Class Prediction from Disparate Biological Data Sources using an Iterative Multi-kernel Algorithm

Yiming Ying†, Colin Campbell†, Theodoros Damoulas‡, and Mark Girolami‡

†, Department of Engineering Mathematics,
University of Bristol, Bristol BS8 1TR, United Kingdom
‡Department of Computer Science,
University of Glasgow,Glasgow, G12 8QQ, United Kingdom
mathying@gmail.com,C.Campbell@bris.ac.uk,
theo@dcs.gla.ac.uk,girolami@dcs.gla.ac.uk

Abstract. For many biomedical modelling tasks a number of different types of data may influence predictions made by the model. An established approach to pursuing supervised learning with multiple types of data is to encode these different types of data into separate kernels and use *multiple kernel learning*. In this paper we propose a simple iterative approach to multiple kernel learning (MKL), focusing on multi-class classification. This approach uses a block L^1-regularization term leading to a jointly convex formulation. It solves a standard multi-class classification problem for a single kernel, and then updates the kernel combinatorial coefficients based on mixed RKHS norms. As opposed to other MKL approaches, our iterative approach delivers a largely ignored message that MKL does not require sophisticated optimization methods while keeping competitive training times and accuracy across a variety of problems. We show that the proposed method outperforms state-of-the-art results on an important protein fold prediction dataset and gives competitive performance on a protein subcellular localization task.

Key words: Multiple kernel learning, multi-class, bioinformatics, protein fold prediction, protein subcellular localization

1 Introduction

Kernel methods [15, 16] have been successfully used for data integration across a number of biological applications. *Kernel matrices* encode the similarity between data objects within a given space. Data objects can include network graphs and sequence strings in addition to numerical data: all of these types of data can be encoded into kernels. The problem of data integration is therefore transformed into the problem of learning the most appropriate combination of candidate kernel matrices and typically a linear combination is used. This is often termed multi-kernel learning (MKL) in Machine Learning and, due to its practical importance, it has recently received increased attention. Lanckriet et al. [9] proposed a semi-definite programming (SDP) approach to automatically learn a

linear combination of candidate kernels for SVMs. This approach was improved by Bach et al. [3] who used sequential minimization optimization (SMO) and by Sonnenburg et al. [18] who reformulated it as a semi-infinite linear programming (SILP) task. In [11], the authors studied the kernel learning problem for a convex set of possibly infinite kernels under a general regularization framework. Other approaches include the COSSO estimate for additive models [10], Bayesian probabilistic models [5, 8], kernel discriminant analysis [20], hyperkernels [12] and kernel learning for structured outputs [21]. Such MKL formulations have been successfully demonstrated in combining multiple data sources to enhance biological inference [5, 9].

Most of the above MKL methods were for binary classification. In Section 2 we build on previous contributions [1, 3, 10, 11, 13, 21] to propose a simple iterative kernel learning approach focusing on multi-class problems. This formulation employs a mixed RKHS norm over a matrix-valued function which promotes common information across classes. We demonstrate that this problem is jointly convex, laying down the theoretical basis for its solution using an extremely simple iterative method. This approach solves a multi-class classification problem for a single kernel, and then updates the kernel combinatorial coefficients based on the mixed RKHS norms. As opposed to other multi-kernel approaches, our iterative approach delivers an important message that MKL does not require sophisticated optimization methods while keeping competitive training times and accuracy across a wide range of problems. In Section 3 we briefly validate our method on UCI benchmark multi-class datasets before applying it to two multi-class multi-feature bioinformatics problems: protein fold recognition and protein subcellular localization.

2 The Learning Method

Let $\mathbb{N}_n = \{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}$ and input/outputs sample $\mathbf{z} = \{(x_i, y_i) : i \in \mathbb{N}_n\}$ with $\mathbf{y} = \{y_i \in [1, C] : i \in \mathbb{N}_n\}$ where C is the number of classes. For input sample x_i, there are m different sources of information (feature spaces), i.e. $x_i = (x_i^1, x_i^2, \ldots, x_i^m)$ with x_i^ℓ from ℓ-th data source for any $\ell \in \mathbb{N}_m$.

To introduce the learning model, we employ a one-versus-all strategy that encodes the multi-class classification problem as a set of binary ones. To this end, we reconstruct the output vector $\mathbf{y}_i = (y_{i1}, \ldots, y_{iC})$ such that $y_{ic} = 1$ if $y_i = c$ and otherwise -1. Hence the outputs are represented by an $n \times C$ *indicator matrix* $\mathbf{Y} = (y_{ic})_{i,c}$ whose c-th column vector is denoted by $\mathbf{y}^c$. For source ℓ and class c, we use a reproducing kernel space $\mathcal{H}_\ell$ with reproducing kernel K^ℓ to represent this dataset. In particular, let $\mathbf{f} = (f_{\ell c})$ be a matrix-valued function[1]. For each class c and data source ℓ we use a function $f_{\ell c} \in \mathcal{H}_\ell$ to learn the output. Then, we simply use the composite function defined by

$$f_c(x_i) = \sum_{\ell \in \mathbb{N}_m} f_{\ell c}(x_i^\ell)$$

[1] We denote with bold type a vector or matrix, e.g. $f_{\ell c}$ is a real function while $\mathbf{f}^c$ denotes a vector of functions and $\mathbf{f}$ denotes a matrix of functions.

to combine m sources. The accuracy of the approximation at sample i can be measured by e.g. $\big(y_{ic} - f_c(x_i)\big)^2$. However, taking the direct minimization of the above empirical error will inevitably lead to overfitting. Hence, we need to enforce some penalty term on $\mathbf{f}$. Since we expect to get good performance after combining multiple sources, the penalty term intuitively should play the role of removing redundant sources (feature spaces) across classes. With this motivation, we introduce a block L^1 regularization on the matrix-valued function $\mathbf{f} = (f_{\ell c})$. This kind of regularization was used in [1] for multi-task linear feature learning and also used in [3, 10, 11, 13] for binary classification kernel learning with block regularization over a vector of functions instead of over a matrix-valued function. More specifically, let $\|\mathbf{f}\|_{(2,1)} = \sum\limits_{\ell\in\mathbb{N}_m} \Big(\sum\limits_{c\in\mathbb{N}_C} \|f_{\ell c}\|^2_{\mathcal{H}_\ell} \Big)^{\frac{1}{2}}$. We now propose the following multi-class multiple kernel learning formulation with least square loss. One can easily extend the method and the followed arguments to other loss functions.

$$\begin{array}{l} \min\limits_{\mathbf{f}} \mu \sum\limits_{i\in\mathbb{N}_n} \sum\limits_{c\in\mathbb{N}_C} \big(y_{ic} - \sum\limits_{\ell\in\mathbb{N}_m} f_{\ell c}(x_i^\ell)\big)^2 + \dfrac{1}{2}\|\mathbf{f}\|^2_{(2,1)} \\ \text{s.t.} \quad f_{\ell c} \in \mathcal{H}_\ell, \quad \forall\, c \in \mathbb{N}_C, \ell \in \mathbb{N}_m \end{array} \tag{1}$$

The mixed $(2,1)$-norm of $\mathbf{f}$ in the regularization term is obtained by first computing the $\mathcal{H}_\ell$-norm of the row vector (across all classes) $\mathbf{f}_\ell = (f_{\ell 1}, \ldots, f_{\ell C})$ and then the 1-norm $\mathcal{F}(\mathbf{f}) = ((\sum_c \|f_{1c}\|^2_{\mathcal{H}_1})^{\frac{1}{2}}, \ldots, (\sum_c \|f_{mc}\|^2_{\mathcal{H}_m})^{\frac{1}{2}})$. Consequently, the 1-norm of vector $\mathcal{F}(\mathbf{f})$ (mixed norm term of $\mathbf{f}$) encourages a sparse representation of the candidate RKHSs $\{\mathcal{H}_\ell : \ell \in \mathbb{N}_m\}$ for the learning task, and thus implies automatically adapting the combination of multiple sources.

In order to deal with the non-differential L^1 regularizer of equation (1), we turn to an equivalent form. To this end, recall [11], for any $w = (w_1, \ldots, w_m) \in \mathbb{R}^m$, that $\big(\sum_{\ell\in\mathbb{N}_m} |w_\ell|\big)^2 = \min\big\{\sum_{\ell\in\mathbb{N}_m} \frac{w_\ell^2}{\lambda_\ell} : \sum_{\ell\in\mathbb{N}_m} \lambda_\ell = 1, \lambda_\ell \geq 0\big\}$. Now, we replace w_ℓ by $\big(\sum_{c\in\mathbb{N}_C} \|f_\ell^c\|^2_{\mathcal{H}_\ell}\big)^{\frac{1}{2}}$ and obtain the following equivalent formulation of equation (1):

$$\begin{array}{ll} \min_{\mathbf{f},\lambda} & \mu \sum\limits_{i\in\mathbb{N}_n} \sum\limits_{c\in\mathbb{N}_C} \Big(y_{ic} - \sum\limits_{\ell\in\mathbb{N}_m} f_{\ell c}(x_i^\ell)\Big)^2 \\ & \quad + \frac{1}{2} \sum\limits_{\ell\in\mathbb{N}_m} \sum\limits_{c\in\mathbb{N}_C} \dfrac{\|f_{\ell c}\|^2_{\mathcal{H}_\ell}}{\lambda_\ell} \\ \text{s.t.} & \sum_{\ell\in\mathbb{N}_m} \lambda_\ell = 1, \ \lambda_\ell \geq 0 \\ & \text{and } f_{\ell c} \in \mathcal{H}_\ell, \quad \forall\, c \in \mathbb{N}_C, \ell \in \mathbb{N}_m\ . \end{array} \tag{2}$$

From the auxiliary regularization term $\sum_{\ell\in\mathbb{N}_n} \sum_{c\in\mathbb{N}_C} \|f_{\ell c}\|^2_{\mathcal{H}_\ell}/\lambda_\ell$ in equation (2), we note that if λ_ℓ is close to zero then $\sum_{c\in\mathbb{N}_C} \|f_{\ell c}\|^2_{\mathcal{H}_\ell}$ should also be close to zero as we are minimizing the objective function. This intuitively explains the role of the auxiliary variable λ.

The following theorem demonstrates the joint convexity of problem (2) which could be shown by adapting the argument in [4]. For completeness, we outline a proof here.

Theorem 1. *The objective function in (2) is jointly convex with respect to* $\mathbf{f}$ *and* λ.

Proof: It suffices to prove the joint convexity of $\|f\|^2_{\mathcal{H}_\ell}/\lambda$ with respect to $f \in \mathcal{H}_\ell$ and $\lambda \in (0,1),\ \forall\, \ell \in \mathbb{N}_m$. The proof is parallel to that in [2]. For completeness, we briefly prove it again here.

We need to show, for any $f_1, f_2 \in \mathcal{H}_\ell$ and $\lambda_1, \lambda_2 \in (0,1)$ and $\theta \in (0,1)$, that

$$\frac{\|\theta f_1 + (1-\theta) f_2\|^2_{\mathcal{H}_\ell}}{\theta\lambda_1 + (1-\theta)\lambda_2} \leq \frac{\|\theta f_1\|^2_{\mathcal{H}_\ell}}{\theta\lambda_1} + \frac{\|(1-\theta) f_2\|^2_{\mathcal{H}_\ell}}{(1-\theta)\lambda_2}$$

Let $a = \frac{1}{\lambda_1\theta}, b = \frac{1}{(1-\theta)\lambda_2}, c = \frac{1}{\theta\lambda_1+(1-\theta)\lambda_2}$ and $F = \theta f_1 + (1-\theta) f_2, G = \theta f_1$. Since f_1, f_2 is arbitrary, the above equation is reduced to the following:

$$c\|F\|^2_{\mathcal{H}_\ell} \leq a\|G\|^2_{\mathcal{H}_\ell} + b\|F-G\|^2_{\mathcal{H}_\ell}, \quad \forall F, G \in \mathcal{H}_\ell.$$

Equivalently,

$$\begin{aligned} c\|F\|^2_{\mathcal{H}_\ell} &\leq \min_{G\in\mathcal{H}_\ell} a\|G\|^2_{\mathcal{H}_\ell} + b\|F-G\|^2_{\mathcal{H}_\ell} \\ &= \|F\|^2_{\mathcal{H}_\ell}\tfrac{b^2 a}{(a+b)^2} + \|F\|^2_{\mathcal{H}_\ell}\tfrac{a^2 b}{(a+b)^2}, \quad \forall F \in \mathcal{H}_\ell. \end{aligned}$$

which is obviously true by the definition of a, b, c. This completes the proof of the convexity. □

Let the composite kernel K_λ be defined by $K_\lambda = \sum_{\ell\in\mathbb{N}_m} \lambda_\ell K^\ell$. Then, the role of λ becomes more intuitive if we use the following dual formulation of (2):

$$\begin{array}{ll} \min_\lambda \max_\alpha & \sum_{i,c} \alpha_{ic} y_{ic} - \frac{1}{4\mu}\sum_{i,c}\alpha^2_{ic} \\ & \quad - \frac{1}{2}\sum_{i,j,c}\alpha_{ic}\alpha_{jc}K_\lambda(x^\ell_i, x^\ell_j) \\ \text{s.t.} & \sum_{\ell\in\mathbb{N}_m}\lambda_\ell = 1, \lambda_\ell \geq 0. \end{array}$$

which can be directly derived from the dual of kernel ridge regression [16] by first fixing λ. It is worth noting that for the equally weighted kernel combination, i.e. $\lambda = \frac{1}{m}$, equation (2) is reduced to a formulation with a plain L^2-regularization term $\sum_{\ell c}\|f_{\ell c}\|^2_{\mathcal{H}_\ell}$. We also note that [14] proposed a multi-class kernel learning algorithm based on one-against strategy starting from the dual formulation of SVM.

We can formulate (2) as a semi-infinite linear programming (SILP) problem, as in [18, 20]. First, however, we propose a conceptually simple implementation based on Theorem 1 which will be referred to as **MCKL-EM** hereafter.

We will initialize $\lambda^{(0)}$ with $\lambda^{(0)}_\ell = \frac{1}{m}$ for any $\ell \in \mathbb{N}_m$. We then solve (2) for this equally weighted kernel coefficient $\lambda^{(0)}$ and get $\mathbf{f}^{(0)}$ which is a least-square ridge regression problem. Next, for any $t \in \mathbb{N}$ we update $\lambda^{(t)}$ for fixed $\mathbf{f}^{(t-1)}$ and update $\mathbf{f}^{(t)}$ for fixed $\lambda^{(t)}$. We repeat the above **EM-type** iteration until convergence. This can reasonably be monitored by the changes of kernel combinatorial coefficients $\sum_{\ell\in\mathbb{N}_m}|\lambda^{\text{old}}_\ell - \lambda_\ell|$ or changes of the objective function, since we are mainly interested in obtaining an optimal kernel combination. Global convergence is expected since the overall problem (2) is jointly convex by Theorem 1. The updates at step $t \in \mathbb{N}$ are listed as follows:

1. For fixed $\mathbf{f}^{(t-1)}$, $\lambda_\ell^{(t)} = \frac{(\sum_c \|f_{\ell c}^{(t-1)}\|_{\mathcal{H}_\ell}^2)^{\frac{1}{2}}}{\sum_\ell (\sum_c \|f_{\ell c}^{(t-1)}\|_{\mathcal{H}_\ell}^2)^{\frac{1}{2}}}$ for any $\ell \in \mathbb{N}_m$. Here we denote the matrix function $\mathbf{f}^{(t-1)} = (f_{\ell c}^{(t-1)})_{\ell c}$.
2. For given $\lambda^{(t)}$, $f_{\ell c}^{(t)}(\cdot) = \lambda_\ell^{(t)} \sum_i \alpha_{ic}^{(t)} K^\ell(x_i^\ell, \cdot)$. Here, $\alpha^{(t)} = (\alpha_{ic}^{(t)})$ is an $n \times C$ matrix given by the equation

$$\alpha^{(t)} = (\mathbf{K}_{\lambda^{(t)}} + \mathbf{I}/2\mu)^{-1}\mathbf{Y} \tag{3}$$

where $\mathbf{K}_{\lambda^t} = \left(\sum_\ell \lambda_\ell^{(t)} K^\ell(x_i^\ell, x_j^\ell)\right)$.

The second update equation follows from standard kernel ridge regression [16] for fixed λ. The first update for λ follows from the fact that $\{|w_1|/\sum_{\ell\in\mathbb{N}_m}|w_\ell|, \ldots |w_m|/\sum_{\ell\in\mathbb{N}_m}|w_\ell|\}$ is the optimizer of the minimization problem $\min\{\sum_{\ell\in\mathbb{N}_m}\frac{w_\ell^2}{\lambda_\ell} : \sum_{\ell\in\mathbb{N}_m}\lambda_\ell = 1, \lambda_\ell \geq 0\}$. Let the convergent solution be $\hat{\mathbf{f}}$. Given a new sample x_*, then we assign its class by $y^* = \arg\max_c \sum_\ell \hat{f}_{\ell c}(x_*)$.

Recently, the SILP approach has been applied to kernel learning problems for large scale datasets, see [18, 20, 21]. Since we later use a SILP approach for comparison (MCKL-SILP) we briefly described this variant here. In a similar fashion to arguments in [18], we can formulate the dual problem as an semi-infinite linear programming. Specifically, let $S_0(\alpha) = \sum_{c,i}\alpha_{ic}y_{ic} - \frac{1}{4\mu}\sum_{c,i}\alpha_{ic}^2$ and, for any $\ell \in \mathbb{N}_m$, $S_\ell(\alpha) = \frac{1}{2}\sum_{c,i,j}\alpha_{ic}\alpha_{jc}K^\ell(x_i, x_j)$. Then, the SILP formulation of algorithm (2) is stated as

$$\begin{array}{ll} \max_{\gamma,\beta}\ \gamma & \\ \text{s.t.} & \sum_{\ell\in\mathbb{N}_m}\lambda_\ell = 1, \quad 0 \leq \lambda \leq 1 \\ & \gamma - \sum_{\ell\in\mathbb{N}_m}\lambda_\ell S_\ell(\alpha) \leq S_0(\alpha), \forall\alpha. \end{array} \tag{4}$$

The SILP can be solved by an iterative algorithm called *column generation* (or exchange methods) which is guaranteed to converge to a global optimum. The basic idea is to compute the optimum (λ, γ) by linear programming for a restricted subset of constraints, and update the constraint subset based on the obtained suboptimal (λ, γ).

Given a set of restricted constraints $\{\alpha_p : p \in \mathbb{N}_P\}$, first we find the intermediate solution (λ, γ) by the following linear programming optimization with P linear constraints

$$\begin{array}{ll} \max_{\gamma,\lambda}\ \gamma & \\ \text{s.t.} & \sum_\ell \lambda_\ell = 1, \quad 0 \leq \lambda \leq 1 \\ & \gamma - \sum_\ell \lambda_\ell S_\ell(\alpha_p) \leq S_0(\alpha), \forall p \in \mathbb{N}_P. \end{array} \tag{5}$$

This problem is often called the *restricted master problem*. Then, we find the next constraint with the maximum violation for the given intermediate solution (λ, γ), i.e. $\min_\alpha \sum_{\ell\in\mathbb{N}_m}\lambda_\ell S_\ell(\alpha) + S_0(\alpha)$. If its optimal α^* satisfies $\sum_\ell \lambda_\ell S_\ell(\alpha^*) + S_0(\alpha^*) \geq \gamma$ then current intermediate solution (λ, γ) is optimal for the optimization (4). Otherwise α^* should be added to the restriction set. We repeat the above

iteration until convergence which is guaranteed to be globally optimal, see e.g. [18]. The convergence criterion for the SILP is usually chosen as

$$\left| 1 - \frac{\sum_\ell \lambda_\ell^{(t-1)} S_\ell(\alpha^{(t)}) + S_0(\alpha^{(t)})}{\gamma^{(t-1)}} \right| \leq \epsilon. \tag{6}$$

3 Experiments

3.1 Validation on UCI datasets

In this section we briefly validate MCKL-EM on UCI datasets [19], to illustrate its performance, before proceeding to bioinformatics datasets. For fairness of comparison, in all kernel learning algorithms we chose the change of kernel weights, $\sum_\ell |\lambda_\ell^{\text{old}} - \lambda_\ell| \leq \varepsilon = 10^{-4}$, as the stopping criterion, and the parameter μ was set at a value of 10.

We compared our iterative approach (MCKL-EM) with the SILP approach (MCKL-SILP) and its doubly cross-validated method (LSR-CV) over μ and σ. The results are based on 10 random data splits into 60% training and 40% test. We can see from Table 1 that there is no significant difference between MCKL-EM and MCKL-SILP with respect to both computation time and test set accuracy (TSA), despite the fact that MCKL-EM is much simpler to implement. These accuracies are also equal to, or better than, the corresponding doubly cross-validated results. The first column of Figure 1 shows that the objective finction value of MCKL-EM quickly becomes stable while MCKL-SILP oscillates during the first few steps. To validate the global convergence of MCKL-EM, in Figure 1 we also depict evolution of the test set accuracy and the largest two kernel combinatorial weights for MCKL-EM and MCKL-SILP for two example datasets. For both methods, we can see from the second column of Figure 1 that the test set accuracy quickly becomes stable.

3.2 Protein Fold Prediction

We now evaluated our algorithm on a well-known protein fold prediction dataset [6]. Prediction of protein three-dimensional structure is a very important problem within computational biology. Protein fold prediction is the sub-task in which we predict a particular class of arrangement of secondary structure components such as alpha-helices or beta-strands. The benchmark dataset is taken from [6] which has 27 SCOP fold classes with 313 proteins for training and 385 for testing. There are 12 different data-types, or feature spaces, including Amino Acid Composition (C), Predicted Secondary Structure (S), Hydrophobicity (H), Polarity (P), van der Waals volume (V), Polarizability (Z), PseAA $\lambda = 1$ (L1), PseAA $\lambda = 4$ (L4), PseAA $\lambda = 14$ (L14), PseAA $\lambda = 30$ (L30), SW with BLOSUM62 (SW1) and SW with PAM50 (SW2). As in [5], we employed linear kernels (Smith-Waterman scores) for SW1 and SW2 and second order polynomial kernels for the others. In [6] and [17], test set accuracies of 56.5% and 62.1%

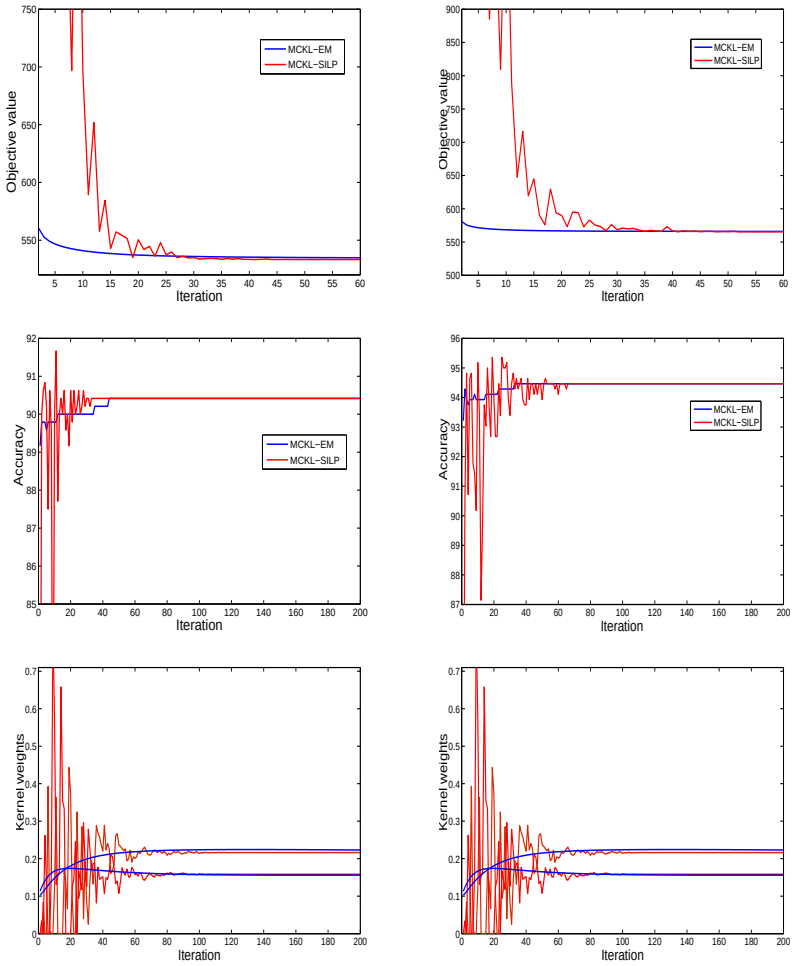

Fig. 1. Evolution of MCKL-EM (blue line) and MCKL-SILP (red line) on the satimage6 (left column) and segment7 (right column) datasets from the UCI Repository [19]. Top: objective function value of MCKL-EM and MCKL-SILP versus iteration; Middle: accuracy of MCKL-EM and MCKL-SILP versus iteration. Bottom: the largest two kernel weights versus iteration, MCKL-EM (blue line) and MCKL-SILP (red line).

wine	MCKL-EM	MCKL-SILP	LSR-CV
TSA	98.19 ± 1.52	98.05 ± 1.17	98.75 ± 1.69
Time	1.20	0.9498	
waveform3	MCKL-EM	MCKL-SILP	LSR-CV
TSA	85.54 ± 1.78	85.95 ± 0.79	86.75 ± 1.77
Time	10.97	2.91	
segment3	MCKL-EM	MCKL-SILP	LSR-CV
TSA	98.66 ± 0.61	98.58 ± 0.65	97.16 ± 1.36
Time	23.24	8.30	
satimage3	MCKL-EM	MCKL-SILP	LSR-CV
TSA	99.58 ± 0.32	99.58 ± 0.34	99.66 ± 0.36
Time	7.02	4.56	
segment7	MCKL-EM	MCKL-SILP	LSR-CV
TSA	93.76 ± 1.14	94.12 ± 0.73	92.71 ± 1.20
Time	106.56	89.77	
satimage6	MCKL-EM	MCKL-SILP	LSR-CV
TSA	90.14 ± 1.45	90.14 ± 1.48	91.14 ± 0.98
Time	40.06	27.93	

Table 1. Test set accuracy (%) and time complexity (seconds) comparison on UCI datasets denoted *wine*, *waveform3*, etc [19]. LSR-CV denotes ridge regression with double cross validation over μ and the Gaussian kernel parameter.

were reported based on various adaptations of binary SVM and neural network. Recently, test performance was greatly improved by Damoulas and Girolami [5] using a Bayesian multi-class multi-kernel algorithm. They reported a best test accuracy of 70% on a single run.

For this problem, we examined the proposed method MCKL-EM, and compared against MCKL-SVM [21] and kernel learning for *regularized kernel discriminant analysis*, RKDA [20] (MCKL-RKDA)[2]. For the first two methods, the parameter μ is tuned by 3-fold cross validation based on a grid search over $\{10^{-2}, 10^{-1}, \ldots, 10^{6}\}$. For RKDA kernel learning [20], we used the SILP approach and the regularization parameter there is also tuned by 3-fold cross validation by a grid search over $\{10^{-6}, 10^{-4}, \ldots, 10^{2}\}$.

Table 2 illustrates the result for MCKL-EM with μ adjusted by 3-fold cross validation. The method achieves a 74.15% test set accuracy (TSA) which outperforms the previously reported state-of-art result of 70% obtained in [5] using a probabilistic Bayesian model, the 68.40% TSA attained by RKDA kernel learning method [20], and the 67.36% TSA by multi-class SVM multi-kernel learning method [21]. The first subfigure of Figure 2 illustrates the performance with each individual feature. The result for MCKL-EM is depicted by a solid line in the first subfigure of Figure 2. The proposed algorithm was also examined with all kernels equally weighted, i.e. $\lambda_\ell = \frac{1}{m}$ for any $\ell \in \mathbb{N}_m$, which as mentioned

[2] The MATLAB code is available from http://www.public.asu.edu/jye02/Software/DKL

	MCKL-EM	MCKL-SVM	MCKL-RKDA
Protein fold (TSA)	**74.15**	67.36	68.40
PSORT+ (Average F1 score)	93.34	**93.8**	**93.83**
PSORT− (Average F1 score)	**96.61**	96.1	96.49

Table 2. Performance comparison (test set accuracy as %) for the protein fold recognition [6, 17] and PSORT protein localization datasets [7, 21]. Results for PSORT are cited from [21].

above is equivalent to a plain L^2-norm regularization. The performance is 70.49% depicted by the dash-dotted line. The second subfigure of Figure 2 shows the kernel combinatorial weights λ. There, the features Amino Acid Composition (C), van der Waals volume (V), SW with BLOSUM62 (SW1), and SW with PAM50 (SW2) are the most prominent sources.

Without using the stopping criterion, MCKL-EM was further examined for up to 2000 iterations after μ was selected by cross-validation. The third subfigure shows the convergence of λ and the fourth subfigure illustrates accuracy versus number of iterations which validates convergence of the iterative algorithm. In Figure 3 the kernel combinatorial weights λ for MCKL-SVM, and MCKL-RKDA are plotted. They both indicate that the first, fifth and last feature are important which is consistent with previous observations. However, the kernel combinations are sparse and quite different from that of MCKL-EM as depicted in the second subfigure of Figure 2. The competing methods also result in worse performance (less than 70%) while MCKL-EM achieves 74.15%. This indicates different combinations of kernel weights lead to significantly different predictions by kernel learning algorithms and sparsity in the kernel weights does not necessarily guarantee good generalization performance. We should note here that the parameter μ in all algorithms is chosen by cross-validation using grid search over the same grid. Moreover, the sparsity usually depends on the parameter μ: the smaller the value μ, the greater the sparsity in kernel weights. This may explain why different kernel weights are obtained for different kernel learning algorithms.

3.3 Prediction of protein subcellular localization

The proposed method (MCKL-EM) was further evaluated on two large datasets for bacterial protein localization [7] where 69 kernels are available. The first problem, derived from the PSORT+ dataset, contains four classes and the other, called PSORT−, has five classes. The results will be based on 30 random partitions into 80% training and 20% test data[3].

In Table 2, test set accuracies for MCKL-EM, MCKL-SVM, MCKL-RKDA are listed. Zien and Ong [21] provided an average F1 score of 93.8% and 96.1% respectively for the PSORT+ and PSORT− datasets after filtering out 81/541 and 192/1444 ambiguous samples. These outperformed the results 90.0% and

[3] http://www.fml.tuebingen.mpg.de/raetsch/suppl/protsubloc

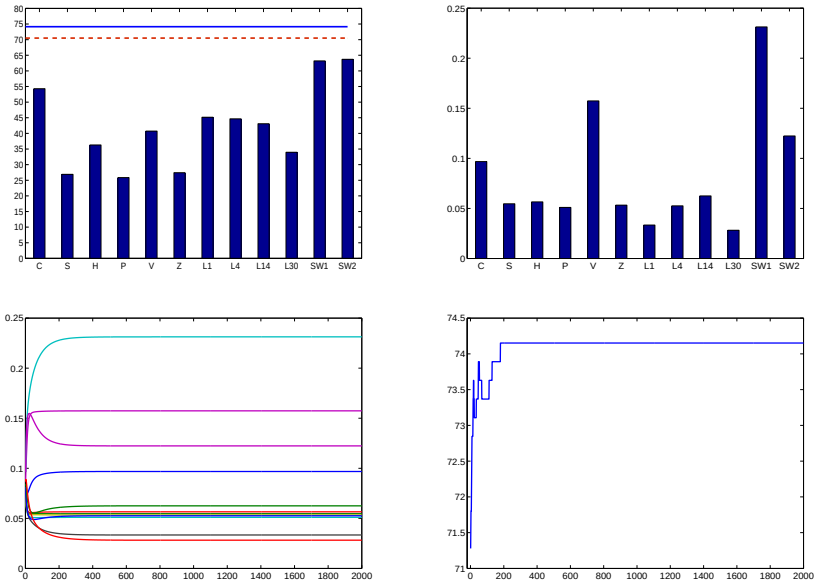

Fig. 2. Performance of MCKL-EM on the protein fold dataset. First subfigure: performance of each individual kernel; dash-dotted red line is for all kernels equally weighted (i.e. plain 2-norm regularization) and the solid blue line is for MCKL-EM. Second one: kernel combinatorial weights i.e. λ. The last two subfigures: evolution of λ and test set accuracy up to 2000 iterations.

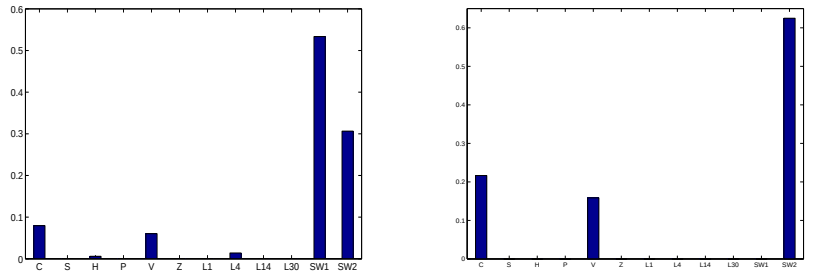

Fig. 3. Kernel weights (i.e. λ) of MCKL-SVM (left subfigure) and MCKL-RKDA (right subfigure) on the protein fold recognition dataset.

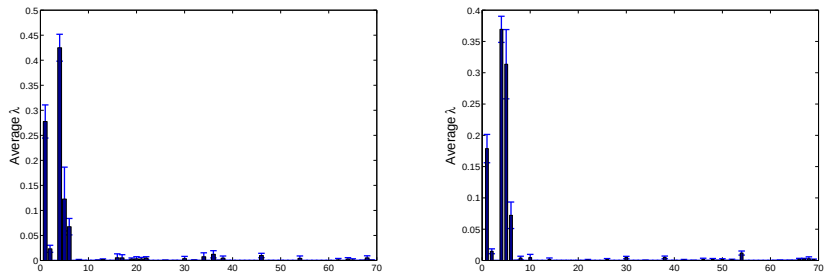

Fig. 4. Averaged kernel combinatorial weights (i.e. λ) with error bars of MCKL-EM on PSORT− (left subfigure) and PSORT+ (right subfigure).

87.5% reported by Gardy et al. [7]. On PSORT+ dataset we got an average F1 score 93.34% for MCKL-EM. For PSORT− dataset, we report an average F1 score 96.61% for MCKL-EM. Hence, our results outperform the results of [7] and are competitive with the methods in [20, 21]. As depicted in Figure 4, the kernel weights for MCKL-EM are quite sparse on this dataset which are consistent with those in [21].

4 Conclusion

In this paper we presented MCKL-EM, a simple iterative algorithm for multiple kernel learning based on the convex formulation of block RKHS norms across classes. As opposed to other MKL algorithms, this iterative approach does not need sophisticated optimization methods while retaining comparable training time and accuracy. The proposed approach yielded state-of-the-art performances on two challenging bioinformatics problems: protein fold prediction and subcellular localization. For the latter we report a competitive performance. For the first one we outperform the previous competitive methods and offer a 4.15% improvement over the state-of-art result which is a significant contribution given the large number of protein fold classes. Future work could include possible extensions of the proposed method for tackling multi-task and multi-label problems.

References

1. Argyriou A., Evgeniou T., and Pontil M. (2006). Multi-task feature learning, *NIPS*.
2. Argyriou A., Micchelli C.A., Pontil M., and Ying Y. (2007). A spectral regularization framework for multi-task structure learning. *NIPS*.
3. Bach F., Lanckriet G.R.G, and Jordan M.I. (2004). Multiple kernel learning, conic duality and the SMO algorithm. *ICML*.
4. Boyd S. and Vandenberghe L. (2004). *Convex Optimization*. Cambridge University Press.
5. Damoulas T. and Girolami M. (2008). Probabilistic multi-class multi-kernel learning: On protein fold recognition and remote homology detection, *Bioinformatics*, **24**(10), 1264-1270.
6. Ding C. and Dubchak I. (2001). Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, **17**, 349–358.
7. J. L. Gardy et al. (2004). PSORTb v.2.0: expanded prediction of bacterial protein subcellular localization and insights gained from comparative proteome analysis, *Bioinformatics*, **21**: 617–623.
8. Girolami M. and Rogers S. (2005). Hierarchic Bayesian models for kernel learning. *ICML*.
9. Lanckriet G.R.G., Cristianini N., Bartlett P., Ghaoui L.E., and Jordan M.I. (2004). Learning the kernel matrix with semidefinite programming. *J. of Machine Learning Research*, **5**, 27–72.
10. Lin Y. and Zhang H. (2006). Component selection and smoothing in multivariate nonparametric regression. *Annals of Statistics*, **34**: 2272–2297.
11. Micchelli C. A. and Pontil M. (2005). Learning the kernel function via regularization, *J. of Machine Learning Research*, 6: 1099–1125.
12. Ong C. S., Smola A. J., and R. C. Williamson R.C. (2005). Learning the kernel with hyperkernels. *J. of Machine Learning Research* **6** 1043–1071.
13. Rakotomamonjy A., Bach F., Canu S., and Grandvalet Y. (2007). More efficiency in multiple kernel learning. *ICML*.
14. Rakotomamonjy A., Bach F., Canu S., and Grandvalet Y. (2008). SimpleMKL, *J. of Machine Learning Research* **9**: 2491–2521.
15. Schölkopf B. and Smola A.J. (2002). *Learning with Kernels*. The MIT Press, Cambridge, MA, USA.
16. Shawe-Taylor J. and Cristianini N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.
17. Shen H. B. and Chou K. C. (2006). Ensemble classifier for protein fold pattern recognition. *Bioinformatics*, **22**, 1717–1722.
18. Sonnenburg S., Rätsch G., Schäfer C., and Schölkopf B. (2006). Large scale multiple kernel learning. *J. of Machine Learning Research*, **7**, 1531–1565.
19. http://archive.ics.uci.edu/ml/
20. Ye J., Ji S., and Chen J. (2008). Multi-class discriminant kernel learning via convex programming, *J. of Machine Learning Research*, **9** 719–758.
21. Zien A. and Ong C. (2007). Multi-class multiple kernel learning, *ICML*.