Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 36 (2009) 3284–3292
Expert Systems with Applications
www.elsevier.com/locate/eswa

Protein functional class prediction with a combined graph

Hyunjung Shin [a,*], Koji Tsuda [b], Bernhard Schölkopf [b]

[a] Department of Industrial and Information Systems Engineering, Ajou University, San 5, Wonchun-dong, Yeoungtong-gu, 443–749 Suwon, Korea
[b] Department of Empirical Inference, Max-Planck-Institute for Biological Cybernetics, Spemannstrasse, 38, 72076 Tübingen, Germany

Abstract

In bioinformatics, there exist multiple descriptions of graphs for the same set of genes or proteins. For instance, in yeast systems, graph edges can represent different relationships such as protein–protein interactions, genetic interactions, or co-participation in a protein complex, etc. Relying on similarities between nodes, each graph can be used independently for prediction of protein function. However, since different graphs contain partly independent and partly complementary information about the problem at hand, one can enhance the total information extracted by combining all graphs. In this paper, we propose a method for integrating multiple graphs within a framework of semi-supervised learning. The method alternates between minimizing the objective function with respect to network output and with respect to combining weights. We apply the method to the task of protein functional class prediction in yeast. The proposed method performs significantly better than the same algorithm trained on any single graph.

Keywords: Bioinformatics; Protein function prediction; Semi-supervised learning

1. Introduction

In bioinformatics, many types of genomic data are frequently represented by using graphs of which nodes correspond to genes or proteins, and edges correspond to different relationships such as physical interactions of proteins (Schwikowski, Uetz, & Fields, 2000; Uetz et al., 2000; von Mering et al., 2002), gene regulatory relationships (Ihmels et al., 2002; Lee et al., 2002; Segal et al., 2003), or similarities between protein sequences (Yona, Linial, & Linial, 1999). One application using a graph representation is the prediction of protein functional class. It can be described as a binary-class classification problem on an undirected graph (see Fig. 1). A protein of known class is labeled either by '+1' or '−1' while a protein yet unknown its class is marked as '?'. The goal is to predict the class of unlabeled proteins relying on similarities between nodes. Prediction of protein functional class has been studied by means of various methods such as diffusion kernel (Tsuda & Noble, 2004), majority vote (Hishigaki, Nakai, Ono, Tanigami, & Takagi, 2001; Schwikowski et al., 2000), graph-based (Vazquez, Flammini, Maritan, & Vespignani, 2003), Bayesian (Deng, Chen, & Sun, 2003), and discriminative learning methods (Lanckriet, Deng, Cristianini, Jordan, & Noble, 2004a; Vert & Kanehisa, 2002). 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 information about the problem at hand. Thus, one can enhance the total information extracted by combining all graphs. Recently, there have arisen several methods for integrating heterogeneous data sources in bioinformatics. Most of them are based on kernel methods which represent data by means of kernel matrices defined by similarities between pairs of genes or proteins (for the kernel methods, refer to Schölkopf & Smola, 2002). Kernel matrices representing

* Corresponding author. Tel.: +82 31 219 2417; fax: +82 31 219 1610.
E-mail addresses: shin@ajou.ac.kr (H. Shin), tsuda@tuebingen.mpg.de (K. Tsuda), bernhard@tuebingen.mpg.de (B. Schölkopf).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.01.006

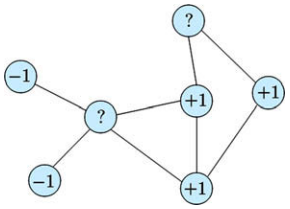

Fig. 1. The functional class prediction on a protein network graph: a protein of known class is labeled either by +1 or −1, and edges represent similarities between proteins. The task is to predict class of unlabeled proteins marked as '?'.

heterogenous data types are then combined into a single matrix by various techniques. Lanckriet, De Bie, Cristianini, Jordan, and Noble (2004c) exploit semi-definite programming (SDP, see also Lanckriet, Cristianini, Bartlett, El Ghaoui, & Jordan, 2004b) techniques to reduce the problem of finding optimizing kernel combinations to a convex optimization problem. This SDP-based approach yields satisfactory results 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, Tsuda, and Asai (2005) differentiate the worth of data sources such as 'expensive', which is data that is informative but difficult to obtain, and 'cheap', which is data that is 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 expectation-maximization (EM) algorithm 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, Akaho, & Asai (2004)). This EM-based method shows promising results when tested on supervised protein network inference and protein superfamily classification. The problem of multiple data sources (not only limited to graph representation) is often described as "data fusion," which is intensely dealt with in the chapter 11–13 of the recent book of Schölkopf, Tsuda, and Vert (2004). Other methods related to integration of data sources can be found in Pavlidis, Weston, Cai, and Grundy (2001), Vert and Kanehisa (2002), and Lanckriet et al. (2004a).

In the meantime, when data is represented as a graph, a more direct state-of-the-art in learning methods is semi-supervised learning. In semi-supervised learning, the labeled nodes provide information about the decision function, while the unlabeled nodes serve to reveal the structure of the data or data manifold by providing additional information (Nigam, McCallum, Thrun, & Mitchell, 1998; Chapelle, Schölkopf, & Weston, 2003b; Seeger, 2000; Zhou, Bousquet, Lal, Weston, & Schölkopf, 2004a). However, the problem of utilizing multiple data sources has yet to be explored in the framework of semi-supervised learning. In this paper, we propose a method for integrating multiple graphs within a framework of semi-supervised learning. The method alternates between minimizing the objective function with respect to network output and with respect to combining weight. We apply the method to the task of protein functional class prediction in yeast 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 remainder of this paper is organized as follows. In Section 2, we briefly introduce semi-supervised learning and review the recent literature. 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 $V = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is the vertex set and E is the edge set. A weight matrix associated with E, denoted as W, represents the magnitude of strength of linkage. W could be simply regarded as a non-negative similarity (or an affinity) matrix. The more similar $\boldsymbol{x}_i$–$\boldsymbol{x}_j$, 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 formulate the idea, let us define a function $f : V \to \Re$ 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 different from f_j's of adjacent vertices (B) under the constraints $f_i \equiv y_i$, $i = 1, \ldots, p$. 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, Weston, and Schölkopf (2003a). For technical convenience, a condition $\sum f_i = 0$ can be added to Eq. (1) (Belkin & Niyogi, 2003a, Belkin, Matveeva, & Niyogi, 2003b). Very often, the quadratic problem of Eq. (1) is represented in terms of matrix

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

where $\boldsymbol{y} = [\boldsymbol{y}_P^{\mathrm{T}} \ \boldsymbol{y}_Q^{\mathrm{T}}]^{\mathrm{T}}$, $y_p \in \{-1, 1\}$, $y_q \in \{0\}$, $p \in P$, $q \in Q$, and $\boldsymbol{f} = [\boldsymbol{f}_P^{\mathrm{T}} \ \boldsymbol{f}_Q^{\mathrm{T}}]^{\mathrm{T}}$, $f \in \Re$. μ is a parameter that trades off loss versus smoothness. The Laplacian is defined as $L = D - W$ where $D = \mathrm{diag}(d_i)$, $d_i = \sum_j w_{ij}$. Instead of L, a 'normalized Laplacian', $\widetilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ 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, Jordan, & Weiss, 2001; Seeger, 2000), graph s–t mincuts (Blum & Chawla, 2001) or multi-way cuts (Kleinberg & Tardos, 1999), co-training (Blum & Mitchell, 1998), random walks (Szummer & Jaakkola, 2001; Zhou & Schölkopf, 2004b; Zhu, Ghahramani, & Lafferty, 2003), and diffusion kernels (Kandola, Shawe-Taylor, & Cristianini, 2002; Kondor & Lafferty, 2002; Smola & Kondor, 2003). 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 (2) after transforming G into a Laplacian L or a normalized Laplacian $\widetilde{L}$. Now, consider the case where a set of graphs $\boldsymbol{G} = \{G_1, G_2, \ldots, G_k\}$ is given, each of which containing different aspects of the data (see Fig. 2. To integrate multiple graphs, we consider parameterized combinations of graphs. In particular, we form the linear combination of Laplacians

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

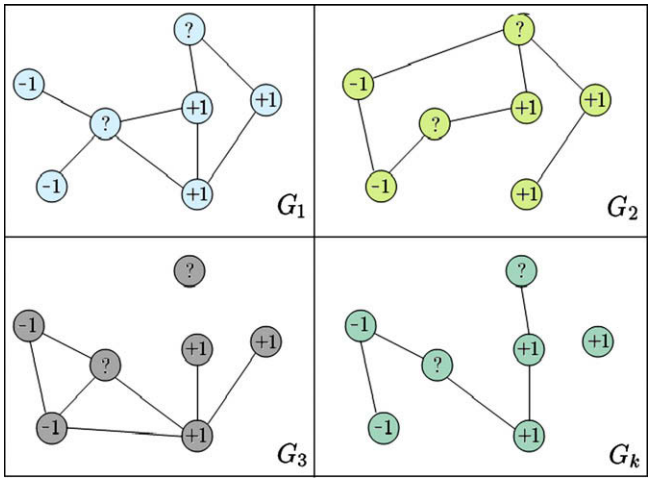

Fig. 2. Multiple graphs: consider the case where a set of graphs $\boldsymbol{G} = \{G_1, G_2, \ldots, G_k\}$ is given, each of which depicts a different aspect of the data. Each graph can solely predict the label of the unlabeled nodes marked as '?', depending on its own similarity measure between nodes.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

where the weights β_k are constrained to be positive to assure that each Laplacian contributes to prediction of $\boldsymbol{f}$. Inserting (3) into (2), we obtain

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

where $\boldsymbol{\beta} = [\beta_1 \beta_2 \ldots \beta_k]^{\mathrm{T}}$. However, since each L_k is positive definite, the value $\boldsymbol{\beta} = \mathbf{0}$ is trivially optimal. This can be avoided with the additional constraint $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{1} = \delta$, which yields

$$\begin{aligned} \min_{\boldsymbol{\beta},\boldsymbol{f}} \quad & \sum_{k=1}^{K} \beta_k \boldsymbol{f}^{\mathrm{T}} L_k \boldsymbol{f} + \mu(\boldsymbol{f}-\boldsymbol{y})^{\mathrm{T}}(\boldsymbol{f}-\boldsymbol{y}), \\ \text{s.t.} \quad & \boldsymbol{\beta} \geqslant 0,\ \boldsymbol{\beta}^{\mathrm{T}}\mathbf{1} = \delta, \end{aligned} \tag{5}$$

where $\mathbf{1} = [1\ 1 \ldots 1]^{\mathrm{T}}$. Nonetheless, the solution for $\boldsymbol{\beta}$ may still be more sparse than desired in that case. Namely, we always get only one non-zero weight, and all the others are zero, $\boldsymbol{\beta} = [0\ 0 \ldots \delta\ 0]^{\mathrm{T}}$. To get a reasonable set of weights, we consider to add an extra regularizer such that the term $\boldsymbol{f}^{\top} L(\beta)\boldsymbol{f}$ penalizes all the directions more equally. If the eigenvalues of $L(\beta)$ are $\lambda_1, \ldots, \lambda_n$, our aim is to regularize $L(\beta)$ such that all the eigenvalues become less variant. One way to achieve this is to penalize large eigenvalues so that they are pulled toward zero. We design the regularization term as

$$-\log\det(I - L(\beta)) = -\sum_{i=1}^{n} \log(1-\lambda_i).$$

Other choices might be possible, but our basic idea is to regularize the eigenvalues instead of the weights β_k's. Then, the optimization problem becomes

$$\begin{aligned} \min_{\boldsymbol{\beta},\boldsymbol{f}} \quad & R(\boldsymbol{\beta},\boldsymbol{f}) = \sum_{k=1}^{K} \beta_k \boldsymbol{f}^{\mathrm{T}} L_k \boldsymbol{f} - \log\det\left(I - \sum_{k=1}^{K} \beta_k L_k\right) \\ & + \mu(\boldsymbol{f}-\boldsymbol{y})^{\mathrm{T}} C(\boldsymbol{f}-\boldsymbol{y}), \\ \text{s.t.} \quad & \boldsymbol{\beta} \geqslant 0,\ \boldsymbol{\beta}^{\mathrm{T}}\mathbf{1} = \delta, \end{aligned} \tag{6}$$

where $\delta < 0.5$. In the third term corresponding to the loss function, a diagonal cost matrix C is incorporated which allows different misclassification costs, i.e., c_1 for $y_i = +1$, and c_2 for $y_j = -1$, $i, j \in P$.

The objective function of (6) is not jointly convex, but has nice properties: by fixing $\boldsymbol{\beta}$, the objective function is convex with respect to $\boldsymbol{f}$, while conversely, fixing $\boldsymbol{f}$ it is convex with respect to $\boldsymbol{\beta}$. Now, we can jointly minimize the objective function on $\boldsymbol{\beta}$ and $\boldsymbol{f}$. We bisect the solution process similar to 'E-step' and 'M-step' of EM algorithm, and alternatively optimize both steps (Dempster, Laird, & Rubin, 1977; McLachlan & Krishnan, 1997). Here, we denote them instead as '$\boldsymbol{\beta}$-step' and '$\boldsymbol{f}$-step,' respectively. The algorithm is presented in Fig. 3.

(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 step-(2) 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.

3.1. Solution of [$\boldsymbol{f}$-step]

When $\boldsymbol{\beta}$ is fixed, the solution $\boldsymbol{f}$ can be obtained by

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

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

$$\boldsymbol{f} = \mu C\left\{\sum_{k=1}^{K}\beta_k L_k + \mu C\right\}^{-1}\boldsymbol{y}. \tag{7}$$

3.2. 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:

$$\boldsymbol{\beta}^{i+1} = \boldsymbol{\beta}^i - \alpha^i P\mathbf{d}\boldsymbol{\beta}^i, \tag{8}$$

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

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

whose kth element is

$$\left.\frac{\partial R(\boldsymbol{\beta},\boldsymbol{f})}{\partial \beta_k}\right|_{(\boldsymbol{f}=\boldsymbol{f}^i,\boldsymbol{\beta}=\boldsymbol{\beta}^i)} = \boldsymbol{f}^{\mathrm{T}}L_k\boldsymbol{f} + \mathrm{tr}\left[\left(I - \sum_{j=1}^{K}\beta_j L_j\right)^{-1}L_k\right]. \tag{9}$$

In (9), $\mathrm{tr}\left[\left(I - \sum_{j=1}^{K}\beta_j L_j\right)^{-1}L_k\right]$ is the derivative of $\frac{\partial}{\partial\beta_k}\left(\log\det\left(I - \sum_k^K \beta_k L_k\right)\right)$ given by the following algebra. Let A be a matrix of which each element is parameterized with respect to t. The derivative of $\frac{\partial}{\partial t}(\log\det A)$ can be drawn by

$$\frac{\partial}{\partial t}(\log\det A) = \sum_{i,j}\frac{\partial}{\partial A_{ij}}(\log\det A)\times\frac{\partial A_{ij}}{\partial t} = \sum_{i,j}A_{ij}^{-1}\frac{\partial A_{ij}}{\partial t} \tag{10}$$

where $\sum_{i,j}\frac{\partial}{\partial A_{ij}}(\log\det A) = \frac{\partial}{\partial A}(\log\det A) = A^{-1}$. And if A and B are symmetric, then $\sum_{i,j}A_{ij}B_{ij} = \mathrm{tr}[AB]$. Thus (10) becomes

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

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

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

Going back to (8), the projection matrix P is defined as

$$P = I - \frac{1}{K}\mathbf{1}\mathbf{1}^{\mathrm{T}}, \tag{11}$$

where $\mathbf{1} = [1\ 1\ \ldots\ 1]^{\mathrm{T}}$. The matrix P enables the next solution of $\boldsymbol{\beta}^i$ to satisfy the constraint $\boldsymbol{\beta}^{\mathrm{T}}\mathbf{1} = \delta$ so that

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

where $\nabla = -\mathbf{d}\boldsymbol{\beta}^i$.

Since $(\boldsymbol{\beta}^i)^{\mathrm{T}}\mathbf{1} = \delta$, ∇ should satisfy

$$\mathbf{1}^{\mathrm{T}}\nabla = 0 \tag{12}$$

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

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

Eq. (11) results from specifying A with $\mathbf{1}^{\mathrm{T}}$ in the formula. With preconditioning of ∇ with P, we now can assure (12)

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

The α^i in (8) determines the learning rate during the update. We begin with α^i set to the maximum value under the condition $\beta_k^{i+1} \geqslant 0, \forall k$, and gradually reduce the magnitude as the iterations increase.

4. Experiments

4.1. Experimental design

Our goal 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.

The input is four different types of protein interaction graphs with proteins as nodes and interactions as edges.

Table 1
13 CYGD functional classes

	Classes
1	Metabolism
2	Energy
3	Cell cycle and DNA processing
4	Transcription
5	Protein synthesis
6	Protein fate
7	Cellular transportation and transportation mechanism
8	Cell rescue, defense and virulence
9	Interaction with cell environment
10	Cell fate
11	Control of cell organization
12	Transport facilitation
13	Others

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) proteins with no interaction, thus only 819 (=3588 − 2769) are available for semi-supervised learning. And no results for the 2769 proteins remaining. Similarly, this situation arises in other graphs when they are considered individually.

In contrast, using a combined graph, a protein can be used if it has at least one non-zero interaction from any graph. It amounts to the size of the union of all proteins with non-zero interaction in all graphs. In the problem at hand, 1529 of 3588 proteins have no interactions in any of the graphs. Consequently, 2059 (=3588 − 1529) proteins are preserved for learning. Combining graphs is also

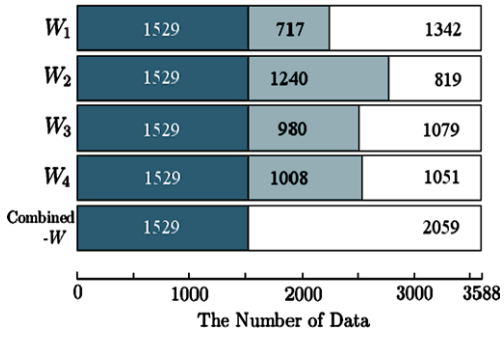

Fig. 4. The number of proteins available to learning: dark gray indicates the number of proteins with no interaction in any of the graphs, hence unavailable to learning. Light gray indicates the number of the proteins in which there is no interaction in the specific graph but available in at least one of the other graphs. These are thus not available to learning in the specified graph but available to that of the combined graph. For each graph, the number of proteins used for learning is depicted in white.

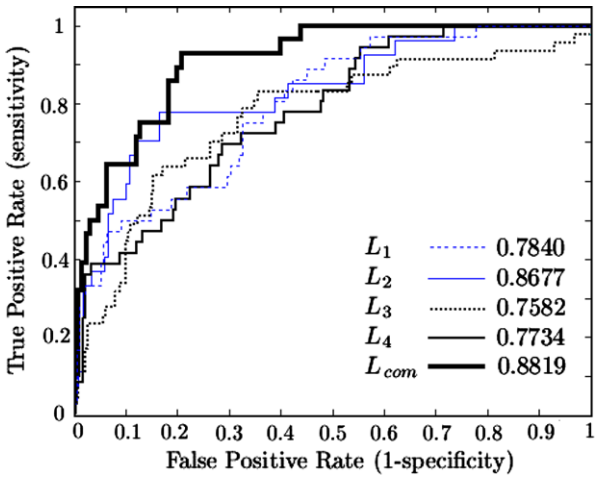

Fig. 5. ROC curve: protein functional class 3. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the classifier.

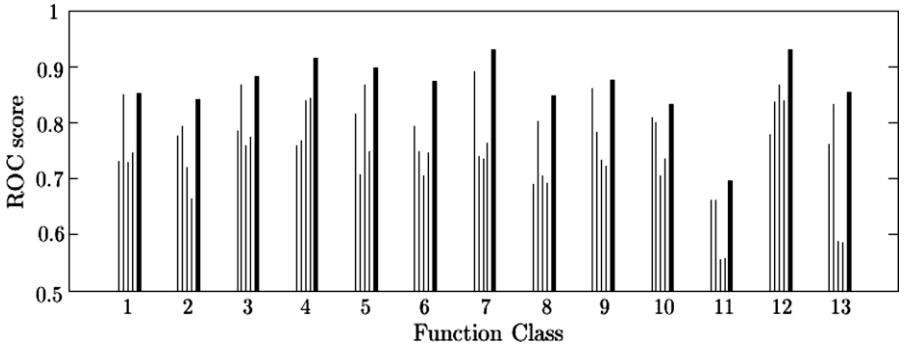

Fig. 6. ROC score for 13 functional protein classes: the height of the stem indicates to the ROC score. Within each group of stems, a thinner stem corresponds to an individual graph in order of such as L_1, L_2, L_3, and L_4, respectively while a thicker one corresponds to L_{com}. Across the 13 classes,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

advantageous for proteins with non-zero interactions particularly when individual graphs cannot reach an accord with each other.

All the matrices W_k $(k = 1, \ldots, 4)$ were transformed to 'normalized' Laplacian L_k's with dimensions of 1342, 819, 1079 and 1051, respectively. Individual Laplacians composing the combined graph, columns and rows were zero-padded up to 2059 after transformation. Hereafter, we indicate each graph with L_k $(k = 1, \ldots, 4)$ 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. The 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 and TP10FP are the rates of true positives at the point that yields 1% and 10% false positive rate on the ROC curve, respectively. Error rate is a conventional performance measurement with a fixed value threshold. Fivefold 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 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 the average ROC score of each class on its test set when performing fivefold 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} outperforms any given single L_k. Overall, L_{com} yields 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. 7a). The performances of TP1FP and TP10FP are depicted in Fig. 7b and 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. 7d, the proportion of the colored bars indicates the relative weights of the different graphs when combined. Fig. 8 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 $k = 1, \ldots, 4$. The error rate of the combined graph is depicted as a square. The performance of L_k differs 'class by class', and the difference between the best and the worst, which is represented as a line, changes significantly as well. Therefore, it is not appropriate to put them in the order of performance. Moreover, since the difference is also large a

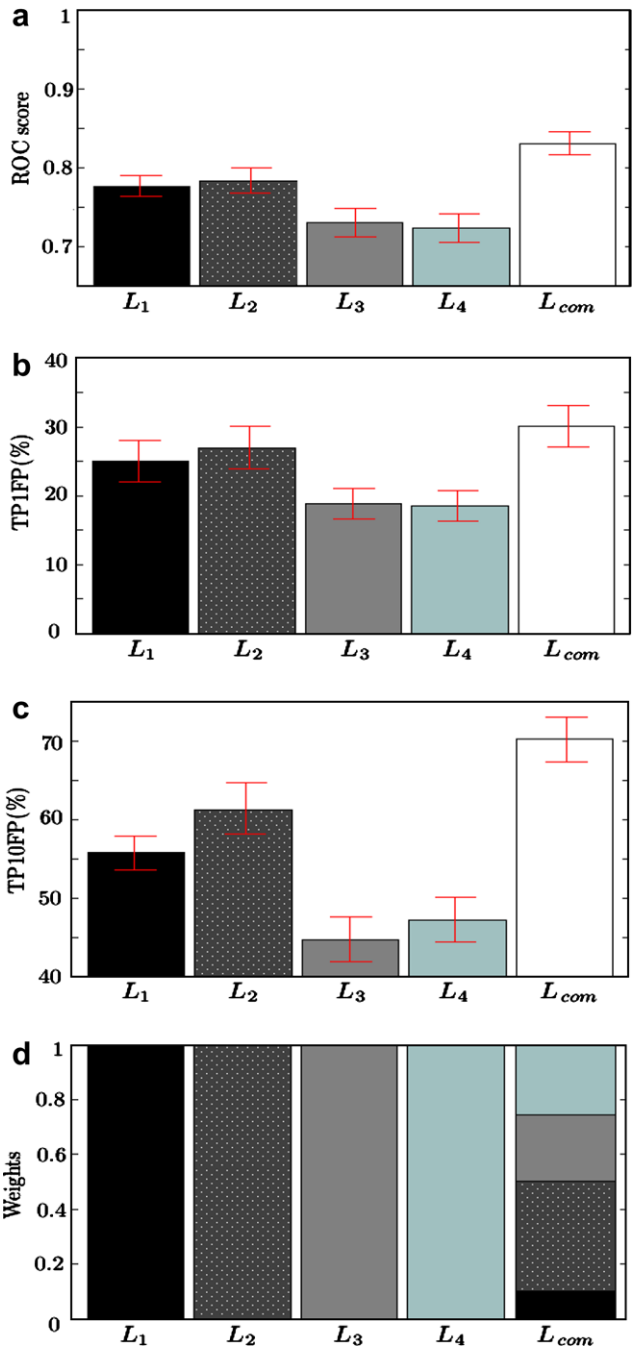

Fig. 7. Overall performance: (a)–(c) corresponds to ROC score, TP1FP and TP10FP, respectively. The height of bars indicates the average value of the measurements on fivefold CV repeated five times across 13 classes, and the error bar indicates the standard error. Seeing the results of (a)–(c), the combined graph yields a better performance. In (d), the proportion of the colored bars indicates the relative weights of the different graphs when combined. (a) ROC score, (b) TP1FP, (c) TP10FP and (d) weight. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

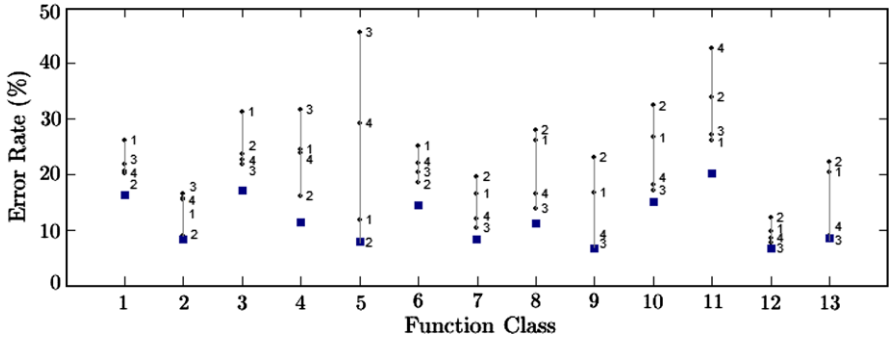

Fig. 8. Error rate for 13 functional protein classes: a dot stands for the error rate of L_k, and the number beside it identifies the individual, $k = 1, \ldots, 4$. The difference between the best and the worst is represented as a line. The error rate of the combined graph is depicted as a square. The performance of L_k differs 'class by class', and the difference changes significantly. 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 specific class.

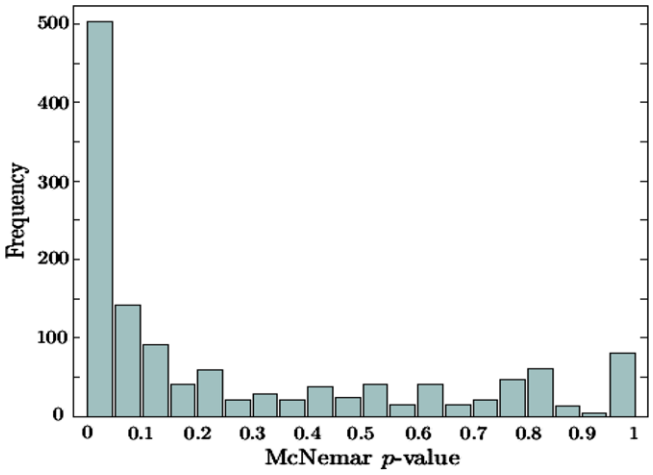

Fig. 9. p-Value distribution of McNemar's test: the smaller p-value indicates a more statistically significant difference between the combined graph versus any single graph, while a p-value of 1 indicates no statistical difference between them. A pairwise test between the combined graph and each of four graphs is conducted during five repetitions of fivefold cross-validation for 13 classes, which amounts to 1300 ($= 4 \times 5 \times 5 \times 13$). For most of 1300 experiments the combined graph outperforms the individual graphs. In 504 out of 1300 McNemar's tests, there is a statistically significant difference between them (at a significance level of $\alpha = 0.05$).

wrong choice of graph may lead to the worst performance in specific class. 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 individual ones, McNemar's test was conducted (Dietterich, 1998). In principle, McNemar's test is used 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. A pairwise test between the combined graph and each of four graphs is conducted during five repetitions of fivefold cross-validation for 13 classes, which amounts to 1300 ($= 4 \times 5 \times 5 \times 13$). 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 for another 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

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

(%)	Functional protein classes													Average
	1	2	3	4	5	6	7	8	9	10	11	12	13	
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

'Error B' contains more proteins which are not available to an individual graph but available to the combined graph. Although Error B is slightly larger than Error A, due to the relative lack of input information, it is nonetheless still a reasonable figure as an error rate.

A, since it contains the proteins of which output is produced with fewer input graphs. Nonetheless, Error B values are still reasonable.

5. Conclusion

In this paper, we have presented a method for combining multiple graphs within a framework of semi-supervised learning. Similar to the EM algorithm, the method alternates between minimizing the objective function with respect to network output and with respect to combining weight. When applied 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 also be used as an alternative to the 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 the tedious process of choosing one out of the candidate parameters, one can combine them with this method.

Acknowledgement

H.S. would be like to gratefully acknowledge support from Post Brain Korea 21 and the research Grant from Ajou University.

References

Belkin, M., & Niyogi, P. (2003a). Using manifold structure for partially labeled classification. *Advances in neural information processing systems (NIPS)* (Vol. 15). Cambridge, MA, USA: MIT Press.

Belkin, M., Matveeva, I., & Niyogi, P. (2003b). Regression and regularization on large graphs. University of Chicago, Computer Science, Technical Report TR-2003-11.

Belkin, M., & Niyogi, P. (2004). Semi-supervised learning on Riemannian manifolds. *Machine Learning, 56*(1–3), 209–239.

Bennett, K. P. (1999). Combining support vector and mathematical programming methods for classification. In B. Schölkopf et al. (Eds.), *Advances in kernel methods: Support vector learning* (pp. 307–326). MIT Press.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th annual conference on computational learning theory* (pp. 92–100). Morgan Kaufmann.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th international conference on machine learning (ICML)* (pp. 19–26). Morgan Kaufmann.

Chapelle, O., Weston, J., & Schölkopf, B. (2003a). Cluster kernels for semi-supervised learning. *Advances in neural information processing systems (NIPS)* (Vol. 15). Cambridge, MA, USA: MIT Press.

Chapelle, O., Schölkopf, B., & Weston, J. (2003). Semi-supervised learning through principal directions estimation. In *ICML workshop, the continuum from labeled to unlabeled data in machine learning and data mining*.

Chung, F. R. K. (1997). Spectral graph theory. In *Regional conference series in mathematics* (Vol. 92).

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B, 39*(1), 1–38.

Deng, M., Chen, T., & Sun, F. (2003). Integrated probabilistic model for functional prediction of proteins. In *Proceedings of the seventh annual international conference on computational biology (RECOMB)* (pp. 95–103). ACM.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation, 10*, 1895–1923.

Gribskov, M., & Robinson, N. L. (1996). Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry, 20*(1), 25–33.

Hanley, J. A., & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 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, 18*(6), 523–531.

Ihmels, J., Friedlander, G., Bergmann, S., Sarig, O., Ziv, Y., & Barkai, N. (2002). Revealing modular organization in the yeast transcriptional network. *Nature Genetics, 31*(4), 370–377.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the 16th international conference on machine learning (ICML)* (pp. 200–209). Morgan Kaufmann.

Joachims, T. (2003). Transductive Learning via Spectral Graph Partitioning. In *Proceedings of the 20th international conference on machine learning (ICML)* (pp. 290–297). AAAI Press.

Kandola, J., Shawe-Taylor, J., & Cristianini, N. (2002). Learning semantic similarity. *Advances in neural information processing systems (NIPS)* (Vol. 15). MIT Press.

Kato, T., Tsuda, K., & Asai, K. (2005). Selective integration of mutiple biological data for kernel matrix completion. *Bioinformatics, 21*(10), 2488–2495.

Kleinberg, J. M., & Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *The 40th annual IEEE symposium on foundations of computer science (FOCS)* (pp. 14–23).

Kondor, R. I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th international conference on machine learning (ICML)* (pp. 315–322). AAAI Press.

Lanckriet, G. R. G., Deng, M., Cristianini, N., Jordan, M. I., & 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, P., El Ghaoui, L., & Jordan, M. I. (2004b). Learning the kernel matrix with semidefinte programming. *Journal of Machine Learning Research, 5*, 27–72.

Lanckriet, G. R. G., De Bie, T., Cristianini, N., Jordan, M. I., & Noble, W. S. (2004c). A statistical framework for genomic data fusion. *Bioinformatics, 20*, 2626–2635.

Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., et al. (2002). Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science, 298*, 799–804.

McLachlan, G., & Krishnan, T. (1997). *The EM algorithm and extensions*. *Wiley series in probability and statistics*. John Wiley & Sons.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems (NIPS)* (Vol. 14). MIT Press, pp. 849–856.

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. In *Proceedings of the 15th national conference on artificial intelligence (AAAI)* (pp. 792–799). AAAI Press.

Pavlidis, P., Weston, J., Cai, J., & Grundy, W. N. (2001). Gene functional classification from heterogeneous data. In *Proceedings of the fifth international conference on computational molecular biology* (pp. 242–248). ACM Press.

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., et al. (2003). Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics, 34*(2), 166–176.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Schölkopf, B., Tsuda, K., & 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, 18*(12), 1257–1261.

Smola, A. J., & Kondor, R. I. (2003). Kernels and regularization on graphs. In *Proceedings of conference on learning theory (COLT)* (pp. 144–158). Springer-Verlag.

Szummer, M., & Jaakkola, T. (2001). Partially labeled classification with Markov random walks. *Advances in neural information processing systems (NIPS)* (Vol. 14). MIT Press, pp. 945–952.

Tsuda, K., & Noble, W. S. (2004). Learning kernels from biological networks by maximizing entropy. *Bioinformatics, 20*, 326–333.

Tsuda, K., Akaho, S., & Asai, K. (2004). The em algorithm for Kernel matrix completion with auxiliary data. *Journal of Machine Learning Research, 4*(1), 67–81.

Uetz, P., Giot, L., Cagney, G., Mansfield, T. A., Judson, R. S., Knight, J. R., et al. (2000). A comprehensive analysis of protein–protein interactions in *Saccharomyces cerevisiae*. *Nature, 403*(6770), 623–627.

Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.

Vazquez, A., Flammini, A., Maritan, A., & Vespignani, A. (2003). Global protein function prediction from protein–protein interaction networks. *Nature Biotechnology, 21*(6), 697–700.

Vert, J. P., & Kanehisa, M. (2002). *Graph-driven feature extraction from microarray data using diffusion kernels and kernel CCA*. *Advances in neural information processing systems (NIPS)* (Vol. 15). Cambridge, MA, USA: MIT Press, pp. 1425–1432.

von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S. G., Fields, S., et al. (2002). Comparative assessment of large-scale data sets of protein–protein interactions. *Nature, 23*(6887), 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, 37*(3), 360–378.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2004a). *Learning with local and global consistency*. *Advances in neural information processing systems (NIPS)* (Vol. 16). MIT Press, pp. 321–328.

Zhou, D., & Schölkopf, B. (2004b). Learning from labeled and unlabeled data using random walks. In *Proceedings of the 26th pattern recognition symposium, (DAGM'04)*. Available from <http://www.kyb.tuebingen.mpg.de/bs/staff.php?gal=all>.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th international conference on machine learning (ICML)* (pp. 912–919). AAAI Press.