BIOINFORMATICS

Vol. 21 Suppl. 2 2005, pages ii59–ii65
doi:10.1093/bioinformatics/bti1110

Protein Structure and Function

Fast protein classification with multiple networks

Koji Tsuda[1,2], HyunJung Shin[1,3,*] and Bernhard Schölkopf[1]

[1]Max Planck Institute for Biological Cybernetics, Spemannstrasse 38, 72076 Tübingen, Germany, [2]Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology (AIST), 2-42 Aomi Koto-ku, Tokyo, Japan and [3]Friedrich Miescher Laboratory, Max Planck Society, Spemannstrasse 37, 72076, Tübingen, Germany

ABSTRACT

Motivation: Support vector machines (SVMs) have been successfully used to classify proteins into functional categories. Recently, to integrate multiple data sources, a semidefinite programming (SDP) based SVM method was introduced. In SDP/SVM, multiple kernel matrices corresponding to each of data sources are combined with weights obtained by solving an SDP. However, when trying to apply SDP/SVM to large problems, the computational cost can become prohibitive, since both converting the data to a kernel matrix for the SVM and solving the SDP are time and memory demanding. Another application-specific drawback arises when some of the data sources are protein networks. A common method of converting the network to a kernel matrix is the diffusion kernel method, which has time complexity of $O(n^3)$, and produces a dense matrix of size $n \times n$.

Results: We propose an efficient method of protein classification using multiple protein networks. Available protein networks, such as a physical interaction network or a metabolic network, can be directly incorporated. Vectorial data can also be incorporated after conversion into a network by means of neighbor point connection. Similar to the SDP/SVM method, the combination weights are obtained by convex optimization. Due to the sparsity of network edges, the computation time is nearly linear in the number of edges of the combined network. Additionally, the combination weights provide information useful for discarding noisy or irrelevant networks. Experiments on function prediction of 3588 yeast proteins show promising results: the computation time is enormously reduced, while the accuracy is still comparable to the SDP/SVM method.

Availability: Software and data will be available on request.

Contact: shin@tuebingen.mpg.de

1 INTRODUCTION

To understand the complex mechanisms of the cell, it is crucial to identify the function of numerous proteins (Alberts *et al*., 1998). However, since identifying the protein function by biological experiments is still costly and difficult, there have been proposed a number of methods for inferring protein function by computational techniques [see Tsuda and Noble (2004) and references therein]. Typically, these methods use various kinds of information sources such as gene expression data, phylogenetic profiles and subcellular locations, because no single source is sufficient to reliably identify protein functions. Recently, it is getting increasingly popular to represent the relations among the proteins as a network. In such a network, nodes represent genes or proteins, and edges represent 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), closeness in a metabolic pathway (Kanehisa *et al*., 2004), similarities between protein sequences (Yona *et al*., 1999), etc. Protein networks have been used for function prediction in a number of approaches, for example, majority vote (Schwikowski *et al*., 2000; Hishigaki *et al*., 2001), graph-based (Vazquez *et al*., 2003), Bayesian (Deng *et al*., 2003), discriminative learning methods (Vert and Kanehisa, 2003; Lanckriet *et al*., 2004a), and probabilistic integration by log-likelihood scores (Lee *et al*., 2004).

Among these approaches, the support vector machine (SVM) has been particularly successful in function prediction using multiple data including networks (Lanckriet *et al*., 2004a,b). In order to combine different data types (e.g. vectors, trees and networks), each dataset is represented by a kernel matrix. When we have n proteins, the kernel matrix is an $n \times n$ positive definite dense matrix, representing similarities between proteins. A kernel matrix is obtained from each dataset. Finally, the kernel matrices are integrated into one matrix and fed into an SVM for inferring the labels of unannotated proteins. For example, the SDP/SVM method by Lanckriet *et al*. (2004a) uses a weighted sum of kernel matrices, where the weights are automatically determined such that irrelevant datasets can be discarded.

However, one of the inherent problems of SDP/SVM is its computational inefficiency. In theory, the method has the time complexity of $O(n^3)$, where n is the number of data or the dimension of kernel matrix. Thus, when the data is large-scale, learning may not be finished in a reasonable time.[1] The inefficiency is mainly caused by the fact that the kernel matrix is dense. Computing the product of two dense matrices already takes $O(n^3)$. In general, it is difficult to make the kernel methods faster than $O(n^3)$ without rather radical approximations (e.g. low rank approximation) (Schölkopf and Smola, 2002). One may attempt to sparsify the kernel matrix by setting small values to zero. But, in general, a kernel matrix artificially 'sparsified' is no longer positive definite, hence it may introduce local minima or convergence problems into the optimization problem for learning.

Another drawback of the SDP/SVM method arises when a protein network is used as an information source. Since the SVM requires

*To whom correspondence should be addressed.

[1]Recently, a fast and greedy approximation method was proposed (Bach *et al*., 2004); but the worst case complexity does not change.

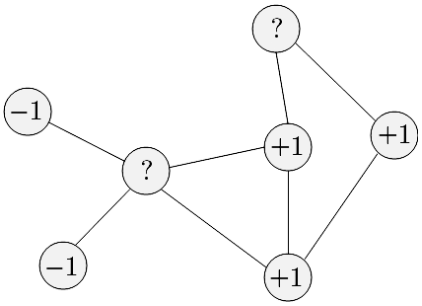

Fig. 1. Functional class prediction on a protein network: Focusing on a particular functional class, the function prediction problem boils down to two-class classification. An annotated protein is labeled either by $+1$ or -1. The positive label indicates that the protein belongs to the class. Edges represent associations between proteins. The task is to predict the class of the unlabeled proteins marked as '?'.

a kernel matrix of the data source, each network has to be converted to a corresponding kernel matrix. Conventionally, the diffusion kernel is used for that purpose (Kondor and Lafferty, 2002). However, it has a time complexity of $O(n^3)$, and produces a dense matrix of $n \times n$, making it thus computationally expensive both in time and memory.

In recent years, we have seen a significant progress of graph-based semi-supervised learning methods in the machine learning community (Zhou *et al*., 2004; Belkin and Niyogi, 2003; Zhu *et al*., 2003; Chapelle *et al*., 2003). As in kernel methods, n proteins are represented as an $n \times n$ symmetric matrix, but now it is sparse and therefore can be depicted as an undirected graph (Fig. 1), where each edge represents a non-zero entry. Our assumption is that an edge represents association of two proteins, thus the labels of two adjacent nodes are likely to be the same (See Section 2 for details). Each edge can have a positive weight, representing the degree of association. Focusing on a particular functional class, the function prediction problem boils down to a two-class classification problem. For annotated proteins, the labels are known ($+1$ for those belonging to the class, -1 otherwise). Our task is to predict the labels of unannotated proteins ('?' in the figure). In graph-based learning algorithms, the prediction can be done by solving a linear system with a sparse coefficient matrix, which is faster than the SVM learning by orders of magnitude (Spielman and Teng, 2004).

One important problem in graph-based learning, which has not yet been addressed, is the combination of multiple graphs (Fig. 2). A graph can be generated from a kernel matrix by thresholding. Also, one can use various kinds of protein networks directly. Since it seems unlikely that one graph contains all the information necessary for function prediction, one has to integrate all the graphs into one. However, some graphs can be harmful for accurate prediction, because they contain a number of false edges, or because the data itself has inherently nothing to do with the function prediction. Therefore, we need an algorithm to select 'good' graphs automatically. The need for automatic selection will get larger, as the number of available data increases due to the progress of biological screening techniques.

In this paper, we propose a new algorithm to assign weights to multiple networks, and thereby select important ones. The selection mechanism is close to the way that the SVM selects support vectors (Schölkopf and Smola, 2002). Label inference and weight assignment are formulated as one convex optimization problem (i.e. no local minima problems).

We applied our approach to functional class prediction of 3588 yeast proteins. We used five networks in all, three of which were from protein networks (co-participation in a protein complex, physical interactions, genetic interactions), and the remaining two were generated from non-network data (Pfam domain structure and gene expression). In comparison with the SDP/SVM approach, we get comparable prediction accuracy in a remarkably short time. When all the networks were combined with uniform weights, the prediction took only 1.4 s on a standard PC. Even when the weights were iteratively optimized, it finished in 49 s.

2 GRAPH-BASED LEARNING

First, we introduce the graph-based learning algorithm for a single network (Zhou *et al*., 2004). Let us assume a weighted graph G with n nodes indexed as $1, \ldots, n$. A symmetric weight matrix, denoted as W, represents the strength of linkage. All weights are nonnegative ($w_{ij} \geq 0$), and if $w_{ij} = 0$, there is no edge between nodes i and j. We assume that the first p training nodes have binary labels, $y_1, y_2, \ldots, y_p$, where $y_i \in \{-1, 1\}$, and the remaining $q = n - p$ test nodes are unlabeled. The goal is to predict the labels $y_{p+1}, \ldots, y_n$ by exploiting the structure of the graph under the assumption that a label of an unlabeled node is more likely to agree with those of more adjacent or more strongly connected nodes. In our case, the label indicates whether a protein belongs to a functional class or not. We solve this binary classification problem for every class to finally predict the functional classes of proteins.

Let us define an n-dimensional score vector $\boldsymbol{f} = (f_1, \ldots, f_n)^\top$. In learning, we determine $\boldsymbol{f}$ using all the available information, and in prediction, the labels are predicted by thresholding the score $f_{p+1}, \ldots, f_n$. We require (A) the score f_i should not be too different from the scores of adjacent vertices, and (B) the scores should be close to the given label y_i in training nodes. One can obtain f by minimizing the following quadratic functional:

$$\sum_{i=1}^{p} (f_i - y_i)^2 + \mu \sum_{i=p+1}^{n} f_i^2 + c \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2, \qquad (1)$$

The first term corresponds to the loss function in terms of condition (B), and the third term describes the *smoothness* of the scores in terms of condition (A). The parameter c trades off loss versus smoothness. The second term is a regularization term to keep the scores of unlabeled nodes in a reasonable range. Alternative choices of smoothness and loss functions can be found in Chapelle *et al*. (2003). From later on, we focus on the special case $\mu = 1$ (Zhou *et al*., 2004). Then, the three terms degenerate to the following two terms,

$$\min_{\boldsymbol{f}} (\boldsymbol{f} - \boldsymbol{y})^\top (\boldsymbol{f} - \boldsymbol{y}) + c \boldsymbol{f}^T L \boldsymbol{f}, \qquad (2)$$

where $\boldsymbol{y} = (y_1, \ldots, y_p, 0, \ldots, 0)^\top$, and the matrix L is called the graph Laplacian matrix (Chung, 1997), which is defined as $L = D - W$ where $D = \mathrm{diag}(d_i)$, $d_i = \sum_j w_{ij}$. Instead of L, the 'normalized Laplacian', $\tilde{L} = D^{-1/2} L D^{-1/2}$ can be used to get a similar result (Chung, 1997). The solution of this problem is obtained as

$$\boldsymbol{f} = (I + cL)^{-1} \boldsymbol{y}, \qquad (3)$$

where I is an identity matrix.

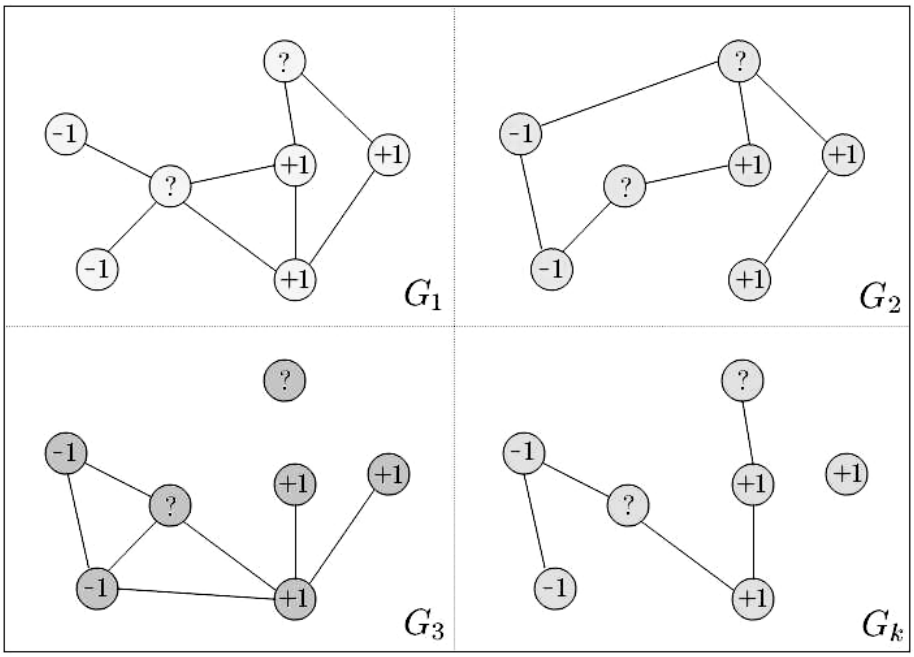

Fig. 2. Multiple graphs: A set of graphs is given, each of which depicts a different aspect of the proteins. Since different graphs contain partly independent and partly complementary pieces of information, one can enhance the total information by combining these graphs.

Actually, the score vector $\boldsymbol{f}$ is obtained by solving a large sparse linear system $\boldsymbol{y} = (I + cL)\boldsymbol{f}$. This numerical problem has been intensively studied, and there exist efficient algorithms, whose computational time is nearly linear in the number of non-zero entries in the coefficient matrix (Spielman and Teng, 2004). Therefore, the computation gets faster as the protein network gets sparser. Moreover, when the linear system solver is parallelized and distributed on a cluster system, the graph-based learning algorithm easily scales to much larger networks.

3 COMBINATION OF MULTIPLE NETWORKS

Since proteins are represented by many aspects (e.g. amino acid sequences, structures and interactions), it is natural to assume multiple networks. However, we do not really know in advance, which networks are important for predicting functional classes. Selecting exactly one network out of m networks would be relatively easy, because one can solve the learning problem using each network, and select the best one in terms of, say, the cross-validation error. However, as the integration of multiple data sources is essential to achieve high accuracy (Lanckriet *et al*., 2004a), our task is rather to choose $m_0 (\leq m)$ networks out of m. To examine every possible combination, we have to solve a combinatorial number of $\binom{m}{m_0}$ learning problems. In this section, we instead propose a convex programming-based algorithm to determine the important networks efficiently. The contents of the last section are already popular in the machine learning community. The novelty of this paper lies in the algorithm described in this section.

Without loss of generality, the optimization problem (2) is rewritten in the constrained form as

$$\min_{\boldsymbol{f},\gamma} (\boldsymbol{f} - \boldsymbol{y})^\top (\boldsymbol{f} - \boldsymbol{y}) + c\gamma, \quad \boldsymbol{f}^\top L \boldsymbol{f} \leq \gamma. \tag{4}$$

When we have multiple Laplacian matrices $L_1, \ldots, L_m$, this problem can be extended to take all of them into account,

$$\min_{\boldsymbol{f},\gamma} (\boldsymbol{f} - \boldsymbol{y})^\top (\boldsymbol{f} - \boldsymbol{y}) + c\gamma, \quad \boldsymbol{f}^\top L_k \boldsymbol{f} \leq \gamma, k = 1, \ldots, m. \tag{5}$$

This amounts to taking the upper bound of the smoothness function $\boldsymbol{f}^\top L_k \boldsymbol{f}$ over all networks and applying it for regularization.

To investigate the properties of the solution of Equation (5), let us derive the dual problem (Schölkopf and Smola, 2002). Our convex optimization problem can be rewritten as the following min-max problem using Lagrange multipliers,

$$\max_{\alpha,\eta} \min_{\boldsymbol{f},\gamma} (\boldsymbol{f} - \boldsymbol{y})^\top (\boldsymbol{f} - \boldsymbol{y}) + c\gamma + \sum_{k=1}^{m} \alpha_k (\boldsymbol{f}^T L_k \boldsymbol{f} - \gamma) - \eta\gamma, \tag{6}$$

where the Lagrange multipliers satisfy $\alpha_k, \eta \geq 0$. If the inner minimization problem is solved analytically, we end up with the maximization problem with respect to the Lagrange multipliers only. This maximization problem is called the 'dual problem', which is often easier to solve. The optimal solution of the original problem is written in terms of the Lagrange multipliers, which assist in the interpretation of the optimal solution. For example, for SVMs, the analysis using the dual problem is effectively used for explaining the basic properties of the discriminant hyperplane (e.g. large margin and support vectors) (Schölkopf and Smola, 2002).

Let us solve the inner optimization problem. Setting the derivative with respect to γ to zero, we get

$$c - \sum_{k=1}^{m} \alpha_k = \eta. \tag{7}$$

Since $\eta \geq 0$, the sum of α_k is constrained as $\sum_{k=1}^m \alpha_k \leq c$. Substituting Equation (7) into Equation (6), we have

$$(\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + \sum_{k=1}^m \alpha_k \boldsymbol{f}^T L_k \boldsymbol{f} \tag{8}$$

Setting the derivative with respect to $\boldsymbol{f}$ to zero, we get

$$\left(I + \sum_{k=1}^m \alpha_k L_k\right)\boldsymbol{f} = \boldsymbol{y}. \tag{9}$$

This is solved as

$$\boldsymbol{f} = \left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1}\boldsymbol{y}. \tag{10}$$

Now the optimal solution of $\boldsymbol{f}$ is written in terms of the Lagrange multipliers α_k. Comparing Equation (10) with the single network solution (3), it is clear that the Lagrange multipliers α_k play a role as the combination weights of the networks. Also, the parameter c constrains the sum of all weights.

Substituting Equation (10) into the Lagrangian (6), we get the following dual problem:

$$\begin{aligned} \max_\alpha \quad & \boldsymbol{y}^\top\boldsymbol{y} - \boldsymbol{y}^\top\left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1}\boldsymbol{y} \\ \text{s.t.} \quad & \textstyle\sum_k \alpha_k \leq c. \end{aligned} \tag{11}$$

Ignoring a constant term, this maximization problem is equivalent to the following minimization problem:

$$\begin{aligned} \min_\alpha \quad & \boldsymbol{y}^\top\left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1}\boldsymbol{y} \\ \text{s.t.} \quad & \textstyle\sum_k \alpha_k \leq c. \end{aligned} \tag{12}$$

Denote by $d(\alpha)$ the dual objective function (12). Due to the Karush–Kuhn–Tucker conditions, we have $\alpha_k(\boldsymbol{f}^\top L_k \boldsymbol{f} - \gamma) = 0$ at the optimal solution. Therefore, $\alpha_k = 0$ iff $\boldsymbol{f}^\top L_k \boldsymbol{f} < \gamma$, and so $\alpha_k > 0$ iff $\boldsymbol{f}^\top L_k \boldsymbol{f} = \gamma$. If the constraint $\boldsymbol{f}^\top L_k \boldsymbol{f} \leq \gamma$ is satisfied as an equality only for several networks, we get a sparse solution for α_k, namely some of α_k's are exactly zero. The networks with zero weight (i.e. $\alpha_k = 0$) are considered as unnecessary, because the optimal score vector $\boldsymbol{f}$ would not change, even if we removed those networks. On the other hand, the networks with non-zero weight satisfy $\boldsymbol{f}^\top L_k \boldsymbol{f} = \gamma$ and play a essential role in determining the score vector.

3.1 Regularized version

In combining networks, one has to balance two contradicting goals: selection and integration. In practical applications, we found the above algorithm too selective (i.e. the maximum weight is too dominant). To make the weights $\{\alpha_k\}_{k=1}^m$ more uniform, we introduce other terms as follows:

$$\begin{aligned} \min_{\boldsymbol{f},\xi,\gamma} \quad & (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + c\gamma + c_0 \sum_{k=1}^m \xi_k \\ \text{s.t.} \quad & \boldsymbol{f}^T L_k \boldsymbol{f} \leq \gamma + \xi_k, \quad \xi_k \geq 0, \gamma \geq 0. \end{aligned} \tag{13}$$

The dual problem then reads

$$\begin{aligned} \min_\alpha \quad & \boldsymbol{y}^\top\left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1}\boldsymbol{y} \equiv d(\alpha) \\ \text{s.t.} \quad & 0 \leq \alpha_k \leq c_0, \ \textstyle\sum_k \alpha_k \leq c. \end{aligned} \tag{14}$$

This extension adds a new parameter c_0, which gives us large flexibility. When $c_0 = c$, we recover the solution of Equation (12), and on the other extreme ($c_0 = c/m$), we obtain uniform weights.

3.2 Optimization

The optimization problem is solved, for example, by gradient descent. This requires the computation of the dual objective $d(\alpha)$ as well as its derivative. The derivative is described as

$$\frac{\partial d}{\partial \alpha_j} = -\boldsymbol{y}^\top\left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1} L_j \left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1}\boldsymbol{y}. \tag{15}$$

Note that we used the relation

$$\frac{\partial}{\partial x}Y^{-1} = -Y^{-1}\left(\frac{\partial}{\partial x}Y\right)Y^{-1}.$$

Although we have the inverse matrix $\left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1}$ in the solution (10), the objective (12), and the derivative (15) as well, we do not need to calculate it explicitly, because it always appears as the vector $\left(I + \sum_{k=1}^m \alpha_k L_k\right)^{-1}\boldsymbol{y}$, which can be obtained as the solution of sparse linear systems. Therefore, the computational cost of the dual objective and the derivative is nearly linear in the number of non-zero entries of $\sum_{k=1}^m \alpha_k L_k$ (Spielman and Teng, 2004).

4 FUNCTION PREDICTION EXPERIMENTS

The proposed method was evaluated in function class prediction of yeast proteins, based on the dataset provided by Lanckriet *et al.* (2004a). The dataset contains 3588 proteins, and the function of each protein is labeled according to the MIPS Comprehensive Yeast Genome Database (CYGD-mips.gsf.de/proj/yeast). It focuses only on the 13 highest-level categories of the functional hierarchy (Table 1). Note that a protein can belong to several functional classes. We solved a two-class classification problem for every functional class, and evaluated the accuracy of each classification.

Table 2 lists the five different types of protein networks used in experiments. The networks W_1 and W_5 are created from non-network data. The networks W_2, W_3, and W_4 have binary edges (i.e. 0/1 weights), and are taken from the database directly. See (Lanckriet *et al.*, 2004a) for more information. The sparsity of the Laplacian matrices (i.e. the fraction of non-zero entries) is shown in the last column of the table. All the matrices are very sparse (0.7% density in maximum), which contributes to memory-saving. If one tries to use diffusion kernel, it will take much more memory ($1/0.007 \approx 142$). In learning, all the networks were transformed to normalized Laplacian matrices L_k's. The prediction accuracy is evaluated by 5-fold cross-validation three times. For each partition of training and test nodes, the receiver operating characteristic (ROC) score is calculated, and then averaged over all the five partitions. The ROC score is calculated as the area under the ROC curve which plots true positive rate (sensitivity) as a function of false positive rate (1-specificity) for differing classification thresholds (Gribskov and Robinson, 1996). 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 the positive examples ahead of all of the

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

Table 2. Protein networks used in the experiment

Matrix	Description	Density (%)
W_1	Network created from Pfam domain structure. A protein is represented by a 4950-dimensional binary vector, in which each bit represents the presence or absence of one Pfam domain. An edge is created if the inner product between two vectors exceeds 0.06. The edge weight corresponds to the inner product	0.7805
W_2	Co-participation in a protein complex (determined by tandem affinity purification, TAP). An edge is created if there is a bait-prey relationship between two protein	0.0570
W_3	Protein–protein interactions (MIPS physical interactions)	0.0565
W_4	Genetic interactions (MIPS genetic interactions)	0.0435
W_5	Network created from the cell cycle gene expression measurements (Spellman *et al*., 1998). An edge is created if the Pearson coefficient of two profiles exceeds 0.8. The edge weight is set to 1. This is identical with the network used in Deng *et al.* (2003)	0.0919

'Density' shows the fraction of non-zero entries in the respective Laplacian matrix.

negatives. The value of parameter c was determined by the results of search over

$$c \in \{0.05, 0.1, 0.25, 0.5, 1, 2.5, 5, 10, 25, 50, 100\}.$$

And the chosen value for each class is as follows:

$$(5, 5, 25, 25, 10, 10, 5, 5, 10, 10, 100, 2.5, 25).$$

The proposed method was compared with individual networks, and with the state-of-the-art SDP/SVM method based on the reported results (Lanckriet *et al*., 2004a). Successively, we compared the proposed method, namely, 'integration by optimized weights' with 'integration by fixed weights'.

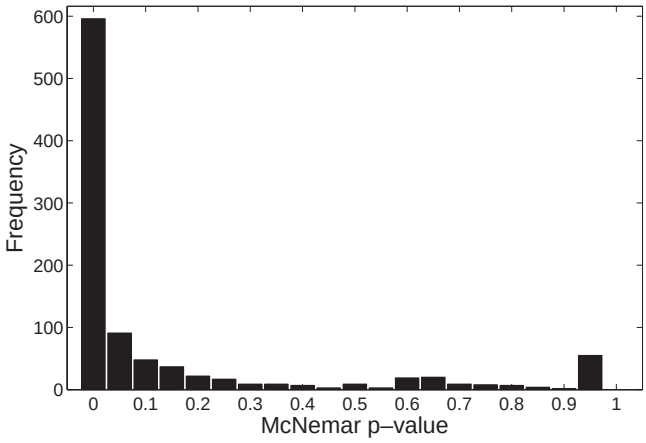

Fig. 3. P-value distribution of McNemar's test: For most of 975 McNemar's test trials, L_{opt} outperforms L_k's. Particularly, for 61% of the total number of trials, there is a statistically significant difference (at a significance level of $\alpha = 0.05$), which corresponds to the leftmost bar.

4.1 Comparison with individual networks

First, we compared the performance between the combined network (L_{opt}) and individual ones (L_k's). McNemar's test has been conducted in order to test the significance of the difference in performance (Dietterich, 1998). In principle, McNemar's test is used to determine whether one learning algorithm outperforms another on a particular learning task. Figure 3 shows empirical P-value distribution of McNemar's test. A small P-value indicates that L_{opt} is better than L_k, while a P-value of 1 means no statistical difference between them. The total number of trials amounts to 975 ($= 3$ repetitions $\times$ 5 pairwise-tests $\times$ 5 CVs $\times$ 13 clsses). In 594 (61%) trials, there is a statistically significant difference (significance level $\alpha = 0.05$), which corresponds to the leftmost bar in Figure 3. Specifically, in each pairwise comparison, L_{opt} significantly outperforms L_k's in 55.31, 58.31, 60.03, 68.21 and 61.03% of the total number of trials, respectively. Figure 4 presents the comparison of ROC scores to the best performing individual network. Note that the reported ROC scores were calculated based on the test nodes with edges in the best individual network (the test nodes with no edges were excluded).

4.2 Comparison with SDP/SVM

4.2.1 Accuracy Figure 5 presents the comparison between the ROC scores of the proposed method and those of SDP/SVM method reported by Lanckriet *et al*. (2004a). We also plot the scores of the Markov random field (MRF) method proposed by Deng *et al*. (2003). For most classes, the proposed method achieves high scores, which are similar to the SDP/SVM methods. In classes 11 and 13 the proposed method was worse (but still better than the MRF method), which is probably due to the superior generalization performance of the SVM. We could not perform tests of significance since it was not available to obtain all the details of experimental results of MRF or SDP/SVM. However, taking into account the simplicity and efficiency (and thus scalability) of the proposed method, we consider the results shown good enough to motivate the use of our method instead of SDP/SVM.

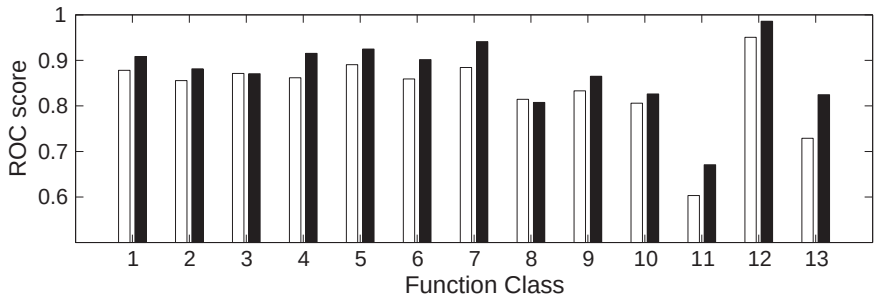

Fig. 4. Comparing ROC scores of combined networks and the best performing individual network. Within each group of bars, a white bar corresponds to the best individual network, while a black bar corresponds to $L_{\rm opt}$. Across the 13 classes, $L_{\rm opt}$ outperforms the best performing individual.

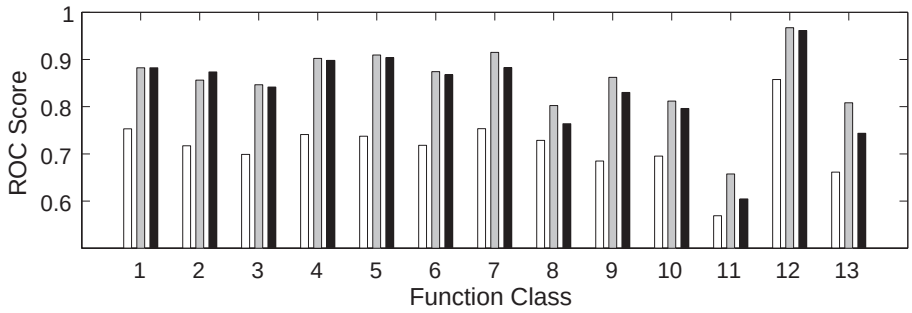

Fig. 5. ROC score comparison between MRF, SDP/SVM, and $L_{\rm opt}$ for 13 functional protein classes: white bars correspond to the MRF method of Deng *et al.* (2003); gray bars correspond to the SDP/SVM method of Lanckriet *et al.* (2004a). Black bars correspond to $L_{\rm opt}$.

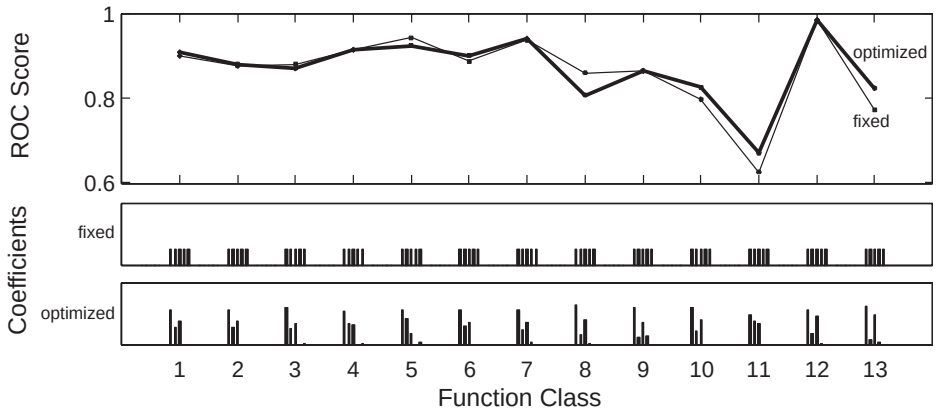

Fig. 6. Prediction accuracy for 13 functional protein classes. The thin and thick lines in the upper figure show the ROC scores of $L_{\rm fix}$ and $L_{\rm opt}$, respectively. In the middle and lower figures, the combination weights of $L_{\rm fix}$ and $L_{\rm opt}$ are described, respectively.

4.2.2 Computational time Solving the sparse linear system which appears in the solution (10), the objective (12), and the derivative (15), only took 1.41 s (standard deviation 0.013) with MATLAB command `mldivide` in a standard 2.2 GHz PC with 1 GByte of memory. Solving the dual problem (14) that includes multiple times of computation for the sparse linear system, took 49.3 s (standard deviation 14.8) with MATLAB command `fmincon`. On the contrary, SDP/SVM method takes several hours [according to discussion with an author of Lanckriet *et al.* (2004a)]. Thus, the shorter computational time will be compromisable on non-significant loss of accuracy against SDP/SVM method.

4.3 Comparison with fixed weight integration

Another combined network was defined as $L_{\rm fix} = \frac{1}{m}\sum_{k=1}^{m} L_k$. Note that the uniform weights corresponds to the solution of Equation (14) when $c_0 = c/m = 0.2c$. The ROC scores for all functional classes are shown in Figure 6, together with the weights of networks. The optimization of weights did not always lead to better ROC scores (except for the classes 10, 11, 13). However, the advantage of $L_{\rm opt}$ is that redundant networks are automatically identified. Looking at the weights of $L_{\rm opt}$ in the figure, W_4 and W_5 almost always have very low weights, which suggests that these two networks can be

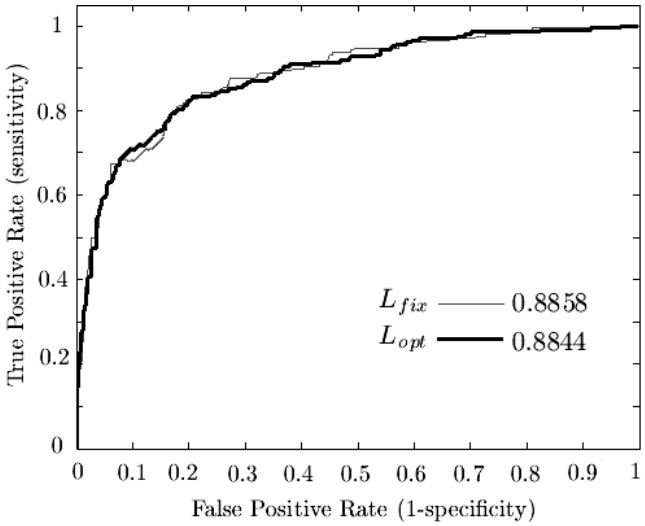

Fig. 7. ROC curve for protein functional class 1. The thin and thick curves correspond to L_{fix} and L_{opt}, respectively.

removed. The weights would be valuable when function prediction experiments are conducted in similar situations, e.g. for different species, because one needs not to prepare the redundant data. There was no statistically significant difference between L_{opt} and L_{fix} in performance (McNemar's test, significance level $\alpha = 0.05$). Figure 7 presents a typical ROC curve for the first class.

5 CONCLUDING REMARKS

We have presented a new algorithm to classify proteins based on multiple networks. The application of this algorithm is not limited to function prediction. Many problems such as subcellular localization and operon detection can also be formulated as classification problems on networks, and solved in a similar way. We believe that graph-based learning algorithms are going to become standard methods in computational biology, because they exhibit very good generalization ability as well as excellent efficiency both in terms of memory and speed. In future work, we will follow this direction and try to solve other problems on multiple networks.

ACKNOWLEDGEMENTS

The authors would like to thank G.R.G. Lanckriet for providing the yeast protein data for comparison. We would like to thank D. Zhou, W.S. Noble and A. Ben-hur for fruitful discussions. H.J.S. was supported by the Korean Science and Engineering Foundation (KOSEF).

Conflict of Interest: none declared.

REFERENCES

Alberts,B., Bray,D., Johnson,A., Lewis,J., Raff,M., Roberts,K. and Walter,P. (1998) *Essential Cell Biology: An Introduction to the Molecular Biology of the Cell*. Garland Science Publishing, New York.

Bach,F., Lanckriet,G. and Jordan,M. (2004) Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, Banff, Canada, ACM Press, pp. 6–13.

Belkin,M. and Niyogi,P. (2003) Using manifold structure for partially labelled classification. In Becker,S., Thrun,S. and Obermayer,K. (eds), *Advances in Neural Information Processing Systems (NIPS) 15*, Vol. 15. MIT Press.

Chapelle,O., Weston,J. and Schölkopf,B. (2003) Cluster kernels for semi-supervised learning. In Becker,S., Thrun,S. and Obermayer,K. (eds), *Advances in Neural Information Processing Systems (NIPS) 15*. MIT Press, pp. 585–592.

Chung,F.R.K. (1997) *Spectral Graph Theory*. Number 92 in Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI.

Deng,M., Chen,T. and Sun,F. (2003) An integrated probabilistic model for functional prediction of proteins. In Miller,W., Vingron,M., Istrail,S., Pevzner,P. and Waterman,M. (eds), *Proceedings of the Seventh Annual International Conference on Computational Biology (RECOMB)*, Berlin, Germany, ACM, pp. 95–103.

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

Gribskov,M. and Robinson,N.L. (1996) Use of receiver operating characteristic (roc) analysis to evaluate sequence matching. *Comput. Chem.*, **20**, 25–33.

Hishigaki,H. *et al*. (2001) Assessment of prediction accuracy of protein function from protein–protein interaction data. *Yeast*, **18**, 523–531.

Ihmels,J. *et al.* (2002) Revealing modular organization in the yeast transcriptional network. *Nat. Genet.*, **31**, 370–377.

Kanehisa,M. *et al*. (2004) The KEGG resources for deciphering genome. *Nucleic Acids Res.*, **32**, D277–D280.

Kondor,I. and Lafferty,J. (2002) Diffusion kernels on graphs and other discrete structures. In Sammut,C. and Hoffmann,A.G. (eds), *Proceedings of the Nineteenth International Conference on Machine Learning, (ICML 2002)*, Sydney, Australia, Morgan Kaufmann, pp. 315–322.

Lanckriet,G.R.G., Deng,M., Cristianini,N., Jordan,M.I. and Noble,W.S. (2004a) Kernel-based data fusion and its application to protein function prediction in yeast. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, Big Island, HI.

Lanckriet,G.R.G. *et al.* (2004b) A statistical framework for genomic data fusion. *Bioinformatics*, **20**, 2626–2635.

Lee,I. *et al.* (2004) A probabilistic functional network of yeast genes. *Science*, **306**, 1555–1558.

Lee,T.I. *et al*. (2002) Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, **298**, 799–804.

Schölkopf,B. and Smola,A.J. (2002) *Learning with Kernels*. MIT Press, Cambridge, MA.

Schwikowski,B. *et al.* (2000) A network of protein–protein interactions in yeast. *Nat. Biotechnol.*, **18**, 1257–1261.

Segal,E. *et al*. (2003) Module networks: identifying regulatory modules and their condition specific regulators from gene expression data. *Nat. Biotechnol.*, **34**, 166–176.

Spellman,P.T. *et al*. (1998) Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, **9**, 3273–3297.

Spielman,D.A. and Teng,S.H. (2004) Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, Montreal, Canada, ACM Press, pp. 81–90.

Tsuda,K. and Noble,W.S. (2004) Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, **20** (suppl. 1), i326–i333.

Uetz,P. *et al*. (2000) A comprehensive analysis of protein–protein interactions in *Saccharomyces cerevisiae*. *Nature*, **403**, 623–627.

Vazquez,A. *et al*. (2003) Global protein function prediction from protein–protein interaction networks. *Nat. Biotechnol.*, **21**, 697–700.

Vert,J.P. and Kanehisa,M. (2003) Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. In Becker,S., Thrun,S. and Obermayer,K. (eds), *Advances in Neural Information Processing Systems 15*. MIT Press, pp. 1425–1432.

von Mering,C. *et al*. (2002) Comparative assessment of large-scale data sets of protein–protein interactions. *Nature*, **417**, 399–403.

Yona,G. *et al*. (1999) Protomap: automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space. *Proteins*, **37**, 360–678.

Zhou,D., Bousquet,O., Weston,J., and Schölkopf,B. (2004) Learning with local and global consistency. In *Advances in Neural Information Processing Systems (NIPS) 16*. MIT Press, pp. 321–328.

Zhu,X., Ghahramani,Z. and Lafferty,J. (2003) Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, Washington, DC, AAAI Press, pp. 912–919.