# Protein Functional Class Prediction with a Combined Graph

Hyunjung Shin[1,2], Koji Tsuda[1,3], Sungzoon Cho[2], and Bernhard Schölkopf[1]

[1] Department of Empirical Inference (Dept.of Dr. Schölkopf), Max-Planck-Institute for Biological Cybernetics, Spemannstr. 38, 72076, Tübingen, Germany
{shin, tsuda, bernhard}@tuebingen.mpg.de
[2] Department of Industrial Engineering, Seoul National University, San 56-1, Shillim-Dong, Kwanak-Gu, 151-744, Seoul, Korea
{hjshin72, zoon}@snu.ac.kr
[3] National Institute of Advanced Industrial Science and Technology (AIST), 2-43 Aomi Koto-ku, Tokyo, Japan

**Abstract.** In bioinformatics, there exist multiple descriptions of graphs for the same set of genes or proteins. For instance of yeast proteins, edges can stand for different kind of relation such as protein-protein interactions, or genetic interactions, or co-participation in a protein complex, etc. Each graph can solely be used for prediction of the proteins unclassified yet, relying on similarities between nodes. However since different graphs contain partly independent and partly complementary pieces of information about the problem at hand, one thus can enhance the total information about the problem by combining those graphs. In this paper, we propose a method for integrating multiple graphs within a framework of semi-supervised learning. The method alternates minimizing the objective function with respect to network output and with respect to combining weight. We demonstrate the method to the task of functional class prediction of yeast proteins. The proposed method performs significantly better than the same algorithm trained on any single graph.

## 1 Introduction

In bioinformatics, it is common that many types of genomic data can be represented using graphs of which nodes correspond to genes or proteins, and of which edges correspond to different kind of relationships such as physical interactions of the proteins (Schwikowski *et al.*, 2000; Uetz *et al.*, 2000; von Mering *et al.*, 2002), gene regulatory relationships (Lee *et al.*, 2002; Ihmels *et al.*, 2002; Segal *et al.*, 2003), and similarities between protein sequences (Yona *et al.*, 1999), etc. One of the applications using graph representation is to predict protein functional class. It can be described as a binary-class classification problem on an undirected graph. Fig.1 illustrates the problem. A protein already known its class is labeled either by '+1' or '−1' while a protein yet unknown its class is marked as '?'. The goal is to predict the label of unlabeled protein relying on similarities between nodes. Prediction of protein functional class has been studied by means of various methods such as diffusion kernel (Tsuda & Noble, 2004a), majority vote (Hishigaki *et al.*, 2001; Schwikowski *et al.*, 2000), graph-based (Vazquez *et al.*, 2003), Bayesian (Deng *et al.*, 2003), and discriminative learning methods (Vert & Kanehisa, 2002; Lanckriet *et al.*, 2004a).

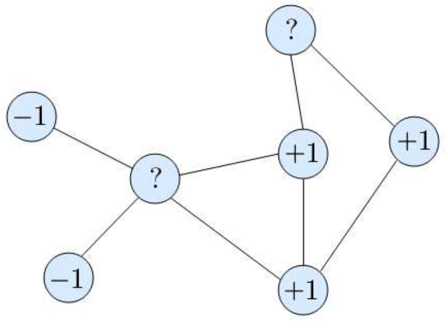


**Fig. 1.** The functional class prediction on a protein network graph. A protein already known its class is labeled either by +1 or −1. The task is to predict labels of unlabeled proteins marked as '?'.

Meanwhile, there can exist multiple descriptions of graphs for the same set of genes or proteins. For instance, nodes of yeast proteins can be connected in many different ways based on heterogeneous information such as protein-protein interactions, or genetic interactions, or co-participation in a protein complex, etc. Different graph sources are likely to contain partly independent and partly complementary pieces of information about the problem at hand. Thus, one can enhance the total information about the problem by combining those graphs. Recently, there have arisen several methods for integrating heterogeneous data sources in bioinformatics. Most of them are based on kernel methods thanks to up-to-date advances of theory and performance. These methods commonly represent data by means of kernel matrix which defines similarities between pairs of genes or proteins. They then propose a method of their own on how to combine those kernel matrices. Lanckriet *et al.* (2004c) exploits semi-definite programming (SDP, see also Lanckriet *et al.* (2004b)) techniques to reduce the problem of finding optimizing kernel combinations to a convex optimization problem. This SDP-based approach yields a satisfactory result when performed on genome-wide data sets, including amino acid sequences, hydropathy profiles, gene expression data and known protein-protein interactions. On the other hand, Kato *et al.* (2004) differentiate the worth of data sources such as 'expensive' data–informative but difficult to obtain, and 'cheap' data–less informative but abundantly available. Since the kernel matrix derived from the expensive data often has missing entries, they attempt to complete them using multiple cheap data. They use an EM (expectation-maximization) algorithm so as to simultaneously optimize the combining weights of data sources and the missing entries of the incomplete kernel matrix (for the methodology about kernel matrix completion, refer to Tsuda *et al.* (2004b)). This EM-based method shows promising results when tested on supervised protein network inference and protein superfamily classification. The problem as to multiple data sources (not only limited to graph representation) is often described as so-called "data fusion," which is intensely dealt with in the chapter 11, 12, and 13 in the recent book of (Schölkopf, Tsuda, & Vert, 2004). Another methods related to integration of data sources can be found (Lanckriet *et al.*, 2004a; Pavlidis *et al.*, 2001; Vert & Kanehisa, 2002).

In the meantime, when data is represented as a graph, and further labeling the nodes unlabeled highly costs but their input can be incorporated in training classifier, a more direct state-of-the-art of learning method is semi-supervised learning. In semi-supervised learning, the labeled nodes provide information about the decision function, while the unlabeled nodes aid to reveal the structure of the data or data manifold by providing additional information (Chapelle *et al.*, 2003b; Zhou *et al.*, 2004a; Seeger, 2000). We will enter into detail about semi-supervised learning in the next section. However, when we encounter the problem on utilization of multiple data sources there has not been yet a research that can deal with the case in semi-supervised learning. In this paper, we propose a method for integrating multiple graphs within a framework of semi-supervised learning. The method alternates minimizing the objective function with respect to network output and with respect to combining weight. We demonstrate the method to the task of functional class prediction of yeast proteins provided by the MIPS Comprehensive Yeast Genome Database (CYGD-mips.gsf.de/proj/yeast). The proposed method performs significantly better than the same algorithm trained on any single graph.

The remaining of this paper is organized as follows. In section 2, we briefly introduce semi-supervised learning and the recent work. Section 3 gives a detailed explanation of our proposed method. In section 4, we show experimental results. We conclude in section 5.

## 2 Semi-Supervised Learning

Let $G = (V, E)$ denote a weighted graph where $V = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ is the vertex set and $E$ is the edge set. And $W$ is denoted as a weight matrix associated with $E$, which represents the magnitude of strength of linkage. $W$ could be simply regarded as a non-negative similarity (or an affinity) matrix. The more similar $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$, the larger a value of $w_{ij}$. Now suppose that $p$ vertices of $V$ are labeled $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_p, y_p)$ where $y_i \in \{-1, 1\}$, and the remaining $q$ vertices $\boldsymbol{x}_{p+1}, \boldsymbol{x}_{p+2}, \ldots, \boldsymbol{x}_{p+q=n}$ are unlabeled. And accordingly, let us define $P = \{1, 2, \ldots, p\}$ for the former and $Q = \{p+1, p+2, \ldots, n\}$ for the latter. The goal of semi-supervised learning is to label those unlabeled vertices by exploiting the structure of the graph under the assumption that a label of an unlabeled vertex is more likely to be that of more adjacent or more strongly connected vertex to it.

To formulate the idea, let us define a function $f : V \rightarrow \Re$ on $G$ that estimates labels with this property. Then, (a) a label $f_i$ or $f(\boldsymbol{x}_i)$ estimated from $f$ should not be too much different from $f_j$'s of adjacent vertices (b) under the constraints $f_i \equiv y_i, i = 1, ..., p$. One can obtain $f$ by minimizing the following quadratic function

$$\sum_{i \sim j} w_{ij}(f_i - f_j)^2 + \mu \sum (f_i - y_i)^2, \tag{1}$$

where $i \sim j$ means $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are adjacent. The first term implies the "smoothness" of (a) and the second term corresponds to the "loss function" of (b). Alternative functions of smoothness or loss can be found in Chapelle *et al.* (2003a). For technical convenience, a condition $\sum f_i = 0$ can be added to Eq.(1). Refer to Belkin & Niyogi (2003a) and Belkin *et al.* (2003b). Very often, the quadratic problem of Eq.(1) is represented in terms of matrix,

$$\min_{\boldsymbol{f}} \;\; \boldsymbol{f}^T L \boldsymbol{f} + \mu \, (\boldsymbol{f} - \boldsymbol{y})^T (\boldsymbol{f} - \boldsymbol{y}), \tag{2}$$

where $\boldsymbol{y} = [\; \boldsymbol{y}_P^T \;\; \boldsymbol{y}_Q^T \;]^T,\; y_p \in \{-1, 1\},\; y_q \in \{0\},\; p \in P,\; q \in Q$, and $\boldsymbol{f} = [\; \boldsymbol{f}_P^T \;\; \boldsymbol{f}_Q^T \;]^T,\; f \in \Re$. $\mu$ is a parameter that trades off loss versus smoothness. The Laplacian is defined as $L = D - W$ where $D = diag(d_i),\; d_i = \sum_j w_{ij}$. Instead of $L$, 'normalized Laplacian', $\tilde{L} = D^{-\frac{1}{2}} L D^{\frac{1}{2}}$ can be used which has many nice properties (Chung, 1997). The solution to the quadratic problem can be obtained in a form

$$\boldsymbol{f} = \mu \; \{L + \mu \, I\}^{-1} \boldsymbol{y}$$

where $I$ is an identity matrix.

There have been various semi-supervised learning algorithms, such as spectral methods and clustering (Belkin & Niyogi, 2004; Chapelle *et al.*, 2003a; Joachims, 2003; Ng *et al.*, 2001; Seeger, 2000), graph s-t mincuts (Blum & Chawla, 2001) or multi-way cuts (Kleinberg and Tardos, 1999), Co-Training (Blum & Mitchell, 1998), random walks (Szummer & Jaakkola, 2001; Zhou & Schölkopf, 2004b; Zhu *et al.*, 2003), diffusion kernels (Kandola *et al.*, 2002; Kondor and Lafferty, 2002; Smola & Kondor, 2003). And see also 'transductive SVM' introduced by Vapnik (1998) which were later refined by Bennett (1999) and Joachims (1999).

## 3 Method of Combining Graphs

Given a single graph $G$, we can predict $\boldsymbol{f}_q$ with Eq.(2) after transforming $G$ into a Laplacian $L$ or a normalized Laplacian $\tilde{L}$ . Now, consider the case that a set

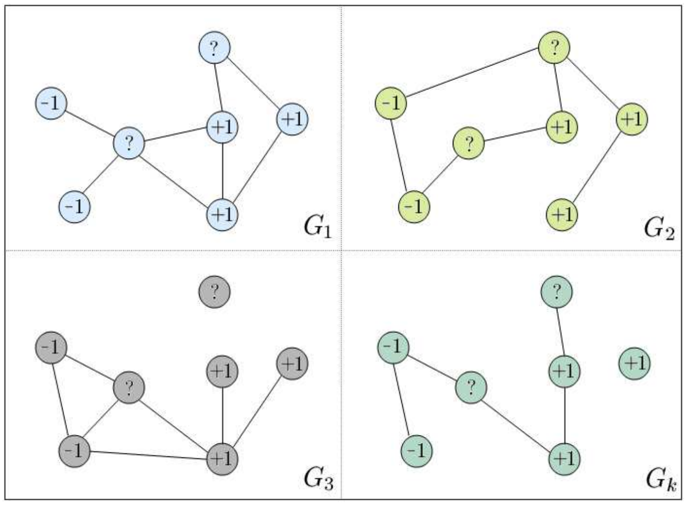


**Fig. 2.** Multiple graphs: consider the case that a set of graphs $\boldsymbol{G} = \{G_1, G_2, \ldots, G_k\}$ is given, and each of which tells different aspect of the data. Each graph can solely predict the label of the unlabeled nodes marked as '?', depending on its own similarity measure between nodes. However, since different graphs contain partly independent and partly complementary pieces of information about the problem at hand, one thus can enhance the total information about the problem by combining those graphs.

of graphs $\boldsymbol{G} = \{G_1, G_2, \ldots, G_k\}$ is given, and each of which tells different aspect of the data (see Fig.2). We can simply formulate $k$ quadratic problems in the form of Eq.(2), and then can obtain $k$ independent outputs $\boldsymbol{f}_k$, $k = 1, \ldots, K$. However, if $\boldsymbol{f}_i \neq \boldsymbol{f}_j$, $i \neq j$, how can we deal with such a disagreement among the outputs? One way to circumvent the situation is to put them all into one objective function, and solve it simultaneously. We can consider parameterized combinations of the graphs. In particular, we can form the linear combination of Laplacians

$$L = \sum_{k=1}^{K} \beta_k L_k, \tag{3}$$

where the weight $\beta_k$ is constrained to be positive to assure that each Laplacian contributes to prediction of $\boldsymbol{f}$. By plugging Eq.(3) into Eq.(2), we obtain

$$\begin{aligned} \min_{\boldsymbol{\beta},\, \boldsymbol{f}} \quad & \sum_{k=1}^{K} \beta_k \boldsymbol{f}^T L_k \boldsymbol{f} + \mu\, (\boldsymbol{f} - \boldsymbol{y})^T (\boldsymbol{f} - \boldsymbol{y}), \\ s.t. \quad & \boldsymbol{\beta} \geq 0, \end{aligned} \tag{4}$$

where $\boldsymbol{\beta} = [\beta_1 \ \beta_2 \ \ldots \ \beta_k]^T$. The solution of Eq.(4) is, however, trivial when $\boldsymbol{\beta} = \mathbf{0}$. Although except the case, the weight vector $\boldsymbol{\beta}$ is prone to be a sparse vector having one or two non-zero elements. For example, $\boldsymbol{\beta} = [0\ 0\ \ldots\ \delta\ 0]^T$ could be the case. In order to prevent the cases aforementioned, we introduce a 'barrier function' (see Bazaraa *et al.* (1993)) to Eq.(4),

$$\min_{\boldsymbol{\beta},\ \boldsymbol{f}} R(\boldsymbol{\beta},\ \boldsymbol{f}) = \sum_{k=1}^{K} \beta_k \boldsymbol{f}^T L_k \boldsymbol{f} - \log\det(I - \sum_{k=1}^{K} \beta_k L_k) + \mu\ (\boldsymbol{f} - \boldsymbol{y})^T C(\boldsymbol{f} - \boldsymbol{y}),$$
$$s.t.\ \boldsymbol{\beta} \geq 0,\ \boldsymbol{\beta}^T \mathbf{1} = \delta, \tag{5}$$

where $\mathbf{1} = [1\ 1\ \ldots\ 1]^T$ and $\delta < 0.5$. The second term plays the role of barrier which spreads a non-zero value on every element of $\boldsymbol{\beta}$. The constraint $\boldsymbol{\beta}^T \mathbf{1} = \delta$ is added for the sake of technical stability (see chap. 1 of Chung (1997)). One can substitute a simpler function such as '$-\sum_k \log \beta_k$' for the current barrier without the extra constraint $\boldsymbol{\beta}^T \mathbf{1} = \delta$. In the third term corresponding to loss function, a diagonal cost matrix $C$ is incorporated so that allows different misclassification cost, i.e., $c_1$ for $y_i = +1$, and $c_2$ for $y_j = -1,\ i, j \in P$.

Eq.(5) has nice properties: fixing $\boldsymbol{\beta}$ the objective function is convex with respect to $\boldsymbol{f}$, while fixing $\boldsymbol{f}$ it is also convex with respect to $\boldsymbol{\beta}$. Now, we can jointly minimize the objective function on $\boldsymbol{\beta}$, and on $\boldsymbol{z}$ as well. We bisect the solving process like 'E-step' and 'M-step' of EM algorithm, and optimize both steps in alternate (Dempster *et al.* (1977); McLachlan & Krishnan (1997)). However, on account of lack of justice yet, we here denote them as '$\boldsymbol{\beta}$-step' and '$\boldsymbol{f}$-step' instead. The algorithm is presented in Fig.3 followed by the solution of each step.

```
(1) Initialize $\boldsymbol{\beta}^i$ $(i = 0)$ with random value under the constraints
    $(\boldsymbol{\beta}^i)^T \mathbf{1} = \delta$.

(2) [$\boldsymbol{f}$-step] Given $\boldsymbol{\beta}^i$, find $\boldsymbol{f}^i$ by minimizing $R(\boldsymbol{\beta}, \boldsymbol{f})$ with
    respect to $\boldsymbol{f}$.

(3) [$\boldsymbol{\beta}$-step] Given $\boldsymbol{f}^i$, find $\boldsymbol{\beta}^{i+1}$ by minimizing $R(\boldsymbol{\beta}, \boldsymbol{f})$ with
    respect to $\boldsymbol{\beta}$.

(4) Return $\boldsymbol{f}^i$ and $\boldsymbol{\beta}^i$ if $\left|\frac{R(\boldsymbol{\beta}^{i+1},\ \boldsymbol{f}^{i+1}) - R(\boldsymbol{\beta}^i,\ \boldsymbol{f}^i)}{R(\boldsymbol{\beta}^i,\ \boldsymbol{f}^i)}\right| < \epsilon$,
    $i = i + 1$ and go to (3) otherwise.
```

**Fig. 3.** Algorithm: By alternating '$\boldsymbol{\beta}$-step' and '$\boldsymbol{f}$-step', the optimal solution of the combining weights and the output can be found simultaneously.

**Solution of [$\boldsymbol{f}$-step]** When $\boldsymbol{\beta}$ is fixed, the solution $\boldsymbol{f}$ can be obtained by

$$\frac{\partial R(\boldsymbol{\beta},\ \boldsymbol{f})}{\partial \boldsymbol{f}}\Big|_{(\boldsymbol{\beta}=\boldsymbol{\beta}^i)} = \left\{\sum_{k=1}^{K}\beta_k L_k + \mu\, C\right\}\boldsymbol{f} - \mu\, C\, \boldsymbol{y} = 0$$

where $C$ is a $(n \times n)$ diagonal cost matrix. And standard linear algebra leads to the solution in the form of

$$\boldsymbol{f} = \mu\, C\, \left\{\sum_{k=1}^{K}\beta_k L_k + \mu\, C\right\}^{-1} \boldsymbol{y} \tag{6}$$

**Solution of [$\boldsymbol{\beta}$-step]** To find the solution of $\boldsymbol{\beta}$ when given $\boldsymbol{f}$, we use the gradient descent method for minimizing $R(\boldsymbol{\beta}, \boldsymbol{f})$ with respect to $\boldsymbol{\beta}$. The current $\boldsymbol{\beta}^i$ is updated to $\boldsymbol{\beta}^{i+1}$ as follows:

$$\boldsymbol{\beta}^{i+1} = \boldsymbol{\beta}^i - \alpha^i P \mathbf{d}\boldsymbol{\beta}^i \tag{7}$$

where $\mathbf{d}\boldsymbol{\beta}^i$ is the gradient vector,

$$\mathbf{d}\boldsymbol{\beta}^i = \frac{\partial R(\boldsymbol{\beta},\ \boldsymbol{f})}{\partial \boldsymbol{\beta}}\Big|_{(\boldsymbol{f}=\boldsymbol{f}^i,\ \boldsymbol{\beta}=\boldsymbol{\beta}^i)}$$

whose $k^{th}$ element is

$$\frac{\partial R(\boldsymbol{\beta},\ \boldsymbol{f})}{\partial \beta_k}\Big|_{(\boldsymbol{f}=\boldsymbol{f}^i,\ \boldsymbol{\beta}=\boldsymbol{\beta}^i)} = \boldsymbol{f}^T L_k \boldsymbol{f} + \mathrm{tr}\big[(I - \sum_{j=1}^{K}\beta_j L_j)^{-1} L_k\big]. \tag{8}$$

In Eq.(8), $\mathrm{tr}\big[(I - \sum_{j=1}^{K}\beta_j L_j)^{-1} L_k\big]$ is the derivative of $\frac{\partial}{\partial \beta_k}\Big(\log\det(I - \sum_{k}^{K}\beta_k L_k)\Big)$ given by the following algebra. Let $A$ be a matrix of which element is parameterized with respect to $t$. The derivative of $\frac{\partial}{\partial t}\Big(\log\det A\Big)$ can be drawn by

$$\begin{aligned}\frac{\partial}{\partial t}\Big(\log\det A\Big) &= \sum_{i,j}\frac{\partial}{\partial A_{ij}}\Big(\log\det A\Big) \times \frac{\partial A_{ij}}{\partial t} \\ &= \sum_{i,j} A^{-1}_{ij}\frac{\partial A_{ij}}{\partial t}\end{aligned} \tag{9}$$

where $\sum_{i,j}\frac{\partial}{\partial A_{ij}}\Big(\log\det A\Big) = \frac{\partial}{\partial A}\Big(\log\det A\Big) = A^{-1}$. And if $A$ and $B$ are symmetric, then $\sum_{i,j} A_{ij}B_{ij} = \mathrm{tr}\big[AB\big]$. Thus Eq.(9) becomes

$$\sum_{i,j} A^{-1}_{ij}\frac{\partial A_{ij}}{\partial t} = \mathrm{tr}\big[A^{-1}\frac{\partial A}{\partial t}\big].$$

By replacing $A$ and $t$ with $(I - \sum_{k=1}^{K} \beta_k L_k)$ and $\beta_k$, respectively, we get the derivative

$$\frac{\partial}{\partial \beta_k}\Big(\log\det(I - \sum_{k=1}^{K} \beta_k L_k)\Big) = \mathrm{tr}\big[(I - \sum_{j=1}^{K} \beta_j L_j)^{-1}(-L_k)\big].$$

Going back to Eq.(7), there is $P$, the projection matrix, defined as

$$P = I - \frac{1}{K}\mathbf{1}\mathbf{1}^T \tag{10}$$

where $\mathbf{1} = [11\ldots1]^T$. The matrix $P$ enables the next solution of $\boldsymbol{\beta}^i$ to satisfy the constraint $\boldsymbol{\beta}^T\mathbf{1} = \delta$ such as

$$(\boldsymbol{\beta}^{i+1})^T\mathbf{1} = (\boldsymbol{\beta}^i + \nabla)^T\mathbf{1} = \delta$$

where $\nabla = -\mathbf{d}\boldsymbol{\beta}^i$. Since $(\boldsymbol{\beta}^i)^T\mathbf{1} = \delta$, $\nabla$ should be content with

$$\mathbf{1}^T\nabla = 0 \tag{11}$$

which implies $\nabla$ has to be projected onto an orthogonal space to $\mathbf{1}^T$. A general formula of orthogonal projection to $A$ when $A\nabla = \mathbf{0}$ is

$$P = I - A^T(AA^T)^{-1}A.$$

Eq.(10) results from specifying $A$ with $\mathbf{1}^T$ in the formula. With preconditioning of $\nabla$ with $P$, we now can assure Eq.(11),

$$\mathbf{1}^T(P\nabla) = \mathbf{1}^T(I - \mathbf{1}(\mathbf{1}^T\mathbf{1})^{-1}\mathbf{1}^T)\nabla = 0.$$

The $\alpha^i$ in Eq.(7) determines the learning rate during the update. We begin with $\alpha^i$ set to the maximum value under the condition $\beta_k^{i+1} \geq 0, \forall k$, and gradually reduce the magnitude as the iteration increases.

## 4 Experiments

### 4.1 Experimental Design

The task is to determine functional classes of yeast proteins. We used as a gold standard the functional catalogue provided by the MIPS Comprehensive Yeast Genome Database (CYGD-mips.gsf.de/proj/yeast). The top-level categories in the functional hierarchy produce 13 classes (see table 1). A protein can belong to several functional classes. In a total of 6355 yeast proteins, however, only 3588 have class labels. The remaining yeast proteins have uncertain function and are therefore not used in evaluation. We dealt with the prediction problem as 'one class-versus-all others' classification tasks, one for each functional class. See Lanckriet *et al.* (2004b) for more detail.

**Table 1.** 13 CYGD functional Classes

| | Classes | Size |
|---|---|---|
| 1 | metabolism | 1048 |
| 2 | energy | 242 |
| 3 | cell cycle and DNA processing | 600 |
| 4 | transcription | 753 |
| 5 | protein synthesis | 335 |
| 6 | protein fate | 578 |
| 7 | cellular transportation and transportation mechanism | 479 |
| 8 | cell rescue, defense and virulence | 264 |
| 9 | interaction with cell environment | 193 |
| 10 | cell fate | 411 |
| 11 | control of cell organization | 192 |
| 12 | transport facilitation | 306 |
| 13 | others | 81 |

The input is four different types of protein interaction graphs with proteins as nodes and interactions as edges. The graphs are represented as mostly binary matrices having non-zero entry if there is interaction between the row and column proteins, 0 otherwise. The followings are the input matrices:

$W_1$ : protein-protein interactions (MIPS physical interactions),
$W_2$ : genetic interactions (MIPS genetic interactions),
$W_3$ : co-participation in a protein complex (determined by tandem affinity purification, TAP), each entry is a count of the number of times two proteins appear together in a complex,
$W_4$ : co-participation in a protein complex, each entry is non-zero if and only if there is a bait-prey relationship.

There are proteins which show no interactions with others. For instance, $W_2$ of Fig.4 has 2769 (=1529+1240) zero entries, thus only 819 (=3588-2769) are available for semi-supervised learning. And no results for 2769 proteins remaining. This situation similarly arises in other graphs when they are considered individually. On the contrary, in a combined graph, more proteins can be used if it has at least one non-zero interaction from any graph. It amounts to the size of union of all non-zero entries in all graphs. In the problem in hand, 1529 of 3588 proteins have no interactions in any of the graphs. Consequently, 2059 (=3588-1529) proteins are preserved for learning, and hence leads to less information loss and more results. For non-zero entries, to combine graphs can also be of more advantage particularly when the outputs of individual graphs cannot reach an accord with each other.

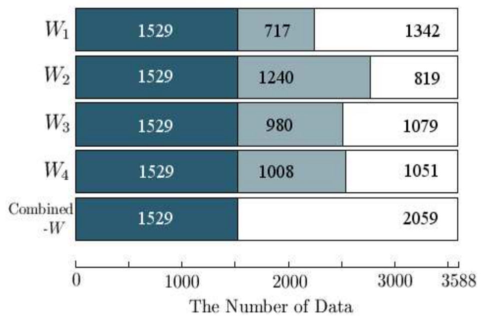


**Fig. 4.** The number of proteins available to learning.

All the matrices $W_k$ $(k = 1, ..., 4)$ were transformed to 'normalized' Laplacian $L_k$'s with dimension of 1042, 819, 1079 and 1051, respectively. For individual Laplacians composing the combined graph, zero columns and rows were inserted up to 2059 after transformation. Hereafter, we indicate each graph with $L_k$ $(k = 1, ..., 4)$ and the combined graph with $L_{com}$. The performance of $L_{com}$ was compared with those of individual $L_k$'s with the receiver operating characteristic (ROC) score, TP1FP, TP10FP, and error rate. ROC score is the area under ROC curve (see Fig.5) that plots true positive rate (sensitivity) as a function of false positive rate (1-specificity) for differing classification thresholds (Gribskov & Robinson, 1996; Hanley & McNeil, 1982). It measures the overall quality of the ranking induced by the classifier, rather than the quality of a single value of threshold in that ranking. An ROC score of 0.5 corresponds to random guessing, and an ROC score of 1.0 implies that the algorithm succeeded in putting all of the positive examples before all of the negatives. TP1FP is the rate of true positives at the point that yields 1% false positive rate on the ROC curve, and similarly, so is TP10FP. Error rate is a conventional performance measurement with a fixed value of threshold. Five-fold cross-validation (CV) was conducted for every class, and repeated five times in order to estimate the variance of the measurement values.

## 4.2 Results

A typical ROC curve, i.e. from the experiment of functional protein class 3, is shown in Fig.5. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the classifier. The figure therefore illustrates that $L_{com}$ is more accurate than any other single $L_k$. Fig.6 presents

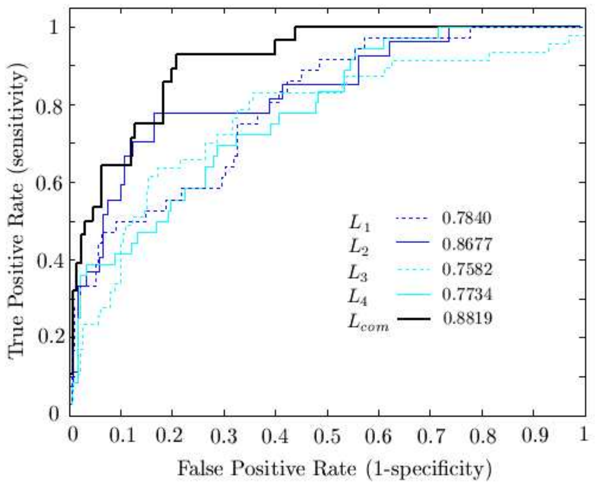


**Fig. 5.** ROC curve: Protein functional class 3. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the classifier.

the average ROC score of each class on its test set when performing five-fold CV five times. The height of the stem indicates to the ROC score. Within each group of stems, a thinner stem corresponds to an individual graph in due order, such as $L_1$, $L_2$, $L_3$, and $L_4$, respectively while a thicker one to $L_{com}$. Across the 13 classes, the combined graph $L_{com}$ outperformed a single one. In overall, $L_{com}$ yielded an ROC score of 0.8313 that surpasses all those of individual $L_k$'s, 0.7777, 0.7836, 0.7310, and 0.7238, respectively. See Fig. 7(a). The performances of TP1FP and TP10FP are depicted in Fig.7(b) and Fig.7(c). Among TP1FP's of $L_k$'s, 26.87% of $L_2$ is the most comparable to 30.07% of $L_{com}$. But the gap between the best and the second best becomes larger in TP10FP by 70.15% of $L_{com}$ and 61.22% of $L_2$. In Fig.7(d), the proportion of the colored bars indicates the relative weights of the different graphs when combined. Fig. 9 presents the error rates of 13 classes. A dot stands for the error rate of $L_k$, and the number beside it identifies the individual such as $k = 1, ..., 4$. The error rate of the combined graph is depicted as a square. The performance of $L_k$ differs 'class by class', and the variation of the difference between the best and the worst, which is represented as a line, changes significantly as well. Therefore, it is hard to put them in the order of which is superior to the other, accordingly we are hardly

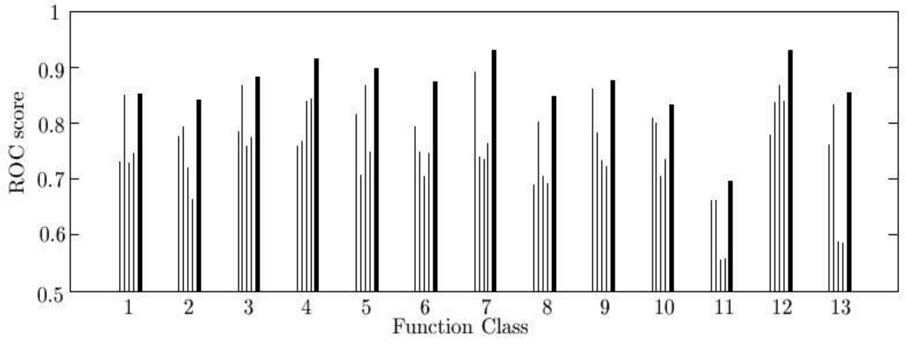
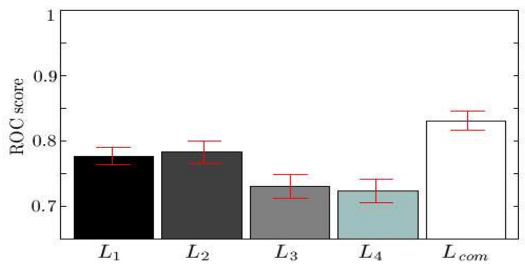


**Fig. 6.** ROC score for 13 functional protein classes: The height of the stem indicates to the ROC score. Within each group of stems, a thinner stem corresponds to an individual graph in due order, such as $L_1$, $L_2$, $L_3$, and $L_4$, respectively while a thicker one to $L_{com}$. Across the 13 classes, the combined graph $L_{com}$ outperformed a single one.

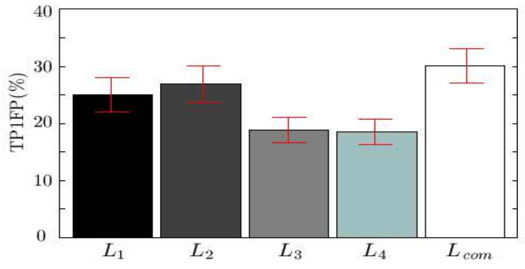
able to pick one out of them. Moreover, since the variation is also large a wrong choice of graph may lead to the worst performance. On the other hand, the error rate of the combined graph is always lower than any of those of individual graphs. In addition, one does not need to take the risk involved in the choice of graphs.

To test the significance of the difference between the combined graph and the individual one, McNemar's test was conducted (Dietterich, 1998). In principle, McNemar's test is to determine whether one learning algorithm outperforms another on a particular learning task. This non-parametric test could be seen as a Sign-Test in disguise. Fig.9 shows $p$-value distribution of McNemar's test. The smaller $p$-value indicates the better the combined graph is than an individual graph, while a $p$-value of 1 means no statistical difference between them. For most of 1300 experiments the combined graph outperforms the individual graphs. And in 504 out of 1300 McNemar's tests, there is a statistically significant difference between them (significance level $\alpha$=0.05).

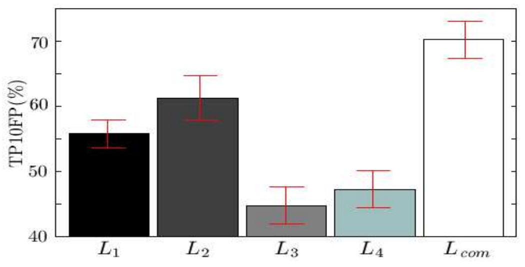
To do the comparison justice, we have only taken into consideration the proteins which are available to learning both for an individual graph and for the combined graph. For instance, when we compared $L_{com}$ with $L_k$, we reported performance only on 1342 proteins. See Fig.4. However, in the combined graph, we are still able to obtain the results of the rest 717 proteins – that is to say, the results of the proteins which are not available in an individual graph but available in the combined graph. Table 2 shows both error rates of the combined graph, 'Error A' for the former and 'Error B' for the latter. Error B is slightly larger than Error A, since it contains the proteins of which output is produced with a less number of input graphs. Nonetheless, it is still a reasonable figure as



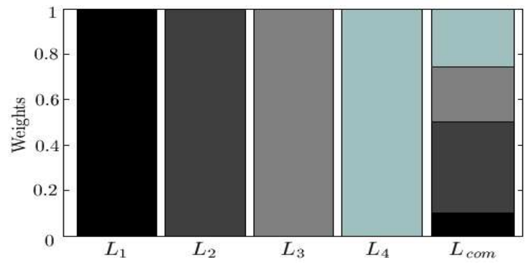
(a) ROC score



(b) TP1FP



(c) TP10FP



(d) Weight

**Fig. 7.** Overall performance: (a), (b), and (c) corresponds to ROC score, TP1FP and TP10FP, respectively. The height of bars indicates the average value of the measurements on five-fold CV repeated five times across 13 classes, and the error bar indicates the standard error. Seeing the results of (a), (b), and (c), the combined graph yields a better performance. In (d), the proportion of the colored bars indicates the relative weights of the different graphs when combined.

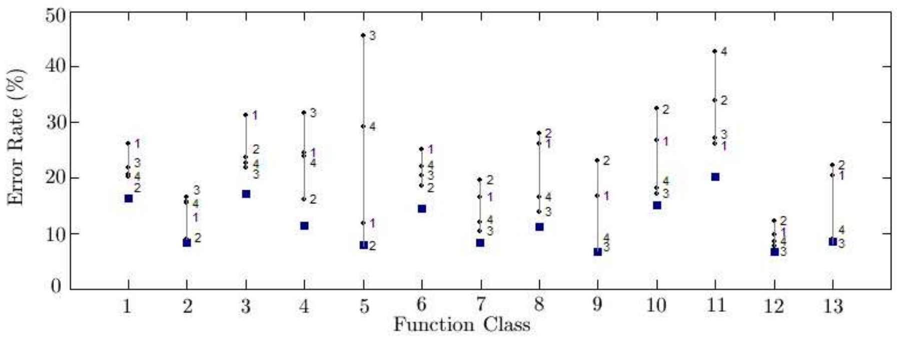


**Fig. 8.** Error rate for 13 functional protein classes: A dot stands for the error rate of $L_k$, and the number beside it identifies the individual, $k = 1, ..., 4$. And the variation of the difference between the best and the worst is represented as a line. The error rate of the combined graph is depicted as a square. The performance of $L_k$ differs 'class by class', and the variation changes significantly as well. On the other hand, the error rate of the combined graph is always lower than any of those of individual graphs. Moreover, one does not need to take the risk involved in the choice of graphs that may lead to the worst performance in some cases (classes).

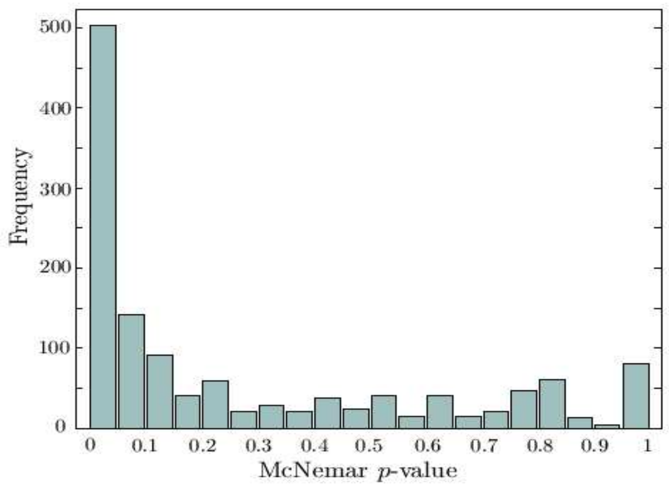


**Fig. 9.** $p$-value distribution of McNemar's test: The smaller $p$-value indicates the better the combined graph is than a single one, while a $p$-value of 1 means no statistical difference between them. For most of 1300 experiments the combined graph outperforms the individual graphs. And in 504 out of 1300 McNemar's tests, there is a statistically significant difference between them (significance level $\alpha$=0.05).

an error rate, and at least better than nothing gained.

**Table 2.** Error rates of the combined graph: 'Error A' is an error rate for the proteins which are available to learning both for an individual graph and for the combined graph. On the contrary, 'Error B' contains more proteins which are not available in an individual graph but available in the combined graph. Although Error B is slightly larger than Error A, because of relative lack of input information, it is nonetheless still a reasonable figure as an error rate.

| (%) | Functional Protein Classes | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | Avg. |
| Error A | 16.32 | 8.23 | 17.08 | 11.39 | 7.92 | 14.42 | 8.29 | 11.11 | 6.65 | 14.98 | 20.18 | 6.70 | 8.49 | 11.67 |
| Error B | 20.39 | 9.98 | 19.46 | 19.71 | 12.99 | 17.52 | 12.85 | 14.26 | 10.71 | 18.19 | 21.12 | 7.29 | 11.24 | 15.05 |

## 5 Conclusion

In this paper, we have presented a novel method for combining multiple graphs within a framework of semi-supervised learning. Similarly to EM algorithm, the method alternates minimizing the objective function with respect to network output and with respect to combining weight. When demonstrated to the task of functional class prediction of yeast proteins, the proposed method performed significantly better than the same algorithm trained on any single graph. The proposed method can be used as the alternative of model selection process. Given a single data source, it is likely to be represented in various ways by means of different parameters, i.e., different similarity measures, leading to different performances. Thus, instead of tedious process choosing one out of the candidate parameters, we can just combine them. From the pilot testing on standard data sets–Breast Cancer and Pima Indian Diabetes (from UCI Repository), we could obtain promising results. Although the method shows good performance, it has not yet compared with the other similar approach such as Lanckriet *et al.* (2004a). Therefore, investigating the merits and the demerits against the other will be the up-coming research.

## Acknowelgement

This work was supported by the Korean Science and Engineering Foundation (KOSEF).

# Bibliography

Bazaraa, M., Sherali, D., and Shetty, C., 1993. $N$onlinear Programming : Theory and Algorithms (2nd eds), Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, pp.385–398.

Belkin, M. and Niyogi, P., 2003a. Using Manifold Structure for Partially Labeled Classification. *Advances in Neural Information Processing Systems (NIPS) 15*, Cambridge, MA, USA, MIT Press.

Belkin, M., Matveeva, I., and Niyogi, P., 2003b. Regression and Regularization on Large Graphs. *University of Chicago, Computer Science, Technical Report TR-2003-11.*

Belkin, M. and Niyogi, P., 2004. Semi-Supervised Learning on Riemannian Manifolds. *Machine Learning Journal*, to appear, *(also Technical Report TR-2001-30, Univ. of Chicago, Computer Science Dept.)*

Bennett, K.P., 1999. Combining Support Vector and Mathematical Programming Methods for Classification, *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf et al. (Eds.), MIT press, pp. 307–326.

Blum, A. and Mitchell, T., 1998. Combining Labeled and Unlabeled Data with Co-Training. *In Proc. of the 11th Annual Conference on Computational Learning Theory*, Morgan Kaufmann, pp. 92–100.

Blum, A., and Chawla, S., 2001. Learning from Labeled and Unlabeled Data Using Graph Mincuts, *In Proc. of the 18th International Conference on Machine Learning (ICML)*, Morgan Kaufmann, pp. 19–26.

Chapelle, O., Weston, J., and Schölkopf, B., 2003. Cluster Kernels for Semi-Supervised Learning. *Advances in Neural Information Processing Systems (NIPS) 15*, MIT Press, Cambridge, MA, USA, pp. 585–592.

Chapelle, O., Schölkopf, B., and Weston, J., 2003. Semi-Supervised Learning through Principal Directions Estimation, *ICML Workshop, The Continuum from Labeled to Unlabeled Data in Machine Learning & Data Mining*

Chung, F.R.K., (1997). *Spectral Graph Theory. Regional Conference Series in Mathematics, no. 92.*

Dempster, A., Laird, N., and Rubin, D., 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society*, Series B, vol. 39, no. 1, pp. 1-38.

Deng, M., Chen, T., and Sun, F., 2003. Integrated Probabilistic Model for Functional Prediction of Proteins, *In Proc. of the Seventh Annual International Conference on Computational Biology (RECOMB)*, ACM, pp. 95–103.

Dietterich, T.G., 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms, *Neural Computation*, vol. 10, pp. 1895–1923.

Gribskov, M. and Robinson, N.L., 1996. Use of Receiver Operating Characteristic (ROC) Analysis to Evaluate Sequence Matching, *Computers and Chemistry*, vol. 20, no. 1, pp. 25-33.

Hanley, J.A. and McNeil, B.J., 1982. The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve,. *Radiology*,(2nd ed), Springer, pp. 29-36.

Hishigaki, H., Nakai, K., Ono, T., Tanigami, A., Takagi, T., 2001. Assessment of Prediction Accuracy of Protein Function from Protein–Protein Interaction Data. *Yeast*, vol. 18, no. 6, pp. 523–531.

Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y., and Barkai, N., 2002 Revealing Modular Organization in the Yeast Transcriptional Network *Nature Genetics*, vol. 31, no. 4, pp. 370–377.

Joachims, T., 1999. Transductive Inference for Text Classification using Support Vector Machines, *In Proc of the 16th International Conference on Machine Learning (ICML)*, Morgan Kaufmann, pp. 200–209.

Joachims, T., 2003. Transductive Learning via Spectral Graph Partitioning, *In Proc. of the Twentieth International Conference on Machine Learning (ICML)*, AAAI press, pp. 290–297.

Kandola, J., Shawe-Taylor, J., and Cristianini, N., 2002. Learning Semantic Similarity, *Advances in Neural Information Processing Systems (NIPS), 15*, MIT Press, pp. 657–664.

Kato, T., Tsuda, K., and Asai, K., 2004. Selective Integration of Mutiple Bilogical Data for Kernel Matrix Completion, *Bioinformatics*, vol. ??, pp. ??–??.

Kleinberg, J.M. and Tardos, E., 1999. Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields, *The 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 14–23.

Kondor, R.I. and Lafferty, J., 2002. Diffusion Kernels on Graphs and Other Discrete Input Spaces, *In Proc. of the Nineteenth International Conference on Machine Learning (ICML)*, AAAI press, pp. 315–322.

Lanckriet, G.R.G., Deng, M., Cristianini, N., Jordan, M.I., and Noble, W.S., 2004. Kernel-based Data Fusion and its Application to Protein Function Prediction in Yeast, *In Pacific Symposium on Biocomputing.*

Lanckriet, G.R.G., Cristianini, N., Bartlett, El Ghaoui, L., and Jordan, M.I., 2004. Learning the Kernel Matrix with Semidefinte Programming, *Journal of Machine Learning Research*, vol 5, pp. 27–72.

Lanckriet, G.R.G., De Bie, T., Cristianini, N., Jordan, M.I., and Noble, W.S., 2004. A Statistical Framework for Genomic Data Fusion, *Bioinformatics*, vol. 20, pp. 2626–2635.

Lee, T.I., Rinaldi, N.J., Robert, F., Odom, D.T., Bar-Joseph, Z., Gerber, G.K., Hannett, N.M., Harbison, C.T., Thompson, C.M., Simon, I., Zeitlinger, J., Jennings, E.G., Murray, H.L., Gordon, D.B., Ren, B., Wyrick, J.J., Tagne, J.B., Volkert, T.L., Fraenkel, E., Gifford, D.K., and Young, R.A., 2002. Transcriptional Regulatory Networks in *Saccharomyces cerevisiae*, *Science*, vol. 298, pp. 799–804.

McLachlan, G. and Krishnan, T., 1997. The EM Algorithm and Extensions, *Wiley Series in Probability and Statistics*, John Wiley & Sons.

Ng, A.Y., Jordan, M.I., and Weiss, Y., 2001. On Spectral Clustering: Analysis and an Algorithm, *Advances in Neural Information Processing Systems (NIPS), 14*, MIT Press, pp. 849-856.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T., 1998. Learning to Classify Text from Labeled and Unlabeled Documents, *In Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI)*, AAAI press, pp. 792–799.

Pavlidis, P., Weston, J., Cai, J., and Grundy, W.N., 2001. Gene Functional Classification from Heterogeneous Data, *In Proc of the Fifth International Conference on Computational Molecular Biology*, ACM press, pp. 242–248.

Seeger, M., 2000. Learning with Labeled and Unlabeled Data, *Technical Report*, Edinburgh University.

Segal, E., Shapira, M., Regev, A., Pe'er, D., Botstein, D., Koller, D., and Friedman, N., 2003. Module Networks: Identifying Regulatory Modules and Their Condition-Specific Regulators from Gene Expression Data, *Nature Genetics*, vol. 34, no. 2, pp. 166–176.

Schölkopf, B., Tsuda, K., and Vert, J.P., 2004. *Kernel Methods in Computational Biology*, MIT Press.

Schwikowski, B., Uetz, P., Fields, S., 2000. A Network of Protein-Protein Interactions in Yeast, *Nature Biotechnology*, vol. 18, no. 12, pp. 1257–1261.

Smola, A.J. and Kondor, R.I., 2003. Kernels and Regularization on Graphs, *In Proc. of Conference on Learning Theory (COLT)*, Springer Verlag, pp. 144–158.

Szummer, M. and Jaakkola, T., 2001. Partially Labeled Classification with Markov Random Walks, *Advances in Neural Information Processing Systems (NIPS), 14*, MIT Press, pp. 945–952.

Tsuda, K. and Noble, W.S., 2004. Learning Kernels from Biological Networks by Maximizing Entropy. *Bioinformatics*, vol. 20, pp.326–333.

Tsuda, K., Akaho, S., and Asai, K., 2004. The *em* Algorithm for Kernel Matrix Completion with Auxiliary Data *Journal of Machine Learning Research*, vol. 4, no. 1, pp. 67–81.

Uetz, P., Giot, L., Cagney, G., Mansfield, T.A., Judson, R.S., Knight, J.R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S., and Rothberg, J.M., 2000. A Comprehensive Analysis of Protein-Protein Interactions in *Saccharomyces cerevisiae*, *Nature*, vol. 403, no. 6770, pp. 623–627.

Vapnik, V., 1998. *Statistical Learning Theory*, Wiley, NY.

Vazquez, A., Flammini, A., Maritan, A., and Vespignani, A., 2003. Global Protein Function Prediction from Protein-Protein Interaction Networks, *Nature Biotechnology*, vol. 21, no. 6, pp. 697–700.

Vert, J.P. and Kanehisa, M., 2002. Graph-Driven Feature Extraction From Microarray Data Using Diffusion Kernels and Kernel CCA, *Advances in Neural Information Processing Systems (NIPS) 15*, Cambridge, MA, USA, MIT Press, pp. 1425–1432.

von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S.G., Fields, S., Bork, P., 2002. Comparative Assessment of Large-Scale Data Sets of Protein-Protein Interactions, *Nature*, vol. 23, no. 6887, pp. 399–403.

Yona, G., Linial, N., Linial, M., 1999. ProtoMap: Automatic Classification of Protein Sequences, a Hierarchy of Protein Families, and Local Maps of the Protein Space. *Proteins*, vol. 37, no. 3, pp. 360–378.

Zhou, D., Bousquet, O., Lal, T.N., Weston, J., and Schölkopf, B., 2004a. Learning with Local and Global Consistency, *Advances in Neural Information Processing Systems (NIPS) 16*, MIT Press, pp. 321–328.

Zhou, D. and Schölkopf, B., 2004b. Learning from Labeled and Unlabeled Data Using Random Walks, *In Proc. of the 26th Pattern Recognition Symposium, (DAGM'04), also available at http://www.kyb.tuebingen.mpg.de/bs/staff.php?gal=all.*

Zhu, X., Ghahramani, Z., and Lafferty, J., 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions, *In Proc. of the Twentieth International Conference on Machine Learning (ICML)*, AAAI press, pp. 912-919