Sparsity · Non-smooth optimization · Super-linear convergence · Proximal minimization

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) (Aronszajn 1950; Schölkopf and Smola 2002). In such a learning framework, the choice of a kernel function can strongly influence the performance. Instead

Editors: Süreyya Özöğür-Akyüz, Dervim Ünay, Alex Smola.

T. Suzuki (✉) · R. Tomioka
Department of Mathematical Informatics, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
e-mail: t-suzuki@mist.i.u-tokyo.ac.jp

R. Tomioka
e-mail: tomioka@mist.i.u-tokyo.ac.jp

of using a fixed kernel, *Multiple Kernel Learning* (*MKL*) (Lanckriet et al. 2004; Bach et al. 2004; Micchelli and Pontil 2005) aims to find an optimal combination of multiple candidate kernels.

More specifically, we assume that a data point $x \in \mathcal{X}$ lies in a space $\mathcal{X}$ and we are given M candidate kernel functions $k_m : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ $(m = 1, \ldots, M)$. Each kernel function corresponds to one data source. A conical combination of k_m $(m = 1, \ldots, M)$ gives the combined kernel function $\bar{k} = \sum_{m=1}^{M} d_m k_m$, where d_m is a nonnegative weight. Our goal is to find a good set of kernel weights based on some training examples.

While the MKL framework has opened up a possibility to combine multiple heterogeneous data sources (numerical features, texts, links) in a principled manner, it is also posing an optimization challenge: the number of kernels can be very large. Instead of arguing whether polynomial kernel or Gaussian kernel fits a given problem better, we can simply put both of them into an MKL algorithm; instead of evaluating which band-width parameter to choose, we can simply generate kernels from all the possible combinations of parameter values and feed them to an MKL algorithm. See Gehler and Nowozin (2009), Tomioka and Suzuki (2009).

Various optimization algorithms have been proposed in the context of MKL. In the pioneering work of Lanckriet et al. (2004), MKL was formulated as a semi-definite programming (SDP) problem. Bach et al. (2004) showed that the SDP can be reduced to a second order conic programming (SOCP) problem. However solving SDP or SOCP via general convex optimization solver can be quite heavy especially for large number of samples.

More recently, *wrapper methods* have been proposed. A wrapper method iteratively solves a single kernel learning problem (e.g., SVM), for a given kernel combination and then updates the kernel weights. A nice property of this type of methods is that it can make use of existing well-tuned solvers for SVM. Semi-Infinite Linear Program (SILP) approach proposed by Sonnenburg et al. (2006) utilizes a cutting plane method for the update of the kernel weights. SILP often suffers from instability of the solution sequence especially when the number of kernels is large, i.e. the intermediate solution oscillates around the optimal one (Rakotomamonjy et al. 2008). SimpleMKL proposed by Rakotomamonjy et al. (2008) performs a reduced gradient descent on the kernel weights. Simple MKL resolves the drawback of SILP, but still it is a first order method. Xu et al. (2009) proposed a novel Level Method (which we call LevelMKL) as an improvement of SILP and SimpleMKL. LevelMKL is rather efficient than SimpleMKL and scales well against the number of kernels by utilizing sparsity but it shows unstable behavior as the algorithm proceeds because LevelMKL solves Linear Programming (LP) and Quadratic Programming (QP) of increasingly large size as its iteration proceeds. HessianMKL proposed by Chapelle and Rakotomamonjy (2008) replaced the gradient descent update of SimpleMKL with a Newton update. At each iteration, HessianMKL solves a QP problem with the size of the number of kernels to obtain the Newton update direction. HessianMKL shows second order convergence, but scales badly against the number of kernels because the size of the QP grows as the number of kernels grows.

The first contribution of this article is an efficient optimization algorithm for MKL based on the block 1-norm formulation introduced in Bach et al. (2005) (see also Bach et al. 2004); see (4). The block 1-norm formulation can be viewed as a kernelized version of group lasso (Yuan and Lin 2006; Bach 2008). For group lasso, or more generally sparse estimation, efficient optimization algorithms have recently been studied intensively, boosted by the development of compressive sensing theory (Candes et al. 2006). Based on this view, we extend the dual augmented-Lagrangian (DAL) algorithm (Tomioka and Sugiyama 2009) recently proposed in the context of sparse estimation to kernel-based learning. DAL is efficient when the number of unknown variables is much larger than the number of samples.

This enables us to scale the proposed algorithm, which we call *SpicyMKL*, to thousands of kernels.

Compared to the original DAL algorithm (Tomioka and Sugiyama 2009), our presentation is based on an application of proximal minimization framework (Rockafellar 1976) to the primal MKL problem. We believe that the current formulation is more transparent compared to the dual based formulation in Tomioka and Sugiyama (2009), because we are not necessarily interested in solving the dual problem. Moreover, we present a theorem on the rate of convergence.

SpicyMKL does not need to solve SVM, LP or QP internally as previous approaches. Instead, it minimizes a smooth function at every step. The cost of inner minimization is proportional to the number of active kernels. Therefore, when we aim for a sparse kernel combination, the proposed algorithm is efficient even when thousands of candidate kernels are used. In fact, we show numerically that we are able to train a classifier with 3000 kernels in less than 10 seconds.

Learning combination of kernels, however, has recently recognized as a more complex task than initially thought. Cortes (2009) pointed out that learning *convex* kernel combination with an ℓ_1-constraint on the kernel weights (see Sect. 2) produces an overly sparse (many kernel weights are zero) solution, and it is often outperformed by a simple uniform combination of kernels; accordingly they proposed to use an ℓ_2-constraint instead (Cortes et al. 2009). In order to search for the best trade-off between the sparse ℓ_1-MKL and the uniform weight combination, Kloft et al. (2009) proposed a general ℓ_p-norm constraint and Tomioka and Suzuki (2009) proposed an elastic-net regularization, both of which smoothly connect the 1-norm MKL and uniform weight combination.

The second contribution of this paper is to extend the block-norm formulation that allows us to view these generalized MKL models in a unified way, and provide an efficient optimization algorithm. We note that while this paper was under review, Kloft et al. (2010) presented a slightly different approach that results in a similar optimization algorithm. However, our formulation provides clearer relationship between the block-norm formulation and kernel-combination weights and is more general.

This article is organized as follows. In Sect. 2, we introduce the framework of block 1-norm MKL through Tikhonov regularization on the kernel weights. In Sect. 3, we propose an extension of DAL algorithm to kernel-learning setting. Our formulation of DAL algorithm is based on a primal application of the proximal minimization framework (Rockafellar 1976), which also sheds a new light on DAL algorithm itself. Furthermore, we discuss how we can carry out the inner minimization efficiently exploiting the sparsity of the 1-norm MKL. In Sect. 4, we extend our framework to general class of regularizers including the ℓ_p-norm MKL (Kloft et al. 2009) and Elastic-net MKL (Tomioka and Suzuki 2009). We extend the proposed SpicyMKL algorithm for the generalized formulation and also present a simple one-step optimization procedure for some special cases that include Elastic-net MKL and ℓ_p-norm MKL. In Sect. 5, we discuss the relations between the existing methods and the proposed method. In Sect. 6, we show the results of numerical experiments. The experiments show that SpicyMKL is efficient for block 1-norm regularization especially when the number of kernels is large. Moreover the one-step optimization procedure for elastic-net regularization shows quite fast convergence. In fact, it is faster than those methods with block 1-norm regularization. Finally, we summarize our contribution in Sect. 7. The proof of super-linear convergence of the proposed SpicyMKL is given in Appendix.

A Matlab® implementation of SpicyMKL is available at the following URL: http://www.simplex.t.u-tokyo.ac.jp/~s-taiji/software/SpicyMKL.

2 Framework of MKL

In this section, we first consider a learning problem with fixed kernel weights in Sect. 2.1. Next in Sect. 2.2, using Tikhonov regularization on the kernel weights, we derive a block 1-norm formulation of MKL, which can be considered as a direct extension of group lasso in the kernel-based learning setting. In addition, we discuss the connection between the current block 1-norm formulation and the *squared* block 1-norm formulation. In Sect. 2.3, we present a finite dimensional version of the proposed formulation and prepare notations for the later sections.

2.1 Fixed kernel combination

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 classification and $\mathcal{Y} = \mathbb{R}$ for regression). 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 that the Gram matrix K_m is positive definite.[1]

We first consider a learning problem with fixed kernel weights. More specifically, we fix non-negative kernel weights $d_1, d_2, \ldots, d_M$ and consider the RKHS $\bar{\mathcal{H}}$ corresponding to the combined kernel function $\bar{k} = \sum_{m=1}^M d_m k_m$. The (squared) RKHS norm of a function $\bar{f}$ in $\bar{\mathcal{H}}$ is written as follows (see Sect. 6 in Aronszajn 1950, and also Micchelli and Pontil 2005):

$$\|\bar{f}\|_{\bar{\mathcal{H}}}^2 := \min_{f_1 \in \mathcal{H}_1, \ldots, f_M \in \mathcal{H}_M} \sum_{m=1}^M \frac{\|f_m\|_{\mathcal{H}_m}^2}{d_m} \quad \text{s.t.} \quad \bar{f} = \sum_{m=1}^M f_m, \tag{1}$$

where $\mathcal{H}_m$ is the RKHS that corresponds to the kernel function k_m. Accordingly, with a fixed kernel combination, a supervised learning problem can be written as follows:

$$\underset{f_1 \in \mathcal{H}_1, \ldots, f_M \in \mathcal{H}_M, b \in \mathbb{R}}{\text{minimize}} \sum_{i=1}^N \ell\left(y_i, \sum_{m=1}^M f_m(x_i) + b\right) + \frac{C}{2} \sum_{m=1}^M \frac{\|f_m\|_{\mathcal{H}_m}^2}{d_m}, \tag{2}$$

where b is a bias term and $\ell(y, f)$ is a loss function, which can be the *hinge loss* $\max(1 - yf, 0)$ or the *logistic loss* $\log(1 + \exp(-yf))$ for a classification problem, or the *squared loss* $(y - f)^2$ or the *SVR loss* $\max(|y - f| - \epsilon, 0)$ for a regression problem. The above formulation may not be so useful in practice, because we can compute the combined kernel function $\bar{k}$ and optimize over $\bar{f}$ instead of optimizing M functions $f_1, \ldots, f_M$. However, explicitly handling the kernel weights allows us to consider various generalizations of MKL in a unified manner.

2.2 Learning kernel weights

In order to learn the kernel weights d_m through the objective (2), there is clearly a need for regularization, because the objective is a decreasing function of the kernel weights d_m. Roughly speaking, the kernel weight d_m corresponds to the complexity allowed for the mth classifier component f_m, without regularization, we can get a serious over-fitting.

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

One way to prevent such overfitting is to penalize the increase of the kernel weight d_m by adding a penalty term. Adding a linear penalty term, we have the following optimization problem:

$$\underset{\substack{f_1\in\mathcal{H}_1,\ldots,f_M\in\mathcal{H}_M,\\ b\in\mathbb{R},\\ d_1\geq 0,\ldots,d_M\geq 0}}{\text{minimize}} \sum_{i=1}^{N}\ell\left(y_i,\sum_{m=1}^{M}f_m(x_i)+b\right)+\frac{C}{2}\sum_{m=1}^{M}\left(\frac{\|f_m\|_{\mathcal{H}_m}^2}{d_m}+d_m\right). \tag{3}$$

The above formulation reduces to the block 1-norm introduced in Bach et al. (2005) by explicitly minimizing over d_m as follows:

$$\underset{\substack{f_1\in\mathcal{H}_1,\ldots,f_M\in\mathcal{H}_M,\\ b\in\mathbb{R}}}{\text{minimize}} \sum_{i=1}^{N}\ell\left(y_i,\sum_{m=1}^{M}f_m(x_i)+b\right)+C\sum_{m=1}^{M}\|f_m\|_{\mathcal{H}_m}, \tag{4}$$

where we used the inequality of arithmetic and geometric means; the minimum with respect to d_m is obtained by taking $d_m=\|f_m\|_{\mathcal{H}_m}$.

The regularization term in the above block 1-norm formulation is the linear sum of RKHS norms. This formulation can be seen as a direct generalization of group lasso (Yuan and Lin 2006) to the kernel-based learning setting, and motivates us to extend an efficient algorithm for sparse estimation to MKL.

The block 1-norm formulation (4) is related to the following squared block 1-norm formulation considered in Bach et al. (2004), Sonnenburg et al. (2006), Zien and Ong (2007), Rakotomamonjy et al. (2008):

$$\underset{f_1\in\mathcal{H}_1,\ldots,f_M\in\mathcal{H}_M,b\in\mathbb{R}}{\text{minimize}} \sum_{i=1}^{N}\ell\left(y_i,\sum_{m=1}^{M}f_m(x_i)+b\right)+\frac{\tilde{C}}{2}\left(\sum_{m=1}^{M}\|f_m\|_{\mathcal{H}_m}\right)^2, \tag{5}$$

which is obtained by considering a simplex constraint on the kernel weights (Kloft et al. 2009) instead of penalizing them as in (3).

The solution of the two problems (4) and (5) can be mapped to each other. In fact, let $\{f_m^\star\}_{m=1}^M$ be the minimizer of the block 1-norm formulation (4) with the regularization parameter C and let $\tilde{C}$ be

$$\tilde{C}=C\left(\sum_{m=1}^{M}\|f_m^\star\|_{\mathcal{H}_m}\right). \tag{6}$$

Then $\{f_m^\star\}$ also minimizes the squared block 1-norm formulation (5) with the regularization parameter $\tilde{\mathrm{C}}$ because of the relation

$$\partial_{f_m}\frac{1}{2}\left(\sum_{m=1}^{M}\|f_m\|_{\mathcal{H}_m}\right)^2=\left(\sum_{m=1}^{M}\|f_m\|_{\mathcal{H}_m}\right)\partial_{f_m}\|f_m\|_{\mathcal{H}_m},$$

where ∂_{f_m} is a subdifferential with respect to f_m.

2.3 Representer theorem

In this subsection, we convert the block 1-norm MKL formulation (4) into a finite dimensional optimization problem via the representer theorem and prepare notation for later sections.

The optimal solution of (4) is attained in the form of $f_m(x) = \sum_{i=1}^N k_m(x, x_i)\alpha_{m,i}$ due to the representer theorem (Kimeldorf and Wahba 1971). Thus, the optimization problem (4) is reduced to the following finite dimensional optimization problem:

$$\underset{\alpha\in\mathbb{R}^{NM},b\in\mathbb{R}}{\text{minimize}} \quad \sum_{i=1}^N \ell\left(y_i, \sum_{m=1}^M \sum_{j=1}^N k_m(x_i, x_j)\alpha_{m,j} + b\right) + C\sum_{m=1}^M \|\alpha_m\|_{K_m}$$

where $\alpha_m = (\alpha_{m,1}, \ldots, \alpha_{m,N})^\top$, $\alpha = (\alpha_1^\top, \ldots, \alpha_M^\top)^\top \in \mathbb{R}^{NM}$, and the norm $\|\cdot\|_{K_m}$ is defined through the inner product $\langle \alpha_m, \beta_m\rangle_{K_m} := \alpha_m^\top K_m \beta_m$ for $\alpha_m, \beta_m \in \mathbb{R}^N$. We also define the norm $\|\cdot\|_{\bar{K}}$ through the inner product $\langle \alpha, \beta\rangle_{\bar{K}} := \sum_{m=1}^M \langle \alpha_m, \beta_m\rangle_{K_m}$ for $\alpha, \beta \in \mathbb{R}^{NM}$.

For simplicity we rewrite the above problem as

$$\underset{\alpha\in\mathbb{R}^{NM},b\in\mathbb{R}}{\text{minimize}} \quad \underbrace{L(\bar{K}\alpha + b\mathbf{1}) + \phi_C(\alpha)}_{=:L_C(\alpha,b)}, \tag{7}$$

where $\bar{K} = (K_1, \ldots, K_M) \in \mathbb{R}^{N\times NM}$, $\mathbf{1} = (1, \ldots, 1)^\top$ and

$$L(z) = \sum_{i=1}^N \ell(y_i, z_i),$$

$$\phi_C(\alpha) = \sum_{m=1}^M \phi_C^{(m)}(\alpha_m) = C\sum_{m=1}^M \|\alpha_m\|_{K_m}.$$

3 A dual augmented-Lagrangian method for MKL

In this section, we first present an extension of dual augmented-Lagrangian (DAL) algorithm to kernel-learning problem through a new approach based on the proximal minimization framework (Rockafellar 1976). Second, assuming that the loss function is twice differentiable, we discuss how we can compute each minimization step efficiently in Sect. 3.3. Finally, the method is extended to the situation where the loss function is not differentiable in Sect. 3.4.

3.1 MKL optimization via proximal minimization

Starting from some initial solution $(\alpha^{(1)}, b^{(1)})$, the proximal minimization algorithm (Rockafellar 1976) iteratively minimizes the objective (7) together with proximity terms as follows:

$$(\alpha^{(t+1)}, b^{(t+1)}) = \underset{\alpha\in\mathbb{R}^{MN},b\in\mathbb{R}}{\text{argmin}} \left(L_C(\alpha, b) + \frac{1}{2\gamma^{(t)}}\left(\|\alpha - \alpha^{(t)}\|_{\bar{K}}^2 + (b - b^{(t)})^2\right)\right), \tag{8}$$

where $0 < \gamma^{(1)} \le \gamma^{(2)} \le \gamma^{(3)} \le \cdots$ is a nondecreasing sequence of *proximity parameters*[2] and $(\alpha^{(t)}, b^{(t)})$ is an approximate minimizer at the tth iteration; L_C is the (regularized) objective function (7). The last two terms in the right-hand side are *proximity terms* that tries

[2]Typically we exponentially increase $\gamma^{(t)}$, e.g. $\gamma^{(t)} = 2^t$. In practice, we can use different values of proximity parameters for each variable (e.g. use $\gamma_m^{(t)}$ for α_m and $\gamma_b^{(t)}$ for b) and choose γ adaptively depending on the scales of the variables.

to keep the next solution $(\alpha^{(t+1)}, b^{(t+1)})$ close to the current solution $(\alpha^{(t)}, b^{(t)})$. Thus, we call the minimization problem (8) a proximal MKL problem. Solving the proximal MKL problem (8) seems as difficult as solving the original MKL problem in the primal. However, when we consider the dual problems, solving the dual of proximal MKL problem (8) is a *smooth* minimization problem and can be minimized efficiently, whereas solving the dual of the original MKL problem (7) is a non-smooth minimization problem and not necessarily easier than solving the primal.

The update equation can be interpreted as an *implicit* gradient method on the objective function $L_C(\alpha, b)$. In fact, by taking the subgradient of the update equation (8) and equating it to zero, we have

$$\alpha_m^{(t+1)} \in \alpha_m^{(t)} - \gamma^{(t)} K_m^{-1} \partial_{\alpha_m} L_C(\alpha^{(t+1)}, b^{(t+1)}),$$
$$b^{(t+1)} \in b^{(t)} - \gamma^{(t)} \partial_b L_C(\alpha^{(t+1)}, b^{(t+1)}).$$

This implies the tth update step is a subgradient of the original objective function L_C at the *next* solution $(\alpha_m^{(t+1)}, b^{(t+1)})$.

The super-linear convergence of proximal minimization algorithm (Rockafellar 1976; Bertsekas 1982; Tomioka et al. 2011) can also be extended to kernel-based learning setting as in the following theorem.

Theorem 1 *Suppose the problem* (7) *has a unique*[3] *optimal solution* α^*, b^*, *and there exist a scalar* $\sigma > 0$ *and a* δ*-neighborhood* $(\delta > 0)$ *of the optimal solution such that*

$$L_C(\alpha, b) \geq L_C(\alpha^*, b^*) + \sigma(\|\alpha - \alpha^*\|_{\bar{K}}^2 + (b - b^*)^2), \tag{9}$$

for all $(\alpha, b) \in \mathbb{R}^{MN} \times \mathbb{R}$ *satisfying* $\|\alpha - \alpha^*\|_{\bar{K}}^2 + (b - b^*)^2 \leq \delta^2$. *Then for all sufficiently large* t *we have*

$$\frac{\|\alpha^{(t+1)} - \alpha^*\|_{\bar{K}}^2 + (b^{(t+1)} - b^*)^2}{\|\alpha^{(t)} - \alpha^*\|_{\bar{K}}^2 + (b^{(t)} - b^*)^2} \leq \frac{1}{(1 + \sigma\gamma^{(t)})^2}.$$

Therefore if $\lim_{t\to\infty} \gamma^{(t)} = \infty$, *the solution* $(\alpha^{(t)}, b^{(t)})$ *converges to the optimal solution super-linearly.*

Proof The proof is given in Appendix. □

3.2 Derivation of SpicyMKL

Although directly minimizing the proximal MKL problem (8) is not a trivial task, its dual problem can efficiently be solved. Once we solve the dual of the proximal minimization update (8), we can update the primal variables $(\alpha^{(t)}, b^{(t)})$. The resulting iteration can be written as follows:

$$\rho^{(t)} := \operatorname*{argmin}_{\rho \in \mathbb{R}^N} \varphi_{\gamma^{(t)}}(\rho; \alpha^{(t)}, b^{(t)}), \tag{10}$$

$$\alpha_m^{(t+1)} = \operatorname{prox}(\alpha_m^{(t)} + \gamma^{(t)} \rho^{(t)} | \phi_{\gamma^{(t)} C}^{(m)}) \quad (m = 1, \ldots, M), \tag{11}$$

[3]The uniqueness of the optimal solution is just for simplicity. The result can be generalized for a compact optimal solution set (see Bertsekas 1982).

$$b^{(t+1)} = b^{(t)} + \gamma^{(t)} \sum_{i=1}^{N} \rho_i^{(t)}, \tag{12}$$

where φ_γ is the dual objective, which we derive in the sequel, and the proximity operator $\mathrm{prox}(\cdot|\phi_C^{(m)})$ corresponding to the regularizer $\phi_C^{(m)}$ is defined as follows:

$$\begin{aligned}\mathrm{prox}(v_m|\phi_C^{(m)}) &= \operatorname*{argmin}_{v'_m \in \mathbb{R}^N} \left(\phi_C^{(m)}(v'_m) + \frac{1}{2}\|v'_m - v_m\|_{K_m}^2 \right) \\ &= \begin{cases} 0 & (\text{if } \|v_m\|_{K_m} \le C), \\ \frac{\|v_m\|_{K_m} - C}{\|v_m\|_{K_m}} v_m & (\text{otherwise}). \end{cases}\end{aligned} \tag{13}$$

The above operation is known as the soft-thresholding function in the field of sparse estimation (see Figueiredo and Nowak 2003; Daubechies et al. 2004) and it has been applied to MKL earlier in Mosci et al. (2008). Intuitively speaking, it thresholds a vector smaller than C in norm to zero but also shrinks a vector whose norm is larger than C so that the transition at $\|v_m\|_{K_m} = C$ is continuous (soft); see Fig. 1 for a one dimensional illustration.

At every iteration we minimize the inner objective φ_γ (the dual of the proximal MKL problem (8)), and use the minimizer $\rho^{(t)}$ to update the primal variables $(\alpha^{(t)}, b^{(t)})$. The overall algorithm is shown in Table 1.

Quick overview of the derivation of the iteration (10)–(12) Consider the following Lagrangian of the proximal MKL problem (8):

$$\mathcal{L} = L(z) + \phi_C(\alpha) + \frac{\|\alpha - \alpha^{(t)}\|_{\bar{K}}^2}{2\gamma^{(t)}} + \frac{(b - b^{(t)})^2}{2\gamma^{(t)}} + \rho^\top (z - \bar{K}\alpha - b\mathbf{1}), \tag{14}$$

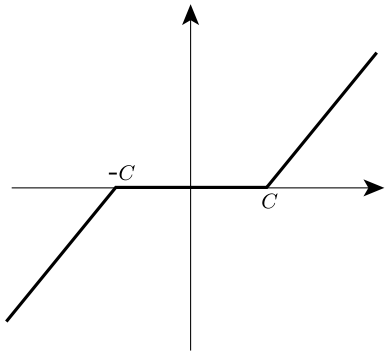


Fig. 1 Illustration of soft thresholding function $\mathrm{prox}(\cdot|\phi_C^{(m)})$

Table 1 Algorithm of SpicyMKL for block 1-norm MKL

1. Choose a sequence $\gamma^{(t)} \to \infty$ as $t \to \infty$.
2. Minimize the augmented Lagrangian with respect to ρ:
   $\rho^{(t)} = \mathrm{argmin}_\rho (L^*(-\rho) + \sum_m \frac{\|\mathrm{prox}(\alpha_m^{(t)} + \gamma^{(t)}\rho^{(t)}|\phi^{(m)}_{\gamma^{(t)}C})\|_{K_m}^2}{2\gamma^{(t)}} + \frac{(b^{(t)} + \gamma^{(t)}\sum_i \rho_i)^2}{2\gamma^{(t)}})$.
3. Update $\alpha_m^{(t+1)} \leftarrow \mathrm{prox}(\alpha_m^{(t)} + \gamma^{(t)}\rho^{(t)}|\phi^{(m)}_{\gamma^{(t)}C})$, $b^{(t+1)} \leftarrow b^{(t)} + \gamma^{(t)}\sum_i \rho_i^{(t)}$.
4. Repeat 2. and 3. until the stopping criterion is satisfied.

where $\rho \in \mathbb{R}^N$ is the Lagrangian multiplier corresponding to the equality constraint $z = \bar{K}\alpha + b\mathbf{1}$. The vector $\rho^{(t)}$ in the first step (10) is the optimal Lagrangian multiplier that *maximizes* the dual of the proximal MKL problem (8) (see (25)). The remaining steps (11)–(12) can be obtained by minimizing the above Lagrangian with respect to α and b for $\rho = \rho^{(t)}$, respectively.

Detailed derivation of the iteration (10)–(12) Let's consider the constraint reformulation of the proximal MKL problem (8) as follows:

$$\begin{aligned} &\underset{\substack{\alpha\in\mathbb{R}^{MN}, b\in\mathbb{R} \\ z\in\mathbb{R}^N}}{\text{minimize}} \quad L(z) + \phi_C(\alpha) + \frac{\|\alpha - \alpha^{(t)}\|_{\bar{K}}^2}{2\gamma^{(t)}} + \frac{(b - b^{(t)})^2}{2\gamma^{(t)}}, \\ &\text{subject to} \quad z = \bar{K}\alpha + b\mathbf{1}. \end{aligned}$$

The Lagrangian of the above constrained minimization problem can be written as in (14).

The dual problem can be derived by minimizing the Lagrangian (14) with respect to the primal variables (z, α, b). See (25) for the final expression. Note that the minimization is separable into minimization with respect to z (15), α (20), and b (24).

First, minimizing the Lagrangian with respect to z gives

$$\min_{z\in\mathbb{R}^N} \left(L(z) + \rho^\top z\right) = -L^*(-\rho), \tag{15}$$

where L^* is the convex conjugate of the loss function L as follows:

$$\begin{aligned} L^*(-\rho) &:= \sup_{z\in\mathbb{R}^N} \left((-\rho)^\top z - L(z)\right) \\ &= \sum_{i=1}^N (-\rho_i z_i - \ell(y_i, z_i)) \\ &= \sum_{i=1}^N \ell^*(y_i, -\rho_i), \end{aligned}$$

where ℓ^* is the convex conjugate of the loss ℓ with respect to the second argument.

For example, the conjugate loss ℓ_L^* for the logistic loss ℓ_L is the negative entropy function as follows:

$$\ell_L^*(y, -\rho) = \begin{cases} (y\rho)\log(y\rho) + (1 - y\rho)\log(1 - y\rho) & (\text{if } 0 \le y\rho \le 1), \\ +\infty & (\text{otherwise}). \end{cases} \tag{16}$$

The conjugate loss ℓ_H^* for the hinge loss ℓ_H is given as follows:

$$\ell_H^*(y, -\rho) = \begin{cases} -y\rho & (\text{if } 0 \le y\rho \le 1), \\ +\infty & (\text{otherwise}). \end{cases} \tag{17}$$

Second, minimizing the Lagrangian (14) with respect to α, we obtain

$$\min_{\alpha\in\mathbb{R}^{MN}} \left(\phi_C(\alpha) + \frac{\|\alpha - \alpha^{(t)}\|_{\bar{K}}^2}{2\gamma^{(t)}} - \rho^\top \bar{K}\alpha\right)$$

$$= \min_{\alpha \in \mathbb{R}^{MN}} \left(\phi_C(\alpha) + \frac{\|\alpha - \alpha^{(t)} - \gamma^{(t)}\rho\|_{\tilde{K}}^2}{2\gamma^{(t)}} \right) - \frac{\|\alpha^{(t)} + \gamma^{(t)}\rho\|_{\tilde{K}}^2}{2\gamma^{(t)}} + \text{const} \tag{18}$$

$$\overset{(21),(23)}{=} -\frac{1}{\gamma^{(t)}} \min_{u \in \mathbb{R}^{MN}} \left(\delta_{\gamma^{(t)}C}(u) + \frac{\|u - \alpha^{(t)} - \gamma^{(t)}\rho\|_{\tilde{K}}^2}{2} \right) + \text{const} \tag{19}$$

$$\overset{(22)}{=} -\frac{1}{\gamma^{(t)}} \sum_{m=1}^{M} \Phi^{(m)}_{\gamma^{(t)}C}(\alpha_m + \gamma^{(t)}\rho) + \text{const}, \tag{20}$$

where const is a term that only depends on $\alpha^{(t)}$ and $\gamma^{(t)}$; the function $\delta_C : \mathbb{R}^{MN} \to \mathbb{R}$ is the convex conjugate of the regularization term $\phi_C^{(m)}(\alpha_m) = C\|\alpha_m\|_{K_m}$ as follows:

$$\delta_C(u) := \sum_{m=1}^{M} \delta_C^{(m)}(u_m),$$

where

$$\begin{aligned} \delta_C^{(m)}(u_m) &:= \sup_{\alpha_m \in \mathbb{R}^N} \left(\langle u_m, \alpha_m \rangle_{K_m} - \phi_C^{(m)}(\alpha_m) \right) \\ &= \sup_{\alpha_m \in \mathbb{R}^N} \left(\|u_m\|_{K_m} \|\alpha_m\|_{K_m} - C\|\alpha_m\|_{K_m} \right) \\ &= \begin{cases} 0 & (\text{if } \|u_m\|_{K_m} \le C), \\ +\infty & (\text{otherwise}). \end{cases} \end{aligned} \tag{21}$$

See Fig. 2 for a one dimensional illustration of the conjugate regularizer $\delta_C^{(m)}$. In addition, the function $\Phi_C^{(m)}$ in the last line is *Moreau's envelope function* (see Fig. 2):

$$\begin{aligned} \Phi_C^{(m)}(v_m) &:= \min_{v'_m \in \mathbb{R}^N} \left(\delta_C^{(m)}(v'_m) + \frac{1}{2}\|v'_m - v_m\|_{K_m}^2 \right), \\ &= \left\| \text{prox}(v_m | \phi_C^{(m)}) \right\|_{K_m}^2. \end{aligned} \tag{22}$$

See (13) and Fig. 1 for the definition of the soft-threshold operation $\text{prox}(\cdot|\phi_C^{(m)})$. Furthermore, we used the following proposition and some algebra to derive (19) from (18).

Proposition 1 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a closed proper convex function and f^* be the convex conjugate of f defined as*

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \left(\langle y, x \rangle_K - f(x) \right),$$

where $K \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix. Then

$$\min_{x \in \mathbb{R}^n} \left(f(x) + \frac{\|x - z\|_K^2}{2} \right) + \min_{y \in \mathbb{R}^n} \left(f^*(y) + \frac{\|y - z\|_K^2}{2} \right) = \frac{\|z\|_K^2}{2}. \tag{23}$$

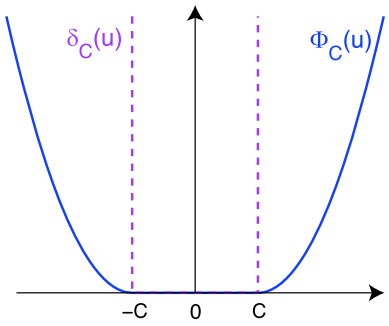


Fig. 2 Comparison of the conjugate regularizer $\delta_C^{(m)}$ and the corresponding Moreau's envelope function $\Phi_C^{(m)}$ in one dimension. The conjugate regularizer $\delta_C^{(m)}$ (21) is nondifferentiable at the boundary of its domain, whereas its envelope function $\Phi_C^{(m)}$ is smooth

Proof It is a straightforward generalization of Moreau's theorem. See Rockafellar (1970, Theorem 31.5) and Tomioka et al. (2011). □

Finally, minimizing the Lagrangian (14) with respect to b, we obtain

$$\min_{b\in\mathbb{R}}\left(\frac{(b-b^{(t)})^2}{2\gamma^{(t)}}-b\rho^\top\mathbf{1}\right)=-\frac{(b^{(t)}+\gamma^{(t)}\rho^\top\mathbf{1})^2}{2\gamma^{(t)}}+\text{const}, \tag{24}$$

where const is a term that only depends on $b^{(t)}$ and $\gamma^{(t)}$.

Combining (15), (20), and (24), the dual of the proximal MKL problem (8) can be obtained as follows:

$$\begin{aligned}\underset{\rho\in\mathbb{R}^N}{\text{maximize}}\ -L^*(-\rho)-\frac{1}{2\gamma^{(t)}}\sum_{m=1}^{M}\left\|\text{prox}(\alpha_m^{(t)}+\gamma^{(t)}\rho|\phi^{(m)}_{\gamma^{(t)}C})\right\|^2_{K_m}\\ -\frac{1}{2\gamma^{(t)}}\left(b^{(t)}+\gamma^{(t)}\sum_{i=1}^{N}\rho_i\right)^2,\end{aligned} \tag{25}$$

where the constant terms are ignored. We denote the maximand in the above dual problem by $-\varphi_{\gamma^{(t)}}(\rho;\alpha^{(t)},b^{(t)})$; see (10).

3.3 Minimizing the inner objective function

The inner objective function (25) that we need to minimize at every iteration is convex and *differentiable* when L^* is differentiable. In fact, the gradient and the Hessian of the inner objective φ_γ can be written as follows:

$$\nabla_\rho\varphi_\gamma(\rho;\alpha,b)=-\nabla_\rho L^*(-\rho)+\left(b+\gamma\sum_i\rho_i\right)\mathbf{1}+\sum_{m\in M_+}K_m\text{prox}\left(\alpha_m+\gamma\rho|\phi^{(m)}_{\gamma C}\right), \tag{26}$$

$$\nabla^2_\rho\varphi_\gamma(\rho;\alpha,b)=\nabla^2_\rho L^*(-\rho)+\gamma\mathbf{1}\mathbf{1}^\top+\gamma\sum_{m\in M_+}\left((1-q_m)K_m+q_mK_m\tilde{v}_m\tilde{v}_m^\top K_m\right), \tag{27}$$

where M_+ is the set of indices corresponding to the *active kernels*; i.e., $M_+=\{m\in\{1,\ldots,M\}\mid\|\alpha_m+\gamma\rho\|_{K_m}>\gamma C\}$; for $m\in M_+$, a scalar q_m and a vector $\tilde{v}_m\in\mathbb{R}^N$ are defined as $q_m:=\frac{\gamma C}{\|\alpha_m+\gamma\rho\|_{K_m}}$ and $\tilde{v}_m:=(\alpha_m+\gamma\rho)/\|\alpha_m+\gamma\rho\|_{K_m}$.

Remark 1 The computation of the objective $\varphi_\gamma(\rho;\alpha,b)$ (25), the gradient (26), and the Hessian (27) is efficient because they require only the terms corresponding to the *active kernels* $\{k_m \mid m \in M_+\}$.

The above sparsity, which makes the proposed algorithm efficient, comes from our dual formulation. By taking the dual, there appears *flat* region (see Fig. 2); i.e. the region $\{\rho \mid \mathrm{prox}(\alpha_m^{(t)} + \gamma^{(t)}\rho|\phi^{(m)}_{\gamma^{(t)}C}) = 0\}$ is not a single point but has its interior.

Since the inner objective function (25) is differentiable, and the sparsity of the intermediate solution can be exploited to evaluate the gradient and Hessian of the inner objective, the inner minimization can efficiently be carried out. We call the proposed algorithm *Sparse Iterative MKL (SpicyMKL)*. The update equations (11)–(12) exactly correspond to the augmented Lagrangian method for the dual of the problem (7) (see Tomioka and Sugiyama 2009) but derived in more general way using the techniques from Rockafellar (1976).

We use the Newton method with line search for the minimization of the inner objective (25). The line search is used to keep ρ inside the domain of the dual loss L^*. This works when the gradient of the dual loss L^* is unbounded at the boundary of its domain, for example the logistic loss (16), in which case the minimum is never attained at the boundary. On the other hand, for the hinge loss (17), the solution lies typically at the boundary of the domain $0 \le y\rho \le 1$. This situation is handled separately in the next subsection.

3.4 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. First we explain this in the case of hinge loss for classification. Next we discuss a generalization to other cases.

3.4.1 Optimization for hinge loss

Here we explain how the augmented Lagrangian technique is applied to the hinge loss. To this end, we introduce two sets of slack variables $\xi = (\xi_1,\dots,\xi_N)^\top \ge 0, \zeta = (\zeta_1,\dots,\zeta_N)^\top \ge 0$ as in standard SVM literature (see e.g., Schölkopf and Smola 2002). The basic update equation (8) is rewritten as follows:[4]

$$
\begin{aligned}
&(\alpha^{(t+1)}, b^{(t+1)}, \xi^{(t+1)}, \zeta^{(t+1)}) \\
&\quad = \operatorname*{argmin}_{\substack{\alpha\in\mathbb{R}^{(MN)}, b\in\mathbb{R} \\ \xi\in\mathbb{R}_+^N, \zeta\in\mathbb{R}_+^N}} \left\{ H_C(\alpha,b,\xi,\zeta) + \frac{1}{2\gamma^{(t)}}\Big(\|\alpha-\alpha^{(t)}\|_{\tilde{K}}^2 + (b-b^{(t)})^2 \right. \\
&\qquad\qquad \left. + \|\xi-\xi^{(t)}\|^2 + \|\zeta-\zeta^{(t)}\|^2 \Big) \right\},
\end{aligned}
$$

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

where

$$H_C(\alpha, b, \xi, \zeta) = \begin{cases} \sum_{i=1}^N \xi_i + \phi_C(\alpha) & (\text{if } y_i((\sum_{m=1}^M K_m \alpha_m)_i + b) = 1 - \xi_i + \zeta_i,\ \xi_i \geq 0,\ \zeta_i \geq 0, \forall i), \\ +\infty & (\text{otherwise}). \end{cases}$$

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

$$H_C(\alpha, b, \xi, \zeta) = \max_{\rho \in \mathbb{R}^N, u \in \mathbb{R}^{MN}} \left\{ -\sum_{i=1}^N (-y_i \rho_i) - b \sum_{i=1}^N \rho_i - \sum_{m=1}^M \delta_C^m(u_m) - \sum_{m=1}^M \langle \alpha_m, \rho - u_m \rangle_{K_m} + \sum_{i=1}^N \xi_i (1 - y_i \rho_i) + \sum_{i=1}^N \zeta_i (y_i \rho_i) \right\}.$$

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 Sect. 3.2). Finally we obtain the following update equations.

$$\alpha_m^{(t+1)} = \mathrm{prox}(\alpha_m^{(t)} + \gamma^{(t)} \rho^{(t)} | \phi^{(m)}_{\gamma^{(t)} C}), \tag{28}$$

$$b^{(t+1)} = b^{(t)} + \gamma^{(t)} \sum_{i=1}^N \rho_i^{(t)}, \tag{29}$$

$$\xi_i^{(t+1)} = \max(0, \xi_i^{(t)} - \gamma_\xi^{(t)} (1 - y_i \rho_i^{(t)})), \tag{30}$$

$$\zeta_i^{(t+1)} = \max(0, \zeta_i^{(t)} - \gamma_\zeta^{(t)} y_i \rho_i^{(t)}), \tag{31}$$

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

$$\varphi_\gamma(\rho; \alpha, b, \xi, \zeta) = -\sum_{i=1}^N y_i \rho_i + \frac{1}{2\gamma} \left(b + \gamma \sum_{i=1}^N \rho_i \right)^2 + \sum_{m=1}^M \frac{1}{2\gamma} \| \mathrm{prox}(\alpha_m + \gamma \rho | \phi^{(m)}_{\gamma C}) \| + \frac{1}{2\gamma} \sum_{i=1}^N \max(0, \xi_i - \gamma(1 - y_i \rho_i))^2 + \frac{1}{2\gamma} \sum_{i=1}^N \max(0, \zeta_i - \gamma y_i \rho_i)^2, \tag{32}$$

and $\gamma \in \mathbb{R}_+$. The gradient and the Hessian of φ_γ with respect to ρ can be obtained in a similar way to (26) and (27). Thus we use the Newton method for the minimization of (32). The overall algorithm is analogous to Table 1 with update equations (28)–(32).

3.4.2 Optimization for general loss functions with constraints

Here we generalize the above argument to a broader class of loss functions. We assume that the dual of the loss function can be written by using twice differentiable convex functions ℓ_0 and h_j as

$$\ell^*(y_i, \rho_i) = \min_{\tilde{\rho}_i} \{\ell_0^*(y_i, (\rho_i, \tilde{\rho}_i)) \mid h_j(y_i, (\rho_i, \tilde{\rho}_i)) \le 0 \ (1 \le j \le B)\}, \tag{33}$$

where $\tilde{\rho}_i \in \mathbb{R}^{B'}$ is an auxiliary variable and we set the right hand side as ∞ if there is no feasible $\tilde{\rho}_i$ satisfying $h_j(y_i, (\rho_i, \tilde{\rho}_i)) \le 0$ $(1 \le j \le B)$. An example is ϵ-sensitive loss for regression that is defined as

$$\ell(y, f) = \begin{cases} |f - y| - \epsilon & (\text{if } |f - y| \ge \epsilon), \\ 0 & (\text{otherwise}). \end{cases}$$

By simple calculation, the dual function of the ϵ-sensitive loss is given as

$$\ell^*(y, \rho) = \begin{cases} y\rho + |\rho|\epsilon & (\text{if } |\rho| \le 1), \\ \infty & (\text{otherwise}). \end{cases} \tag{34}$$

This is not differentiable, but is written as

$$\ell^*(y, \rho) = \begin{cases} \min_{\tilde{\rho} \in \mathbb{R}} \{y\rho + (\rho + 2\tilde{\rho})\epsilon \mid -\tilde{\rho} \le 0, -\rho - \tilde{\rho} \le 0\} & (\text{if } -1 \le \rho \le 1), \\ \infty & (\text{otherwise}). \end{cases} \tag{35}$$

Thus in this example, $\ell_0^*(y, (\rho, \tilde{\rho})) = y\rho + (\rho + 2\tilde{\rho})$, $h_1(y, (\rho, \tilde{\rho})) = \rho - 1$, $h_2(y, (\rho, \tilde{\rho})) = -1 - \rho$, $h_3(y, (\rho, \tilde{\rho})) = -\tilde{\rho}$, and $h_4(y, (\rho, \tilde{\rho})) = -\rho - \tilde{\rho}$ in the formulation of (33).

The update equation becomes as following:

$$(\alpha^{(t+1)}, b^{(t+1)}, \xi^{(t+1)}) \tag{36}$$

$$= \operatorname*{argmin}_{\substack{\alpha \in \mathbb{R}^{(MN)}, b \in \mathbb{R} \\ \xi \in \mathbb{R}_+^{NB}}} \left\{ H_C(\alpha, b, \xi) + \frac{1}{2\gamma^{(t)}} \left(\|\alpha - \alpha^{(t)}\|_{\bar{K}}^2 + (b - b^{(t)})^2 + \|\xi - \xi^{(t)}\|^2 \right) \right\}, \tag{37}$$

where H_C is expressed in terms of maximum over $\rho \in \mathbb{R}^N$, $\tilde{\rho} \in \mathbb{R}^{NB'}$, $u \in \mathbb{R}^{MN}$ as follows:

$$H_C(\alpha, b, \xi)$$

$$= \max_{\rho \in \mathbb{R}^N, \tilde{\rho} \in \mathbb{R}^{NB'}, u \in \mathbb{R}^{MN}} \left\{ -\sum_{i=1}^N \ell_0^*(y_i, (-\rho_i, \tilde{\rho}_i)) - \sum_{m=1}^M \delta_C^m(u_m) - \sum_{m=1}^M \langle \alpha_m, \rho - u_m \rangle_{K_m} \right.$$

$$\left. - b \sum_{i=1}^N \rho_i - \sum_{i=1}^N \sum_{j=1}^B \xi_{i,j} h_j(y_i, (-\rho_i, \tilde{\rho}_i)) \right\}.$$

Note that the minimum of $H_C(\alpha, b, \xi)$ with respect to ξ is the primal objective function $L(\bar{K}\alpha + b\mathbf{1}) + \phi_C(\alpha)$ because by exchanging min and max we have

$$
\begin{aligned}
&\min_{\xi\in\mathbb{R}_+^{NB}} H_C(\alpha,b,\xi)\\
&= \max_{\substack{\rho\in\mathbb{R}^N\\ \tilde\rho\in\mathbb{R}^{NB'},u\in\mathbb{R}^{MN}}} \inf_{\xi\in\mathbb{R}_+^{NB}} \left\{ -\sum_{i=1}^N \ell_0^*(y_i,(-\rho_i,\tilde\rho_i)) - \sum_{m=1}^M \delta_C^m(u_m) - \sum_{m=1}^M \langle \alpha_m, \rho - u_m\rangle_{K_m} \right.\\
&\qquad\qquad \left. - b\sum_{i=1}^N \rho_i - \sum_{i=1}^N\sum_{j=1}^B \xi_{i,j} h_j(y_i,(-\rho_i,\tilde\rho_i)) \right\}\\
&= \max_{\rho\in\mathbb{R}^N\tilde\rho\in\mathbb{R}^{NB'}} \left\{ -\sum_{i=1}^N \ell_0^*(y_i,(-\rho_i,\tilde\rho_i)) + \rho^\top(\bar K\alpha + b\mathbf{1}) \mid h_j(y_i,(-\rho_i,\tilde\rho_i)) \le 0\ (\forall i,j) \right\}\\
&\qquad + \max_{u\in\mathbb{R}^{MN}} \left\{ -\sum_{m=1}^M \delta_C^m(u_m) + \sum_{m=1}^M \langle \alpha_m, u_m\rangle_{K_m} \right\}\\
&= L(\bar K\alpha + b\mathbf{1}) + \phi_C(\alpha).
\end{aligned}
$$

However it is difficult to directly optimize the primal objective. Therefore we add the penalty term as in (37) and solve its dual problem. We exchange the order of minimization and maximization as in the last section (see also Sect. 3.2) and remove α, b, ξ, and u_m by explicitly minimizing or maximizing over them. Finally we obtain the following update equations.

$$
\alpha_m^{(t+1)} = \mathrm{prox}(\alpha_m^{(t)} + \gamma^{(t)}\rho^{(t)} | \phi_{\gamma^{(t)}C}^{(m)}), \tag{38}
$$

$$
b^{(t+1)} = b^{(t)} + \gamma^{(t)} \sum_{i=1}^N \rho_i^{(t)}, \tag{39}
$$

$$
\xi_{i,j}^{(t+1)} = \max(0, \xi_{i,j}^{(t)} - \gamma^{(t)} h_j(y_i,(-\rho_i^{(t)},\tilde\rho_i^{(t)}))), \tag{40}
$$

where $(\rho^{(t)},\tilde\rho^{(t)}) \in \mathbb{R}^N \times \mathbb{R}^{NB'}$ is the minimizer of the function $\varphi_{\gamma^{(t)}}(\rho,\tilde\rho;\alpha^{(t)},b^{(t)},\xi^{(t)})$ defined as follows:

$$
\begin{aligned}
&\varphi_\gamma(\rho,\tilde\rho;\alpha,b,\xi)\\
&\quad = \sum_{i=1}^N \ell_0^*(y_i,(-\rho_i,\tilde\rho_i)) + \frac{1}{2\gamma}\left(b + \gamma\sum_{i=1}^N \rho_i\right)^2 + \sum_{m=1}^M \frac{1}{2\gamma}\|\mathrm{prox}(\alpha_m + \gamma\rho|\phi_{\gamma C}^{(m)})\|_{K_m}^2\\
&\qquad + \frac{1}{2\gamma}\sum_{i=1}^N\sum_{j=1}^B \max(0,\xi_{i,j} - \gamma h_j(y_i,(-\rho_i,\tilde\rho_i)))^2,
\end{aligned} \tag{41}
$$

and $\gamma \in \mathbb{R}_+$. The gradient and the Hessian of φ_γ with respect to $(\rho,\tilde\rho)$ can be obtained in a similar way to (26) and (27). Thus we use the Newton method for the minimization of $\varphi_{\gamma^{(t)}}$ (32). The overall algorithm is summarized in Table 2.

Table 2 Algorithm of SpicyMKL for a general regularization term and a general loss function

1. Choose a sequence $\gamma^{(t)} \to \infty$ as $t \to \infty$.
2. Minimize the augmented Lagrangian with respect to ρ and $\tilde{\rho}$:

$$(\rho^{(t)}, \tilde{\rho}^{(t)}) = \mathrm{argmin}_{\rho,\tilde{\rho}} \Big(\sum_i \ell_0^*(y_i, (-\rho_i, \tilde{\rho}_i)) + \frac{1}{2\gamma^{(t)}} \sum_m \|\mathrm{prox}(\alpha_m^{(t)} + \gamma^{(t)} \rho | \phi^{(m)}_{\gamma^{(t)}C})\|^2_{K_m} + \frac{1}{2\gamma^{(t)}} \sum_{i=1}^N \sum_{j=1}^B \max(0, \xi_{i,j}^{(t)} - \gamma h_j(y_i, (-\rho_i, \tilde{\rho}_i)))^2 + \frac{1}{2\gamma^{(t)}} (b^{(t)} + \gamma^{(t)} \sum_i \rho_i)^2\Big).$$

3. Update $\alpha_m^{(t+1)} \leftarrow \mathrm{prox}(\alpha_m^{(t)} + \gamma^{(t)} \rho^{(t)} | \phi^m_{\gamma^{(t)}C})$, $b^{(t+1)} \leftarrow b^{(t)} + \gamma^{(t)} \sum_i \rho_i^{(t)}$, $\xi_{i,j}^{(t+1)} \leftarrow \max(0, \xi_{i,j}^{(t)} - \gamma^{(t)} h_j(y_i, (\rho_i^{(t)}, \tilde{\rho}_i^{(t)})))$.
4. Repeat 2. and 3. until the stopping criterion is satisfied.

3.5 Technical details of computations

3.5.1 Back-tracking in the Newton update

We use back-tracking line search with *Armijo's rule* to find a step size of the Newton method. During the back-tracking, the computational bottle-neck is the computation of the norm $\|\rho + c\Delta\rho + \frac{\alpha_m}{\gamma_m}\|_{K_m}$ $(m = 1, \ldots, M)$ where $\Delta\rho$ is the Newton update direction and $0 < c \leq 1$ is a step size. However, the above computation is necessary only for the active kernels M_+; for example, the active kernels M_+ is included in $\{m \mid \|\rho + \frac{\alpha_m}{\gamma_m}\|_{K_m} > C$ or $\|\rho + \Delta\rho + \frac{\alpha_m}{\gamma_m}\|_{K_m} > C\}$ for block 1-norm MKL and $\{m \mid \|\rho\|_{K_m} > (1-\lambda)C$ or $\|\rho + \Delta\rho\|_{K_m} > (1-\lambda)C\}$ for Elastic-net MKL because of the convexity of $\|\cdot\|_{K_m}$. This reduces the computation time considerably.

3.5.2 Parallelization in the computation of $\|\alpha_m^{(t)} + \frac{\rho^{(t)}}{\gamma_m^{(t)}}\|_{K_m}$

The computation of $\|\alpha_m^{(t)} + \frac{\rho^{(t)}}{\gamma_m^{(t)}}\|_{K_m}$ for $m = 1, \ldots, M$ is necessary at every outer iteration. Thus when the number of kernels is large, its cost cannot be ignored. Fortunately, the computation with respect to the m-th kernel is independent of the others and the outer loop is easily parallelizable. In our experiments, we implemented the parallel computing by OpenMP in mex-files of our Matlab® code. We observed twenty or thirty percent improvement of overall computational time by the parallelization.

4 Generalized block-norm formulation

Motivated by the recent interest in non-sparse regularization for MKL, we consider a generalized block norm formulation in this section. The proposed formulation uses a concave function g and it subsumes previously proposed MKL models, such as ℓ_p-norm MKL (Kloft et al. 2009) and Elastic-net MKL (Tomioka and Suzuki 2009), as different choices of the concave function g. Moreover, using a convex upper-bounding technique (Palmer et al. 2006) we show that the proposed formulation can be written also as a Tikhonov regularization problem (3). Thus the solution can be easily mapped to kernel weights as in previous approaches. We extend SpicyMKL algorithm for the generalized formulation in Sect. 4.2. Furthermore we preset an efficient one-step optimization procedure for Elastic-net MKL and ℓ_p-norm MKL in Sect. 4.3.

4.1 Proposed formulation

We consider the following generalized block-norm formulation:

$$\underset{f_1\in\mathcal{H}_1,\ldots,f_M\in\mathcal{H}_M,b\in\mathbb{R}}{\text{minimize}} \sum_{i=1}^{N}\ell\left(y_i,\sum_{m=1}^{M}f_m(x_i)+b\right)+C\sum_{m=1}^{M}g(\|f_m\|_{\mathcal{H}_m}^2), \tag{42}$$

where g is a non-decreasing concave function defined on the nonnegative reals, and we assume that $\tilde{g}(x)=g(x^2)$ is a convex function of x. For example, taking $g(x)=\sqrt{x}$ gives the block 1-norm MKL in (4) and taking $g(x)=x^{q/2}/q$ gives the block q-norm MKL (Kloft et al. 2009; Nath et al. 2009). Moreover by choosing $g(x)=(1-\lambda)\sqrt{x}+\frac{\lambda}{2}x$, we obtain the following Elastic-net MKL (Tomioka and Suzuki 2009):

$$\begin{aligned}\underset{f_1\in\mathcal{H}_1,\ldots,f_M\in\mathcal{H}_M,b\in\mathbb{R}}{\text{minimize}} &\sum_{i=1}^{N}\ell\left(y_i,\sum_{m=1}^{M}f_m(x_i)+b\right)\\ &+C\sum_{m=1}^{M}\left((1-\lambda)\|f_m\|_{\mathcal{H}_m}+\frac{\lambda}{2}\|f_m\|_{\mathcal{H}_m}^2\right),\end{aligned} \tag{43}$$

which reduces to the block 1-norm regularization (4) for $\lambda=0$ and the uniform-weight combination ($d_m=1$ $(\forall m)$ in (2)) for $\lambda=1$.

Note that the generalized regularization term in the above formulation (42) is separable into each kernel component; thus it can be more efficiently handled than the squared regularization used in previous studies (Kloft et al. 2009, 2010).

The first question we need to answer is how the generalized block-norm formulation (42) is related to the Tikhonov regularization formulation (3). This can be shown using a convex upper-bounding technique; see Palmer et al. (2006). For any given concave function g, the *concave conjugate* g^* of g is defined as follows:

$$g^*(y)=\inf_{x\geq 0}\left(xy-g(x)\right).$$

The definition of concave conjugate immediately implies that the following inequality is true:

$$g\left(\|f_m\|_{\mathcal{H}_m}^2\right)\leq\frac{\|f_m\|_{\mathcal{H}_m}^2}{2d_m}-g^*\left(\frac{1}{2d_m}\right).$$

Note that the equality is obtained by taking $d_m=1/(2g'(\|f_m\|_{\mathcal{H}_m}^2))$, where g' is the derivative of g. Therefore, the generalized block-norm formulation (42) is equivalent to the following Tikhonov regularization problem:

$$\underset{\substack{f_1\in\mathcal{H}_1,\ldots,f_M\in\mathcal{H}_M,\\ b\in\mathbb{R},\\ d_1\geq 0,\ldots,d_M\geq 0}}{\text{minimize}} \sum_{i=1}^{N}\ell\left(y_i,\sum_{m=1}^{M}f_m(x_i)+b\right)+\frac{C}{2}\sum_{m=1}^{M}\left(\frac{\|f_m\|_{\mathcal{H}_m}^2}{d_m}+h(d_m)\right), \tag{44}$$

where we define the regularizer $h(d_m)=-2g^*(1/(2d_m))$. It is easy to obtain the regularizer h corresponding to the block 1-norm MKL, block q-norm MKL, and the Elastic-net MKL, and it is shown in Table 3. See Tomioka and Suzuki (2011) for more detailed and generalized

Table 3 Correspondence of the concave function g in (42) and the regularizer h in (44). For block q-norm MKL, the exponent q in the block norm formulation and the exponent p in the Tikhonov regularization problem correspond as $p := q/(q-2)$. $I_{[0,1]}$ denotes the indicator function of the interval $[0, 1]$; i.e., $I_{[0,1]}(x) = 0$ (if $x \in [0, 1]$), and $I_{[0,1]}(x) = \infty$ (otherwise)

MKL model	$g(x)$	$h(d_m)$
Block 1-norm MKL	$\sqrt{x}$	d_m
Block q-norm MKL	$\frac{1}{q}x^{q/2}$	$\frac{1}{p}d_m^p$
Elastic-net MKL	$(1-\lambda)\sqrt{x} + \frac{\lambda}{2}x$	$\frac{(1-\lambda)^2 d_m}{1-\lambda d_m}$
Uniform-weight MKL	$x/2$	$I_{[0,1]}(d_m)$

discussions about the relations between the regularizations on the RKHS norm $\{\|f_m\|_{\mathcal{H}_m}\}_m$ and the kernel weight $\{d_m\}_m$.

Once we obtain the optimizer $\{f_m^*\}_m$ for the generalized block-norm formulation (42), then we can easily recover the corresponding kernel weight by the relation

$$d_m = \frac{1}{2g'(\|f_m^*\|_{\mathcal{H}_m}^2)} = \frac{1}{2g'(\|\alpha_m^*\|_{K_m}^2)},$$

where α_m^* is the optimal coefficient vector corresponding to f_m^*. For the Elastic-net MKL, for example, the kernel weight is recovered as

$$d_m = \begin{cases} 0 & (\|\alpha_m^*\|_{K_m} = 0), \\ \frac{\|\alpha_m^*\|_{K_m}}{1-\lambda+\lambda\|\alpha_m^*\|_{K_m}} & (\text{otherwise}). \end{cases}$$

Note that we have a freedom of a constant multiplication for the kernel weight.

4.2 SpicyMKL for generalized block-norm formulation

Now we are ready to extend SpicyMKL algorithm to the generalized block-norm formulation (42). The original iteration (10)–(12) needs three modifications, namely the proximity operator $\mathrm{prox}(\cdot|\phi_C^{(m)})$, the conjugate regularizer $\delta_C^{(m)}$, and Moreau's envelope function $\Phi_C^{(m)}$.

First the proximity operator $\mathrm{prox}(\cdot|\phi_C^{(m)})$ is redefined using the generalized regularization term $g(\|\cdot\|_{K_m}^2)$ as follows:

$$\mathrm{prox}(v_m|\phi_C^{(m)}) := \operatorname*{argmin}_{v_m' \in \mathbb{R}^N} \left(Cg(\|v_m'\|_{K_m}^2) + \frac{1}{2}\|v_m' - v_m\|_{K_m}^2 \right).$$

Note that the above minimization reduces to a one-dimensional minimization along $v_m' = cv_m$. Let $\tilde{g}_C(x) = Cg(x^2)$. In fact, due to the Cauchy-Schwarz inequality, we have

$$\tilde{g}_C(\|v_m'\|_{K_m}) + \frac{1}{2}\|v_m' - v_m\|_{K_m}^2 \geq \tilde{g}_C(\|v_m'\|_{K_m}) + \frac{1}{2}(\|v_m'\|_{K_m} - \|v_m\|_{K_m})^2.$$

The minimum is obtained when

$$\begin{aligned} \|v_m'\| &= \mathrm{prox}(\|v_m\|_{K_m}|\tilde{g}_C) \\ &= \operatorname*{argmin}_{x \geq 0} \left(\tilde{g}_C(x) + \frac{1}{2}(x - \|v_m\|_{K_m})^2 \right). \end{aligned}$$

For example, for Elastic-net MKL, the regularizer $\tilde{\mathrm{g}}$ is written as $\tilde{g}(x) = (1-\lambda)x + \lambda x^2/2$, and the one dimensional proximity operator $\mathrm{prox}(\cdot|\tilde{g}_C)$ is obtained as follows:

$$\mathrm{prox}(x|\tilde{g}_C) = \begin{cases} 0 & (\text{if } 0 \le x \le C(1-\lambda)), \\ \frac{x - C(1-\lambda)}{C\lambda + 1} & (\text{otherwise}). \end{cases}$$

Therefore, the proximity operator $\mathrm{prox}(v_m|\phi_C^{(m)})$ is obtained as follows:

$$\mathrm{prox}(v_m|\phi_C^{(m)}) = \begin{cases} 0 & (\text{if } \|v_m\|_{K_m} \le C(1-\lambda)), \\ \frac{\|v_m\|_{K_m} - C(1-\lambda)}{(C\lambda+1)\|v_m\|_{K_m}} v_m & (\text{otherwise}). \end{cases}$$

Second, the convex conjugate of the regularizer $\delta_C^{(m)}$ is redefined as follows:

$$\begin{aligned} \delta_C^{(m)}(u_m) &:= \sup_{\alpha_m \in \mathbb{R}^N} \left(\langle u_m, \alpha_m \rangle_{K_m} - \tilde{g}_C(\|\alpha_m\|_{K_m}) \right) \\ &= \sup_{\alpha_m \in \mathbb{R}^N} \left(\|u_m\|_{K_m} \|\alpha_m\|_{K_m} - \tilde{g}_C(\|\alpha_m\|_{K_m}) \right) \\ &= \tilde{g}_C^*(\|u_m\|_{K_m}). \end{aligned} \tag{45}$$

For example, for the same Elastic-net regularizer, the convex conjugate $\tilde{g}_C^*(y)$ is obtained as follows:

$$\begin{aligned} \tilde{g}_C^*(y) &= \sup_{x \ge 0} \left(xy - C(1-\lambda)x - \frac{C\lambda}{2}x^2 \right) \\ &= \begin{cases} 0 & (\text{if } 0 \le y \le C(1-\lambda)), \\ \frac{(y - C(1-\lambda))^2}{2C\lambda} & (\text{otherwise}). \end{cases} \end{aligned} \tag{46}$$

Third the envelope function $\Phi_C^{(m)}$ is redefined in terms of the above new conjugate regularizer $\delta_C^{(m)}$ as follows:

$$\begin{aligned} \Phi_C^{(m)}(v_m) &:= \min_{v'_m \in \mathbb{R}^N} \left(\tilde{g}_C^*(\|v'_m\|_{K_m}) + \frac{1}{2}\|v'_m - v_m\|_{K_m}^2 \right) \\ &= \min_{v'_m \in \mathbb{R}^N} \left(\tilde{g}_C^*(\|v'_m\|_{K_m}) + \frac{1}{2}(\|v'_m\|_{K_m} - \|v_m\|_{K_m})^2 \right) \\ &= \widehat{\tilde{g}_C^*}(\|v_m\|_{K_m}), \end{aligned}$$

where we again used Cauchy-Schwarz inequality in the second line, and $\widehat{\tilde{g}_C^*}$ is the Moreau's envelope function of $\tilde{g}_C^*$ as follows:

$$\widehat{\tilde{g}_C^*}(y) = \min_{y' \ge 0} \left(\tilde{g}_C^*(y') + \frac{1}{2}(y' - y)^2 \right).$$

With the above three modifications, the SpicyMKL iteration is rewritten as follows:

$$\begin{aligned}
\rho^{(t)} &:= \operatorname*{argmin}_{\rho\in\mathbb{R}^N}\left(L^*(-\rho)+\frac{1}{\gamma^{(t)}}\sum_{m=1}^{M}\widehat{\tilde{g}_C^*}(\|\alpha_m+\gamma^{(t)}\rho\|_{K_m})+\frac{1}{2\gamma^{(t)}}\left(b+\gamma^{(t)}\sum_{i=1}^{N}\rho_i\right)^2\right),\\
\alpha_m^{(t+1)} &= \operatorname{prox}\left(\alpha_m^{(t)}+\gamma^{(t)}\rho^{(t)}\middle|\phi^{(m)}_{\gamma^{(t)}C}\right)\quad (m=1,\ldots,M),\\
b^{(t+1)} &= b^{(t)}+\gamma^{(t)}\sum_{i=1}^{N}\rho_i^{(t)}.
\end{aligned}\tag{47}$$

Note that if $\tilde{g}_C^*(0)=0$ (this is the case if $\tilde{g}_C(0)=0$), the envelope function $\widehat{\tilde{g}_C^*}(\cdot)$ in (47) becomes zero whenever the norm $\|\alpha_m+\gamma^{(t)}\rho\|_{K_m}$ is zero. Thus the inner objective (47) inherits the computational advantage provided by sparsity of the 1-norm SpicyMKL presented in Sect. 3.3. Moreover, the gradient and Hessian of the generalized inner objective (47) can be computed in a similar manner as (26) and (27).

A generalization to the general loss functions with constraints is also straightforward; we only need to replace $\|\operatorname{prox}(\alpha_m+\gamma\rho|\phi^{(m)}_{\gamma C})\|^2_{K_m}$ in (41) (the definition of $\varphi_\gamma(\rho,\tilde{\rho};\alpha,b,\xi)$) with $\widehat{\tilde{g}_C^*}(\|\alpha_m+\gamma\rho\|_{K_m})$.

Implementation of general loss and regularization Our Matlab implementation is suited to general loss and block-norm regularization functions. The code runs under general settings if one plugs in scripts of the following information: the primal and dual functions of the loss, the gradient and Hessian of the dual loss, the primal and dual of the regularization function, the corresponding proximity operator and the derivative of the proximity operator.

4.3 One step optimization for a regularization function with smooth dual

When the convex conjugate $\delta_C^{(m)}$ of the regularization term $\phi_C^{(m)}$ is smooth (twice differentiable), the optimization needs only *one* step iteration of the outer loop. We illustrate the optimization method for smooth dual function in a generalized formulation and show two useful examples, Elastic-net MKL for $\lambda>0$ and block q-norm MKL for $q>1$.

The primal problem (7) is equivalent to the following dual problem by Fenchel's duality theorem (Rockafellar 1970, Theorem 31.2):

$$\operatorname*{maximize}_{\substack{\rho\in\mathbb{R}^N\\ \mathbf{1}^\top\rho=0}}\left(-L^*(-\rho)-\sum_{m=1}^{M}\delta_C^{(m)}(\rho)\right),\tag{48}$$

where L^* and $\delta_C^{(m)}$ are the convex conjugate of L and $\phi_C^{(m)}$. Note that the constraint $\mathbf{1}^\top\rho=0$ is due to the bias term b so that if there is no bias term this constraint is removed. Using the expression for the convex conjugate $\delta_C^{(m)}$ in (45), (48) is rewritten as follows:

$$\operatorname*{maximize}_{\substack{\rho\in\mathbb{R}^N\\ \mathbf{1}^\top\rho=0}}\left(-L^*(-\rho)-\sum_{m=1}^{M}\tilde{g}_C^*(\|\rho\|_{K_m})\right).$$

Thus if L^* and $\tilde{g}^*$ are differentiable, the optimization problem can be easily solved by gradient descent or Newton method with the constraint $\mathbf{1}^\top\rho=0$. By a simple calculation, we

obtain the gradient and Hessian of the dual regularization term as

$$\nabla_\rho \tilde{g}_C^*\left(\|\rho\|_{K_m}\right) = \frac{K_m\rho}{\|\rho\|_{K_m}} \frac{\mathrm{d}\tilde{g}_C^*(y)}{\mathrm{d}y}\bigg|_{y=\|\rho\|_{K_m}}, \tag{49}$$

$$\nabla\nabla_\rho^\top \tilde{g}_C^*\left(\|\rho\|_{K_m}\right) = \left(\frac{K_m}{\|\rho\|_{K_m}} - \frac{K_m\rho\rho^\top K_m}{\|\rho\|_{K_m}^3}\right) \frac{\mathrm{d}\tilde{g}_C^*(y)}{\mathrm{d}y}\bigg|_{y=\|\rho\|_{K_m}} + \frac{K_m\rho\rho^\top K_m}{\|\rho\|_{K_m}^2} \frac{\mathrm{d}^2\tilde{g}_C^*(y)}{\mathrm{d}y^2}\bigg|_{y=\|\rho\|_{K_m}}. \tag{50}$$

To deal with the constraint $\mathbf{1}^\top\rho = 0$, we added a penalty function $C_b(\mathbf{1}^\top\rho)^2$ to the objective function in our numerical experiments with large C_b (in our experiments we used $C_b = 10^5$).

The primal optimal solution can be computed from the dual optimal solution as follows. Given the dual optimal solution $\rho^\star$, the primal optimal solution $\alpha^\star = (\alpha_1^{\star\top}, \ldots, \alpha_M^{\star\top})^\top$ and $b^\star$ satisfy

$$\bar{K}\alpha^\star + \mathbf{1}b^\star = -\nabla_\rho L^*(-\rho^\star),$$

see Rockafellar (1970, Theorem 31.3). By KKT-condition, there is a real number c such that

$$\begin{aligned}\nabla_\rho L^*(-\rho^\star) &= -\sum_{m=1}^M \nabla_\rho \tilde{g}_C^*(\|\rho\|_{K_m})\Big|_{\rho=\rho^\star} + c\mathbf{1} \\ &= -\sum_{m=1}^M \frac{K_m\rho^\star}{\|\rho^\star\|_{K_m}} \frac{\mathrm{d}\tilde{g}_C^*(y)}{\mathrm{d}y}\bigg|_{y=\|\rho^\star\|_{K_m}} + c\mathbf{1}.\end{aligned}$$

Therefore the primal optimal solution is given as

$$\alpha_m^\star = \frac{\rho^\star}{\|\rho^\star\|_{K_m}} \frac{\mathrm{d}\tilde{g}_C^*(y)}{\mathrm{d}y}\bigg|_{y=\|\rho^\star\|_{K_m}}, \qquad b^\star = -c.$$

When L^* is not differentiable (e.g., hinge loss), combination of the techniques of this subsection and Sect. 3.4 can resolve the non-differentiability of L^*.

In the following we give two examples; Elastic-net and the block q-norm regularization ($q > 1$).

4.3.1 Efficient optimization of Elastic-net MKL

In Elastic-net MKL, the regularization term is $\tilde{g}_C(x) = C(1-\lambda)x + \frac{C\lambda}{2}x^2$. The convex conjugate of $\tilde{g}_C$ is given in (46). Therefore

$$\frac{\mathrm{d}\tilde{g}_C^*(y)}{\mathrm{d}y} = \begin{cases} 0 & (|y| \le C(1-\lambda)), \\ \frac{y - C(1-\lambda)}{C\lambda} & (\text{otherwise}), \end{cases} \qquad \frac{\mathrm{d}^2\tilde{g}^*(y)}{\mathrm{d}y^2} = \begin{cases} 0 & (|y| \le C(1-\lambda)), \\ \frac{1}{C\lambda} & (\text{otherwise}). \end{cases}$$

Substituting the gradient and the Hessian to (49) and (50), we obtain the Newton method for the dual of Elastic-net MKL. It should be noted that the gradient and Hessian need to be computed only on active kernels as in Sect. 3.3, i.e. $\nabla_\rho \tilde{g}^*(\|\rho\|_{K_m}) = 0$, $\nabla\nabla_\rho^\top \tilde{g}^*(\|\rho\|_{K_m}) = 0$ for all m such that $\|\rho\|_{K_m} \le C(1-\lambda)$.

4.3.2 Efficient optimization of block q-norm MKL

When $q > 1$, block q-norm MKL can be solved in one outer step. The regularization term of the block q-norm MKL is written as $\tilde{g}_C(x) = \frac{C}{q}x^q$. By a simple calculation, we obtain the convex conjugate $\tilde{g}_C^*$ as:

$$\tilde{g}_C^*(y) = C^{1-r}\frac{1}{r}y^r,$$

where $r = \frac{q}{q-1}$. Note that $r > 1$ when $q > 1$. Then we have

$$\frac{\mathrm{d}\tilde{g}^*(y)}{\mathrm{d}y} = C^{1-r}y^{r-1}, \qquad \frac{\mathrm{d}^2\tilde{g}^*(y)}{\mathrm{d}y^2} = C^{1-r}(r-1)y^{r-2}.$$

Therefore we can apply the Newton method to the dual of ℓ_p-norm MKL using the formulae (49) and (50).

5 Relations with the existing methods

We briefly illustrate the relations between our approach and the existing methods. Basically the existing methods (such as SILP (Sonnenburg et al. 2006), SimpleMKL (Rakotomamonjy et al. 2008), LevelMKL (Xu et al. 2009), and HessianMKL (Chapelle and Rakotomamonjy 2008)) rely on the equation (Micchelli and Pontil 2005):

$$\left(\sum_{m=1}^M \|f_m\|_{\mathcal{H}_m}\right)^2 = \inf_{d_m\geq 0, \sum_m d_m = 1}\left\{\sum_{m=1}^M \frac{\|f_m\|_{\mathcal{H}_m}^2}{d_m}\right\}.$$

An advantage of this formulation is that we have a smooth upper bound $\sum_{m=1}^M \frac{\|f_m\|_{\mathcal{H}_m}^2}{d_m}$ of the non-smooth ℓ_1 regularization. Moreover (1) says that minimizing this upper bound with respect to $\{f_m\}_{m=1}^M$ under the constraint $f = \sum_{m=1}^M f_m$ for a given function f, the upper bound becomes the RKHS norm of the function f in the RKHS $\mathcal{H}_{\bar{k}(d)}$ corresponding to the averaged kernel $\bar{k}(d) = \sum_{m=1}^M d_m k_m$, i.e.,

$$\|f\|_{\mathcal{H}_{\bar{k}(d)}}^2 = \inf_{f=\sum_{m=1}^M f_m}\left\{\sum_{m=1}^M \frac{\|f_m\|_{\mathcal{H}_m}^2}{d_m}\right\}$$

(see Aronszajn 1950 for the proof). Thus we don't need to consider M functions $\{f_m\}_{m=1}^M$, instead we only need to deal with one function f on the averaged kernel function $\bar{k}(d)$. That is, the MKL learning scheme can be converted to

$$\inf_{f_1\in\mathcal{H}_1,\ldots,f_M\in\mathcal{H}_M}\sum_{i=1}^N \ell\left(y_i, \sum_{m=1}^M f_m(x_i)\right) + C\left(\sum_{m=1}^M \|f_m\|_{\mathcal{H}_m}\right)^2$$
$$= \inf_{d_m\geq 0, \sum_m d_m = 1}\left\{\inf_{f\in\mathcal{H}_{\bar{k}(d)}}\sum_{i=1}^N \ell\left(y_i, f(x_i)\right) + C\|f\|_{\mathcal{H}_{\bar{k}(d)}}^2\right\}.$$

One notes that fixing $\{d_m\}_{m=1}^M$ the problem of the inner minimization is a standard single kernel learning. Thus the inner minimization is easily solved by using publicly available

efficient solvers for a single kernel learning. Based on these relations, the existing methods consist of the following two parts, the single kernel learning part and the updating part of $\{d_m\}_m$:

1. Minimize $L(\tilde{K}\alpha) + C\alpha^\top \tilde{K}\alpha$ where $\tilde{K} = \sum_{m=1}^M d_m K_m$.
2. Update $\{d_m\}_{m=1}^M$ so that the objective function decreases,

where the update step differs depending on methods.

On the other hand, our approach is totally different from the existing approaches. We don't utilize the kernel weight to obtain a smoothed approximation of the objective problem. Our approach utilize the proximal minimization update (8) that can be interpreted as a gradient descent of the Moreau's envelope of the objective function. In general, for a convex function f, the Moreau's envelope $\hat{f}$ defined below is differentiable:

$$\begin{aligned} \hat{f}(x) &:= \min_y \left\{ f(y) + \frac{1}{2\gamma}\|y - x\|^2 \right\}, \\ \nabla \hat{f}(x) &= \frac{1}{\gamma}(x - y^*(x)) \in \nabla f(y^*(x)) \\ &\text{where } y^*(x) = \arg\min_y \left\{ f(y) + \frac{1}{2\gamma}\|y - x\|^2 \right\}. \end{aligned} \tag{51}$$

One can check that $\min \hat{f} = \min f$ and $\arg\min \hat{f} = \arg\min f$ because $f(x^*) \le f(y^*(x^*)) + \frac{1}{2\gamma}\|y^*(x^*) - x^*\|^2 = \hat{f}(x^*) \le f(x^*) + \frac{1}{2\gamma}\|x^* - x^*\|^2 = f(x^*)$ for all x and $x^* \in \arg\min f$, and if $x \notin \arg\min f$, we have $f(x^*) < \hat{f}(x)$. Thus the minimization of $\hat{f}$ leads to the minimization of f. Using the relation (51), the proximal minimization update (8) corresponds to

$$x^{(t+1)} \leftarrow y^*(x^{(t)}) = x^{(t)} - \gamma \nabla \hat{f}(x^{(t)}) \in x^{(t)} - \gamma \nabla f(x^{(t+1)}).$$

Therefore the updating rule is a gradient descent of the Moreau's envelope of the objective function (and the increment is also the subgradient of f at the *next solution* $x^{(t+1)}$). Fortunately the minimization required to obtain the Moreau's envelope is obtained by a smooth optimization procedure as described in Sect. 3.3. More general and detailed discussions about the use of the Moreau's envelope for machine learning settings can be found in Tomioka et al. (2011).

Regarding the optimization of block q-norm MKL, Kloft et al. (2010) considered a unifying regularization including block q-norm MKL where the regularization term is

$$C_1 \left(\sum_{m=1}^M \|f_m\|_{\mathcal{H}_m}^q \right)^{\frac{2}{q}} + C_2 \sum_{m=1}^M \|f_m\|_{\mathcal{H}_m}^2,$$

(there is additional operation $(\cdot)^{\frac{2}{q}}$ at the q-norm term compared with our block q-norm MKL formulation, however there is one-to-one correspondence between both formulations as in the same reason described in the end of Sect. 2.2). They also noticed that the dual of the above unifying regularization is smooth unless $C_2 = 0$ and $q = 1$, and suggest to utilize L-BFGS to solve the dual problem. This corresponds to the special case of our setting $g(x) = (1 - \lambda)x^{q/2} + \lambda x$ with $0 \le \lambda \le 1$. As shown in Sect. 4.3, we can solve the MKL problem with this regularization by one step outer iteration because of the smoothness of its dual.

6 Numerical experiments

In this section, we experimentally confirm the efficiency of the proposed SpicyMKL on several binary classification tasks from UCI and IDA machine learning repository.[5] We compared our algorithm SpicyMKL to four state-of-the-art algorithms namely SILP (Sonnenburg et al. 2006), SimpleMKL (Rakotomamonjy et al. 2008), LevelMKL (Xu et al. 2009) and HessianMKL (Chapelle and Rakotomamonjy 2008). As for SpicyMKL, we report the results of hinge and logistic losses with block 1-norm regularization. We also report the result of the one step optimization method for elastic-net regularization (the method described in Sect. 4.3.1) with hinge and logistic losses. To distinguish the iterative method and the one step method, we call the one step optimization method for elastic-net regularization as ElastMKL.

6.1 Performances on UCI benchmark datasets

We used 5 datasets from the UCI repository (Asuncion and Newman 2007): '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 ... 19 20) and polynomial kernels of degree 1 to 3. All of 27 different kernel functions (24 Gaussian kernels and 3 polynomial kernels) 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 ($K_m \leftarrow K_m/\mathrm{trace}(K_m)$), and were precomputed prior to running the algorithms. These experimental settings were borrowed from the paper (Rakotomamonjy et al. 2008) of SimpleMKL, but we used a larger number of kernels.

For each dataset, we randomly chose 80% of samples for training and the remaining 20% for testing. This procedure was repeated 10 times. Experiments were run on 3 different regularization parameters $C = 0.005$, 0.05 and 0.5; for SimpleMKL, LevelMKL and HessianMKL, the regularization parameter was converted by (6). We employed the *relative duality gap*, (primal obj − dual obj)/primal obj with tolerance 0.01 as the stopping criterion for all algorithms. The primal objective for SpicyMKL and ElastMKL can be computed by using $\alpha^{(t)}$ and $b^{(t)}$. In order to compute the dual objective of SpicyMKL and ElastMKL, we project the multiplier vector ρ to the equality constraint $\tilde{\rho} = \rho - \mathbf{1}(\sum \rho_i)/N$ and in addition, for SpicyMKL with block 1-norm regularization, we project $\tilde{\rho}$ to the domain of $\delta_C^{(m)}$ (21) by $\tilde{\rho}' = \tilde{\rho}/\max\{\max_m\{\|\tilde{\rho}\|_{K_m}/C\}, 1\}$. Then we compute the dual objective function as $-L^*(-\tilde{\rho})$. The same technique can be found in Tomioka and Sugiyama (2009) and Wright et al. (2009). The primal objective of SILP, SimpleMKL, HessianMKL and LevelMKL is computed as $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j\sum_m d_m K_m(x_i, x_j)$ and the dual objective as $\sum_i \alpha_i - \frac{1}{2}\max_m \sum_{i,j}\alpha_i\alpha_j K_m(x_i, x_j)$ where α_i is the dual variable of SVM solved at each iteration and d_m is the kernel weight (see Sonnenburg et al. 2006 and Rakotomamonjy et al. 2008).

For SpicyMKL and ElastMKL, we report the result from two loss functions; the hinge loss and the logistic loss. We used $\lambda = 0.5$ for ElastMKL. For SILP, SimpleMKL, LevelMKL and HessianMKL, we used the hinge loss. For SILP, we used Shogun implementa-

[5] All the experiments were executed on Intel Xeon 3.33 GHz with 48 GB RAM.

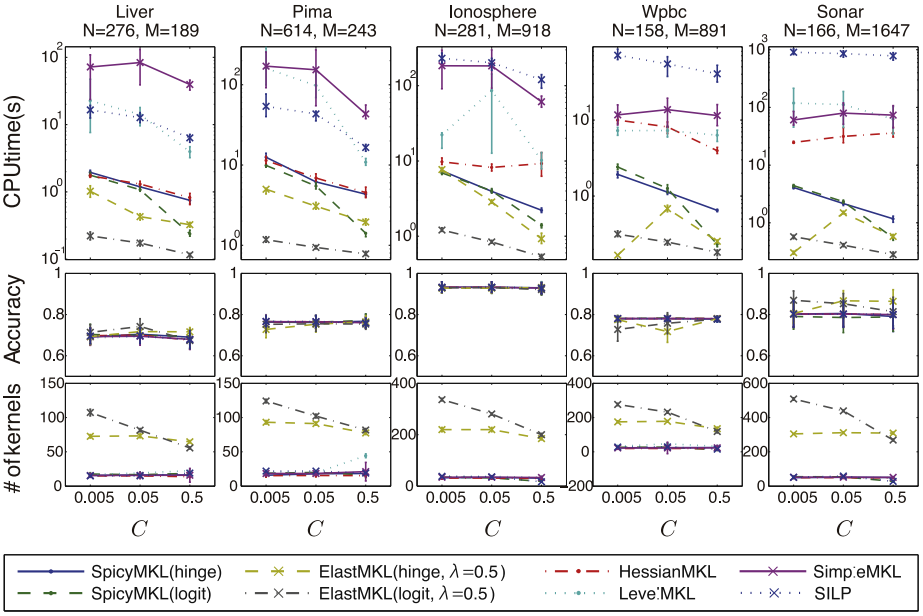


Fig. 3 Mean and standard deviation of performance measures for each MKL method for UCI datasets

tion[6] written in C++, and for SimpleMKL,[7] LevelMKL[8] and HessianMKL,[9] we used publicly available Matlab® codes. We replaced the Mosek® solver with the CPLEX® solver for the linear programming and the quadratic programming required inside LevelMKL.

The performance of each method is summarized in Fig. 3. The average CPU time, test accuracy, and final number of active kernels, are shown from top to bottom. In addition, standard deviations are also shown. We can see that SpicyMKL tends to be faster than SILP (by factors of 4 to 670) and SimpleMKL (by factors of 6 to 60) and LevelMKL (by factors of 3 to 50), and faster than HessianMKL when the number of kernels M is large (Ionospher, Wpbc & Sonar). In all datasets, SpicyMKL becomes faster as the regularization parameter C increases. This is because the larger C becomes, the smaller the number of active kernels during the optimization. ElastMKL is even faster than SpicyMKL in all datasets. In particular, ElastMKL with the logistic loss shows the best performance among all methods. This is because the one step optimization method explained in Sect. 4.3 does not require iterative procedure for logistic loss. Accuracies of all methods for block 1-norm regularization are nearly identical. Thus from the classification accuracy, the block 1-norm MKL using logistic and hinge loss seem to perform similarly. The accuracy for elastic-net regularization varies slightly depending on problems.

SpicyMKL using the logistic loss tends to be faster than that using the hinge loss. This could be explained by the strong convexity of the conjugate of the logistic loss, which is not the case for the hinge loss. In fact, when L^* is strongly convex, the inner Newton method

[6] http://www.shogun-toolbox.org.

[7] http://asi.insa-rouen.fr/enseignants/~arakotom/code/mklindex.html.

[8] http://appsrv.cse.cuhk.edu.hk/~zlxu/toolbox/level_mkl.html.

[9] http://olivier.chapelle.cc/ams/.

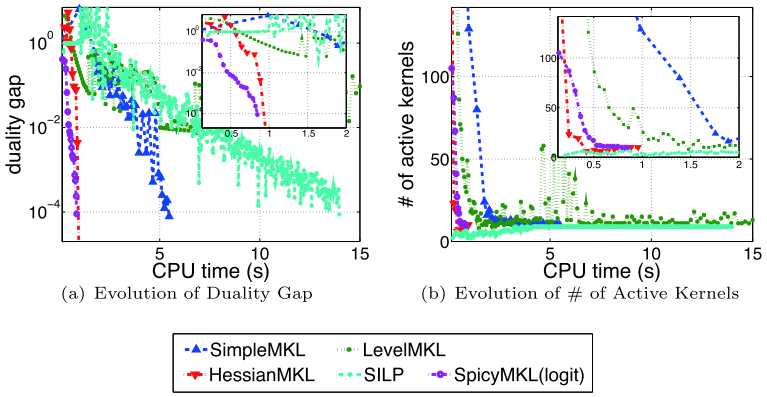
Table 4 Average number of outer iterations. Average number of SVM evaluations is also shown in the brace for SimpleMKL

Dataset	C	Spicy (hinge)	Spicy (logit)	Simple	Hessian	Level	SILP
Liver	0.005	22.6	20.1	200.3(3770.2)	13.2	171.6	718.8
	0.05	14.1	14.5	222.2(4800.0)	11.8	138.0	517.2
	0.5	6.7	3.3	134.2(2200.5)	9.4	50.1	237.8
Pima	0.005	31.5	21.5	76.2(1488.9)	15.4	290.7	1020.1
	0.05	13.6	13.9	74.9(1511.7)	12.2	197.9	727.1
	0.5	7.9	3.2	24.6(470.9)	10.2	30.9	258.0
Ionosphere	0.005	38.0	24.6	184.5(2864.1)	11.3	100.9	1810.3
	0.05	18.3	17.1	188.6(2983.1)	10.3	135.6	1580.1
	0.5	7.8	5.4	57.8(1374.4)	14.4	65.5	972.0
Wpbc	0.005	24.0	24.4	45.4(1014.2)	11.6	61.3	683.6
	0.05	13.2	14.4	49.1(1134.7)	10.7	60.8	514.2
	0.5	6.0	2.0	40.9(964.4)	7.2	55.4	367.0
Sonar	0.005	35.2	27.2	158.3(2736.8)	10.3	193.0	4484.6
	0.05	16.6	16.8	204.8(3535.3)	10.6	194.2	4356.4
	0.5	7.7	4.2	173.3(3511.4)	19.6	154.4	3862.3

(minimization of φ_γ with respect to ρ) converges rapidly. Although the logistic loss is often faster to train, yet the accuracy is nearly identical to that of the hinge loss.

When the regularization is elastic-net, the hinge loss gives sparser solution than the logistic loss. The number of kernels selected under elastic-net regularization is much lager than that under block 1-norm regularization as expected, and decreases as the strength of regularization C increases. Under block 1-norm regularization, the number of kernels slightly decreases as C increases.

In Table 4, a comparison of the average numbers of outer iterations and SVM evaluations on the UCI datasets is reported. The existing methods (HessianMKL, LevelMKL, SimpleMKL, SILP) iterate the update of the kernel weight $\{d_m\}_{m=1}^M$ and the SVM evaluation of the objective function at the current weight as illustrated in Sect. 5. We refer to the combination of these two steps as one outer iteration. Since SimpleMKL requires several SVM evaluations during the line search in gradient descent, the number of outer iterations and that of SVM evaluations are different. As for the other methods, both quantities are same. Obviously ElastMKL requires only one outer iteration, thus the result for ElastMKL is not reported in the table. We can see that in all methods the number of outer iterations decreases as the regularization parameter C increases. This is because as the regularization becomes strong, the required number of kernels becomes small and accordingly the intrinsic problem size becomes small. SpicyMKL and HessianMKL require much smaller number of iterations than others. This shows that these methods have fast convergence rate and thus the solution drastically approaches to the optimal one at each iteration. Comparing the CPU time required by SpicyMKL and HessianMKL, SpicyMKL tends to require light computation per iteration especially for large number of kernels while HessianMKL requires QP to derive the update direction that is heavy for large number of kernels.



(a) Evolution of Duality Gap (b) Evolution of # of Active Kernels

Fig. 4 Duality gap and # of active kernels against CPU time

In Fig. 4(a), we plot the relative duality gaps against CPU time for a single run of SpicyMKL (with logistic loss and block 1-norm regularization), HessianMKL, LevelMKL and SimpleMKL on the 'Wpbc' dataset. We can see that the duality gap of SpicyMKL drops rapidly and it is faster than linear. That supports the super-linear convergence of our method. HessianMKL decreases the duality gap quickly because HessianMKL is a second order method. The duality gap of LevelMKL gradually drops in the early stage, but after several steps, the behavior of duality gap becomes unstable. After all, the duality gap of LevelMKL did not drop below 10^{-2} in 500 iteration steps. This is because the size of QP required in LevelMKL increases as the algorithm proceeds so that it gets hard to obtain a precise solution. SILP shows fluctuations of the duality gap so that the duality gap does not decrease monotonically. This is because the sequence of the solutions generated by cutting plane method shows oscillation behavior. Figure 4(b) shows the number of active kernels as a function of the CPU time spent by the algorithm. Here we again observe rapid decrease in the number of kernels for SpicyMKL. This reduces huge amount of computation per iteration. We see a relation between the speed of convergence and the number of kernels in SpicyMKL; as the number of kernels decreases, the time spent per iteration becomes small.

6.2 Scaling against the sample size and the number of kernels

Here we investigate the dependency of CPU time on the number of kernels and the sample size. We used 4 datasets from IDA benchmark repository (Rätsch et al. 2001): 'Ringnorm', 'Splice', 'Twonorm' and 'Waveform'. The same relative duality gap criterion with tolerance 0.01 was used. We generated a set of basis kernels by randomly selecting subsets of features and applying Gaussian kernels with random width $\sigma = 5\chi^2 + 0.1$, where χ^2 is a chi-squared random variable. Here we report ElastMKL with $\lambda = 0.1, 0.5$ with hinge and logistic loss in addition to SpicyMKL with block 1-norm regularization with hinge and logistic loss, SimpleMKL, HessianMKL, LevelMKL and SILP.

In Fig. 5, the number of kernels is increased from 50 to 6000. The graph shows the median of CPU times with 25 and 75 percentiles over 10 random train-test splitting where the size of training set was fixed to 200. We observe that for small number of kernels the CPU time of HessianMKL is the fastest among all methods with block 1-norm regularization. However when the number of kernels is greater than 1000, our method SpicyMKL is clearly the fastest among the methods with block 1-norm regularization. In fact, SpicyMKL

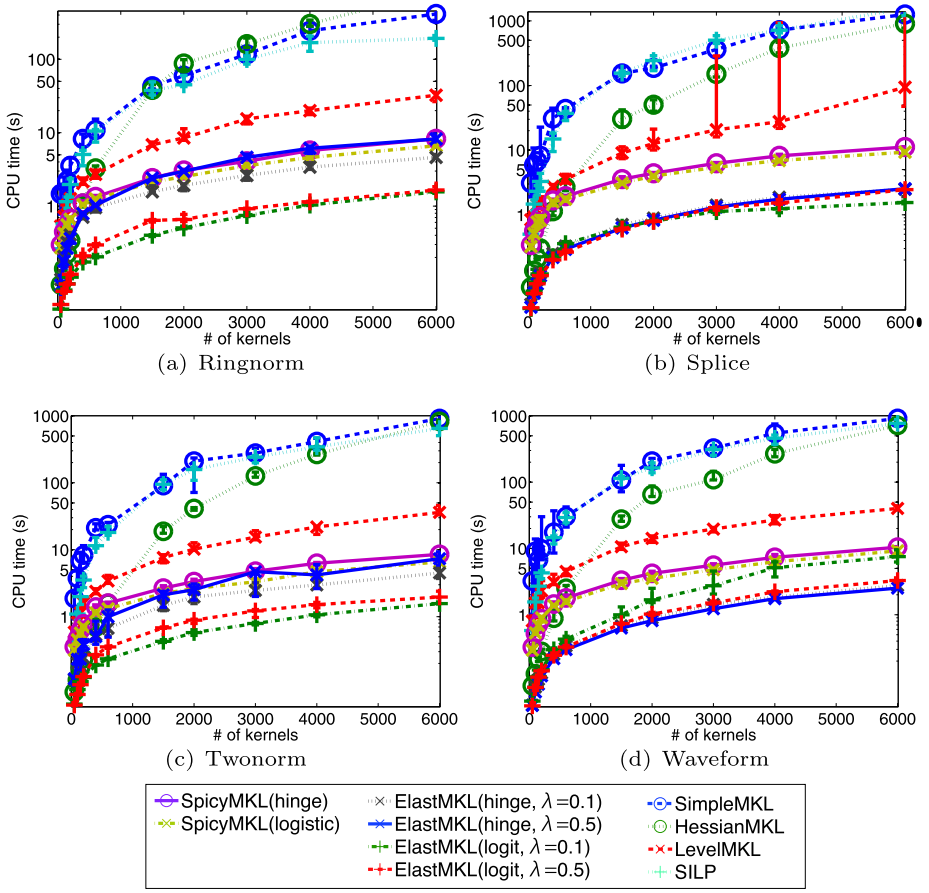


Fig. 5 CPU time as a function of the number of kernels

is roughly 100 times faster than SimpleMKL, HessianMKL and SILP when the number of kernels is 6000. LevelMKL scales almost same as SpicyMKL, but shows unstable performances in Splice dataset. One reason is that LevelMKL shows the oscillation behavior found in Fig. 4(a). In particular, the quadratic programming required in LevelMKL often does not converge until the maximum number of iterations available in CPLEX. The scaling property of SILP against the number of kernels is similar to that of SimpleMKL. Those methods do not have as good scaling property as SpicyMKL against the number of kernels. Here again we observe that ElastMKL regularization is even faster than that with block 1-norm regularization, and shows the best performance among all methods.

In Fig. 6 the number of training samples is increased from 500 to 3000. The number of kernels is fixed to 20. SILP shows fairly good computational efficiency among the methods with block 1-norm regularization. Comparing the result of Fig. 5, while SILP is not fast for a large number of kernels, SILP converges relatively fast for a small number of kernels and scales well against the number of samples. The CPU time of SpicyMKL and ElastMKL is comparable to that of other methods. In particular, ElastMKL shows the best performance in Splice, Twonorm and Waveform.

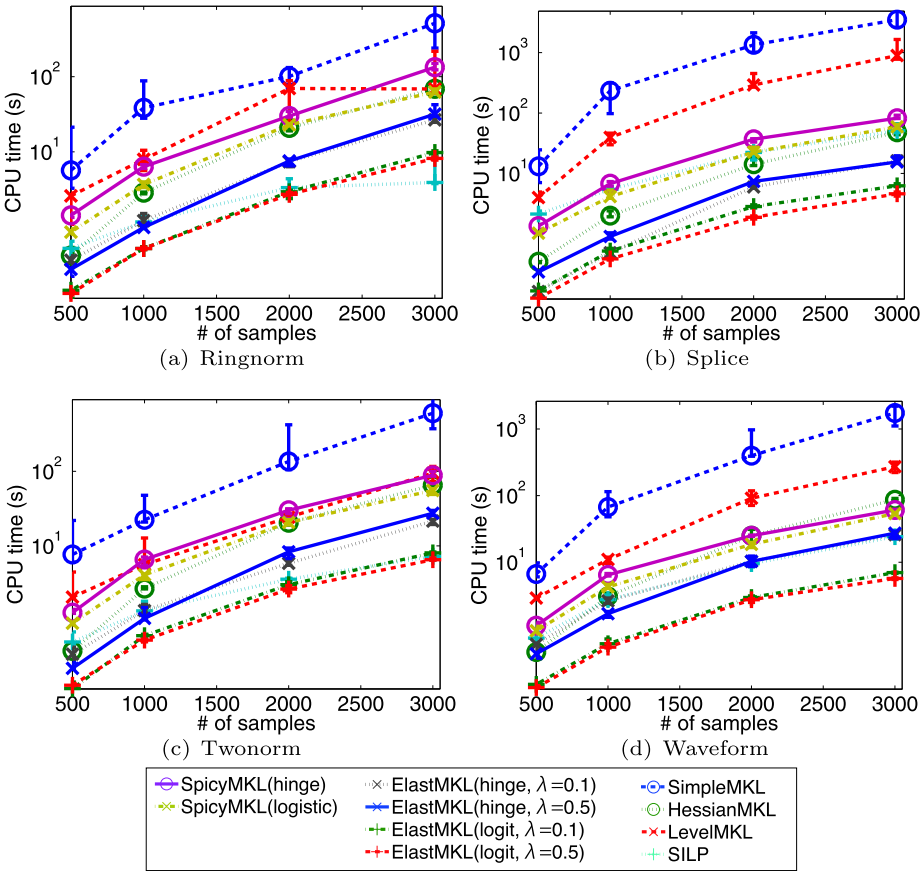


Fig. 6 CPU time as a function of the sample size

7 Conclusion and future direction

In this article, we have proposed a new efficient training algorithm for MKL with general convex loss functions and general regularizations. The proposed SpicyMKL algorithm generates a sequence of primal variables by iteratively optimizing a sequence of smooth minimization problems. 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. We introduced a generalized block norm formulation of MKL regularization that includes non-sparse regularizations such as Elastic-net MKL and block q-norm MKL, and derived the connections between our block norm formulation and the kernel weights. Then we derived a general optimization method that is applicable to the general block norm framework. We also gave an efficient one step optimization method for regularizations with smooth dual (e.g., elastic-net and block q-norm).

We have shown through numerical experiments that SpicyMKL scales well with increasing number of kernels and it has similar scaling behavior against the number of samples to conventional methods. It is worth noting that SpicyMKL does not rely on solving LP or QP problems, which itself can be a challenging task; accordingly SpicyMKL can reliably obtain

a precise solution. SpicyMKL with the logistic loss and elastic-net regularization has shown the best performance while showing fairly good accuracies.

Future work includes a second order modification of the update rule of the primal variables, and combination of the techniques developed in SpicyMKL and wrapper methods (Sonnenburg et al. 2006; Rakotomamonjy et al. 2008; Chapelle and Rakotomamonjy 2008).

It would also be interesting to develop an efficient decomposition method to deal with a large sample size problem. Since the dual problem (25) of the inner loop is an N dimensional optimization problem, when the number of samples N is large, the naive Newton method or other gradient descent type methods might be hard to be applied. In such a situation, the block-coordinate descent algorithm might be useful. That is, decompose N variables $\{\rho_i\}_{i=1}^N$ into some groups (say B groups $\{\rho_{I_i}\}_{i=1}^B$) and iteratively minimize the objective function with respect to one group with other groups fixed:

$$\rho_{I_i}^{k+1} \leftarrow \operatorname*{arg\,min}_{\rho_{I_i}} \varphi_{\gamma^{(t)}}((\rho_{I_1}^{k+1}, \ldots, \rho_{I_{i-1}}^{k+1}, \rho_{I_i}, \rho_{I_{i+1}}^{k}, \ldots, \rho_{I_B}^{k}); \alpha^{(t)}, b^{(t)})$$

(see Sect. 5.4.3 of Zangwill 1969 and Proposition 2.7.1 of Bertsekas 1999). In a similar sense, Sequential Minimal Optimization (SMO) algorithm (Platt 1999) might be useful for some loss function classes such as the hinge loss. We leave these important issues for future work.

Acknowledgements We would like to thank anonymous reviewers for their constructive comments, which improved the quality of this paper. We would also like to thank Manik Varma, Marius Kloft, Alexander Zien, and Cheng Soon Ong for helpful discussions. This work was partially supported by MEXT KAKENHI 22700289 and 22700138.

Appendix: Proof of Theorem 1

Combining (15), (19), and (24), we can rewrite the dual of the proximal MKL problem (8) as follows:

$$\begin{aligned}\underset{\substack{\rho\in\mathbb{R}^N\\ u\in\mathbb{R}^{MN}}}{\text{minimize}}\Bigg\{ & L^*(-\rho) + \sum_{m=1}^M \delta_C^{(m)}(u_m) \\ & + \sum_{m=1}^M \left({\alpha_m^{(t)}}^\top K_m(\rho - u_m) + \frac{\gamma^{(t)}}{2}\|\rho - u_m\|_{K_m}^2 \right) + b^{(t)}\mathbf{1}^\top\rho - \frac{\gamma^{(t)}}{2}\left(\mathbf{1}^\top\rho\right)^2 \Bigg\},\end{aligned}$$

where the maximization is turned into minimization and we redefined $u_m/\gamma^{(t)}$ as u_m. This formulation is known as the *augmented Lagrangian* for the dual of the MKL optimization problem (7), which can be expressed as follows:

$$\begin{aligned}\underset{\substack{\rho\in\mathbb{R}^N,\\ u\in\mathbb{R}^{MN}}}{\text{minimize}} \quad & L^*(-\rho) + \sum_{m=1}^M \delta_C^{(m)}(u_m), \\ \text{subject to} \quad & K_m^{1/2}(\rho - u_m) = 0 \quad (m = 1, \ldots, M), \\ & \mathbf{1}^\top\rho = 0.\end{aligned}$$

$K_m^{1/2}\alpha_m^{(t)}$ and $b^{(t)}$ are the Lagrangian multipliers corresponding to the above $M+1$ equality constraints (Hestenes 1969; Powell 1969; Tomioka and Sugiyama 2009). We apply Proposition 5.11 of Bertsekas (1982) so that we obtain

$$\sum_{m=1}^{M} \|\alpha_m^{(t)} - \alpha_m^*\|_{K_m}^2 + (b^{(t)} - b^*)^2 \to 0 \quad (\text{as } t \to \infty).$$

Thus there is sufficiently large T such that for all $t \geq T$, $(\alpha^{(t)}, b^{(t)})$ is inside the δ-neighborhood of (α^*, b^*). Therefore we can assume that (9) holds at $(\alpha^{(t)}, b^{(t)})$ for all $t \geq T$. Finally Theorem 3 of Tomioka et al. (2011) (and its proof) gives the assertion (see also the proof of Proposition 5.22 of Bertsekas 1982 where $\phi(t) = t^2$ and $\nabla\phi(t) = t$ are substituted for our setting).

References

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, *68*, 337–404.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bach, F. R. (2008). Consistency of the group Lasso and multiple kernel learning. *Journal of Machine Learning Research*, *9*, 1179–1225.

Bach, F. R., Lanckriet, G., & Jordan, M. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st international conference on machine learning* (pp. 41–48).

Bach, F. R., Thibaux, R., & Jordan, M. I. (2005). Computing regularization paths for learning multiple kernels. In *Advances in neural information processing systems* (Vol. 17, pp. 73–80). Cambridge: MIT Press.

Bertsekas, D. P. (1982). *Constrained optimization and Lagrange multiplier methods*. New York: Academic Press.

Bertsekas, D. P. (1999). *Nonlinear programming*. Nashua: Athena Scientific.

Candes, E. J., Romberg, J., & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, *52*(2), 489–509.

Chapelle, O., & Rakotomamonjy, A. (2008). Second order optimization of kernel parameters. In *NIPS workshop on kernel learning: automatic selection of optimal kernels*, Whistler.

Cortes, C. (2009). *Can learning kernels help performance?* Invited talk at International Conference on Machine Learning (ICML 2009), Montréal, Canada.

Cortes, C., Mohri, M., & Rostamizadeh, A. (2009). L_2 regularization for learning kernels. In *Proceedings of the 25th conference on uncertainty in artificial intelligence (UAI 2009)*, Montréal, Canada.

Daubechies, I., Defrise, M., & Mol, C. D. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, *LVII*, 1413–1457.

Figueiredo, M., & Nowak, R. (2003). An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, *12*, 906–916.

Gehler, P. V., & Nowozin, S. (2009). Let the kernel figure it out; principled learning of pre-processing for kernel classifiers. In *Proceedings of the IEEE computer society conference on computer vision and pattern (CVPR2009)*.

Hestenes, M. (1969). Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, *4*, 303–320.

Kimeldorf, G. S., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, *33*, 82–95.

Kloft, M., Brefeld, U., Sonnenburg, S., Laskov, P., Müller, K. R., & Zien, A. (2009). Efficient and accurate ℓ_p-norm multiple kernel learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 22, pp. 997–1005). Cambridge: MIT Press.

Kloft, M., Rückert, U., & Bartlett, P. L. (2010). *A unifying view of multiple kernel learning*. arXiv:1005.0437.

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

Micchelli, C. A., & Pontil, M. (2005). Learning the kernel function via regularization. *Journal of Machine Learning Research*, *6*, 1099–1125.

Mosci, S., Santoro, M., Verri, A., & Villa, S. (2008). A new algorithm to learn an optimal kernel based on Fenchel duality. In *NIPS 2008 workshop: kernel learning: automatic selection of optimal kernels*, Whistler.

Nath, J. S., Dinesh, G., Raman, S., Bhattacharyya, C., Ben-Tal, A., & Ramakrishnan, K. R. (2009). On the algorithmics and applications of a mixed-norm based kernel learning formulation. In *Advances in neural information processing systems* (Vol. 22, pp. 844–852). Cambridge: MIT Press.

Palmer, J., Wipf, D., Kreutz-Delgado, K., & Rao, B. (2006). Variational EM algorithms for non-Gaussian latent variable models. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems* (Vol. 18, pp. 1059–1066). Cambridge: MIT Press.

Platt, J. C. (1999). Using sparseness and analytic QP to speed training of support vector machines. In *Advances in neural information processing systems* (Vol. 11, pp. 557–563). Cambridge: MIT Press.

Powell, M. (1969). A method for nonlinear constraints in minimization problems. In R. Fletcher (Ed.), *Optimization* (pp. 283–298). London: Academic Press.

Rakotomamonjy, A., Bach, F., & Canu, S. Y. G. (2008). SimpleMKL. *Journal of Machine Learning Research*, *9*, 2491–2521.

Rätsch, G., Onoda, T., & Müller, K. R. (2001). Soft margins for adaboost. *Machine Learning*, *42*(3), 287–320.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton: Princeton University Press.

Rockafellar, R. T. (1976). Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, *1*, 97–116.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge: MIT Press.

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, *7*, 1531–1565.

Tomioka, R., & Sugiyama, M. (2009). Dual augmented lagrangian method for efficient sparse reconstruction. *IEEE Signal Processing Letters*, *16*(12), 1067–1070.

Tomioka, R., & Suzuki, T. (2009). *Sparsity-accuracy trade-off in MKL*. arXiv:1001.2615.

Tomioka, R., & Suzuki, T. (2011). *Regularization strategies and empirical Bayesian learning for MKL*. arXiv:1011.3090.

Tomioka, R., Suzuki, T., & Sugiyama, M. (2011). Super-linear convergence of dual augmented lagrangian algorithm for sparse learning. *Journal of Machine Learning Research*, *12*, 1501–1550.

Wright, S. J., Nowak, R. D., & Figueiredo, M. A. T. (2009). Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, *57*(7), 2479–2493. doi:10.1109/TSP.2009.2016892.

Xu, Z., Jin, R., King, I., & Lyu, M. R. (2009). An extended level method for efficient multiple kernel learning. In *Advances in neural information processing systems* (Vol. 21, pp. 1825–1832). Cambridge: MIT Press.

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

Zangwill, W. I. (1969). *Nonlinear programming: a unified approach*. New York: Prentice Hall.

Zien, A., & Ong, C. (2007). Multiclass multiple kernel learning. In *Proceedings of the 24th international conference on machine learning* (pp. 11910–1198). New York: ACM.