社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－BIO－1（1）
2005／7／25

複数のネットワークを用いたタンパク質の高速分類

津田 宏治 †

† 産総研 生命情報科学研究センター

要 旨

本論文では、物理的相互作用ネットワークや、代謝ネットワークといった複数のタンパク質ネットワークを用いた高速なタンパク質の機能分類法を提案する。この方法では、各ネットワークに重みが自動的に割り振られ、機能分類に有効な情報をもつネットワークを特定することができる。Lanckriet et al.(2004) の提案した SVM に基く手法と実験的に比較した結果、提案手法は、はるかに高速であり、また予測精度は同程度であることが分った。

Fast Protein Classification with Multiple Networks

Koji Tsuda†

†Computational Biology Research Center,
National Institute of Advanced Industrial Science and Technology

Abstract

We propose an efficient method of protein classification using multiple protein networks. Multiple networks, such as physical interaction networks or metabolic networks, are combined to classify proteins, e.g., for function prediction. The combination weights are automatically determined by convex optimization so that noisy or irrelevant networks can be automatically discarded. We compared our method with the SVM-based approach by Lanckriet et al. (2004) in function prediction of 3588 yeast proteins, and obtained favorable results in computation time and prediction accuracy.

1 Introduction

To understand the complex mechanisms of the cell, it is crucial to identify the function of numerous proteins. 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 that the relations among the proteins are represented as a *network*. In such a network, nodes represent genes or proteins, and edges represent physical interactions of the proteins [von Mering et al., 2002], gene regulatory relationships [Lee et al., 2002], closeness in a metabolic pathway [Kanehisa et al., 2004], similarities between protein sequences, etc. Protein networks have been used for function prediction in a number of approaches, for example, majority vote [Schwikowski et al., 2000], graph-based [Vazquez et al., 2003], Bayesian [Deng et al., 2003], discriminative learning methods [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 data set 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 data set. 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 data sets 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). 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 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

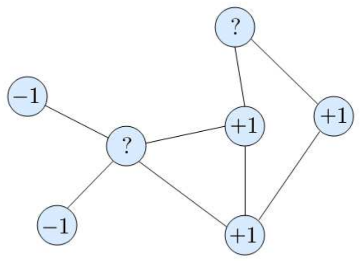

図 1: Functional class prediction on a protein network: Focusing on a particular functional class, the function prediction problem boils down to two-class classification. An annotated protein is labeled either by $+1$ or -1. The positive label indicates that the protein belongs to the class. Edges represent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

methods in the machine learning community [Zhou et al., 2004]. 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 (Figure 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 (Figure 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 con-

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

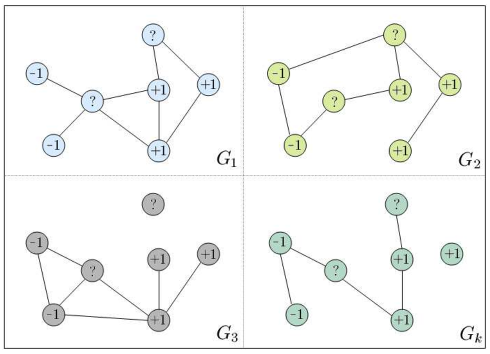

図 2: Multiple graphs: A set of graphs is given, each of which depicts a different aspect of the proteins. Since different graphs contain partly independent and partly complementary pieces of information, one can enhance the total information by combining these graphs.

tain 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. 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 total, three of which come from protein networks (co-participation in a protein complex, physical interactions, genetic interactions), and the remaining two are 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 are combined with uniform weights, the prediction took only 1.4 seconds on a standard PC. Even when the weights are iteratively optimized, it finished in 49 seconds.

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 $1, \dots, n$. 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, $y_1, y_2, \dots, y_p$, where $y_i \in \{-1, 1\}$, and the remaining $q = n - p$ test nodes are unlabeled. The goal is to predict the labels $y_{p+1}, \dots, y_n$ 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 $\boldsymbol{f} = (f_1, \cdots, f_n)^\top$. In learning, we determine $\boldsymbol{f}$ using all the available information, and in prediction, the labels are predicted by thresholding the score $f_{p+1}, \dots, f_n$. 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:

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

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,

$$\min_{\boldsymbol{f}} \ (\boldsymbol{f} - \boldsymbol{y})^\top (\boldsymbol{f} - \boldsymbol{y}) + c \boldsymbol{f}^T L \boldsymbol{f}, \qquad (1)$$

where $\boldsymbol{y} = (y_1, \dots, y_p, 0, \dots, 0)^\top$, 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', $\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ can be used to get a similar result [Chung, 1997]. The solution of this problem is obtained as

$$\boldsymbol{f} = (I + cL)^{-1}\boldsymbol{y} \tag{2}$$

where I is an identity matrix.

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 nonzero 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 (1) is rewritten in the constrained form as

$$\min_{\boldsymbol{f},\gamma} (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + c\gamma, \quad \boldsymbol{f}^\top L \boldsymbol{f} \leq \gamma. \tag{3}$$

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

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

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 (4), let us derive the dual problem. Our convex optimization problem can be rewritten as the following min-max problem using Lagrange multipliers,

$$\begin{aligned} \max_{\alpha,\eta} \min_{f,\gamma} \quad & (\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + c\gamma \\ & + \textstyle\sum_{k=1}^m \alpha_k(\boldsymbol{f}^T L_k \boldsymbol{f} - \gamma) - \eta\gamma, \end{aligned} \tag{5}$$

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 support vector machines, 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).

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

$$c - \sum_{k=1}^m \alpha_k = \eta. \tag{6}$$

Since $\eta \geq 0$, the sum of α_k is constrained as $\sum_{k=1}^m \alpha_k \leq c$. Substituting (6) into (5), we have

$$(\boldsymbol{f}-\boldsymbol{y})^\top(\boldsymbol{f}-\boldsymbol{y}) + \sum_{k=1}^m \alpha_k \boldsymbol{f}^T L_k \boldsymbol{f} \tag{7}$$

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

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

This is solved as

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

Now the optimal solution of $\boldsymbol{f}$ is written in terms of the Lagrange multipliers α_k. Comparing (9) with the single network solution (2), 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 (9) into the Lagrangian (5), we get the following dual problem:

$$\begin{array}{ll} \max_\alpha & \boldsymbol{y}^\top \boldsymbol{y} - \boldsymbol{y}^\top (I + \sum_{k=1}^m \alpha_k L_k)^{-1} \boldsymbol{y} \\ & \sum_k \alpha_k \le c. \end{array} \qquad (10)$$

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

$$\begin{array}{ll} \min_\alpha & \boldsymbol{y}^\top (I + \sum_{k=1}^m \alpha_k L_k)^{-1} \boldsymbol{y} \\ & \sum_k \alpha_k \le c. \end{array} \qquad (11)$$

Denote by $d(\alpha)$ the dual objective function (11). Due to the Karush-Kuhn-Tucker (KKT) 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 $\boldsymbol{f}^\top L_k \boldsymbol{f} \le \gamma$ 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 nonzero 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:

$$\begin{array}{ll} \min\limits_{\boldsymbol{f},\xi,\gamma} & (\boldsymbol{f} - \boldsymbol{y})^\top (\boldsymbol{f} - \boldsymbol{y}) + c\gamma + c_0 \sum_{k=1}^m \xi_k \\ & \boldsymbol{f}^T L_k \boldsymbol{f} \le \gamma + \xi_k, \;\; \xi_k \ge 0, \gamma \ge 0. \end{array} \qquad (12)$$

The dual problem then reads

$$\begin{array}{ll} \min_\alpha & \boldsymbol{y}^\top (I + \sum_{k=1}^m \alpha_k L_k)^{-1} \boldsymbol{y} \equiv d(\alpha) \\ & 0 \le \alpha_k \le c_0, \; \sum_k \alpha_k \le c. \end{array} \qquad (13)$$

This extension adds a new parameter c_0, which gives us large flexibility. When $c_0 = c$, we recover the solution of (11), 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 $\frac{\partial d}{\partial \alpha_j} = -\boldsymbol{y}^\top (I + \sum_{k=1}^m \alpha_k L_k)^{-1} L_j (I + \sum_{k=1}^m \alpha_k L_k)^{-1} \boldsymbol{y}$. Note that we used the relation $\frac{\partial}{\partial x} Y^{-1} = -Y^{-1} (\frac{\partial}{\partial x} Y) Y^{-1}$. Although we have the inverse matrix $(I + \sum_{k=1}^m \alpha_k L_k)^{-1}$ in the solution (9), the objective (11), and the derivative as well, we do not need to calculate it explicitly, because it always appears as the vector $(I + \sum_{k=1}^m \alpha_k L_k)^{-1} \boldsymbol{y}$, 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 nonzero 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 labelled 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. Notice 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 1 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 in-

表 1: Protein networks used in the experiment. ‘Density’ shows the fraction of non-zero entries in the respective Laplacian matrix.

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 proteins.	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

formation. The sparsity of the Laplacian matrices (i.e. the fraction of nonzero 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 five-fold cross-validation three times. For each partition of training and test nodes, the ROC (receiver operating characteristic) score is calculated, and then averaged over all the five partitions. 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 the state of the art SDP/SVM method based on the reported results [Lanckriet et al., 2004a]. Succeedingly, we compared the proposed method, namely, ‘integration by optimized weights’ with ’integration by fixed weights.’

4.1 Comparison with SDP/SVM

Figure 3 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 detail 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 shown results good enough to motivate the use of our method instead of SDP/SVM. Solving the sparse linear system took only 1.41 seconds (standard deviation 0.013) with MATLAB command `mldivide` in a standard 2.2Ghz PC with 1GByte of memory. Solving the dual problem (13) that includes multiple times of computation for the sparse linear system, took 49.3 seconds (standard deviation 14.8) with MATLAB command `fmincon`. On the contrary, SDP/SVM method

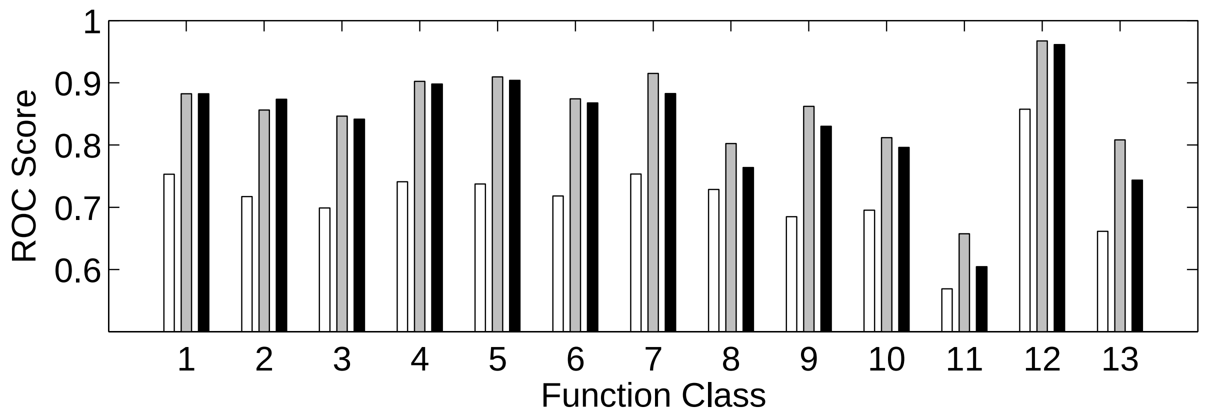

図 3: ROC score comparison between MRF, SDP/SVM, and L_{opt} for 13 functional protein classes: White bars correspond to the MRF method of Deng et al. [2003]; gray bars correspond to the SDP/SVM method of Lanckriet et al. [2004a]. Black bars correspond to L_{opt}.

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.

4.2 Comparison with Fixed Weight Integration

Another combined network was defined as $L_{fix} = \frac{1}{m}\sum_{k=1}^{m} L_k$. Note that the uniform weights corresponds to the solution of (13) when $c_0 = c/m = 0.2c$. The ROC scores for all functional classes are shown in Figure 4, 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_{opt} is that redundant networks are automatically identified. Looking at the weights of L_{opt} in the figure, W_4 and W_5 almost always have very low weights, which suggests that these two networks can be 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 α=0.05).

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.

参考文献

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

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

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

M. Deng, T. Chen, and F. Sun. An integrated probabilistic model for functional prediction of proteins.

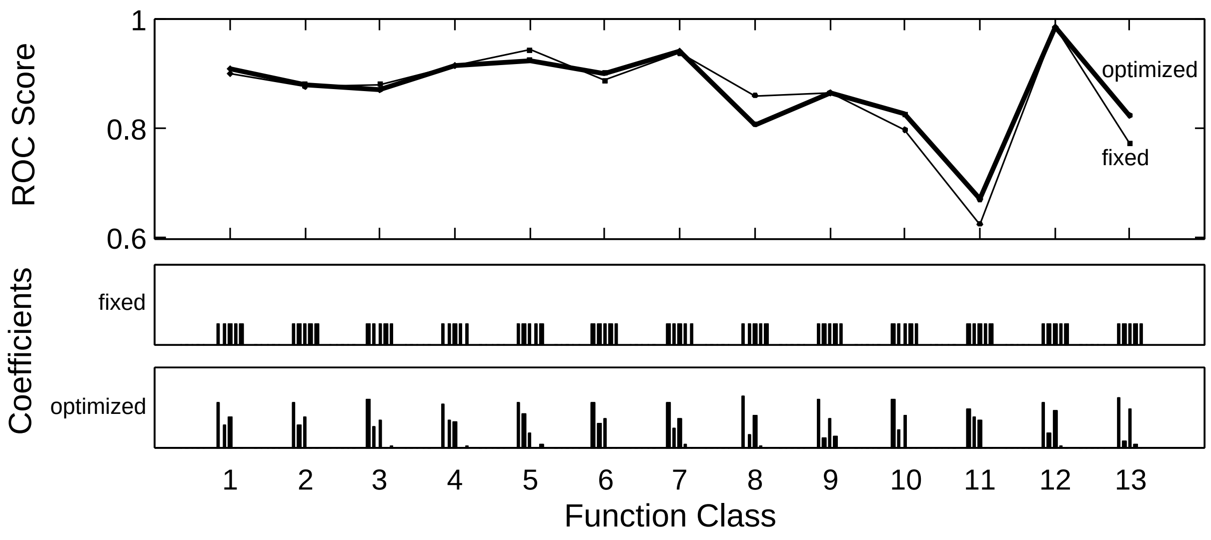

図 4: Prediction accuracy for 13 functional protein classes. The thin and thick lines in the upper figure show the ROC scores of L_{fix} and L_{opt}, respectively. In the middle and lower figures, the combination weights of L_{fix} and L_{opt} are described, respectively.

In W. Miller, M. Vingron, S. Istrail, P. Pevzner, and M. Waterman, editors, *Proceedings of the Seventh Annual International Conference on Computational Biology (RECOMB)*, pages 95–103. ACM, 2003.

M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M .Hattori. The KEGG resources for deciphering genome. *Nucleic Acids Res.*, 32:D277–D280, 2004.

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

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

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

I. Lee, S.V. Date, A.T. Adai, and E.M. Marcotte. A probabilistic functional network of yeast genes. *Science*, 306 (5701):1555–1558, 2004.

T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. R. Harbison, C. M. Thompson, I. Simon, et al. Transcriptional regulatory networks in *Saccharomyces cerevisiae. Science*, 298:799–804, 2002.

B. Schwikowski, P. Uetz, and S. Fields. A network of protein-protein interactions in yeast. *Nature Biotechnology*, 18:1257–1261, 2000.

P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell*, 9:3273–3297, 1998.

D.A. Spielman and S.H. Teng. 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*, pages 81–90. ACM Press, 2004.

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

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

C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Olivier, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

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