Graph-based Protein Functional Classification

Hyunjung Shin*
Dept. of Industrial & Information Systems Engineering, Ajou University, Suwon, 443–749, Korea
shin@ajou.ac.kr

Andreas Martin Lisewski*
Dept. of Molecular & Human Genetics, Baylor College of Medicine, Houston, Texas, 77030, USA
lisewski@bcm.edu

Olivier Lichtarge
Dept. of Molecular & Human Genetics, Baylor College of Medicine, Houston, Texas, 77030, USA
lichtarge@bcm.edu

Abstract— *Predicting protein function is a central problem in bioinformatics, and many approaches have been suggested that use partially or fully automated methods based on various combination of sequence, structure, and other information on proteins or genes. Such information establishes relationships between proteins that can be modeled most naturally as edges in graphs. A priori, however, it is often unclear which edges from which graph may contribute most to accurate predictions. For that reason, one established strategy is to integrate all available sources, or graphs, in the hope that the positive signals will add to each other. However, in the problem of functional prediction, noise, i.e. the presence of inaccurate or false edges, can still be large enough that integration alone has little effect on prediction accuracy. In order to reduce noise levels and to improve integration efficiency, we present here a method in graph-based learning, graph sharpening, which provides a theoretically firm yet intuitive and practical approach for disconnecting undesirable edges from protein similarity graphs. This approach has several attractive features: it is quick, scalable in the number of proteins, robust with respect to errors, and tolerant of very diverse types of protein similarity measures. We tested the classification accuracy in a test set of 599 proteins with remote sequence homology spread over 20 Gene Ontology (GO) functional classes. When compared to integration alone, graph sharpening plus integration of four vastly different molecular similarity measures improved the overall classification by nearly 30% (0.17 average increase in ROC score). Moreover, and partially through the increased sparsity of the graphs induced by sharpening, this gain in accuracy came at negligible computational cost: sharpening and integration took on average* $4.66(\pm 4.44)$ *CPU seconds.*

* Both authors contributed equally to this work.

1. INTRODUCTION

The prediction of biological function is a central challenge in modern biology since few genomic or proteomic data have detailed annotations [1]. A new class of computer aided solutions to this problem is currently emerging that aims to integrate diverse sources of relevant information, rather than relying only on single methods, which have specific limitations [2], [3], [4]. Function information may come in many forms, ranging from genomic and molecular to cellular and tissue contexts. The unprecedented availability of data from primary to tertiary protein structure in particular motivated many methods that use computational comparisons of specific molecular attributes to define functional relationships [5]. From this a clearer signal for function prediction may emerge by considering multiple relevant relationships.

A natural model of relationships between proteins is a network (or graph), where nodes depict genes or proteins and edges represent their possible interactions or correlations [6], [7], [8], [9] (see Fig.1). Among these there are networks

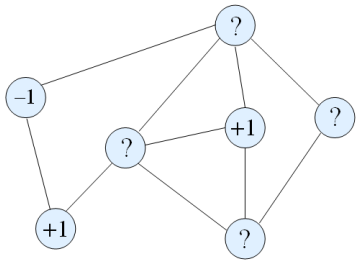


Fig. 1. A graph model of relationships between proteins. Nodes depict genes or proteins and edges represent their possible interactions or correlations, e.g., molecular similarities extracted from sequence or structure comparisons. An annotated protein is labelled either by '$+1$' or '-1', indicating either it belongs to a particular functional class or not. Graph-based function prediction seeks to classify the unannotated (unlabelled) proteins marked as '? '.

of weighted edges that value molecular similarity between protein pairs extracted from sequence or structure comparisons and classify or predict protein function [10], [11], [12], [13]. Recently, it was shown that graphs based on semi-supervised learning [14], [15] have better classification performance for sequence similarity networks over conventional local sequence comparison [16]. This suggests that graph based semi-supervised learning can capture global network information that is functionally relevant. Later, and in a more general approach, multiple protein networks were then combined into a single computationally efficient semi-supervised learning problem and this further improved functional classification over individual networks [17].

Here, we further extend these ideas to improve the annotation of protein function based on graphs. Starting from a diverse set of protein similarity measures, some that are standard and based on sequence and others that are more novel and based on structure, we apply a graph-based algorithm to integrate them. Compared to previous work [17], the novelty of this approach lies in taking into explicit account the directionality of edges in each graph through a method that we call "graph sharpening" [18]. With a test set of 15 out of 20 functional GO terms among 599 non-redundant protein structures from the PDBselect25 list [19], we show that sharpening and integration raise the overall classification performance by nearly 30%. In absolute values, an average increase of the Receiver Operating Characteristic (ROC) score

by 0.17 up to a level of 0.75.

This paper is organized as follows. We introduce graph-based semi-supervised learning in section 2. In section 3, the basic idea and some elements of the mathematical framework behind graph sharpening are explained [18]. In section 4, we present a specific graph-integration approach which uses linear combinations of graph Laplacian matrices [17]. Then, we demonstrate how these methods can be successfully applied to protein function prediction (section 5). We conclude with some future work remarks.

2. GRAPH-BASED LEARNING

In the graph-based Semi-Supervised Learning algorithm [14], a data point $\boldsymbol{x}_i \ \ (i = 1, \ldots, n)$ is represented as a node i in a graph, and the relationship between data points is represented by an edge where the connection strength from each node j to each other node i is encoded in element w_{ij} of a weight matrix W. A weight w_{ij} can take a binary value (0 or 1) in the simplest case. Often, a Gaussian function of Euclidean distance between points, with length scale σ, is used to specify connection strength:

$$w_{ij} = \begin{cases} \exp\Big(-\frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j)}{\sigma^2} \Big) & \text{if} \quad i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

The $i \sim j$ stands for node i and j having an edge between them which can be established either by k nearest neighbors or by Euclidean distance within a certain radius r, $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 < r$. [1] The labelled nodes have labels $\boldsymbol{y}_l \in \{-1, 1\}$, while the unlabeled nodes have zeros $\boldsymbol{y}_u = \mathbf{0}$. Our algorithm will output an n-dimensional real-valued vector $\boldsymbol{f} = [\boldsymbol{f}_l^\top \ \boldsymbol{f}_u^\top]^\top = (f_1, \cdots, f_l, f_{l+1}, \cdots, f_{n=l+u})^\top$ which can be thresholded to make label predictions on $f_{l+1}, \ldots, f_n$ after learning. It is assumed that (a) f_i should be close to the given label y_i in labelled nodes, and (b) overall, f_i should not be too different from the f_j of adjacent nodes ($i \sim j$). One can obtain $\boldsymbol{f}$ by minimizing the following quadratic functional [20], [21], [14]:

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

where $\boldsymbol{y} = (y_1, \ldots, y_l, 0, \ldots, 0)^\top$, and the matrix L, called the *graph Laplacian matrix* [22], is defined as $L = D - W$ where $D = \mathrm{diag}(d_i)$, $d_i = \sum_j w_{ij}$. The first term corresponds to the *loss function* in terms of condition (a), and the second term represents the *smoothness* of the predicted outputs in terms of condition (b). The parameter μ trades off loss versus smoothness. The solution of this problem is obtained as

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

where I is the identity matrix.

The values of $\boldsymbol{f}$ are obtained by solving *a large sparse linear system* $\boldsymbol{y} = (I + \mu L)\boldsymbol{f}$. This numerical problem has been intensively studied, and there exist efficient algorithms, of which computational time is nearly linear in the number of nonzero entries in the coefficient matrix [23]. Therefore, computation gets faster as the Laplacian matrix gets sparser.

[1] We represent scalars as lower case, vectors as boldface lower case, and martrices are uppercase. $\mathbf{0}$ (or $\mathbf{1}$) is a vector or matrix of variable-dependent size containing of all zeros (or ones).

3. GRAPH SHARPENING

Next we present a method which is directly applicable to the weight matrix W, and motivated by the following intuition: in an undirected graph as in Fig.1, all connections are reciprocated and so the matrix of edge weights W is symmetric as shown in Fig.2(a). However, when W describes relationships between labelled and unlabelled points, it is not necessarily desirable to regard all such relationships as symmetric. That is, we may differentiate the importance of information flow so far equally weighing all edges. *First*, some edges may convey more useful information in one direction (e.g. from labelled to unlabelled) than in the reverse direction. Propagating information in the reverse direction, from unlabelled to labelled, may be undesirable since it allows points about which information is uncertain to corrupt the source of information in the system. Since we are already using the language of "points" and "edges", we will say that this causes the point and its outgoing edges to be "blunted", reducing their effectiveness. There are many problem settings (for example, protein function prediction and other applications in the field of bioinformatics) in which (a) there is a high degree of certainty about the input space representation of each labelled point and its label, and (b) the number of labelled points is low. In this situation, it is indicated to avoid blunting, thus to preserve the effectiveness of information sources. *Second*, edges directly connecting oppositely labelled points may propagate unhelpful information in either direction. The smoothness condition in Eq.(1) plays the role of forcing both predicted scores to be similar to each other, and again, both important points, the very sources of information, are blunted. Further, those edges unnecessarily incur a conflict of the opposite flows in the area of a high certainty in the system. *Third*, propagation of information between unlabelled points is different—while some edges of the graph may be more helpful than others in solving the overall problem, a priori we do not know which these might be. Allowing the unlabelled points to harmonize with their neighbours (thus implementing smoothness condition) common to most such learning approaches) is a desirable process.

Fig.2 illustrates how this intuition is realized by graph sharpening. Fig.2(a) shows an original graph of seven nodes (points) and its (so far) symmetric weight matrix W. For simplicity, we set the value of '1' as a weight on every edge. The row index in W denotes the destination and the column index the source, so w_{ij} reads "the weight of the edge *from* j *to* i ($j \longrightarrow i$)." In Fig.2(b), the edges from unlabelled to labelled points w_{ij} (denoted as dotted arrows) are disconnected where i belongs to the set of labelled points $l := \{1, 2, 3\}$ and j to the set of unlabelled points $u := \{4, 5, 6, 7\}$. Fig.2(c) presents the case of connection between oppositely labelled points,

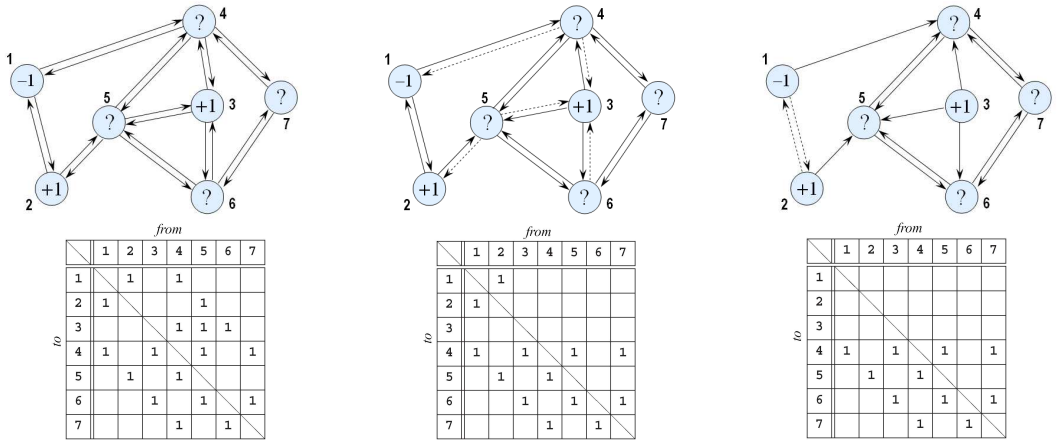


Fig. 2. Graph sharpening: (a) an original undirected (bi-directed) graph and its symmetric weight matrix W. Note that in the interpretation of W, the row index denotes the destination and the column index the source —so for example w_{ij} should be read as "the weight of the edge *from* j *to* i ($j \longrightarrow i$). (b) Edges from unlabelled to labelled points (denoted as dotted arrows) are disconnected. (c) Edges between oppositely labelled points are further removed. Sharpening leaves the edges between unlabelled points intact. In contrast to the original in Fig.1, the *sharpened* graph is no longer fully reciprocated and so the matrix of edge weights W becomes asymmetric.

w_{12} and w_{21}. Therefore, both edges are further removed from Fig.2(b). Finally, we obtain a *sharpened* graph in Fig.2(c). Note that the weight matrix becomes asymmetric, and the edges between unlabelled points remain intact. [2] A detailed mathematical foundation of *sharpening* is given in [18]. Let us give a schematic flow of the proof. We pose the general question: what if W is not considered fixed? Is it possible to change some or all of the w_{ij} such that our algorithm performs better? We begin by re-formulating the objective function Eq.(1) in terms of W, which is based on the formulation of [21]. Blockwise consideration of the weight matrix,

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}$$

allows us to state a condition which solutions W must satisfy if the objective function is to be optimized —there are many such solutions. Exploring every class of solutions is currently regarded as intractable and is left as an open problem. However, we can specify one simple *ad hoc* solution, concordant with the intuition already stated. By setting W_{ll} to a non-negative diagonal matrix (including null matrix) and W_{lu} to $\mathbf{0}$ such as

$$W_s = \begin{bmatrix} \text{diagonal matrix} & \mathbf{0} \\ W_{ul} & W_{uu} \end{bmatrix} \tag{3}$$

(see the final W in Fig.2(c)), we obtain the output prediction for unlabelled data points

$$\boldsymbol{f}_u \quad = \quad \mu(I + \mu(D_{uu} - W_{uu}))^{-1} W_{ul} \boldsymbol{y}_l. \tag{4}$$

In spreading activation network terms, Eq.(4) is equivalent to activity being propagated from labelled to unlabelled data *once* ($W_{ul}\boldsymbol{y}_l$) to set the initial condition for subsequent spreading activation among $u \leftrightarrow u$, analogous to Eq.(2) but now *excluding* the labelled points. This also has intuitive appeal. First, for labelled points, it assures $\boldsymbol{f}_l = \boldsymbol{y}_l$— *there is no loss of information on labelled data points*. By disconnecting unnecessary and unhelpful edges, we allow the labelled points and their outgoing edges to stay "sharp" in their influence on the rest of the graph. Second, for unlabelled points, it preserves an important principle of SSL, namely *exploitation of the manifold structure inferred from unlabelled data points*, by keeping the edges, $u \leftrightarrow u$ and $l \rightarrow u$, of W.

4. GRAPH INTEGRATION

Given a node set of proteins, there are several ways to represent edges. For example, edges can be defined from from amino acid sequences, or from structures, or from protein-protein interactions. Therefore, many possible graphs exist, and each graph can be partly independent from and partly complementary to others. It can be often difficult to decide which graph will perform best for function prediction for unlabelled proteins, and individual graphs (from single data sources) are often not sufficient for reliable prediction. One way to enhance reliability may be to integrate the given multiple graphs. Integrating multiple graphs stands for finding an optimum value of the linear combination coefficient for the individual graphs. In the semi-supervised learning framework, this translates to finding the combination coefficients $\boldsymbol{\alpha}$ for the individual Laplacians, as shown in Fig.3. Recently, [17] proposed to extend Eq.(1) of a single graph to multiple graphs

$$\begin{aligned} \min_{\boldsymbol{f},\gamma} \quad & (\boldsymbol{f} - \boldsymbol{y})^\top (\boldsymbol{f} - \boldsymbol{y}) + \mu\gamma, \\ \text{s.t.} \quad & \boldsymbol{f}^\top L_k \boldsymbol{f} \leq \gamma, \quad k = 1, \ldots, K, \end{aligned} \tag{5}$$

where K is the number of graphs and an L_k is the corresponding *graph Laplacian* to graph G_k. This formulation is related

[2] Instead of disconnecting or removing an edge from W, we can penalize an undesirable edge weights by introducing a penalty term, $w_{ij}^{\text{NEW}} = w_{ij} - \delta_{ij}$ where $0 \leq \delta_{ij} \leq w_{ij}$.

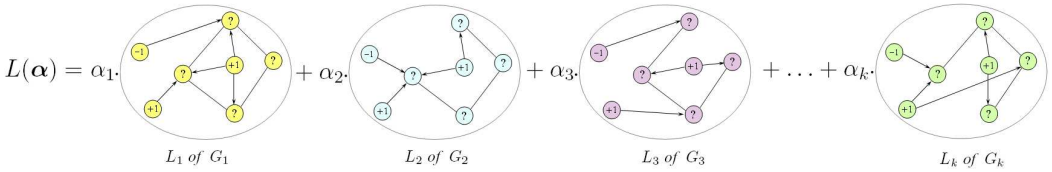


Fig. 3. Graph integration: Every single graph G_k can solely be used for label prediction. However since different graphs contain partly independent and partly complementary pieces of information, integrating them into one may increase the reliability of predictions. In semi-supervised learning framework, graph integration stands for finding an optimum value of the linear combination coefficient $\boldsymbol{\alpha}$ for individual graph Laplacians L_k.

to the illustration in Fig.3 through its dual problem

$$\min_{\boldsymbol{\alpha}} \quad \boldsymbol{y}^\top (I + \sum_{k=1}^{K} \alpha_k L_k)^{-1} \boldsymbol{y}, \quad \text{s.t.} \quad \sum_k \alpha_k \leq \mu \quad (6)$$

and its output prediction

$$\boldsymbol{f} \quad = \quad (I + \sum_{k=1}^{K} \alpha_k L_k)^{-1} \boldsymbol{y}. \quad (7)$$

5. Function Prediction Experiments

5.1. Data

In automated protein functional class classification, accurate detection of functional relationships beyond close sequence homology is desirable. Thus, we have restricted our selection to protein chains from the PDBselect25 list (version October 2004), where any pair shares no more than 25% sequence identity. Functional information was assigned in terms of the Gene Ontology (GO) in the category 'Molecular function' and mapped to PDB structures through the gene ontology annotation database (GOA PDB 24.0). We chose the 20 highly populated GO categories as in [13]. Of all PDBselect25 chains, 599 proteins shares at least one of these functional GO terms (Table I). Therefore, we took them as our experimental protein set.

To generate weighted edges between nodes (or proteins), we used four different computational measures of molecular similarity. Each measure assigned to every protein pair (i, j) a positive weight ($0 \leq w_{ij} \leq 1$) meaning the degree of molecular similarity between chain i and j. The four different similarity measures are Basic Local Alignment and Search Tool (BLAST, [24]), the standard approach to alignment of primary sequence which resulted in a degree of sequence identity; length-corrected Contact Metric (CM, [25]), a metric based similarity score by considering vector representations of contacts from the entire tertiary structure; Fast Alignment and Search Tool (FAST, [26]), an accurate and computationally efficient 3D geometrical alignment algorithm calculating positive similarity scores; Evolutionary Trace Annotation (ETA, [27]), a method that measures similarity based on local similarity of protein substructure, specifically 3D-templates that are small structural motifs of evolutionarily important residues. Our choice represented all currently and commonly used approaches to protein similarity measurement [5]: Sequence alignment (BLAST), global 3D alignment (FAST), local 3D alignment of small motifs (ETA), and alignment-independent vector based approaches (CM). An edge-weight from BLAST/CM/FAST is a continuous value while that of ETA is a binary value from a support vector classifier, i.e., the edge-weight between two proteins is set to 1 ($w_{ij} = 1$) if they were predicted functionally related; no edge ($w_{ij} = 0$) otherwise.

TABLE I

20 functional Gene Ontology terms from the category 'Molecular function'. GO terms with less than ten proteins were excluded (asterisk mark).

GO term	Molecular function	Number of proteins
0003677	DNA binding	137
0005515	Protein binding	49
0016491	Oxidoreductase activity	39
0005524	ATP binding	95
0003723	RNA binding	50
0006118	Electron transport	87
0003676	Nucleic acid binding	63
0003824	Catalytic activity	57
0005198	Structural molecule activity	22
0005509	Calcium ion binding	32
0000287	Magnesium ion binding	3*
0008270	Zinc ion binding	56
0005489	Electron transporter activity	37
0004872	Receptor activity	7*
0016798	Hydrolase activity	0*
0004519	Endonuclease activity	9*
0004871	Signal transducer activity	18
0004672	Protein kinase activity	29
0004518	Nuclease activity	5*
0004867	Serine-type endopeptidase inhibitor activity	15

5.2. Results

Since one protein can belong to several GO categories, we posed a binary-class classification problem for each GO term in TableI. The GO categories having only a few labelled proteins (less than 10) were excluded; therefore, our experiment became 15 binary-class classification problems, determining membership or non-membership of unannotated (unlabelled) proteins for the respective GO terms. For a GO term, we calculated the five-fold cross-validation (5 CV) ROC (receiver operating characteristic) score as a performance measurement. The ROC score indicates the area under the ROC curve which plots true positive rate (sensitivity) as a function of false positive rate (1-specificity) for all possible thresholds, see Fig.6. A ROC score of 0.5 corresponds to

random guessing, while an ROC score of 1.0 implies the algorithm successfully outputs a higher predicted value for any positive example than that for any negatives example (perfect classification).

We compared the proposed method–*integration on sharpened graphs*, with the four individual graphs obtained from BLAST, FAST, CM, and ETA, respectively, and also with the previous method of [17]–*integration on original graphs*. In every comparison, we examined the performance change in terms of *original vs. sharpened* and *individual vs. integrated*. This setting enabled us to see the effect of the proposed method from two separate viewpoints, *sharpening* and *integration*. The value of smoothing parameter μ–from Eq.(2) of *original*, Eq.(4) of *sharpening*, and Eq.(7) of *integration*– was obtained by 5CV searching over $\mu \in \{0.1, 1, 5, 10, 50, 100\}$. For each of the available settings per GO category, we compared the results at their best parameters. A table with the best parameters is published at the Internet address (`http://mammoth.bcm.tmc.edu/biocomp2007/`), were the software and supplementary material can be retrieved. Remarkably, *sharpening* does not include any additional parameters, nor *integration*–the linear combination coefficient $\boldsymbol{\alpha}$ in Eq.(7) is automatically set as a solution of the optimization.

1) Graph Sharpening on Individual Graphs: Fig.4(a) shows the distribution of 300 (= 15 GO terms $\times$ 5 CVs $\times$ 4 graphs) ROC score pairs from the original (unsharpened) and sharpened graphs. Most dots are scattered above the diagonal, thus indicating better performance of the sharpened graphs against originals. A *sign test* statistically rejected the null hypothesis, "the distribution would be evenly scattered below and above the diagonal," with significant confidence ($p < 9 \times 10^{-16}$). Table II confirms this result: on average, sharpening increases the original ROC scores by 0.03 (ETA) upto 0.12 (FAST).

2) Integrated Graph vs. Individual Graphs: The next test was whether individual prediction in sharpened graphs could be further improved through their *integration* as stated in section 4. Fig. 4(b) shows that this was the case: 222 of the 300 ROC score pairs are above the diagonal ($p < 9 \times 10^{-16}$). In Fig. 5 the average over 5CV ROC scores are assigned to their function classes. The figure assures that *integration* on sharpened graphs achieves the highest ROC scores when compared with the *individuals* (sharpened), except for GO 0003677 (DNA binding) and GO 0008270 (Zinc ion binding). Integration effect on sharpened individual graphs show more significance and less volatility than the effect of integration on the original (unsharpened) networks (Table II): 0.09 ($\pm$ 0.04) vs. 0.02 ($\pm$ 0.08) in average ROC score increase.

3) Proposed Method vs. Previous Method: In the third test, we verified how integration on sharpened graphs upgrades the performance of the previous method [17], *integration alone*, so to speak. The results show that sharpening significantly contributes to ROC score improvement, see Fig.4(c); 70 out of 75 ROC pairs are in the upper diagonal ($p < 3 \times 10^{-18}$). Integration alone can be insignificant and even worse (see, value 0.02 ± 0.08 in Table II), particularly if given graphs

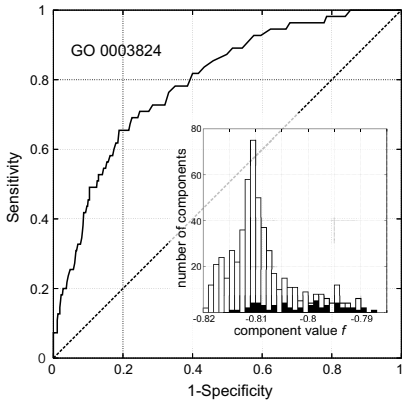


Fig. 6. ROC curve for GO 0003824 ('Catalytic activity') shows that graph sharpening and integration reliably discriminates enzymes (catalytic proteins) from non-enzymes. Insert: the distribution of predicted values (scores), $\boldsymbol{f}$, for the unlabelled proteins– an enrichment of enzymes toward larger values is evident.

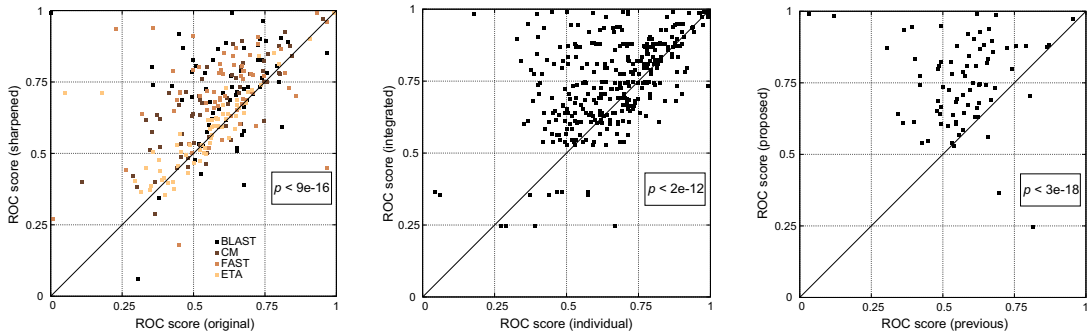
are noisy, which can often be the case in protein similarity networks. But with sharpening, integration becomes more effective, which is reflected by a ROC score increase of 0.17 ± 0.08, see Table II. Hence, given the performance of single unsharpened (original) graphs, one can raise the ROC score as much as 0.17 ± 0.08 by applying integration with sharpening.

4) Enrichment: Fig.6 shows a typical ROC curve of the proposed method for GO 0003824 'Catalytic activity'. The curve shows that sharpening and integration can reliably discriminate enzymes (catalytic proteins) from non-enzymes among proteins with less than 25% sequence similarity. The inner figure shows the distribution of predicted values (scores), $\boldsymbol{f}$, for the unannotated proteins. An enrichment of enzymes toward larger scores is evident.

5) Computation Time: Sharpening does not require computation. The computation time of an original graph, the time for solving the sparse linear system in Eq.(2), was nearly trivial (less than 0.001 cpu second with MATLAB in a standard 1500 Mhz PC with 1 GByte of memory). It became faster for a *sharpened* graph since the Laplacian matrix gets sparser. *Integration* in Eq.(7) took avg. $24.27(\pm 10.86)$ CPU seconds for original, while it was avg. $4.66(\pm 4.44)$ for sharpened weights, and graph sparsity through sharpening benefitted the computation time for integration.

6. Conclusion and Discussion

We applied to the protein function prediction problem two recent developments in machine learning, sharpening [18] and integration [17]. Graph sharpening is a formal yet intuitive approach for disconnecting undesirable edges in a graph. The result yields sparser graphs that contain less noise and, in turn, reduce computational expense and improve prediction without need for additional parameters. Using an established optimization framework, graph integration can then be applied with greater efficiency to pool information from multiple sources. Both strategies, graph sharpening and graph integration, lend themselves well to the protein function prediction



(a) sharpened vs. original (b) integrated vs. individuals (c) proposed(2007) vs. previous(2005)

Fig. 4. Positive effect of sharpening in pairwise ROC score comparison for two competing methods in three different cases. More dots in upper diagonal half indicate that the method in vertical axis outperforms the other assigned in the horizontal axis; *p*-values (from a sign test) represent the statistical deviation from a random distribution of dots in the upper and lower half. (a) Sharpened vs. original: for 221 out of 300 pairs in total, *sharpening* gave a higher ROC score; (b) integrated (sharpened) vs. individuals (sharpened): for 212 out of 300 experiments, *integration* gave a higher ROC score; (c) integration with sharpening (2007) vs. without sharpening (2005): for 70 out of 75 experiments, *integration with sharpening* had a higher ROC score.

TABLE II

ROC SCORES COMPARISON FOR ALL THE COMBINATION OF *original vs. sharpened* AND *individual vs. integrated*.

Graphs		GO terms 0003677	0005515	0016491	0005524	0003723	0006118	0003676	0003824	0005198	0005509	0008270	0005489	0004871	0004672	0004867	ROC increase by sharpening
BLAST	original	0.71	0.70	0.41	0.56	0.60	0.68	0.70	0.60	0.65	0.59	0.74	0.59	0.68	0.60	0.83	**0.09**
	sharpened	0.71	0.76	0.72	0.56	0.59	0.74	0.72	0.80	0.71	0.75	0.76	0.84	0.74	0.55	0.92	± 0.10
CM	original	0.64	0.46	0.75	0.49	0.41	0.61	0.56	0.76	0.67	0.50	0.59	0.36	0.39	0.51	0.75	**0.10**
	sharpened	0.64	0.73	0.76	0.52	0.56	0.63	0.68	0.75	0.74	0.56	0.65	0.59	0.56	0.66	0.93	± 0.08
FAST	original	0.65	0.53	0.64	0.55	0.43	0.67	0.55	0.70	0.68	0.71	0.60	0.62	0.37	0.61	0.76	**0.12**
	sharpened	0.65	0.78	0.73	0.63	0.62	0.72	0.69	0.81	0.64	0.82	0.67	0.82	0.84	0.73	0.88	± 0.13
ETA	original	0.51	0.43	0.53	0.57	0.46	0.57	0.45	0.45	0.53	0.56	0.50	0.59	0.75	0.64	0.53	**0.03**
	sharpened	0.50	0.49	0.55	0.58	0.46	0.60	0.48	0.45	0.57	0.58	0.52	0.59	0.74	0.67	0.56	± 0.07
Integrated	original	0.68	0.49	0.75	0.51	0.41	0.64	0.56	0.76	0.73	0.63	0.64	0.49	0.39	0.58	0.79	**0.16**
	sharpened	0.66	0.78	0.80	0.59	0.61	0.73	0.71	0.83	0.77	0.83	0.71	0.81	0.84	0.73	0.96	± 0.12
ROC increase by integration																	
original	avg.	**0.05**	**-0.04**	**0.17**	**-0.03**	**-0.06**	**0.01**	**0.00**	**0.13**	**0.10**	**0.04**	**0.03**	**-0.05**	**-0.16**	**-0.01**	**0.07**	**0.02**
	std.	± 0.09	± 0.12	± 0.14	± 0.04	± 0.09	± 0.05	± 0.10	± 0.13	± 0.07	± 0.09	± 0.10	± 0.12	± 0.19	± 0.06	± 0.13	± 0.08
sharpened	avg.	**0.03**	**0.09**	**0.11**	**0.01**	**0.05**	**0.06**	**0.06**	**0.13**	**0.11**	**0.15**	**0.05**	**0.10**	**0.12**	**0.08**	**0.13**	**0.09**
	std.	± 0.09	± 0.14	± 0.09	± 0.05	± 0.07	± 0.07	± 0.11	± 0.17	± 0.08	± 0.13	± 0.10	± 0.14	± 0.12	± 0.08	± 0.18	± 0.04
ROC increase by sharpening and integration																	
	avg.	**0.03**	**0.25**	**0.22**	**0.05**	**0.14**	**0.10**	**0.14**	**0.20**	**0.14**	**0.24**	**0.10**	**0.27**	**0.29**	**0.13**	**0.24**	**0.17**
	std.	± 0.08	± 0.10	± 0.12	± 0.03	± 0.07	± 0.05	± 0.09	± 0.11	± 0.06	± 0.08	± 0.09	± 0.10	± 0.17	± 0.05	± 0.11	± 0.08

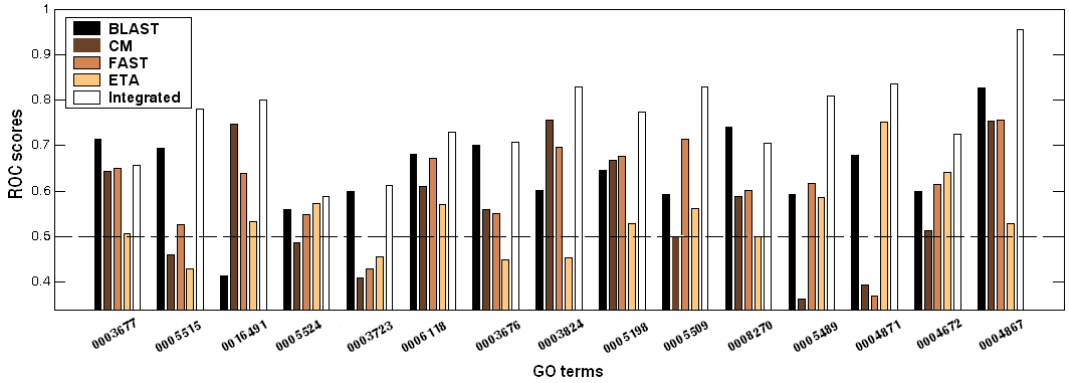


Fig. 5. ROC score comparison (avg. of five cross-validation) between individual and the integrated graphs: Bars within a group correspond to BLAST, CM, FAST, ETA, and the integrated graph in due order. For 13 out of 15 categories, *integration with sharpening* significantly surpasses individual performance.

problem. Graph sharpening offers a general framework to address the noise that is pervasive among functionally relevant measures of protein similarity, while graph integration allows overall predictions to take into account a wide variety of complementary types of information on protein similarity graphs, each one representing a different aspect or type of functional information. While either strategy can be applied alone, with sharpening having a greater effect than integration, the best result was reached when sharpening and integration are used together and thus yield a synergestic effect on function prediction performance-at lesser computational cost. This improvement is noteworthy for its size (0.17, or nearly 30% average increase in the area under the ROC curve) and in view of the diversity of similarity scores that were integrated: BLAST is used over sequences, CM and FAST are applied over whole structures, and ETA is applied to a local structural motif (and has only been optimized for enzymes, rather than for the GO annotations used here).

This work motivates possible future studies. The method is general and its full application for function prediction will still require a continued refinement of individual methods, as well as broadening the number of similarity measures whose graphs are sharpened and then integrated together. Towards this goal, we plan an Internet accessible software tool for sharpening and integration to allow large scale function prediction using these methods.

ACKNOWLEDGEMENT

HS was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) and by the grant for Post Brain Korea 21. Financial support was provided by a training fellowship from the Gulf Coast Consortia through the W. M. Keck Center for Computational and Structural Biology (AML). This work was also supported by grants from the National Science Foundation (DBI-0547695), National Institutes of Health (R01 GM066099), and March of Dimes (1-FY06-371) to OL.

REFERENCES

[1] I. Friedberg, "Automated protein function prediction - the genomic challenge," *Briefings in Bioinformatics*, vol. 7, no. 3, pp. 225–242, 2006.

[2] R. A. Laskowski, J. D. Watson, and J. M. Thornton, "Profunc: a server for predicting protein function from 3d structure," *Nucleic Acids Research*, vol. 33, pp. W89–W93, 2005.

[3] D. Pal and D. Eisenberg, "Inference of protein function from protein structure," *Structure*, vol. 13, no. 1, pp. 121–130, 2005.

[4] I. Friedberg, T. Harder, and A. Godzik, "Jafa: a protein function annotation meta-server," *Nucleic Acids Res*, vol. 34, no. Web Server issue, pp. W379–81, 2006.

[5] J. D. Watson, R. A. Laskowski, and J. M. Thornton, "Predicting protein function from sequence and structural data," *Current Opinion in Structural Biology*, vol. 15, no. 3, pp. 275–284, 2005.

[6] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, *et al.*, "A comprehensive analysis of protein-protein interactions in *Saccharomyces cerevisiae*," *Nature*, vol. 403, no. 6770, pp. 623–627, 2000.

[7] 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*, vol. 417, pp. 399–403, 2002.

[8] I. Lee, S. Date, A. Adai, and E. Marcotte, "A probabilistic functional network of yeast genes," *Science*, vol. 306 (5701), pp. 1555–1558, 2004.

[9] M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. .Hattori, "The KEGG resources for deciphering genome," *Nucleic Acids Res.*, vol. 32, pp. D277–D280, 2004.

[10] G. Yona, N. Linial, and M. Linial, "Protomap: Automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space," *Proteins: Structure, Function, and Genetics*, vol. 37, pp. 360–678, 1999.

[11] A. Adai, S. Date, S. Wieland, and E. Marcotte, "Connecting the protein structure universe by using sparse recurring fragments," *Journal of Molecular Biology*, vol. 340, no. 1, pp. 179–190, 2004.

[12] I. Friedberg and A. Godzik, "Connecting the protein structure universe by using sparse recurring fragments," *Structure*, vol. 13, no. 8, pp. 1213–1224, 2005.

[13] J. T. Hou, S. R. Jun, C. Zhang, and S. H. Kim, "Global mapping of the protein structure space and application in structure-based inference of protein function," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 10, pp. 3651–3656, 2005.

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

[15] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[16] W. Noble, R. Kuang, C. Leslie, and J. Weston, "Idetifying remote protein homologs by network propagation," *FEBS Journal*, vol. 272, no. 20, pp. 5099–5100, 2005.

[17] K. Tsuda, H. Shin, and B. Schölkopf, "Fast protein classification with multiple networks," *Bioinformatics*, pp. 59–65, 2005, also appeared as *Prediction of Protein Function from Networks*, authored by H. Shin and K. Tsuda, Chapter 20 in book *Semi-Supervised Learning* edited by O. Chapelle, B. Schölkopf, and A. Zien, MIT press, 2006.

[18] H. Shin, N. Hill, and G. Raetsch, "Graph-based semi-supervised learning with sharper edges," *Proc. of the 17th European Conference on Machine Learning and 10th European on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2006.

[19] U. Hobohm and C. Sander, "Enlarged representative set of protein structures," *Protein Science*, vol. 3, pp. 522–524, 1994.

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

[21] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and regression on large graphs," *Lecture Notes in Computer Science (In COLT)*, pp. 624–638, 2004.

[22] F. Chung, *Spectral Graph Theory*, ser. Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1997, no. 92.

[23] D. Spielman and S. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proc. of the 26th annual ACM symposium on Theory of computing*. ACM Press, 2004, pp. 81–90.

[24] S. Altschul, W. Gish, W. Miller, E. Myers, and D. lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, pp. 403–410, 1990.

[25] A. M. Lisewski and O. Lichtarge, "Rapid detection of similarity in protein structure and function through contact metric distances," *Nucleic Acids Research*, vol. 34, no. 22, p. e152, 2006.

[26] J. Zhu and Z. Weng, "Fast: a novel protein structure alignment algorithm," *Proteins*, vol. 14, pp. 417–423, 2005.

[27] D. M. Kristensen, B. Y. Chen, V. Y. Fofanov, R. M. Ward, A. M. Lisewski, M. Kimmel, L. E. Kavraki, and O. Lichtarge, "Recurrent use of evolutionary importance for functional annotation of proteins based on local structural similarity," *Protein Science*, vol. 15, no. 6, pp. 1530–1536, 2006.

[28] O. Lichtarge, H. Bourne, and F. Cohen, "An evolutionary trace method defines binding surfaces common to protein families," *Journal of Molecular Biology*, vol. 257, no. 2, pp. 342–358, 1995.

[29] M. Gribskov and N. Robinson, "Use of receiver operating characteristic (roc) analysis to evaluate sequence matching," *Computers and Chemistry*, vol. 20, no. 1, pp. 25–33, 1996.