# Protein Function Prediction by Integrating Multiple Kernels [*]

**Guoxian Yu[1,4], Huzefa Rangwala[2], Carlotta Domeniconi[2], Guoji Zhang[3], Zili Zhang[4]**
[1] School of Computer Sci. and Eng., South China University of Technology, Guangzhou, China
[2] Department of Computer Science, George Mason University, VA, USA
[3] School of Sciences, South China University of Technology, Guangzhou, China
[4] School of Computer and Information Science, Southwest University, Chongqing, China
[1]guoxian85@gmail.com, [2]{rangwala, carlotta}@cs.gmu.edu

## Abstract

Determining protein function constitutes an exercise in integrating information derived from several heterogeneous high-throughput experiments. To utilize the information spread across multiple sources in a combined fashion, these data sources are transformed into kernels. Several protein function prediction methods follow a two-phased approach: they first optimize the weights on individual kernels to produce a composite kernel, and then train a classifier on the composite kernel. As such, these methods result in an optimal composite kernel, but not necessarily in an optimal classifier. On the other hand, some methods optimize the loss of binary classifiers, and learn weights for the different kernels iteratively. A protein has multiple functions, and each function can be viewed as a label. These methods solve the problem of optimizing weights on the input kernels for each of the labels. This is computationally expensive and ignores inter-label correlations.

In this paper, we propose a method called *Pro*tein Function Prediction by Integrating *M*ultiple *K*ernels (ProMK). ProMK iteratively optimizes the phases of learning optimal weights and reducing the empirical loss of a multi-label classifier for each of the labels simultaneously, using a combined objective function. ProMK can assign larger weights to smooth kernels and downgrade the weights on noisy kernels. We evaluate the ability of ProMK to predict the function of proteins using several standard benchmarks. We show that our approach performs better than previously proposed protein function prediction approaches that integrate data from multiple networks, and multi-label multiple kernel learning methods.

## 1 Introduction

Understanding biological mechanisms constitutes an exercise in integrating information derived from several heterogeneous high-throughput experiments. In modern day biology, for a given protein, the different forms of collected data can be its sequence (linear chain of amino acids), its three-dimensional structure, various interactions (e.g., protein-protein interactions) and gene co-expression. Determining the function of a protein using experimental approaches is time consuming and expensive. As such, several computational approaches have been proposed to predict the function of a protein by integrating different sources of available data and have shown superior empirical performance in comparison to training a protein function prediction model only on one of the data sources [Lanckriet *et al.*, 2004; Mostafavi and Morris, 2010].

Several protein function prediction approaches involve representing different data sources as individual kernels (or graphs) and integrating the different kernels within a multiple kernel learning framework [Lanckriet *et al.*, 2004]. Each data source is represented by a kernel function $\mathcal{K}$ that measures the pairwise similarities between proteins. $\mathcal{K}$ also captures the underlying biological complexity associated with the data. Multiple kernels are integrated by finding optimal weights within a semi-definite programming framework [Lanckriet *et al.*, 2004]. Tsuda *et al.* [Tsuda *et al.*, 2005] determine the optimal combination of networks and predictions by taking advantage of the dual problem. Mostafavi *et al.* [Mostafavi *et al.*, 2008] construct the optimal composite graph by solving a linear regression problem. Alternatively, another set of approaches use classifier ensembles to integrate the predictions from models trained on individual sources [Cesa-Bianchi *et al.*, 2012; Yu *et al.*, 2012]. In this paper, we focus on protein function prediction by integrating multiple kernels.

Proteins are multi-functional and each *function* can be viewed as a *label*. Therefore, the protein function prediction problem can be viewed as a multi-label learning problem [Tsoumakas *et al.*, 2010]. The above described methods [Tsuda *et al.*, 2005; Mostafavi *et al.*, 2008] divide the multi-label learning problem into multiple *binary* classification problems and ignore label correlations, which are known to be beneficial for multi-label classification [Tsoumakas *et al.*, 2010]. To make use of inter-label dependencies, Mostafavi *et al.* [Mostafavi and Morris, 2010] proposed an approached called Simultaneous Weights (SW). SW first determines the optimal combination of weights by considering a group of functions instead of a single one, and then trains multiple

[*]This work is partially supported by grants from NSF IIS (0905117, 1252318), NSFC (61070090, 61003174) and China Scholarship Council.

transductive *binary* classifiers on the composite kernel. Experimental results show that SW outperforms methods that optimize separate composite kernel for each of the labels. Tang *et al.* [Tang *et al.*, 2009] proposed a unified framework, in which selecting a specific composite kernel for each label and using the same composite kernel for all the labels are the two extreme cases. Their empirical study shows that if there is enough training data, it is better to select a common composite kernel for all the labels, since selecting a specific kernel for each label is likely to lead to over-fitting. We refer to this approach as Multi-label Multiple Kernel Learning with Sum (MKL-Sum). Bucak *et al.* [Bucak *et al.*, 2010] developed Multi-label Multiple Kernel Learning by Stochastic Approximation (MKL-SA), and the time complexity of MKL-SA is sub-linear with respect to the number of labels. In this paper, we propose a method called *Pro*tein Function Prediction by Integrating *M*ultiple *K*ernels (ProMK). ProMK iteratively optimizes the process of learning weights for multiple kernels with respect to all the labels and reduces the empirical loss of a multi-label classifier simultaneously. We summarize our key contributions as follows:

1. ProMK unifies composite kernel learning and classification optimization within the same objective function, and takes into account all the labels in the objective function.
2. ProMK can selectively integrate kernels, and is able to assign smaller weights to noisy kernels, which are not beneficial to the classification performance.
3. ProMK outperforms other related methods on various multi-label evaluation metrics, across publicly available protein datasets.

## 2 Problem Formulation

We assume that we are given $R$ different data sources that describe the set of $N$ proteins, which have one or more functions (labels) from a total of $C$ possible functions. Different data sources provide different representations for proteins (e.g., vectors, sequences, or networks). To overcome their structural differences, the $R$ different sources of proteins can be transformed into $R$ kernels $[K_r]_{r=1}^{R} (K_r \in \mathcal{R}^{N\times N})$ by domain-specific kernel functions, i.e., string kernel [Leslie and Kuang, 2004] for protein sequences and cluster kernel [Weston *et al.*, 2005] for PPI networks. Among the $N$ proteins, the first $l$ proteins have known functions and the remaining $u$ proteins are not annotated ($l + u = N$). Our objective is to integrate these kernels into a composite kernel and predict the multiple functions associated with the $u$ unlabeled proteins.

### 2.1 Unified Objective Function

The objective function minimized by the traditional transductive classification problem is [Zhou *et al.*, 2004]:

$$\Phi(\mathbf{f}) = \frac{1}{2}\sum_{i,j=1}^{N} \|\frac{\mathbf{f}_i}{\sqrt{D_{ii}}} - \frac{\mathbf{f}_j}{\sqrt{D_{jj}}}\|_2^2 K_{ij} + \lambda \sum_{i=1}^{l} \|\mathbf{f}_i - \mathbf{y}_i\|_2^2 \quad (1)$$

where $D$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{N} K_{ij}$ and $K_{ij}$ is the similarity (or weight of interaction) between proteins $i$ and $j$. $\mathbf{y}_i \in \{0|1\}^C$ is the initial label assignment with $y_{ic} = 1$ if sample $i$ belongs to the $c$-th class, and $y_{ik(k\neq c)} = 0$; if sample $j$ is unlabeled, $y_{jk(1\leq k\leq C)} = 0$. $\mathbf{f}_i \in \mathcal{R}^C$ is the predicted likelihood vector for sample $i$.

Eq. (1) is equivalent to

$$\Phi(F) = tr(F^T LF) + \lambda tr((F - Y)^T U(F - Y)) \quad (2)$$

where $L = I - D^{-\frac{1}{2}} K D^{-\frac{1}{2}}$ is the Laplacian matrix, $F = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N]$ and $tr(\cdot)$ is the matrix trace operation. $\lambda > 0$ is a scalar value that controls the tradeoff between the first and second terms. $U$ is a diagonal matrix with $U_{ii} = 1$ if $i \leq l$ and $U_{ii} = 0$ otherwise. The first term on the right-hand side of Eq. (2) is the *smoothness constraint*, which enforces that the difference between predictions $\mathbf{f}_i$ and $\mathbf{f}_j$ should be small for two similar proteins $i$ and $j$ (i.e., proteins with similar sequences or pairs interacting within the PPI network). The second term on the right-hand side of Eq. (2) is the *fitting constraint*, which enforces that the predictions in $F$ do not change too much for the initially labeled $l$ proteins.

We propose an objective function for protein function prediction by integrating multiple kernels as follows:

$$\Psi(\mathbf{f}, \boldsymbol{\alpha}) = \frac{1}{2}\sum_{r=1}^{R}\sum_{i,j=1}^{N} \alpha_r \|\frac{\mathbf{f}_i}{\sqrt{D_{r,ii}}} - \frac{\mathbf{f}_j}{\sqrt{D_{r,jj}}}\|_2^2 K_{r,ij} + \lambda tr((F - Y)^T U(F - Y))$$

which can be rewritten as:

$$tr(F^T(\sum_{r=1}^{R} \alpha_r L_r)F) + \lambda tr((F - Y)^T U(F - Y)) \qquad s.t. \quad \alpha_r \geq 0, \boldsymbol{\alpha}^T \mathbf{1} = 1 \quad (3)$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_R]^T$ are the weights on the $R$ kernels, $D_r$ is a diagonal matrix with $D_{r,ii} = \sum_{j=1}^{N} K_{r,ij}$ , $L_r$ is the Laplacian matrix on the $r$-th kernel, which is defined in the same way as $L$ in Eq. (2), and $\mathbf{1} \in \mathcal{R}^{R\times 1}$ is a vector with all elements equal to 1. In case of single-label learning, $\mathbf{y}_i$ is encoded such that only one of the $C$ elements is set to 1. In case of multi-label learning, we will use $k$ of $C$ encoding, i.e., set $k$ elements of $\mathbf{y}_i$ to 1 if protein $i$ have $k$ different functions, and different proteins may have different $k$ values. This coding scheme seems simple but yet effective. It was also used in *TRA*nsductive *M*ulti-label learning by label set propagation (TRAM) [Kong *et al.*, 2013] and worked well. By using $k$ of $C$ coding, Eq. (3) can produce a composite kernel that is coherent and optimal for all the labels, and can constrain the prediction matrix $F$ to be optimized across all the labels. In addition, it also alleviates the small positive samples problem, which is often more serious when casting a multi-label learning problem as multiple binary classification problems. By setting $k = 1$, Eq. (3) can also be applied to single-label and multi-class multiple kernels integration. Eq. (3) can also be viewed as combination of multiple graph Laplacians [Argyriou *et al.*, 2005].

By combining $R$ kernels into a composite kernel, Eq. (3) can exploit the complimentary information spread across different data sources. By minimizing Eq. (3), larger weights can be assigned to the kernels that produce a smooth prediction $F$ (i.e., smaller value for $tr(F^T L_r F)$), and smaller weights are given to the kernels that output non-smooth predictions (i.e., larger value for $tr(F^T L_r F)$).

However, Eq. (3) has a trivial solution given by $\alpha_r = 1$, when $tr(F^T L_r F)$ is the smallest. In this case, the weights on all the other kernels are set to zero and only the $r$-th kernel is selected. Since, complementary information (beneficial for protein function prediction) often exists in more than one kernel [Mostafavi and Morris, 2010; Yu *et al.*, 2012], this trivial solution should be avoided. To this end, we add an $l_2$-norm based regularization penalty on $\boldsymbol{\alpha}$ in Eq. (3) and obtain:

$$\begin{aligned} H(F, \boldsymbol{\alpha}) &= \sum_{r=1}^{R} tr(F^T \alpha_r L_r F) \\ &+ \lambda_1 tr((F-Y)^T U (F-Y)) + \lambda_2 \|\boldsymbol{\alpha}\|_2^2 \\ &st. \quad \alpha_r \geq 0, \boldsymbol{\alpha}^T \mathbf{1} = 1 \end{aligned} \tag{4}$$

where $\lambda_1, \lambda_2 > 0$ are used to control the loss on the annotated proteins and the complexity of $\boldsymbol{\alpha}$, respectively. Kloft *et al.* [Kloft *et al.*, 2009] used $l_p$-norm ($p > 1$) for non-sparse multiple kernels integration, but their method is limited to binary classification. By adding the $l_2$-norm, Eq. (4) avoids the trivial solution, and will also assign larger weights to smooth kernels and smaller (or zeros) weights to non-smooth (or noisy) kernels. These advantage will be verified in experimental evaluation (Section 4).

## 2.2 Optimization Solution

Eq. (4) has two variables $\boldsymbol{\alpha}$ and $F$ and can be solved using an EM-style [Dempster *et al.*, 1977] algorithm, in which $\boldsymbol{\alpha}$ and $F$ are iteratively considered as constants.

**Step 1**. We fix $\boldsymbol{\alpha}$ and compute the partial derivative of $H(F, \boldsymbol{\alpha})$ with respect to $F$:

$$\frac{\partial H(F, \boldsymbol{\alpha})}{\partial F} = 2(\sum_{r=1}^{R} (\alpha_r L_r) F + \lambda_1 U (F - Y)) \tag{5}$$

Setting $\frac{\partial H(F, \boldsymbol{\alpha})}{\partial F} = 0$ and solving for $F$, we obtain:

$$F = (\lambda_1 U + \sum_{r=1}^{R} \alpha_r L_r)^{-1} U Y \tag{6}$$

We observe that ProMK computes the inverse of $(\lambda_1 U + \sum_{r=1}^{R} \alpha_r L_r)$ once in each iteration, whereas previous approaches [Tsuda *et al.*, 2005; Wang *et al.*, 2009] optimize the optimal set of weights ($\boldsymbol{\alpha}$) for each label separately, in each iteration.

**Step 2**. We fix $F$ and compute the partial derivative of $H(F, \boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$. In this case, the second term in Eq. (4) is a constant and can be ignored. We can rewrite $H(F, \boldsymbol{\alpha})$ as follows:

$$\begin{aligned} \tilde{H}(F, \boldsymbol{\alpha}) &= \sum_{r=1}^{R} \alpha_r tr(F^T L_r F) + \lambda_2 \|\boldsymbol{\alpha}\|_2^2 \\ &st. \quad \alpha_r \geq 0, \boldsymbol{\alpha}^T \mathbf{1} = 1 \end{aligned} \tag{7}$$

Let $u_r = tr(F^T L_r F)$, $\boldsymbol{\mu} = [u_1, u_2, \cdots, u_R]^T$, thus the equation can be transformed:

$$\begin{aligned} H'(\boldsymbol{\mu}, \boldsymbol{\alpha}) &= \boldsymbol{\alpha}^T \boldsymbol{\mu} + \lambda_2 \boldsymbol{\alpha}^T \boldsymbol{\alpha} \\ s.t. \quad & \alpha_r \geq 0, \boldsymbol{\alpha}^T \mathbf{1} = 1 \end{aligned} \tag{8}$$

Eq. (8) is a quadratic optimization problem with respect to $\boldsymbol{\alpha}$, and can be formulated as a minimization problem:

$$J(\boldsymbol{\alpha}, \boldsymbol{\beta}, \eta) = \boldsymbol{\alpha}^T \boldsymbol{\mu} + \lambda_2 \boldsymbol{\alpha}^T \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \boldsymbol{\beta} - \eta(\boldsymbol{\alpha}^T \mathbf{1} - 1) \tag{9}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_R]^T \geq \mathbf{0}$ and $\eta \geq 0$ are the Karush-Kuhn-Tucker (KKT) multipliers [Boyd and Vandenberghe, 2004]. The optimal $\boldsymbol{\alpha}^*$ should satisfy the following four conditions [Boyd and Vandenberghe, 2004]:

*i) Stationary condition*: $\frac{\partial J(\boldsymbol{\alpha}^*, \boldsymbol{\beta}, \eta)}{\partial \boldsymbol{\alpha}^*} = \boldsymbol{\mu} + 2\lambda_2 \boldsymbol{\alpha}^* - \boldsymbol{\beta} - \eta \mathbf{1} = \mathbf{0}$
*ii) Feasible condition*: $\alpha_r^* \geq 0$ , $\sum_{r=1}^{R} \alpha_r^* - 1 = 0$
*iii) Dual feasibility*: $\beta_r \geq 0, 1 \leq r \leq R$
*iv) Complementary slackness*: $\beta_r \alpha_r^* = 0, 1 \leq r \leq R$
From the stationary condition, $\alpha_r$ can be computed as:

$$\alpha_r = \frac{\beta_r + \eta - \mu_r}{2\lambda_2} \tag{10}$$

$\alpha_r$ depends on the specification of $\beta_r$ and $\eta$, where the specification of $\beta_r$ can be divided into three cases:

(i) If $\eta - \mu_r > 0$, since $\beta_r \geq 0$, we get $\alpha_r > 0$; because of the complementary slackness, $\beta_r \alpha_r = 0$, $\beta_r = 0$ and $\alpha_r = (\eta - \mu_r)/2\lambda_2$.

(ii) If $\eta - \mu_r < 0$, since $\alpha_r \geq 0$, it requires $\beta_r > 0$. Due to $\beta_r \alpha_r = 0$, it follows $\alpha_r = 0$.

(iii) If $\eta - \mu_r = 0$, since $\beta_r \alpha_r = 0$ and $\alpha_r = \beta_r / 2\lambda_2$, $\beta_r = 0$ and $\alpha_r = 0$.

From these cases and assuming $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_R$, there exists $\eta > 0$ such that $\eta - \mu_p > 0$ and $\eta - \mu_{p+1} \leq 0$. $\alpha_r$ has the following solution:

$$\alpha_r = \begin{cases} \frac{\eta - \mu_r}{2\lambda_2}, & \text{if } r \leq p \\ 0, & \text{if } r > p \end{cases} \tag{11}$$

Since $\sum_{r=1}^{p} \alpha_r = 1$, we can obtain the value for $\eta$ as follows:

$$\eta = \frac{2\lambda_2 + \sum_{r=1}^{p} \mu_r}{p} \tag{12}$$

From the solution of $\alpha_r$, we find that, if $\mu_r$ is smaller than $\mu_s$, $\eta - \mu_s > 0$ and $\eta - \mu_r > 0$, the $r$-th kernel will get a larger weight than the $s$-th kernel, because the $r$-th kernel enforces the smoothness constraint better than the $s$-th kernel.

From Eq. (11) we can see that $\alpha_r = 0$ if $r > p$, which implies that the corresponding kernels are discarded. In fact, these kernels have larger $u_r$ values, possibly due to distortion of noisy features (or false positive interactions). Thus, by adding $\|\boldsymbol{\alpha}\|_2^2$ in Eq. (4), ProMK can identify noisy kernels and assigns small or zero weights to them. In Eq. (11), it is easy to observe that if $\lambda_2$ is set to a very small value, $\eta \approx \sum_{r=1}^{p} \mu_r / p$, then at least one kernel (corresponding to the smallest $\mu_r$) will be selected. When $\lambda_2$ is very large, all the kernels will be selected and assigned nearly equal weights. To find $\eta$ that satisfies $\eta - \mu_p > 0$ and $\eta - \mu_{p+1} \leq 0$, we decrease step by step $p$ from $R$ to 1, as stated in **Algorithm 1**. The procedure for ProMK is outlined in **Algorithm 2**. In **Algorithm 2**, $maxit$ is the specified maximum number of iterations, $\theta$ is the specified threshold, $F^t$ and $\boldsymbol{\alpha}^t$ are the learned $F$ and $\boldsymbol{\alpha}$ in the $t$-th iteration.

**Algorithm 1** A method to seek $p$ and compute $\boldsymbol{\alpha}$

```
Input:
    sorted $\mu_1, \mu_2, ..., \mu_R$ in increasing order, $\lambda_2$
Output:
    output $p, \boldsymbol{\alpha}$
 1: Initialize $p = R, \eta = 0$.
 2: while $p > 0$ do
 3:     $\eta \leftarrow (2\lambda_2 + \sum_{r=1}^{p} \mu_r)/p$.
 4:     if $\eta - \mu_p > 0$ then
 5:         break.
 6:     else
 7:         $p \leftarrow p - 1$.
 8:     end if
 9: end while
10: $\alpha_r \leftarrow (\eta - \mu_r)/2\lambda_2$, for $r = 1, ..., p$.
11: $\alpha_r \leftarrow 0$, for $r = p + 1, ..., R$.
```

**Algorithm 2** ProMK: Protein Function Prediction by Integrating Multiple Kernels

```
Input:
    $\{K_r\}_{r=1}^{R}$ from $R$ data sources, $Y \leftarrow [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_l]$
    $\lambda_1, \lambda_2, maxit, \theta$
Output:
    Predicted likelihood score vectors $\{F(j)\}_{j=l+1}^{N}$
1: Initialize $\alpha_r^1 \leftarrow 1/R$, for $r = 1, 2, ..., R$, $t \leftarrow 1$.
2: while $t < maxit$ and $|\varepsilon| > \theta$ do
3:     $t \leftarrow t + 1$.
4:     Compute $F^t$ using Eq. (6).
5:     Seek $p$ and compute $\boldsymbol{\alpha}^t$ using Algorithm 1.
6:     $\varepsilon \leftarrow H(F^t, \boldsymbol{\alpha}^t) - H(F^{t-1}, \boldsymbol{\alpha}^{t-1})$.
7: end while
```

## 3 Experiment Setup

### 3.1 Datasets

We evaluate our approach on two protein datasets (Yeast and Mouse) with multiple heterogeneous data sources. These two datasets are obtained from the study of Mostafavi *et al.* [Mostafavi and Morris, 2010][1], and annotated in the Gene Ontology (GO) database [Ashburner *et al.*, 2000]. Yeast includes 44 kernels derived from 44 different sources (i.e., protein sequences, PPI networks). Mouse includes 10 kernels derived from 10 different data sources. We filtered the Yeast dataset to include only those GO functions that have at least 100 and at most 300 proteins. We filtered the Mouse dataset to include only those GO functions that have at least 30 and at most 100 proteins. Table 1 shows the statistics of these two datasets.

To investigate the ability of ProMK on discarding noisy kernels, we generated synthetic *valid* kernels and synthetic *noisy* kernels from the Gene expression dataset[2]. The Gene dataset was used in [Kong *et al.*, 2013] and includes 2417 genes described by 103 features in 14 function classes. We generate 5 kernels from the $k = 5$ nearest neighbor graph based on 5 different distance metric options of the function *pdist2* in Matlab ( (i) 'Euclidean', (ii) 'Standardized Euclidean', (iii) 'Cosine', (iv) 'Correlation' and (v) 'Spearman'). The similarity between two connected nodes is specified by the Gaussian heat kernel, and the Guassian kernel width $\sigma$ is equal to the average distance between the nodes and their $k$-th neighbors. We refer to these 5 kernels derived from 5 distance functions as *valid* kernels. We also added 15 synthetic *noisy* kernels. For each node in a noisy kernel, we randomly choose $k$ points from all the data, and consider them as the 'neighbors'; the similarity between them is set as in the valid kernel. We repeat this process three times for each distance function, thus producing 15 noisy kernels. The statistics of this dataset is listed in Table 1.

Table 1: Dataset statistics (Avg±Std means average number of functions for each sample and its standard deviation)

| Dataset | #Networks | #Samples | #Labels | Avg±Std |
|---|---|---|---|---|
| Yeast | 44 | 1809 | 57 | $4.35 \pm 3.28$ |
| Mouse | 10 | 3443 | 239 | $3.72 \pm 4.24$ |
| Gene | 20 | 2417 | 14 | $4.24 \pm 1.57$ |

### 3.2 Comparative Methods and Parameter Setting

We compared our method with five state-of-the-art methods: (i) SW [Mostafavi and Morris, 2010], (ii) OMG [Wang *et al.*, 2009], (iii) TRAM [Kong *et al.*, 2013], (iv) MKL-Sum [Tang *et al.*, 2009] and (v) MKL-SA [Bucak *et al.*, 2010]. ProMK and the first three approaches are transductive methods and the last two approaches are inductive methods. OMG was initially proposed for binary classification, and is computational expensive for our datasets with a large number of labels. We modified OMG to a multi-label version by using the trace norm, and extended the label vector to a label matrix as in ProMK. TRAM is proposed for multi-label learning on a single kernel. To adapt it for our experiments, we train TRAM on the composite kernel obtained by combining the individual kernels with equal weight. MKL-SA and MKL-Sum optimize the composite kernel for all the labels, and then apply SVM on the composite kernel for each label.

Here we use four widely-used multi-label evaluation metrics, namely MicroF1, MacroF1, Ranking Loss and Average Precision [Tsoumakas *et al.*, 2010]. To maintain consistency with the other evaluation metrics, we report *1-RankLoss* instead of *RankingLoss*. Thus, similarly to the other metrics, the higher the value of *1-RankLoss*, the better the performance. *MicroF1* and *MacroF1* depend on transforming the predicted likelihood score vector $\mathbf{f}_i$ into a indicative binary label vector. We choose the top $m$ most probable label assignments. For each instance, the $m$ largest likelihood scores are chosen as predicted labels [Yu *et al.*, 2012], and $m$ is specified as the average number of functions (rounded to the next integer) of each protein in each dataset. From Table 1: $m$ is set to 5 for Yeast, to 4 for Mouse and to 5 for Gene. Note, *1-RankingLoss* and *Average Precision* directly uses the predicted likelihoods to evaluate the performance.

We adapted the original code of SW[3], TRAM[4], MKL-Sum[5] and MKL-SA[6] for our experiments. Five-fold cross

[1]http://morrislab.med.utoronto.ca/∼sara/SW/

[2]http://cse.seu.edu.cn/people/zhangml/files/Yeast.zip

[3]http://morrislab.med.utoronto.ca/∼sara/SW/

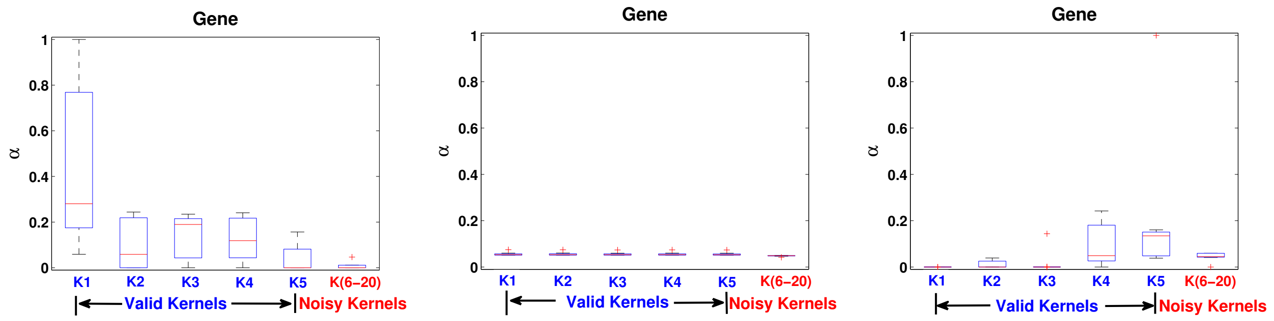
[4]http://lamda.nju.edu.cn/files/TRAM.zip

[5]http://www.public.asu.edu/∼ltang9/code/mkl-multiple-label/

[6]http://www.cse.msu.edu/∼bucakser/ML-MKL-SA.rar

Table 2: Experimental results on **Yeast (44 kernels)**, **Mouse (10 kernels)** and **Gene (20 kernels)**(avg±std).

| Dataset | Metric | ProMK | OMG | SW | MKL-Sum | MKL-SA | TRAM |
|---|---|---|---|---|---|---|---|
| Yeast | MicroF1 | **34.25±1.16** | **34.23±0.96** | 26.27±2.22 | 26.74±3.70 | 29.00±1.17 | 30.57±1.09 |
| | MacroF1 | **31.60±1.13** | **31.69±0.95** | 23.11±2.12 | 26.35±3.35 | 28.64±1.26 | 26.86±1.12 |
| | 1-RankLoss | **78.59±0.88** | **78.83±0.81** | 71.17±1.66 | 68.92±3.05 | 71.37±1.12 | **78.83±0.93** |
| | AvgPrec | **48.56±1.26** | **48.71±1.16** | 35.95±2.50 | 38.11±4.83 | 40.60±1.50 | 45.88±1.44 |
| Mouse | MicroF1 | **25.89±0.82** | **25.68±0.62** | 23.79±0.86 | 19.24±0.94 | 22.44±0.83 | 22.93 ±2.32 |
| | MacroF1 | **21.72±0.76** | **21.47±0.68** | 19.82±0.81 | 16.87±0.87 | 19.03±0.81 | 18.56 ±2.00 |
| | 1-RankLoss | 79.34±0.69 | 78.94±0.61 | 78.95±0.77 | 69.27±0.93 | 69.36±0.86 | **81.49 ±1.85** |
| | AvgPrec | **39.62±1.01** | **39.29±0.83** | 35.19±1.16 | 29.00±1.37 | 34.02±1.04 | 37.59 ±3.34 |
| Gene | MicroF1 | **65.79±0.64** | 64.06±0.72 | 58.13±0.74 | 63.14±0.61 | 60.37±0.71 | 50.30±0.59 |
| | MacroF1 | **42.67±0.69** | 36.89±0.67 | 24.42±0.38 | 41.27±0.79 | 41.35±0.78 | 26.40±0.47 |
| | 1-RankLoss | **83.47±0.53** | 81.98±0.55 | 78.86±0.56 | 79.01±0.68 | 76.27±0.71 | 65.05±0.47 |
| | AvgPrec | **76.51±0.73** | 74.23±0.81 | 70.32±0.83 | 72.86±0.72 | 70.56±0.77 | 59.00±0.64 |



(a) ProMK (b) OMG (c) Sum

Figure 1: Distribution of $\boldsymbol{\alpha}$(learned weights) on the 20 kernels: the first 5 kernels (K1 to K5) are synthetic valid kernels and the other 15 (K6 to K20) are synthetic noisy kernels.

validation is used to select the optimal parameter values for each comparing method (except for SW and TRAM, which do not require explicit parameter tuning). For ProMK, we vary $\lambda_1$ and $\lambda_2$ in $\{10^{-4}, 10^{-3}, ..., 10^3, 10^4\}$. OMG has a similar optimization function as ProMK, but it uses $\alpha_r^q$ ($q$ is the exponent of $\alpha_r$) instead of $\alpha_r$. For OMG, we choose $q$ in $\{1.2, 1.5, 1.8, 2.0, 2.5, 3, 4, 5, 6\}$. For MKL-Sum, we vary the soft margin parameter $C$ of SVM in $\{10^{-4}, 10^{-3}, ..., 10^3, 10^4\}$. For MKL-SA, we use the same optimal $C$ value found for MKL-Sum, and fix the approximation step size as $\eta = 0.0001$. We set $maxit = 20$ and $\theta = 0.001$ for ProMK. Although the objective function of ProMK is not convex, ProMK often converges to local minimum in no more than 10 iterations.

## 4 Experimental Analysis

### 4.1 Performance on Predicting Protein Function

We investigate the performance of ProMK by randomly sampling different percentages (50%,60%,70% and 80%) of the instances as labeled data and the remaining as unlabeled data for testing. Each experiment (sampling) is performed 20 times. In Table 2, we report the multi-label performance results averaged across the 20 runs for the different samplings of labeled and unlabeled instances on the Yeast, Mouse and Gene datasets. Note, the best results and its comparable results are in **boldface** (statistical significance is examined via pairwise $t$-test at 95% significance level).

We observe that ProMK has the best (or comparable to the best) performance. For the experiment on the Yeast dataset (Table 2), ProMK on average is 23.32% better than SW, 20.53% better than MKL-Sum, 13.79% better than MKL-SA and 5.96% better than TRAM on the four metrics. ProMK and OMG have similar performance on Yeast and Mouse datasets (Table 2), but ProMK outperforms OMG on the Gene benchmark, which includes noisy kernels (more discussion on this is in Section 4.2). Both OMG and ProMK combine the phases of composite kernel learning and multi-label classification loss optimization within a unified objective function. In contrast, SW, MKL-Sum and MKL-SA first learn a composite kernel and then train individual binary classification models on the composite kernel. ProMK, OMG and TRAM have better performance than MKL-Sum and MKL-SA on Yeast and Mouse. The reason are two folds: (i) the former three methods do multi-label classification and the latter two perform binary classification; (ii) the former methods are transductive and the latter two are inductive. ProMK and OMG also outperform TRAM, which does not utilize composite kernel learning (uses equal weights for different kernels). These results corroborate the effectiveness of ProMK in integrating multiple kernels for protein function prediction.

### 4.2 Kernel Relevance Estimation

We assess the ability of different approaches on estimating the usefulness of kernels by performing additional experiments on Gene benchmark. We set 80% of the examples for training and 20% for testing. For ProMK, we vary $\lambda_2$ in $\{10^{-4}, 10^{-3}, \cdots, 10^3, 10^4\}$ and record the corresponding learned weights ($\boldsymbol{\alpha}$) across the 20 kernels. We also record the weights of OMG (by varying $q$ in $\{1.2, 1.5, 1.8, 2.0, 2.5, 3, 4, 5, 6\}$) and MKL-Sum (by varying $C$ in $\{10^{-4}, 10^{-3}, \cdots, 10^3, 10^4\}$) on the 20 kernels. The distribution of learned weights across the 20 kernels are shown in Figure 1. For concise representation, we averaged the distributions of the weights for the noisy kernels. We observe that ProMK can identify the noisy kernels and discard them by assigning zero weights. On the contrary, neither OMG nor MKL-Sum can identify and discard these noisy kernels. OMG always gives similar weights to all the kernels, whether valid or noisy. MKL-Sum sometimes gives larger weights to the noisy kernels. In contrast, the weights given to valid kernels by ProMK are never smaller than those assigned to noisy kernels. This also explains the superiority of ProMK to MKL-Sum and OMG on multiple kernel learning with noisy kernels (as shown in Table 2 for **Gene**). We also note that from the solution $\alpha_r$ of OMG [Wang *et al.*, 2009] ($q > 1$), OMG cannot assign zero weights to kernels:

$$\alpha_r^{OMG} = \frac{\left(\frac{1}{\lambda_1\|F-Y\|_2^2 + tr(F^T L_r F)}\right)^{\frac{1}{q-1}}}{\sum_{r=1}^{R}\left(\frac{1}{\lambda_1\|F-Y\|_2^2 + tr(F^T L_r F)}\right)^{\frac{1}{q-1}}}$$

We also study the normalized trace values $\mu_r = tr(F^T L_r F)$ using ProMK and OMG for the Gene benchmark (Due to space constraint, we do not report these results here). We observe that the $\mu_r$ values for ProMK on valid kernels are smaller than the $\mu_r$ values for OMG on valid kernels. Also, the $\mu_r$ values for ProMK on noisy kernels are larger than the $\mu_r$ values for OMG on noisy kernels. As seen in Eq. (11), $\mu_r$ determines the value of the weight $\alpha_r$. As such, ProMK can assign larger weights to valid kernels and smaller weights to noisy kernels (see Figure 1) than OMG. For the Yeast and Mouse benchmarks, ProMK and OMG both learn nearly equal weights for all the kernels. We notice that the $\mu_r$ values for the different kernels are very close to each other, which makes it harder to determine the importance of individual kernels. In both these datasets, the kernels represent heterogeneous networks and they seem to provide complementary information as determined by experimental results reported in Section 4.3.

## 4.3 On the Benefit of Optimizing Kernel Weights

We also performed experiments to investigate the benefit of optimizing kernel weights. For these experiments, we first construct a composite kernel by equally-weighting and adding individual kernels. Then we apply the proposed multi-label classifier (see Eq. (3)) on the composite kernel. We refer to this approach as ProMK-Same and compare the performance of ProMK with ProMK-Same. We also apply the ProMK approach on each of the individual kernels, and report the *best* and *worst* performance in Table 3.

We observe that ProMK always shows the best (or comparable to the best) performance. ProMK always outperforms ProMK-Same on Gene dataset. This is because ProMK can identify the synthetic noisy kernels, and discard or assign zero weights to them. Simply combining kernels with equal weights often results in a sub-optimal composite kernel. ProMK achieves a performance that is no worse than the ProMK on any individual kernels.

ProMK-Same outperforms ProMK on any single kernel of Yeast and Mouse benchmarks. ProMK(Worst) always loses to ProMK(Best) and other comparing methods. These facts are indicative of the complementary information spread across different kernels, and that different kernels hold different quality of information. ProMK and ProMK-Same have comparable performance on the Yeast and Mouse datasets. This also explains why ProMK sets nearly equal weights to all the kernels for these benchmarks.

Table 3: ProMK vs. ProMK-Same on **Yeast**, **Mouse** and **Gene**.

| Dataset | Metric | ProMK | ProMK-Same | Best | Worst |
|---|---|---|---|---|---|
| Yeast | MicroF1 | **36.09±1.12** | 34.69±1.25 | 19.58±1.16 | 8.65±0.60 |
| | MacroF1 | **33.35±0.91** | 32.04±1.11 | 17.72±1.25 | 2.53±0.25 |
| | 1-RankLoss | **80.04±1.01** | **79.61±1.19** | 65.85±1.35 | 3.54±0.50 |
| | AvgPrec | **50.49±1.28** | **49.98±1.63** | 30.71±1.32 | 15.77±0.56 |
| Mouse | MicroF1 | **27.19±1.20** | **26.80±0.81** | 23.96±0.80 | 1.64±0.40 |
| | MacroF1 | **22.68±0.91** | **22.16±0.58** | 20.66±0.58 | 1.45±0.55 |
| | 1-RankLoss | **80.35±1.06** | **80.42±0.63** | 73.66±0.53 | 2.22±0.39 |
| | AvgPrec | **41.65±1.48** | 40.67±0.93 | 37.04±0.93 | 6.90±0.46 |
| Gene | MicroF1 | **65.87±1.16** | 61.70±1.24 | **65.71±0.85** | 56.60±1.04 |
| | MacroF1 | **43.40±0.90** | 31.12±0.83 | **43.41±0.83** | 28.10±0.42 |
| | 1-RankLoss | **83.50±0.85** | 80.55±0.80 | **83.47±0.67** | 77.59±0.73 |
| | AvgPrec | **76.54±1.31** | 72.33±0.99 | **76.59±0.92** | 69.13±0.85 |

## 4.4 Run Time Analysis

We also record the average run time on the three datasets in Table 4. The experiments are conducted on Windows 7 platform with Intel E8400 processor and 4GB memory. We can see that the run time of ProMK is ranking in the median amongst all the comparing methods. However, ProMK generally achieves the best (or comparable to the best) performance amongst the six methods. SW does not compute the inverse of a matrix during the learning of a composite kernel, and thus takes less time than OMG and ProMK. TRAM does not learn weights on kernels and is the fastest. MKL-Sum and MKL-SA apply SVM on the composite kernel, they cost much more time than other methods.

Table 4: Runtime comparison (in seconds).

| Dataset | ProMK | OMG | SW | MKL-Sum | MKL-SA | TRAM |
|---|---|---|---|---|---|---|
| Yeast | 40.82 | 45.23 | 13.25 | 175.20 | 129.22 | 1.94 |
| Mouse | 89.28 | 111.97 | 24.48 | 517.45 | 220.15 | 5.97 |
| Gene | 45.51 | 42.79 | 7.35 | 65.17 | 112.62 | 2.70 |
| Total | 175.61 | 199.99 | 45.08 | 757.82 | 461.99 | 10.61 |

# 5 Conclusion

In this paper, we proposed a protein function prediction method using multiple kernels integration to exploit heterogeneous data sources. Different from traditional multiple kernel learning methods, which consist of two separate steps (kernel integration and classifier training), or seeking a composite kernel and training a binary classifier for each label independently, ProMK can integrate multiple kernels into a composite kernel, and train a multi-label classifier on the composite kernel for all the labels simultaneously. In addition, ProMK can identify and give zero weights to noisy kernels. Our experimental results show that ProMK performs better than other related methods.

## References

[Argyriou *et al.*, 2005] A. Argyriou, M. Herbster, and M. Pontil. Combining graph laplacians for semi–supervised learning. In *18th Advances in Neural Information Processing Systems*, pages 1–8, 2005.

[Ashburner *et al.*, 2000] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25, 2000.

[Boyd and Vandenberghe, 2004] S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[Bucak *et al.*, 2010] S.S. Bucak, R. Jin, and A.K. Jain. Multi-label multiple kernel learning by stochastic approximation: Application to visual object recognition. *Advances in Neural Information Processing Systems*, 24:325–333, 2010.

[Cesa-Bianchi *et al.*, 2012] N. Cesa-Bianchi, M. Re, and G. Valentini. Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference. *Machine Learning*, 88(1-2):209–241, 2012.

[Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[Kloft *et al.*, 2009] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.R. Müller, and A. Zien. Efficient and accurate lp-norm multiple kernel learning. In *22th Advances in Neural Information Processing Systems*, pages 997–1005, 2009.

[Kong *et al.*, 2013] X. Kong, M. Ng, and Z. Zhou. Transductive multi-label learning via label set propagation. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):704–719, 2013.

[Lanckriet *et al.*, 2004] G.R.G. Lanckriet, T. De Bie, N. Cristianini, M.I. Jordan, and W.S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[Leslie and Kuang, 2004] C. Leslie and R. Kuang. Fast string kernels using inexact matching for protein sequences. *The Journal of Machine Learning Research*, 5:1435–1455, 2004.

[Mostafavi and Morris, 2010] S. Mostafavi and Q. Morris. Fast integration of heterogeneous data sources for predicting gene function with limited annotation. *Bioinformatics*, 26(14):1759–1765, 2010.

[Mostafavi *et al.*, 2008] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris. Genemania: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, 9(Suppl 1):S4, 2008.

[Tang *et al.*, 2009] L. Tang, J. Chen, and J. Ye. On multiple kernel learning with multiple labels. In *21st International Joint Conference on Artificial Intelligence*, pages 1255–1260, 2009.

[Tsoumakas *et al.*, 2010] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. *Data Mining and Knowledge Discovery Handbook*, pages 667–685, 2010.

[Tsuda *et al.*, 2005] K. Tsuda, H.J. Shin, and B. Schölkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21(suppl 2):ii59, 2005.

[Wang *et al.*, 2009] M. Wang, X.S. Hua, R. Hong, J. Tang, G.J. Qi, and Y. Song. Unified video annotation via multigraph learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(5):733–746, 2009.

[Weston *et al.*, 2005] J. Weston, C. Leslie, E. Ie, D. Zhou, A. Elisseeff, and W.S. Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, 2005.

[Yu *et al.*, 2012] G. Yu, C. Domeniconi, H. Rangwala, G. Zhang, and Z. Yu. Transductive multi-label ensemble classification for protein function prediction. In *18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1077–1085. ACM, 2012.

[Zhou *et al.*, 2004] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.