Improvements in Protein Function Prediction Using Confidence in Protein Interactions

Kathryn Doroschak
University of Minnesota
kdoro@cs.umn.edu

Lenore Cowen
Tufts University
lenore.cowen@tufts.edu

Abstract

Characterizing protein function is a crucial part of understanding biological systems. Here we improve protein function prediction by accounting for data quality issues inherent in protein-protein interaction (PPI) databases.

To accomplish this, we incorporate confidence information into the function prediction pipeline. The model pipeline uses weighted majority voting on the protein-protein interaction network, with weights defined by shortest paths distance, confidence, diffusion state distance (DSD), or DSD with confidence. The result is that incorporating confidence weights in general significantly helps improve protein function prediction. Confidence with DSD performs especially well, improving by 11.9 pp over Majority Vote with ordinary shortest paths distance and no confidence weights.

Through this study, we have determined that incorporating confidence as weights improves protein function prediction, yielding greater accuracy than previously possible.

1. Introduction

A major part of understanding biological systems requires understanding and determining protein function. For this reason, our primary goal is to further improve protein function prediction. Most computational methods for function prediction have two main constraints: first, indistinct functional neighborhoods from using shortest paths in small world protein-protein interaction (PPI) networks, and second, most methods ignore data quality issues inherent in PPI databases.

Cao et al. [5] attempted to address the first problem by combining majority voting and diffusion state distance (DSD). DSD is a new metric that leverages graph diffusion to better capture inter-network distances.

Here our focus is on the second limitation, addressing data quality. We do so by introducing confidence to function prediction, assigning a score to each edge in the PPI network and integrating these scores into the function prediction pipeline in a variety of ways.

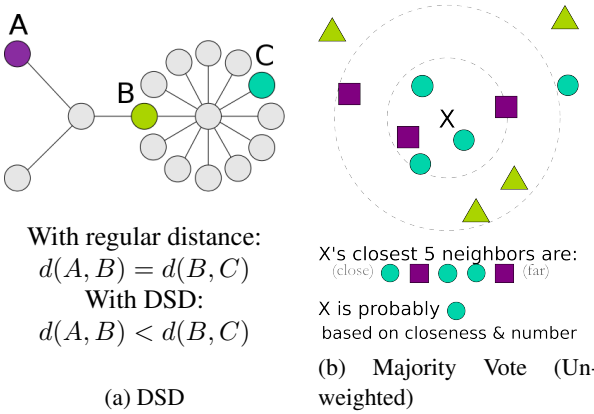
2. Data and Problem

2.1 Understanding the PPI Network

The heart of our data is the PPI network itself. At minimum, such a network consists of a list of interactions, each specified by two proteins in the network. We chose to use a PPI network for *S. Cerevisiae* from the BioGRID database v3.2.101 [5], as this is one of the most complete and well connected PPI networks in existence. This network originally consisted of 6,379 nodes and 324,743 interactions, but we filtered it down to 5,792 nodes 216,842 undirected interactions by removing non-ORF proteins and duplicate interactions and by using only the largest connected component of the network. Additionally, each interaction has experimental metadata with it, as provided by the BioGRID database. It is this network that we used to develop and test our protein function prediction improvements.

2.2 Assigning Functional Annotations

The true problem stems from the combination of the PPI network and the proteins' functional category labels. Our overarching goal is to use the properties of the PPI network to predict the functions of proteins contained within the network. To achieve the latter, we applied the MIPS functional catalog (FunCat) v2.1 [3] to the PPI as the source for our labels. MIPS FunCat provides annotations in a leveled, hierarchical fashion, where the deeper levels are more numerous and specific than those at the higher levels. We provide results for MIPS levels one, two and three, predicting functions through 2-fold cross validation. In this setup, the set of labeled proteins is randomly split in half and the labels on the training half are used to predict the labels on



(a) DSD

(b) Majority Vote (Unweighted)

Figure 1: Depictions of the two steps in the function prediction pipeline - DSD and Majority Vote.

the testing half. Then the two halves are swapped and we repeat this process.

2.3 Measuring Performance

Performance is measured based on two characteristics, accuracy and F1 score. Both characteristics were used by Cao et al. [1].

Accuracy: If the highest scoring prediction matches any function actually assigned to the protein, we count it as correct. Accuracy reflects the percentage of correctly labeled proteins.

F1 score: Each label in the list of predictions is considered correct if it matches any of the protein's actual functions. Precision is the percentage of correct labels relative to the number of predictions. Recall is the percentage of correct labels relative to the number of actual functions.

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

3. Prior Work: Function Prediction Using DSD and Majority Vote

In general, computational protein function prediction is carried out by taking advantage of the properties of a PPI network. Typically, this means calculating shortest paths distance between pairs in the network, using the result within the function prediction algorithm as a basis of comparison between proteins. However, with the recent development of DSD, we are able to better capture the distance between proteins and therefore improve protein function prediction. The DSD paper utilizes the same data sources (although we use newer versions of the data sets) and same pipeline as we use here [1]. We repeat the results, but also vary the usage of DSD and Majority Vote (described below) to highlight the effects of confidence in protein function prediction.

3.1 How DSD Works

Many PPIs contain hubs (high degree nodes), which create a small world property in the network. This property hurts algorithms that rely on shortest paths, e.g. Majority Vote, because most paths are similarly short. To offset this, DSD uses graph diffusion to distinguish paths via hubs from paths via lower degree nodes [1]. This has the effect of creating distances between proteins that are better spread out.

Overview of the DSD metric:

Excerpt from Cao et al. [1]:

"Consider the undirected graph *G(V,E)* on the vertex set $V = v_1, v_2, v_3, ..., v_n$ and $|V| = n$. $He^{\{k\}}(A,B)$ is defined as the expected number of times that a random walk starting at node A and proceeding for k steps, will visit node B. In what follows, assume k is fixed, and when there is no ambiguity in the value of k, we will denote $He^{\{k\}}(A,B)$ as $He(A,B)$. We further define a *n – dimensional* vector $He(v_i), \forall v_i \in V$, where

$$He(v_i) = (He(v_i, v_1), He(v_i, v_2), ..., He(v_i, v_n)).$$

"Then, the Diffusion State Distance (DSD) between two vertices *u* and *v*, $\forall u, v \in V$ is defined as:

$$DSD(u,v) = ||He(u) - He(v)||_1$$

where $||He(u) - He(v)||_1$ denote the L_1 norm of the He vectors of u and v."

The original DSD paper goes on to prove that DSD is indeed a metric, but this exercise has been omitted here for brevity.

It is important to note that when we talk about adding confidence as weights within DSD, this is not the same as calculating "weighted DSD." What we are doing is using the confidence values assigned to the edges to weight the He random walk inside DSD. This is in contrast to weighted DSD, in which "all new neighbors get a vote proportional to the reciprocal of their DSD distance" [5].

3.2 How Majority Vote Works

Majority Vote [4] can be thought of as "guilt by association" – protein functions are predicted based on the closest neighbors' functions. To do this, we allocate votes to each neighbor, either unweighted (equal vote for each neighbor) or weighted (based on neighbor weights, which in this case are assigned by confidence). The node is assigned the function with the most votes, as depicted in Fig. 1b.

Overview of Majority Vote:

1. Gather the t nearest neighbors of node u.
2. Allocate votes for each neighbor.
   - Unweighted (equal vote for each neighbor)
   - Weighted (vote based on neighbor weights, as specified below)
3. Assign u the function with the most votes when each neighbor votes for its own function.

4. Generating Confidence Scores

Confidence values are derived from the volume and type of experiments conducted in support of each edge. Multiple publications for an edge serve as verification for that edge. Additionally, high-throughput experiments tend to be less reliable due to their tendency to produce false positives. The cutoff for high and low throughput is 100 interactions [2].

#/Experiments	Low*	High*
0	0	0
1	80	25
2	90	50
3	95	75
4+	95	85

Table 1: Confidence score assignments, where 100 is very confident. The threshold between high and low throughput experiments is 100 interactions.
*Low and high are short for low throughput experiments and high throughput experiments.

We experimented with various other confidence methods, including literature-based without distinguishing between high and low throughput, Gene Ontology (GO) semantic similarity scores, and MINT scores. These methods did not fare quite as well, and have been omitted for brevity.

5. Applying Confidence Scores

There are two steps in the function prediction pipeline and accordingly, there are two ways to add confidence:

1. Majority voting with confidence as weights
2. DSD with confidence as probability in random walks

For Majority Vote with confidence as weights, we allocate votes by the edge's associated confidence. The more confident we are that two nodes are connected, the more votes the pair gets.

For DSD with confidence, we weight the random walk within DSD so that higher confidence edges are taken more frequently. We normalize the confidence weights with respect to all edges extending from a given node.

6. Results and Conclusions

Adding confidence weights to majority voting helps significantly, but DSD still performs even better, both with and without confidence weights as shown in Table 2. DSD with confidence weights performs best by far, likely because it addresses the issues of both shortest path distribution and data quality.

Confidence weights significantly improved both DSD and ordinary-distance majority voting by accounting for data unreliability. Accuracy improved 11.9 pp for MIPS Level 1 from original majority voting to majority voting using DSD with confidence weights. The F1 score improved 6.6 pp using the same comparison. Performance even increases relative to the already well-performing original DSD. Accuracy improved 3.5 pp for MIPS Level 1 from majority vote using original DSD to majority vote using DSD with confidence weights. The F1 score improved 1.8 pp accordingly.

In the future, these confidence techniques can be easily integrated with other function prediction methods; DSD with confidence weights can be used with any shortest-path function prediction method simply by replacing ordinary distance. Additionally, some preliminary work has been done with combining confidence information for protein networks with genetic interactions corresponding to specific proteins. We hope to continue work with this in the future, and improve protein function prediction even further.

Acknowledgments

The author would like to thank her mentor, Professor Lenore Cowen of Tufts University for her support and direction. She would also like to thank fellow undergraduate Thomas Schaffner at Tufts University for dedicating his summer to this project, and Professor Benjamin Hescott of Tufts University for his additional support.

This work is supported in part by the Distributed Research Experiences for Undergraduates (DREU) program, a joint project of the CRA Committee on the Status of Women in Computing Research (CRA-W) and the Coalition to Diversify Computing (CDC).

The author would also like to thank Professor Maria Gini of the University of Minnesota for her encouragement in applying for DREU.

	MIPS 1		MIPS 2		MIPS 3	
	Accuracy	F1 score	Accuracy	F1	Accuracy	F1
Majority Vote (MV)	58.0	45.0	45.3	32.7	40.7	30.4
MV, confidence as weights	65.6	49.2	52.8	38.0	48.4	34.6
MV, weighted original DSD	66.4	49.8	54.7	38.9	50.3	36.1
MV, weighted DSD with conf	69.9	51.6	58.1	41.6	54.5	39.3

Table 2: Summary of MV performance improvements using various confidence techniques, 2-fold cross-validation, and 10 voting neighbors.

References

[1] M. Cao, H. Zhang, N. Daniels, J. Park, M. Crovella, L. Cowen, and B. Hescott, "Going the distance for protein function prediction," 2013, In Review.

[2] Y. Chen, S. Rajagopala, T. Stellberger, and P. Uetz, "Exhaustive benchmarking of the yeast two-hybrid system," *Nature Methods*, vol. 7, 2010, pp. 667–668.

[3] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, and et al, "The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes," *Nucleic Acids Research*, vol. 32, 2004, pp. 5539–5555.

[4] B. Schwikowski, P. Uetz, and S. Fields, "A network of protein-protein interactions in yeast," *Nature Biotechnology*, vol. 18, 2000, pp. 1257–1261.

[5] C. Stark, B. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and et al, "Biogrid: a general repository for interaction datasets," *Nucleic Acids Research*, vol. 34, 2006, pp. D535–D539.