Bilinear Formulated Multiple Kernel Learning for Multi-class Classification Problem

Takumi Kobayashi and Nobuyuki Otsu

National Institute of Advanced Industrial Science and Technology,
1-1-1 Umezono, Tsukuba, Japan

Abstract. In this paper, we propose a method of multiple kernel learning (MKL) to inherently deal with multi-class classification problems. The performances of kernel-based classification methods depend on the employed kernel functions, and it is difficult to predefine the optimal kernel. In the framework of MKL, multiple types of kernel functions are linearly integrated with optimizing the weights for the kernels. However, the multi-class problems are rarely incorporated in the formulation and the optimization is time-consuming. We formulate the multi-class MKL in a bilinear form and propose a scheme for computationally efficient optimization. The scheme makes the method favorably applicable to large-scaled samples in the real-world problems. In the experiments on multi-class classification using several datasets, the proposed method exhibits the favorable performance and low computation time compared to the previous methods.

Keywords: Kernel methods, multiple kernel learning, multi-class classification, bilinear form.

1 Introduction

The kernel-based methods have attracted keen attentions, exhibiting the state-of-the-art performances, such as in support vector machines (SVM) [10] and kernel multivariate analyses [8]. These methods are applied in various real-world tasks, e.g., in the fields of computer vision and signal processing. In the kernel-based methods, the input vectors are implicitly embedded in a high dimensional space (called kernel feature space) via kernel functions which efficiently compute inner products of those vectors in the kernel feature space. Thus, the performance of the kernel-based methods depends on how to construct the kernel functions.

In recent years, Lanckriet et al. [5] proposed the method to integrate different kernel functions with optimizing the weights for the kernels, which is called multiple kernel learning (MKL). By combining multiple types of kernels, the heterogeneous information, which is complementary to each other, can be effectively incorporated, possibly improving the performance. The composite kernel is successfully applied to, for example, object recognition [11].

In MKL, the weights for combining the kernels are obtained via the optimization processes based on a certain criterion, mainly for classification. Since the criterion can be defined in different formula, various methods for MKL have been

K.W. Wong et al. (Eds.): ICONIP 2010, Part II, LNCS 6444, pp. 99–107, 2010.

proposed by treating different optimization problems in different approaches; e.g., semi-definite programming [5] and semi-infinite linear program [9,13]. Most of the methods, however, are intended for classifying binary classes, while real-world problems contain multi classes in general. In addition, for application to practical problems, the optimization process should be computationally efficient.

In this paper, we propose a MKL method for multi-class classification problems. Without decomposing the multi-class problem into several binary class problems, the proposed method inherently deals with it based on the formulation of Crammer & Singer [2] which first proposed multi-class SVM using a single kernel. The contributions of this paper are as follows:

- We extend the formulation of multi-class classification in [2] to cope with multiple kernel functions, and formulate multi-class MKL (MC-MKL) in a bilinear form. In the formulation, the optimal weights for kernel functions are obtained in respective classes.
- We propose a scheme to effectively optimize the bilinear formulated problem, which makes the method applicable to large-scaled samples.
- In the experiments on various datasets, we demonstrate the effectiveness of the proposed method, compared to the existing MKL methods [7,13].

While Zien & Ong [13] proposed the method of MC-MKL based on a similar formulation, we employ a different criterion for the margin of the multi-class classifiers and propose a more efficient optimization scheme.

2 Bilinear Formulation for MC-MKL

To consider multiple kernels, we introduce multiple types of features $\boldsymbol{x}^{(r)}$ ($r \in \{1, .., R\}$, where R is the number of feature types). The inner products of those features can be replaced with respective types of kernels via kernel tricks: $\boldsymbol{x}_i^{(r)\prime}\boldsymbol{x}_j^{(r)} \rightarrow k_r(\boldsymbol{x}_i^{(r)}, \boldsymbol{x}_j^{(r)})$. Crammer & Singer [2] have proposed a formulation for multi-class SVM, considering only a single type of feature $\boldsymbol{x}$. We extend the formulation to incorporate the multiple types of features (kernels). We additionally introduce the weights $\boldsymbol{v}$ for feature types as well as the weights $\boldsymbol{w}$ within features similarly in MKL methods [13]. These two kinds of weights are mathematically integrated into the following bilinear form to constitute multi-class classification [2]:

$$\overset{*}{c} = \arg \max_{c \in \{1,..,C\}} \left\{ \sum_{r=1}^{R} v_c^{(r)} \boldsymbol{w}_c^{(r)\prime} \boldsymbol{x}^{(r)} = \boldsymbol{w}_c' \boldsymbol{X} \boldsymbol{v}_c = \langle \boldsymbol{X}, \boldsymbol{w}_c \boldsymbol{v}_c' \rangle_F \right\}, \tag{1}$$

where C is the number of classes, $\langle\ ,\ \rangle_F$ indicates Frobenius inner product, $v_c^{(r)}$ is a weight for the r-th type of feature, $\boldsymbol{w}_c^{(r)}$ is a classifier vector for the r-th type of feature vector in class c, and these variables are concatenated into long vectors, respectively;

$$\boldsymbol{v}_c \triangleq \left[v_c^{(1)}, \cdots, v_c^{(R)} \right]', \ \boldsymbol{X} \triangleq \begin{bmatrix} \boldsymbol{x}^{(1)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{x}^{(R)} \end{bmatrix}, \ \boldsymbol{w}_c \triangleq \begin{bmatrix} \boldsymbol{w}_c^{(1)} \\ \vdots \\ \boldsymbol{w}_c^{(R)} \end{bmatrix}. \tag{2}$$

Then, we can consider the margin of the above-defined bilinear classifiers (projections) by using the Frobenius norm $||\boldsymbol{w}_c\boldsymbol{v}_c'||_F$, and define the following optimization problem based on a large margin criterion:

$$\begin{aligned} \text{OP:} \min_{\{\boldsymbol{w}_c,\boldsymbol{v}_c\},\{\xi_i\}} \ & \sum_{c=1}^{C} ||\boldsymbol{w}_c\boldsymbol{v}_c'||_F + \kappa \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad \forall i, c \ \ & \langle \boldsymbol{X}_i, \boldsymbol{w}_{y_i}\boldsymbol{v}_{y_i}' \rangle_F - \langle \boldsymbol{X}_i, \boldsymbol{w}_c\boldsymbol{v}_c' \rangle_F + \delta_{y_i,c} \geq 1 - \xi_i, \end{aligned} \tag{3}$$

where N is the number of samples, y_i indicates the class label of the i-th sample; $y_i \in \{1, .., C\}$, $\delta_{y_i,c}$ equals to 1 if $c = y_i$ and 0 otherwise, ξ_i is a slack variable for soft margin, and κ is a parameter to control the trade-off between the margin and the training errors. Note that we minimize the Frobenius norm, not squared one in SVM. Since this problem is difficult to directly optimize, we employ the upper bound of the Frobenius norm:

$$||\boldsymbol{w}_c\boldsymbol{v}_c'||_F = ||\boldsymbol{w}_c||||\boldsymbol{v}_c|| \leq \frac{1}{2}(||\boldsymbol{w}_c||^2 + ||\boldsymbol{v}_c||^2). \tag{4}$$

Therefore, the optimization problem OP is modified to

$$\begin{aligned} \text{P':} \min_{\{\boldsymbol{w}_c,\boldsymbol{v}_c\},\{\xi_i\}} \ & \frac{1}{2}\left(\sum_{c=1}^{C} ||\boldsymbol{w}_c||^2 + ||\boldsymbol{v}_c||^2\right) + \kappa \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad \forall i, c \ \ & \boldsymbol{w}_{y_i}'\boldsymbol{X}_i\boldsymbol{v}_{y_i} - \boldsymbol{w}_c'\boldsymbol{X}_i\boldsymbol{v}_c + \delta_{y_i,c} \geq 1 - \xi_i. \end{aligned} \tag{5}$$

The weights $\boldsymbol{w}$ and $\boldsymbol{v}$ separately emerge as standard squared norm, which facilitates the optimization. It can be shown that the OP and P' have the identical optimum solution by using rescaling technique described in Sec. 3.3.

In the problem P', if the optimal weights $\overset{*}{\boldsymbol{v}}$ are obtained, the optimal classifier vectors are represented as $\overset{*}{\boldsymbol{w}}_c = \sum_{i=1}^{N} \overset{*}{\tau}_{ic}\boldsymbol{X}_i\overset{*}{\boldsymbol{v}}_c$, where $\overset{*}{\tau}_{ic}$ are the optimal dual variables [2]. Thus, the multi-class bilinear classifier in Eq.(1) results in

$$\begin{aligned} \overset{*}{\boldsymbol{w}}_c{}'\boldsymbol{X}\overset{*}{\boldsymbol{v}}_c &= \sum_{i=1}^{N} \overset{*}{\tau}_{ic}\overset{*}{\boldsymbol{v}}_c{}'\boldsymbol{X}_i'\boldsymbol{X}\overset{*}{\boldsymbol{v}}_c = \sum_{i=1}^{N} \overset{*}{\tau}_{ic} \sum_{r=1}^{R} \overset{*}{v}_c^{(r)\,2} \boldsymbol{x}_i^{(r)\prime}\boldsymbol{x}^{(r)} \\ &\rightarrow \sum_{i=1}^{N} \overset{*}{\tau}_{ic} \sum_{r=1}^{R} \overset{*}{v}_c^{(r)\,2} k_r(\boldsymbol{x}_i^{(r)}, \boldsymbol{x}^{(r)}), \end{aligned} \tag{6}$$

where $k_r(\boldsymbol{x}_i^{(r)}, \boldsymbol{x}^{(r)})$ is a kernel function on behalf of the inner-product of the r-th type of features, $\boldsymbol{x}_i^{(r)\prime}\boldsymbol{x}^{(r)}$, in kernel tricks. Note that the kernel functions can be differently defined for respective feature types. The squared weights $v_c^{(r)\,2}$ play a role to weight the kernel functions as in MKL, and produce the composite kernel function specialized to class c. In this case, we can introduce alternative nonnegative variables $d_c^{(r)} = v_c^{(r)\,2} \geq 0$ without loss of generality. The variables $\boldsymbol{d}$ are the weights for kernel functions, and therefore the above bilinear formulation is applicable to MC-MKL. The primal problem P' is reformulated to

$$\text{P:} \min_{\{\boldsymbol{w}_c, \boldsymbol{d}_c\}, \{\xi_i\}} \frac{1}{2}\left(\sum_{c=1}^{C} ||\boldsymbol{w}_c||^2 + \mathbf{1}'\boldsymbol{d}_c\right) + \kappa \sum_{i=1}^{N} \xi_i \tag{7}$$
$$\text{s.t. } \forall i, c \;\; \boldsymbol{w}'_{y_i} \boldsymbol{X}_i \boldsymbol{d}_{y_i}^{\frac{1}{2}} - \boldsymbol{w}'_c \boldsymbol{X}_i \boldsymbol{d}_c^{\frac{1}{2}} + \delta_{y_i,c} \geq 1 - \xi_i, \;\; \boldsymbol{d}_c \geq 0,$$

where $\boldsymbol{d}_c = [d_c^{(1)}, .., d_c^{(R)}]'$, and $\boldsymbol{d}_c^{\frac{1}{2}}$ is a component-wise square root of the vector $\boldsymbol{d}_c$. In the problem P, non-negativity constraint is additionally introduced to the problem P' (or OP). The bilinear classifier is finally obtained by

$$\overset{*}{\boldsymbol{w}}{}'_c \boldsymbol{X} \overset{*}{\boldsymbol{d}}{}_c^{\frac{1}{2}} = \sum_{i=1}^{N} \overset{*}{\tau}_{ic} \sum_{r=1}^{R} \overset{*}{d}{}_c^{(r)} \, k_r(\boldsymbol{x}_i^{(r)}, \boldsymbol{x}^{(r)}). \tag{8}$$

We describe the scheme to efficiently optimize P in the following section.

3 Optimization Methods

The primal problem P in Eq.(7) has the following dual form, similarly to [11]:

$$\max_{\{\boldsymbol{\tau}_i\}} \sum_{i=1}^{N} \boldsymbol{e}'_{y_i} \boldsymbol{\tau}_i, \;\; \text{s.t. } \forall i \;\; \boldsymbol{\tau}_i \leq \kappa \boldsymbol{e}_{y_i}, \;\; \mathbf{1}'\boldsymbol{\tau}_i = 0, \;\; \forall r, c \;\; \frac{1}{2} \sum_{i,j} k_r(\boldsymbol{x}_i^{(r)}, \boldsymbol{x}_j^{(r)}) \tau_{ic} \tau_{jc} \leq \kappa.$$

where $\boldsymbol{\tau}_i$ is the i-th C-dimensional dual variable, $\boldsymbol{e}_{y_i}$ is a C-dimensional vector in which only the y_i-th element is 1 and the others are 0, and $\mathbf{1}$ is a C-dimensional vector of which all elements are 1. This is a convex problem having the global optimum. However, it is actually solved by second order cone programming, which requires exhaustive computational cost, and it is not applicable to large-scaled samples. Therefore, we take an alternative scheme to optimize the primal problem P in a manner similar to [7,11]. The scheme is based on the iterative optimization for $\boldsymbol{w}$ and $\boldsymbol{d}$, with applying projected gradient descent.

3.1 Optimization with Respect to $\boldsymbol{w}$

At the t-th iteration with fixing the variable $\boldsymbol{d}$ to $\boldsymbol{d}^{[t]}$, in a manner similar to [2], the problem P results in the dual form:

$$\max_{\{\boldsymbol{\tau}_i\}} \; -\frac{1}{2} \sum_{i,j=1}^{N} \sum_{c=1}^{C} (\boldsymbol{v}_c^{[t]\prime} \boldsymbol{X}_i' \boldsymbol{X}_j \boldsymbol{v}_c^{[t]}) \tau_{ic} \tau_{jc} + \sum_{i=1}^{N} \boldsymbol{e}'_{y_i} \boldsymbol{\tau}_i$$
$$\Leftrightarrow \; \text{D}_w\text{: } \max_{\{\boldsymbol{\tau}_i\}} \; -\frac{1}{2} \sum_{i,j=1}^{N} \boldsymbol{\tau}_i' \boldsymbol{\Lambda}_{ij} \boldsymbol{\tau}_j + \sum_{i=1}^{N} \boldsymbol{e}'_{y_i} \boldsymbol{\tau}_i, \quad \text{s.t. } \forall i \;\; \boldsymbol{\tau}_i \leq \kappa \boldsymbol{e}_{y_i}, \;\; \mathbf{1}'\boldsymbol{\tau}_i = 0, \tag{9}$$

where $\boldsymbol{\Lambda}_{ij}$ is a C-dimensional diagonal matrix, $\{\boldsymbol{\Lambda}_{ij}\}_{cc} = \sum_{r=1}^{R} d_c^{(r)[t]} k_r(\boldsymbol{x}_i^{(r)}, \boldsymbol{x}_j^{(r)})$. In this dual problem, the constants derived from $\boldsymbol{d}^{[t]}$ are omitted. This is optimized by iteratively solving the decomposed small subproblem [2,3], as follows.

Algorithm 1. Optimization for subproblem SubD_w

Require: Reindex $\tilde{b}_c = \frac{b_c}{\lambda_c}$ and λ_c such that $\tilde{b}_c$ are sorted in decreasing order

Initialize $c = 2$, $\zeta_{num} = \lambda_1^2 \tilde{b}_1 - \kappa$, $\zeta_{den} = \lambda_1^2$.

while $c \leq C$, $\zeta = \frac{\zeta_{num}}{\zeta_{den}} \leq \tilde{b}_c$ **do**

$\zeta_{num} \leftarrow \zeta_{num} + \lambda_c^2 \tilde{b}_c$, $\zeta_{den} \leftarrow \zeta_{den} + \lambda_c^2$, $c \leftarrow c + 1$.

end while

Output: $\tilde{\tau}_c = \min(b_c, \zeta \lambda_c)$, $\therefore \tau_c = \min\{\kappa \delta_{y,c}, \lambda_c^2(\zeta - \beta_c)\}$.

The dual problem D_w is decomposed into N small subproblems; the i-th subproblem focuses on the dual variable $\boldsymbol{\tau}_i$ associated with the i-th sample, while fixing the others $\boldsymbol{\tau}_j$ $(j \neq i)$:

$$\mathrm{SubD}_w\text{: } \max_{\boldsymbol{\tau}_i} \; -\frac{1}{2}\boldsymbol{\tau}_i' \boldsymbol{\Lambda}_{ii} \boldsymbol{\tau}_i - \boldsymbol{\beta}' \boldsymbol{\tau}_i - \gamma, \quad \text{s.t. } \boldsymbol{\tau}_i \leq \kappa \boldsymbol{e}_{y_i}, \; \mathbf{1}' \boldsymbol{\tau}_i = 0, \tag{10}$$

where

$$\boldsymbol{\beta} = \sum_{j \neq i} \boldsymbol{\Lambda}_{ij} \boldsymbol{\tau}_j - \boldsymbol{e}_{y_i}, \quad \gamma = \frac{1}{2} \sum_{j \neq i, k \neq i} \boldsymbol{\tau}_j \boldsymbol{\Lambda}_{jk} \boldsymbol{\tau}_k - \sum_{j \neq i} \boldsymbol{e}_{y_j}' \boldsymbol{\tau}_j.$$

For optimization in D_w, the process to solve SubD_w works in rounds for all i and the dual variables $\boldsymbol{\tau}_i$ are updated until convergence. The subproblems are rather more complex than those in [2] since they include not scalar value $\boldsymbol{x}_i' \boldsymbol{x}_j$ but the diagonal matrix $\boldsymbol{\Lambda}_{ij}$ derived from multiple features (kernels). However, it is noteworthy that they are solved at a quite low computational cost, as follows.

Optimization for Subproblem SubD_w

In the following, we omit the index i for simplicity. By ignoring the constant, the subproblem SubD_w in Eq.(10) is reformulated to

$$\min_{\tilde{\boldsymbol{\tau}}} \; \frac{1}{2}||\tilde{\boldsymbol{\tau}}||^2, \quad \text{s.t. } \tilde{\boldsymbol{\tau}} \leq \kappa \lambda_y^{-1} \boldsymbol{e}_y + \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{\beta}, \quad \boldsymbol{\lambda}' \tilde{\boldsymbol{\tau}} = \boldsymbol{\lambda}' \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{\beta},$$

where $\tilde{\boldsymbol{\tau}} = \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\tau} + \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{\beta}$, $\boldsymbol{\lambda}$ is a C-dimensional vector composed of diagonal elements of $\boldsymbol{\Lambda}^{-\frac{1}{2}}$, and λ_y is the y-th element of the vector $\boldsymbol{\lambda}$. By using $\boldsymbol{b} = \kappa \lambda_y^{-1} \boldsymbol{e}_y + \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{\beta}$, the constraints are rewritten as

$$\text{s.t. } \tilde{\boldsymbol{\tau}} \leq \boldsymbol{b}, \quad \boldsymbol{\lambda}' \tilde{\boldsymbol{\tau}} = \boldsymbol{\lambda}' \boldsymbol{b} - \kappa. \tag{11}$$

The Lagrangian for this problem is

$$L = \frac{1}{2}||\tilde{\boldsymbol{\tau}}||^2 - \boldsymbol{\alpha}'(\boldsymbol{b} - \tilde{\boldsymbol{\tau}}) - \zeta(\boldsymbol{\lambda}' \tilde{\boldsymbol{\tau}} - \boldsymbol{\lambda}' \boldsymbol{b} + \kappa), \tag{12}$$

where $\boldsymbol{\alpha} \geq 0$, ζ are Lagrangian multipliers. When the subproblem is optimized, the followings hold:

$$\frac{\partial L}{\partial \tilde{\boldsymbol{\tau}}} = \tilde{\boldsymbol{\tau}} + \boldsymbol{\alpha} - \zeta \boldsymbol{\lambda} = 0, \quad \text{KKT: } \forall c \;\; \alpha_c(b_c - \tilde{\tau}_c) = 0. \tag{13}$$

Therefore, we obtain

$$\alpha_c = 0 \Rightarrow \tilde{\tau}_c = \zeta\lambda_c,\ \ \zeta \leq \frac{b_c}{\lambda_c}, \qquad \alpha_c > 0 \Rightarrow \tilde{\tau}_c = b_c,\ \zeta > \frac{b_c}{\lambda_c}. \tag{14}$$

By using the above, the second constraint in Eq.(11) results in

$$\boldsymbol{\lambda}'\tilde{\boldsymbol{\tau}} = \zeta \sum_{c|\alpha_c=0} \lambda_c^2 + \sum_{c|\alpha_c>0} \lambda_c b_c = \sum_{c=1}^{C} \lambda_c b_c - \kappa, \quad \therefore \zeta = \frac{\sum_{c|\alpha_c=0} \lambda_c b_c - \kappa}{\sum_{c|\alpha_c=0} \lambda_c^2}. \tag{15}$$

Thus, for solving the subproblem, we only seek ζ satisfying Eq.(14) and (15), and the simple algorithm is constructed in Algorithm 1.

The optimization of D_w is the core and most exhaustive process for the whole optimization in P. Therefore, the effective algorithm (Algorithm 1) to solve the subproblem SubD_w makes the whole optimization process computationally efficient.

3.2 Optimization with Respect to $\boldsymbol{d}$

Then, the optimization of P is performed with respect to $\boldsymbol{d}$. In this study, we simply employ projected gradient descent approach, although the other method such as in [12] would be applicable. In this approach, the objective cost function is minimized by a line search [6] along the projected gradient under the constraints $\boldsymbol{d} \geq 0$. Based on the principle of strong duality, the primal P is represented by using D_w in Eq.(9) with the optimal dual variables $\boldsymbol{\tau}^{[t]}$ as

$$\min_{\{\boldsymbol{d}_c\}} \left\{ \left(\sum_{c=1}^{C} \frac{1}{2}\mathbf{1}'\boldsymbol{d}_c - \boldsymbol{\theta}_c'\boldsymbol{d}_c \right) + \sum_{i=1}^{N} \boldsymbol{e}_{y_i}'\boldsymbol{\tau}_i^{[t]} = W(\boldsymbol{d}) \right\}, \qquad \text{s.t. } \forall c\ \ \boldsymbol{d}_c \geq 0,$$

where $\boldsymbol{\theta}_c$ is a R-dimensional vector of $\theta_c^{(r)} = \frac{1}{2}\sum_{i,j} \tau_{ic}^{(r)[t]} \tau_{jc}^{(r)[t]} k_r(\boldsymbol{x}_i^{(r)}, \boldsymbol{x}_j^{(r)})$. In this case, W is differentiable with respect to $\boldsymbol{d}$ (ref. [7]), and thus the gradients are obtained as $\nabla W = \frac{1}{2}\mathbf{1} - \boldsymbol{\theta}_c$. Thereby, the optimization in P is performed by using projected gradient descent, $\boldsymbol{d}^{[t+1]} = \boldsymbol{d}^{[t]} - \epsilon\nabla W$. We apply a line search [6] to greedily seek the parameter ϵ such that W, i.e., the objective cost function in P, is minimized while ensuring $\boldsymbol{d} \geq 0$. Note that, in this greedy search, the cost function is evaluated several times via optimization of D_w.

3.3 Rescaling

After optimization of D_w with the fixed $\boldsymbol{d}$, the cost function is further decreased by simply rescaling the variables of $\boldsymbol{\tau}$ and $\boldsymbol{d}$, so as to reach the lower bound in Eq.(4). The rescaling, $\hat{\tau}_{ic} = s_c\tau_{ic}$, $\hat{\boldsymbol{d}}_c = \frac{1}{s_c}\boldsymbol{d}_c$, does not affect the bilinear projection in Eq.(8) and thus the constraints in P are kept:

$$\hat{\boldsymbol{w}}_c'\boldsymbol{X}\hat{\boldsymbol{d}}_c^{\frac{1}{2}} = \sum_{i=1}^{N} s_c\tau_{ic} \sum_{r=1}^{R} \frac{d_c^{(r)}}{s_c} k_r(\boldsymbol{x}, \boldsymbol{x}_i) = \boldsymbol{w}_c'\boldsymbol{X}\boldsymbol{d}_c^{\frac{1}{2}}, \tag{16}$$

while the first term in the cost function is transformed to

$$\frac{1}{2}\sum_{c=1}^{C}||\hat{\boldsymbol{w}}_c||^2 + \mathbf{1}'\hat{\boldsymbol{d}}_c = \frac{1}{2}\sum_{c=1}^{C} s_c||\boldsymbol{w}_c||^2 + \frac{1}{s_c}\mathbf{1}'\boldsymbol{d}_c. \tag{17}$$

The optimal rescaling that minimizes the above is analytically obtained as $\overset{*}{s}_c = \sqrt{\mathbf{1}'\boldsymbol{d}_c}/||\boldsymbol{w}_c||$, and Eq.(17) equals to the lower bound (the Frobenius norm):

$$\frac{1}{2}\sum_{c=1}^{C}||\hat{\boldsymbol{w}}_c||^2 + \mathbf{1}'\hat{\boldsymbol{d}}_c = \sum_{c=1}^{C}\sqrt{\mathbf{1}'\boldsymbol{d}_c}||\boldsymbol{w}_c|| = \sum_{c=1}^{C}||\boldsymbol{w}_c\boldsymbol{d}_c^{\frac{1}{2}\prime}||_F. \tag{18}$$

Although the rescaled $\hat{\boldsymbol{\tau}}$ is not necessarily the solution of the problem D_w with the rescaled $\hat{\boldsymbol{d}}$, in the greedy optimization for $\boldsymbol{d}$, the gradients using $\hat{\boldsymbol{\tau}}$ are employed as the approximation for $\nabla W(\hat{\boldsymbol{d}})$. This rescaling contributes to fast convergence.

4 Experimental Result

We show the classification performances and computation time of the proposed methods in comparison with the other MKL methods [7,13] on various datasets. We employed the 1-vs-all version of [7] to cope with multi-class problems. The proposed method is implemented by using MATLAB with C-mex on Xeon 3GHz PC. For the methods of [7,13], we used the MATLAB codes provided by the authors and combined them with libsvm [1] and MOSEK optimization toolbox in order to speed up those methods as much as possible. In this experiment, the parameter values in the all methods are set as follows: κ is determined from $\kappa \in \{0.5, 1, 10\}$ based on 3-fold cross validation and the maximum number of iterations is set to 40 iterations for fair comparison of computation time. These methods almost converge on various datasets within 40 iterations.

First, we used four benchmark datasets: *waveform* from UCI Machine Learning Repository, *satimage* and *segment* from the STATLOG project, and *USPS* [4]. The multiple RBF kernels with 10 σ's (uniformly selected on the logarithmic scale over $[10^{-1}, 10^2]$) were employed. We drew 1000 random training samples and classified the remained samples. The trial is repeated 10 times and the average performance is reported. Fig. 1(a,b) shows the classification results (error rates) and computation time on those datasets. While the performances of the proposed method are competitive to the others, the computation time is much more reduced; especially, more than 20 times faster than the method of [13].

Next, we applied the proposed method to the other practical classification problems in cell biology. The task is to predict the sub-cellular localizations of proteins, and in this case it results in multi-class classification problems. We employed a total of 69 kernels of which details are described in [13]. MKL would

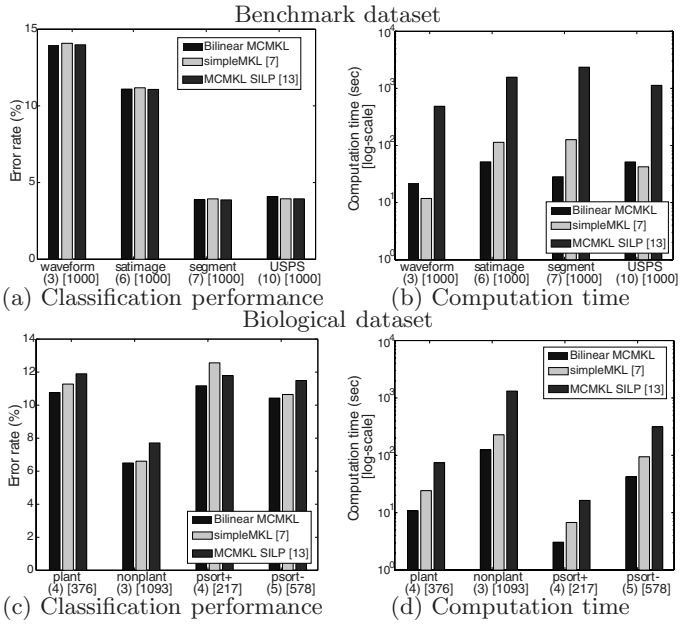

Fig. 1. The classification performances (error rates) and computation time on the four benchmark datasets. The number of classes is indicated in parentheses and that of training samples is in brackets. The left bar shows the result of the proposed methods.

be effectively applied to these substantial types of kernel. In this experiment, we used four biological datasets [13]: *plant*, *nonplant*, *psort+*, and *psort-*. We randomly split the dataset into 40% for training and 60% for testing. The trial is repeated 10 times and the average performance is reported. The results are shown in Fig. 1(c,d), demonstrating that the proposed method is quite effective; the proposed method is superior and faster to the methods of [7,13]. The experimental result shows that the proposed method effectively and efficiently combines a lot of heterogeneous kernel functions.

5 Conclusion

We have proposed a multiple kernel learning (MKL) method to deal with multi-class problems. In the proposed method, the multi-class classification using multiple kernels is formulated in the bilinear form, and the computationally efficient optimization scheme is proposed in order to be applicable to large-scaled samples. In the experiments on the benchmarks and the biological datasets, the proposed method exhibited the favorable performances and computation time compared to the previous methods of MKL.

References

1. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
2. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research 2, 265–292 (2001)
3. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. Journal of Machine Learning Research 9(6), 1871–1874 (2008)
4. Hull, J.: A database for handwritten text recognition research. IEEE Trans. Pattern Analysis and Machine Intelligence 16(5), 550–554 (1994)
5. Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research 5, 27–72 (2004)
6. Nocedal, J., Wright, S. (eds.): Numerical optimization. Springer, New York (1999)
7. Rakotomamonjy, A., Bach, F.R., Canu, S., Grandvalet, Y.: Simplemkl. Journal of Machine Learning Research 9, 2491–2521 (2008)
8. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2001)
9. Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B.: Large scale multiple kernel learning. Journal of Machine Learning Research 7, 1531–1565 (2006)
10. Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)
11. Varma, M., Ray, D.: Learning the discriminative power-invariance trade-off. In: Proceedings of IEEE 11th International Conference on Computer Vision (2007)
12. Xu, Z., Jin, R., King, I., Lyu, M.R.: An extended level method for efficient multiple kernel learning. In: Advances in Neural Information Processing Systems, vol. 21, pp. 1825–1832 (2008)
13. Zien, A., Ong, C.S.: Multiclass multiple kernel learning. In: Proceedings of the 24th International Conference on Machine Learning (2007)