

Introduction

The clear understanding of biological processes is the key to solving the mystery of life. To solve this mystery we need to first understand the complex biological functions in higher organisms' like humans. Recent studies have shown that interaction between protein sets in higher organisms' like humans drives our biological functions. Thus, if we can identify pairs of proteins which are likely to interact and separate them from those which don't, then we have a mechanism to construct a protein interaction network.

The GO project has developed three structured controlled vocabularies (ontologies) that describe gene products in terms of their associated "biological processes, cellular components and molecular functions." This vocabulary enables us to form uniform queries across the database. Each protein is associated with a set of terms which are a part of several hierarchies. All the algorithms use "is_a" and "part_of" relationship tags to construct GO annotation hierarchy. "The hierarchy is a single rooted, gene ontology, structure such that each node can have multiple parents and children." Function of a protein is defined by its structure. SO, structure comparison is used to find similarity between proteins. The study of protein-protein interactions is essential to define the molecular networks that contribute to maintain homeostasis of an organism's body functions. Since these terms have been arranged in a hierarchical structure and try to maintain the property of moving from less specialized to more specialized case(not always true), it can be exploited to find proximity between them and thus predict the chance of them interacting directly or participating in a pathway. Proteins located in same cellular components or having close molecular function or close biological process have higher chance of participating in a common network though not totally true. This project is aimed at exploiting this very idea.

Goal

Classify a pair of protein as positively interactive or negatively interactive.

Data Sets

Annotation definition: This flat file defines GO annotations with a set of tags. There are 20403 terms for all three categories, biological process, molecular functions, and cellular component. We only process biological process for C. elegans annotations and there are 10693 of them.

Gene hierarchy definitions: This gene hierarchy file defines each gene with a number of GO annotations. Each gene only has one protein, so we interchange the terms "protein" and "gene". There are 9820 genes defined in this file.

Positive training dataset: This dataset contains pairs of proteins that we believe are of a higher likelihood to be interacting than not interacting. This file was downloaded

<http://tenaya.caltech.edu:8000/predict/> . The data set contains pairs of proteins which are likely to interact.

Negative training dataset: This dataset contains pairs of proteins that we believe are of a higher likelihood of being non-interacting than interacting. This file was downloaded <http://tenaya.caltech.edu:8000/predict/> . The data set contains pairs of proteins which are unlikely to interact.

Positive testing dataset: This dataset contains pairs of proteins that we believe are of a higher likelihood to be interacting than not interacting. This file was downloaded <http://tenaya.caltech.edu:8000/predict/>. The data set contains pairs of proteins which are likely to interact.

Negative testing dataset: This dataset contains pairs of proteins that we believe are of a higher likelihood of being non-interacting than interacting. This file was downloaded <http://tenaya.caltech.edu:8000/predict/> . The data set contains pairs of proteins which are unlikely to interact.

Methods

We have constructed 3 single rooted DAGs, each for “Biological Process”, “Molecular Function”, and “Cellular Component”. To build the DAG we have used “is_a” and “part_of” relationship tags in the Gene Ontology such that both are parents of the current gene. This results in three single-rooted DAGs where a node can have multiple parents.

A DAG would look something similar to Figure 1.

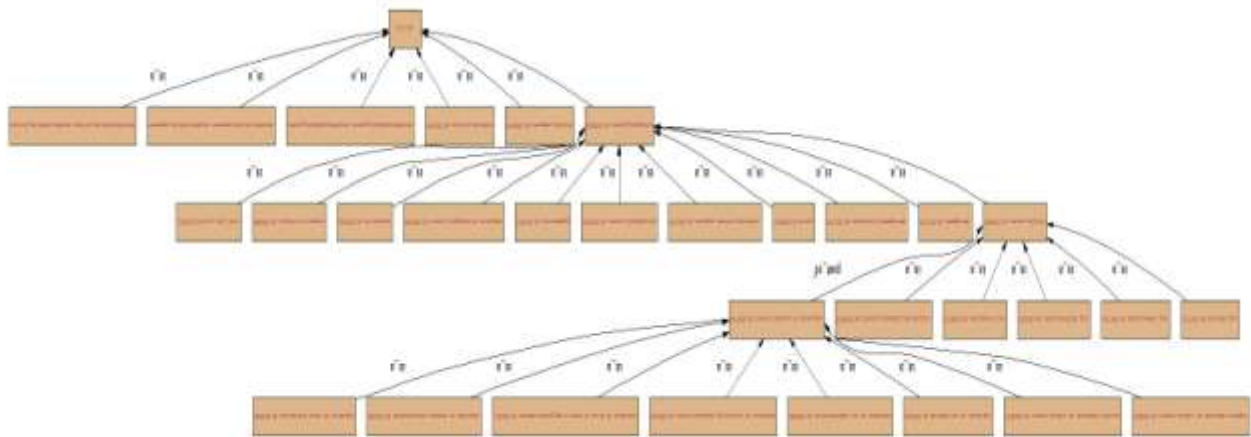
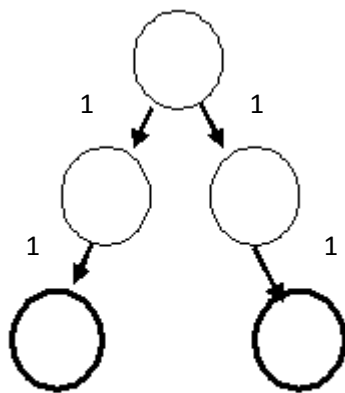


Figure 1

Each protein contains a set of GO terms. These GO terms can be from any of the three categories, i.e. Biological Process, Molecular Function or Cellular Component. We need to find the distance between the two Proteins based on the GO terms they contain. Thus our goal is to calculate three score values when a pair of proteins is submitted. The three scores will represent the difference between the two proteins in the three categories. According to our algorithm a high score would represent a protein pair where there the distance between the two proteins is really high. Distance between two GO terms can be calculated only if they are from the same category.

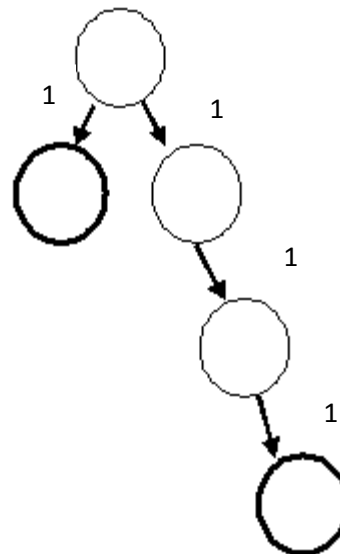
To calculate the score between two GO terms we have taken into account:

1. **The distance between the two terms through the nearest ancestor:** This is obviously a factor that affects the similarity between two terms. If the go terms have separated just one generation before, then they should be more similar than two go terms that have separated 8 generations ago. Thus the simple approach would be to first find the nearest ancestor. Then calculate the distance of go terms from the nearest ancestor and add them up. However, there can be various paths to the nearest ancestor since a go term can have multiple paths. We consider the minimum distance to the nearest ancestor because with this distance measure if a pair of protein is not interactive then it won't be for any of the other paths. But this distance measure is not sufficient since it will be totally dependent on the sum of the distance and not the individual distances of the go terms from the ancestor. Consider figure 2a and 2b. While both will return the same value, they are different configurations. Intuitively the go terms in 2a are more similar since they both are equally distant from the nearest ancestor. But in 2b the go terms are more dissimilar as one is closer to the nearest ancestor than the other. This problem can be solved if we weight the edges. Thus with weighted edges we modify the DAG as shown in 2c and 2d.



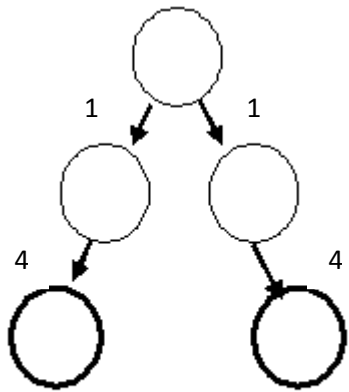
Distance: $(1+1) + (1+1) = 4$

Figure 2a



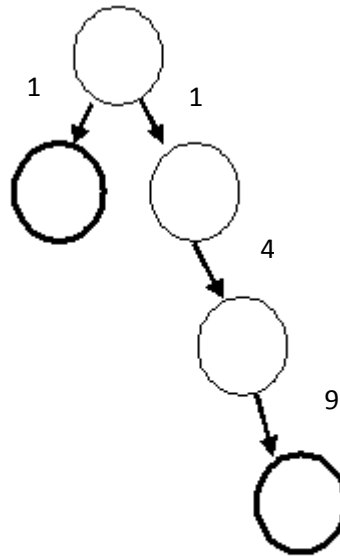
Distance: $(1) + (1+1+1) = 4$

Figure 2b



Distance: $(1+4) + (1+4) = 10$

Figure 2c



Distance: $(1) + (1+4+9) = 15$

Figure 2d

2. **The distance of the nearest ancestor from the root:** We take this factor into account as two go terms which have separated very early have spent very less time together. However go terms which have separated a long distance away from the root have spent a lot of time together, thus likely to more similar.

Based on the above two factors, our scoring formula is:

$$S = \frac{\text{distance}_a + \text{distance}_b}{\text{distance of ancestor from root}}$$

The next step is to get three score values for the three categories for a protein pair. To calculate this, the first step is to decide how we want to form pairs between the go terms in each of the proteins. We have used two algorithms here:

1. **Normalized score of all possible pairs within the same GO DAG:** Form all possible pairs as shown in Figure 3, add the scores for each of the GO terms and return the average of that.

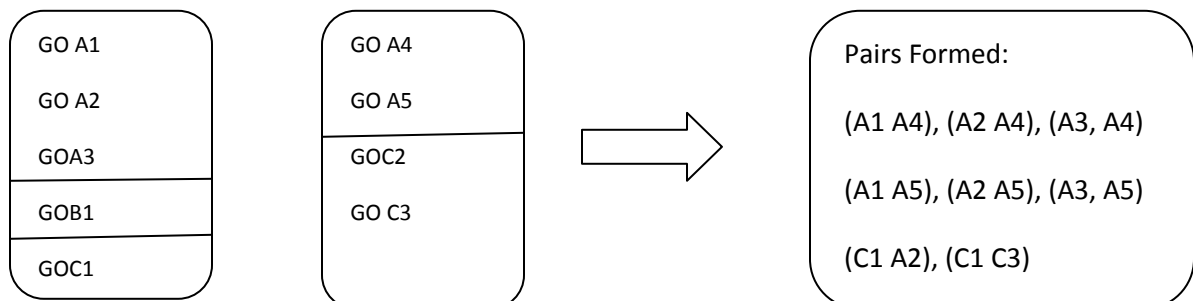


Figure 3

2. **Normalized score of pairs formed between the closest GO terms:** The idea behind this algorithm is to get a score that reflects how much we need to change one protein to make it similar to the other one. So our algorithm is:
1. Form all possible pairs of go terms and calculate the distance between them
 2. Sort the list of pairs in ascending order according to the distance
 3. Go through the sorted list and the pair is taken into account only if both the go terms have not been used yet
 4. Repeat 3 till all the go terms in one of the proteins are mapped
 5. Form pairs for the unmapped go terms in the other protein with the closes go term(which is already mapped now)

Figure 4 below illustrated this algorithm. Step 5 is required to make the algorithm symmetric.

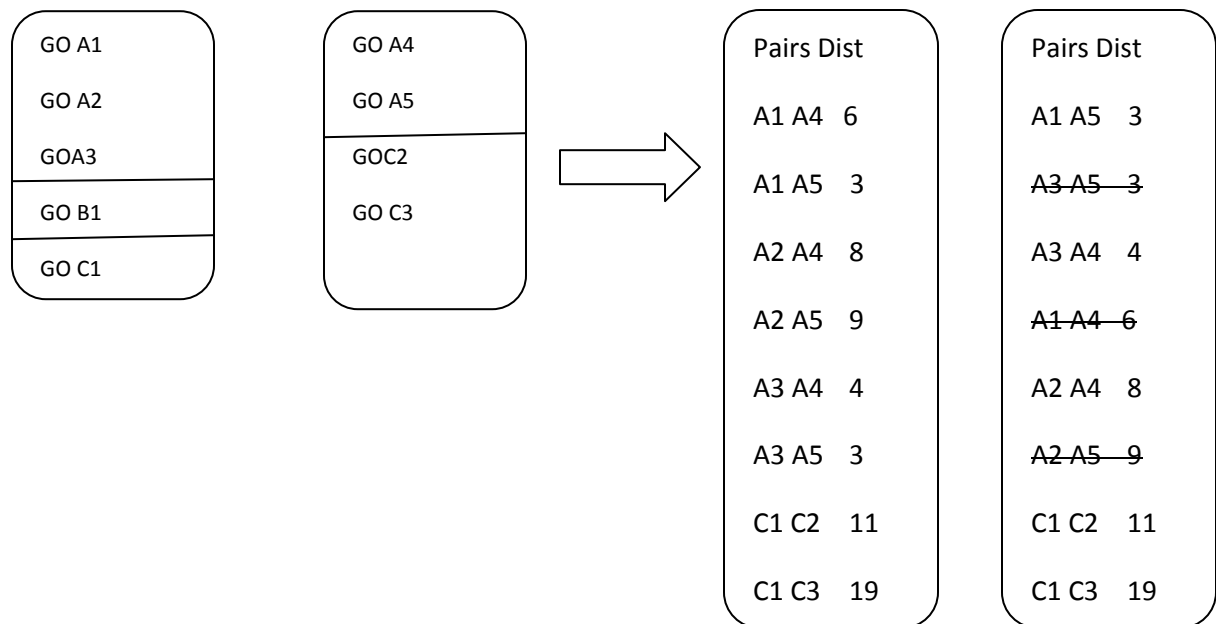


Figure 4

Experiments

We calculate the scores for the protein pairs in the 4 datasets we have, which are the positive training and testing dataset and the negative training and testing dataset. Then we use classifiers to train them, and check the accuracy on the testing set. We have used two classifiers, Naïve Bayes, and Support Vector Machine. The results are reported below in Table 1.

Method	Accuracy			
	Naïve Bayes Classifier		Support Vector Machine	
	5 fold Cross Validation	Training Set	5 fold Cross Validation	Training Set
All possible Pairs (Without weighted edges)	62%	57%	60%	68%
All possible Pairs	58%	56%	77%	86%
Closest Pairs	63%	58%	74%	83%

Table 1

References

- [1] Edward M. Marcotte, Matteo Pellegrini, Ho-Leung Ng, Danny W. Rice, Todd O. Yeates and David Eisenberg. Detecting protein functions and protein-protein interactions from genome sequences. *www.sciencemag.org*, vol 285, 30 July, 1999.
- [2] Joel R. Bock, and David A. Gough. Predicting protein protein interactions from primary structure. *Bioinformatics, Oxford Journals*, August 22, 2000.
- [3] Jones,S. and Thornton,J.M. Principles of protein-protein interactions. *Proc. Natl Acad. Sci. USA*, 93, 1320, 1996.
- [4]. <http://www.geneontology.org/> (Read)
- [5]. MS Thesis Report, Patty Lu, Department of Computer Science, UCSB, June 2006. (Read)
- [6] M.A. Huynene J.O. Korbel, B. Snel and P. Brok. Shot: a web server for the construction of genome phylogenies. *Trends Genet*, pages 158–162, 2002.
- [7] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24(5):513–523, 1988.
- [8] W. Zhong and P.W. Sternberg. An efficient indirect branch predictor. *Genome-wide prediction of C. elegans genetic interactions*, pages 1481–4, March 2006.