Random Forest Similarity for Protein-Protein Interaction Prediction from Multiple Sources

Y. Qi, J. Klein-Seetharaman, and Z. Bar-Joseph

Pacific Symposium on Biocomputing 10:531-542(2005)

RANDOM FOREST SIMILARITY FOR PROTEIN-PROTEIN INTERACTION PREDICTION FROM MULTIPLE SOURCES

YANJUN QI

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

JUDITH KLEIN-SEETHARAMAN

Department of Pharmacology, University of Pittsburgh School of Medicine, Pittsburgh, PA 15213, USA

ZIV BAR-JOSEPH

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

One of the most important, but often ignored, parts of any clustering and classification algorithm is the computation of the similarity matrix. This is especially important when integrating high throughput biological data sources because of the high noise rates and the many missing values. In this paper we present a new method to compute such similarities for the task of classifying pairs of proteins as interacting or not. Our method uses direct and indirect information about interaction pairs to constructs a random forest (a collection of decision tress) from a training set. The resulting forest is used to determine the similarity between protein pairs and this similarity is used by a classification algorithm (a modified kNN) to classify protein pairs. Testing the algorithm on yeast data indicates that it is able to improve coverage to 20% of interacting pairs with a false positive rate of 50%. These results compare favorably with all previously suggested methods for this task indicating the importance of robust similarity estimates.

1 Background

Protein-protein interactions play key role in many biological systems. These involve complex formation and various pathways which are used to carry out biological processes. Correctly identifying the set of interacting proteins can shed new light on the functional role of various proteins within the complex environment of the cell.

High throughput methods including two-hybrid screens [3,4] and mass spectrometry [5,6] have been used in an attempt to characterize the large set of interacting proteins in yeast. While some promising results were generated from these large scale experiments, these data sets are often incomplete and exhibit high false positive and false negative rates [1]. In addition to the high throughput experimental datasets that specifically look for protein interaction, other datasets provide indirect information about interaction pairs. For example, it has been shown that many interacting pairs are co-expressed [1] and that proteins in the same complex are in some cases bound by the same transcription factor(s) [18]. Sequence data was also be used to infer such interactions (for example by relying on domain-domain interactions [14]). Each of these datasets provides partial information about the interacting pairs and thus, in order to reliably infer protein-protein interactions one needs to integrate evidence from these different biological sources. Deriving such an accurate and complete set of interactions from these data sources is an important computational and biological problem.

When combining these disparate datasets one needs to solve a number of computational problems. First, the data is noisy and contains many missing values (for example, for two-hybrid (Y2H) system, the interactions involving membrane proteins may be undetectable and the system also suffers from high false positive rate due to factors like fortuitous activation of reporter genes [3,4]). In addition, some of these data sources are categorical (for example, synthetic lethal [1]) while others are continuous (for example, mRNA co-expression). Finally, there is the issue of weighting the different data sources. Some should have more weight than others (for example, intuitively the direct information should be weighted higher than the indirect information).

In this paper we present a method that overcomes the above problems by using random forest [19] to compute similarity between protein interaction pairs. We construct a set of decision trees such that each tree contains a random subset of the attributes. Next, protein pairs are propagated down the trees and a similarity matrix based on leaf occupancy is calculated for all pairs. Decision trees and the randomization strategy within random forest can handle categorical data and can automatically weight the different data sources based on their ability to distinguish between interacting and non interacting pairs. Because the trees are generated from random subsets of the possible attributes, missing values can be handled by filled in by an iterative algorithm. Finally, a weighted k nearest neighbor algorithm, where distances are based on the computed similarity, is used to classify pairs as interacting or not.

Our method was tested on yeast. We used several direct and indirect data sources and compared our results to previously suggested algorithms and to many other classification algorithms. As we show in Results, the method described above outperformed all other methods.

2 Related work

von Mering [1] is one of the first to discuss the problem of accurately inferring protein interactions from high throughput data sources. In that paper they have relied on the intersection of four direct experiments to identify interacting pairs.

While this method resulted in a low false positive rate, the coverage was also very low. Less than 3% of interacting pairs were recovered using this method compared to a reference set.

In order to improve coverage, Jansen et al. [11] combined direct and indirect data sources using naïve Bayes and a fully-connected Bayesian network. Unlike our method naïve Bayes assumes conditional independence between the attributes, which is clearly a problem for these datasets (for example, co-expression and co-binding are clearly correlated attributes). Indeed, as we show in section 4 (results), our method outperforms naïve Bayes for this task.

Gilchrist et al. [13] proposed a Bayesian method to integrate information from direct high-throughput experiment [5, 6]. However, while their method works well it requires a large number of direct measurements for each protein pair. Such data is not available for most pairs. In contrast, by integrating indirect information our method can correctly detect interacting pairs if only limited direct information is available.

Lan et al. [12] constructed a decision tree to predict co-complexed protein pairs by integrating direct and indirect genomic and proteomic data. While our method also relies on decision trees, it constructs many such trees and not one. This is important when missing values are an issue. Consider two (dependent) attributes A and B that are both useful for the classification task. Assume A is slightly better than B. In that case A can be chosen as the root node of the tree and since B is highly correlated with A, it will not be used at a high level in the tree. Now, if a protein pair lacks a measurement for A but has a value for B it might not be classified correctly based on the low weight assigned to B. In contrast, when using the random forest approach, B will be selected by many trees as a high level split (see Methods). This allows our method to deal more effectively with these noisy datasets.

3 Method

We use multiple high throughput datasets to construct a d -dimensional vector $\boldsymbol{X_i}$ for every pair of proteins. Each item in this vector summarizes one of these datasets (for example, are these two proteins bound by the same transcription factor? what is their expression correlation? see [21] for a complete list of attributes in each vector). Given these vectors the task of protein interaction prediction can be presented as a binary classification problem. That is, given $\boldsymbol{X_i}$ does the i*th* pair interact (Y_i=1) or not (Y_i= -1).

Note that there are a number of unique characteristic to this classification task. First, the number of non interacting pairs is much larger than the number of interacting pairs (as much as ~600 to 1 [section 4.2]) and so false negatives may be more costly than false positives. Second, there is no negative training dataset available for this task. Third, the features are of different types. Some of the entries in $\boldsymbol{X_i}$ are categorical while others are numerical. Finally, many entries in each of these vectors are missing.

In order to overcome these difficulties we divide the classification task into two parts (see Figure 1). We first compute a similarity measure between genes pairs (overcoming noise, missing values and the different types of data) and then use this similarity to classify protein pairs taking into account the skewed distribution of positives and negatives. Below we discuss in detail each of these two parts.

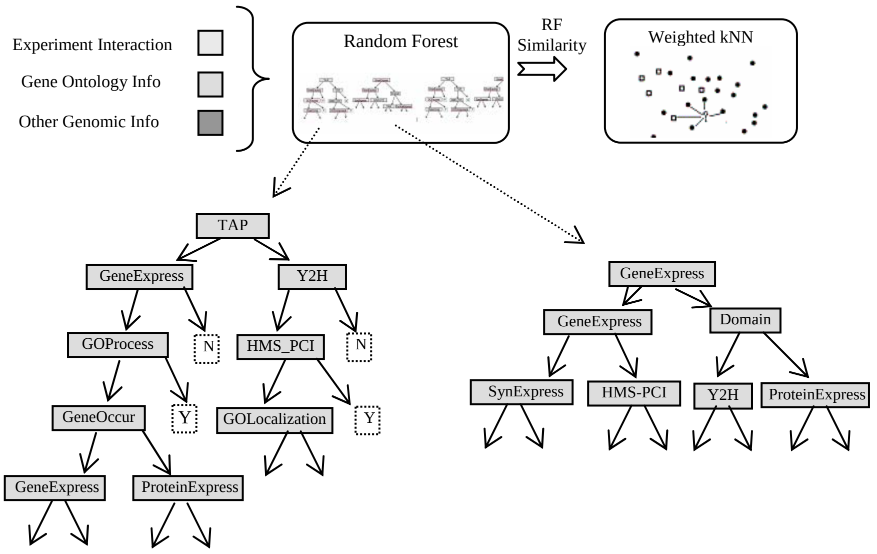

Figure 1. Classification process. To generate the random forest we select for each tree a subset of training data. Next, for every node in these trees a random subset of the attributes is chosen and the attribute achieving the best division is selected. Once trees are grown, all protein pairs (remaining training and test sets) are propagated down the trees and similarity is computed based on leaf occupancy (see text). Using the computed similarity a weighted KNN algorithm is used to rank pairs by the resulting interaction score.

3.1 Random Forest (RF) and RF Similarity

We use random forest [19] to determine the similarity of protein pairs. Random forest (RF) was initially introduced as a classification algorithm, though it can also be used to compute similarities. RF constructs a large set of independent decision trees (see below). Results from these trees are combined for the classification or similarity calculation task.

Decision tree: A decision tree is a binary tree with nodes corresponding to attributes in the input vectors. Tree nodes are used to determine how to propagate a given attribute set down the tree. Nodes can either be threshold nodes or categorical nodes. Decision trees also contain terminal (or leaf) nodes that are labeled as -1 or 1. In order to classify a protein pair as interacting or not, this pair is propagated down the tree and decision is made based on the terminal node that is reached. Decision trees are grown using a training set. At each node the algorithm searches for an attribute that best separates all instances in that node. If the attribute perfectly classifies all instances so that all instances in one of the two descendent nodes have the same label then this node becomes a terminal node with the appropriate label. Otherwise, the above process is repeated until all instances are at terminal nodes.

Random forest: Random forest [19] uses a collection of independent decision trees instead of one tree. Denote by Θ the set of possible attributes (or variables on which nodes can be split) and by $h(x,\Theta)$ a tree grown using Θ to classify a vector *x*. Using these notations a random forest *f* is defined as:

$$f = \{h(x,\Theta_k)\}, k = 1,2,...,K \qquad (1)$$

Where $\Theta_k \subseteq \Theta$. That is, a random forest is a collection of trees, where each tree is grown using a subset of the possible attributes. For the *k*th tree Θ_k is randomly selected, and is independent of the past random vectors $\Theta_1,...,\Theta_{k-1}$. In order to classify, using each of the trees 'votes' for one of the classes and the most popular class is assigned to input **x**.

One of the main reasons random forests perform better than a single decision tree is their ability to utilize redundant features and the independence of the different classifiers (trees) used. This is also important in our case since if a pair has values for one redundant feature but not the other, we can still use this feature for the similarity calculation process.

Specifically we have grown the random forest in the following way: Each tree is grown on a bootstrap sample of the training set (this helps in avoiding overfitting). A number $m << M$ (M is the total number of attributes) is specified, and for each node in the tree the split is chosen from *m* variables that are selected at random out of the total M attributes. Once the trees are grown, they can be used to estimate missing values and to compute similarities as follows.

Handling missing values: Random forest can be used to estimate missing values. For *training* data missing values are first assigned the median of all values in the same class (or the most frequent value for categorical data). Next, the data vectors are run on the forest, and missing data is re-estimated based on pairs that share the same terminal leaves with this pair. This process can be iterated until these estimates converge.

For *test* data, we first replicate the attribute vector and then apply a similar procedure to each of the two replicas. Initially, missing values in the first and second replicas are set to the mean values of the positive and negative classes, respectively. Next, these two replicas are propagated down the trees and the values for each are re-estimated based on neighboring pairs. This process is iterated and the final values are determined from the class that receives the most number of votes in the different trees.

Random Forest Similarity: For a given forest *f*, we compute the similarity between two pairs of proteins pairs X_1 and X_2 in the following way. For each of the two pairs we first propagate their values down all trees within *f*. Next, the terminal node position for each pair in each of the trees is recorded. Let $\mathbf{Z_1} = (Z_{11}, \ldots, Z_{1K})$ be these tree node positions for X_1 and similarly define $\mathbf{Z_2}$. Then the similarity between pair X_1 and X_2 is set to: (*I* is the indicator function.)

$$S(X_1, X_2) = \frac{1}{K}\sum_{i=1}^{K} I(Z_{1i} == Z_{2i}) \qquad (2)$$

As we discuss in Results, we partition our training set to two parts. The first is used to generate the random forest. The second is used to train the kNN algorithm. In order to compute similarities efficiently, the following algorithm is used. Given a random forest with *K* trees and up to *N* terminal nodes in each tree we first generate a $N*K$ vector *V* where each entry in *V* contains a linked list of the kNN training set pairs that reside in that node. Given a new test pair we first propagate it down all trees (in $O(N*K)$ time) and for each of the terminal nodes it arrives at we find the corresponding set of training pairs from *V*. For each such pair we increase their similarity count by one. Thus, for a given test pair it takes only $O(|S_{train}|+N*K)$ to compute its similarity to all the training points, where S_{train} is the training set and $|S|$ represents the number of elements in *S*.

3.2 Classifying protein pairs

We use a weighted version of the k-Nearest Neighbor (kNN) algorithm to classify pairs as interacting or not. While we have tried a number of classifiers for this data (see Results) the main advantage of kNN for this task is its ability to classify based on both, similarity and dissimilarity (as opposed to similarity alone). As can be seen in Figure 2, while non interacting pairs are similar to each other, the main distinguishing feature of interacting pairs is their distance from (or dissimilarity with) non interacting pairs. Due to the highly skewed distribution of interacting and non interacting pairs, it is likely that the closest pair to an interacting pair will be a non interacting pair (though their similarity might not be high). Decision trees (or RF) may use these to incorrectly classify an interacting pair as non interacting.

However, kNN can take into account the magnitude of the similarity, and if it is too weak can still classify the pair as an interacting pair.

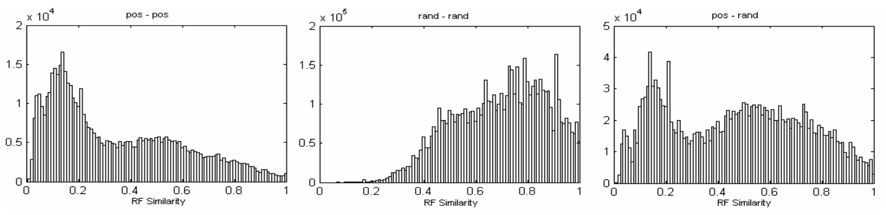

Figure 2. The pairwise RF similarity. Three histograms of pairwise similarities between all positive pairs (left) all random pairs (center) and between all positive and all random pairs. Note that while the random set is fairly tight, the positive set exhibits a greater diversity and is also far (on average) from most random samples. kNN can utilize this fact by relying on the actual distance to the closest neighbors (see text for details).

Specifically, given a set of training examples (X_i ,Y_i), and a query X_q , we calculate the interaction score for *q* using the weighted mean of its neighbor's Y_i values, where the weight depends on the similarity of each of training pairs to *q:*

$$f(q) = \sum_{p=1}^{k} S(X_q, X_{neighbor(p)}) * Y_{neighbor(p)} \qquad (3)$$

Here $S(X_i, X_q)$ is the similarity between *i* and *q* as computed by RF and $Y \in \{1,-1\}$. The *test* set can then be ranked by the *f(q)* interaction scores. A cutoff *t* can be derived using a training set to threshold this ranking list such that *q* is classified as interacting if $f(q)>t$. In particular, *t* does not have to be equal to 0. In fact, in our case $t<0$ meaning that even though this pair is (on average) closer to non interacting pairs, since it is not close enough, it is classified as an interacting pair.

4 Results

We first discuss the biological features we used for the attributes vectors. Next we present results for applying our classification method for determining protein interaction pairs in yeast.

4.1 Attribute set

As mentioned in the introduction, there are many high throughput biological data sources related to protein-protein interaction. The method described in this paper is general and can be used with any type of biological data. Thus, while we have tried to use as much data sources as we could, when a new data source (such as protein expression arrays) becomes available, the method discussed in this paper can take advantage of that data as well. For the results presented in this section we used a total of 15 attributes for each protein pair (see website [21]). Overall, these data sources can be divided into three categories: Direct experimental data sets (two-hybrid screens and mass spectrometry), indirect high throughput data sets (gene expression, protein-DNA binding etc.) and sequence based data sources (domain information, gene fusion etc.). In addition to combining these data sources, our method can also indicate which of the different data sources is better for predicting protein interactions as discussed below.

4.2 Reference set

We need a reference set to train/test the algorithm. For the positive set (or the interacting pairs) ~4000 yeast protein pairs are derived from the database of interacting proteins (DIP [17]). This set is composed of interacting protein pairs which have been experimentally validated, and thus can serve as a reliable positive set. Unlike positive interactions, it is rare to find a confirmed report on non interacting yeast pairs. Here we follow [12] which have used a random set of protein pairs as their negative set instead. This selection is justified because of the small fraction of interacting pairs in the total set of potential protein pairs. It is estimated Supplementary [21] that only ~1 in 600 possible protein pairs actually interact [10, 13] and thus, over 99.8% of our random data is indeed non interacting which is probably better than the accuracy of most training negative data. Actually this extremely unbalanced class distribution of our reference set motives the weighted kNN ranking step in our algorithm.

4.3 Important attributes

Biologically, it is of particular interest to identify the attributes and data sources that contribute the most to our ability to classify protein pairs. Such an analysis can help uncover relationships between different data sources which are not directly apparent. In addition, it can help identify what data sources should be generated for determining interaction in other species (for example, in humans). One way to determine such a set using random forest is to score attributes based on the levels of nodes that use them to split the data. Since each node splits the data using the best available attribute, attributes used in higher levels in the tree contribute more than those used in lower levels.

As a rough estimate for the contribution of each attribute we have counted the percentage of nodes that use this attribute in the top four levels of all trees in our trained random forest model. Of the 15 features we used, gene co-expressed had the highest score with 18% of top nodes using it to split the input data. Next came three features: protein co-expression, domain-domain interaction and GO co-process, each with ~11% of the nodes. These were followed by TAP mass spectrometry data (8%) GO co-localization (6%), Y2H screens (4%) and HMS-PCI (4%). (see

Supplementary [21] complete list). Interestingly, indirect information played a very important role in the decision process though it is likely that this results from the fact that direct experiments cover less than 30% of all protein pairs. However, mass spectrometry data are clearly more significant than Y2H data, consistent with the notion that mass spectrometric identification of protein-protein interaction is less prone to artifacts than Y2H experiments. It is particularly encouraging that co-expression and GO features contribute such strong components to the prediction, clearly supporting the notion that a large amount of indirect data that measures biologically relevant information is helpful in predicting interaction partners.

4.4 Performance Comparison

We use precision vs. recall curve to perform the comparisons completely.
Precision: Among the pairs identified as interacting by the classifier, what is the fraction (or percentage) that is truly interacting?
Recall: For the known interaction pairs, what is the percentage that is identified?

Let d be the number pairs identified as interacting by the classifier, T be the number of pairs labeled as interacting, and c be the number of pairs correctly identified as interacting, then precision and recall are defined as:

$$\text{Precision} = c/d \quad \text{Recall} = c/T \tag{4}$$

In other words, precision is the accuracy of our predictor whereas recall is the coverage of the classifier. Note that even 50% precision can useful. For example, biologists studying a specific protein can extract a list of potential interacting partners computationally first and carry out further experiments knowing that on average 50% of their experiments will identify true interacting pair. This is a much better ratio than if the set of potential pairs was randomly selected.

For our algorithm, in each cross validation run, we divided our training set into two equal parts. The first was used to construct the random forest and the second was used by kNN algorithm. Thus, our algorithm uses the same amount of training data as the other algorithms we compare to (see below).

In order to generate a precision / recall curve we use different thresholds as discussed above. For the other classification methods, we generate the curve in a similar manner. For instance, for the naïve Bayes classifier, we can use the naïve Bayes prediction probability of a test point to arrive at a ranked list.

Figure 3 shows a comparison between our method and a number of other classification algorithms. The figure on the left compares our method with a weighted kNN that uses Euclidean distance instead of the random forest similarity, with the naïve Bayes method and with a single decision tree. In general, for a wide range of high precision values our method outperforms the other methods. It is especially interesting to compare our method with the kNN method using Euclidian distance. As can be seen, using the robust similarity estimates from the random forest results greatly improves the classification results. We have also compared our algorithm to a classification method that only uses the resulting random forest (based on popular vote) to classify protein pairs and to a number of other popular classifiers including Support Vector Machine (SVM), logistic regression and Adaboost (right figure). In all cases our algorithm performed better for precision values that are higher than 0.32. Specifically, holding precision fixed at 0.5 (or 50%) our algorithm achieved a recall rate of 20% while logistic regression achieved 14% recall, random forest and SVM achieved 11% and Adaboost had a 7% recall rate. Finally, we note that while the methods we have used to compare our algorithm with were inspired by previous work (such as single decision tree [12] and naïve Bayes [11]), we have used a slightly different feature set and a different training set compared to each of these two papers. Thus, the results reported for these methods here are different from the ones reported in these papers. See website [21] for details about the implementation of the other classification methods.

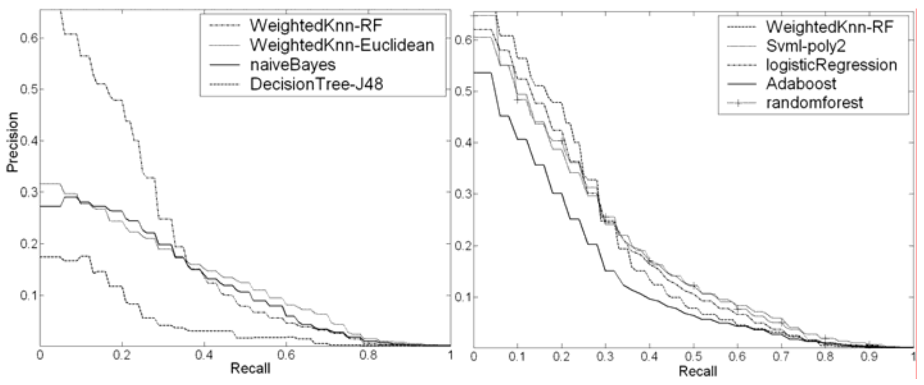

Figure 3. Precision vs. Recall curves. Left: Comparison of weighted kNN using random forest similarity, weighted kNN using Euclidean distance, naïve Bayes and a single decision tree (J48); Right: Comparison of weighted kNN using random forest similarity, logistic regression, support vector machine, Adaboost and random forest classifier.

4.5 Predicting interactions for the yeast pheromone response pathway

To analyze the results for their utility in the design of new experiments, we compared the predictions of our method to their labels for one specific pathway, the yeast pheromone pathway. The yeast mating factors MATα/a bind to their cognate membrane receptors, Ste2/3, members of the G protein coupled receptor family.

Subsequent binding and activation of the G protein induces a MAP kinase signaling pathway via the G protein βγ subunit. We selected 25 proteins that are known to participate in this pathway. We applied our algorithm (using a different training set) to classify the 300 (25*24/2) potential interacting pairs. Our algorithm classified 44 of these pairs as interacting. 31 of these pairs (70.45%) were known to interact while only 2 (4.55%) are verified to be wrong predictions. The remaining 11 pairs (25%) are new predictions that are reasonable and would functionally make sense. They form two clusters: The first involves the possible interaction between the STE5 anchor protein and the receptors. The receptor would then make additional interactions due to STE5s' anchoring function. The second cluster is a possible interaction between the most downstream components of the signaling cascade including BEM1, BNI1 and FUS1, mediating cell fusion (see website [21] for details). These new biological hypothesis can be used to design future lab experiments.

5 Discussion and future work

In this paper we presented a method for predicting protein-protein interactions by integrating diverse high-throughput biological datasets. Our method works in two steps. First, a similarity measure is computed between protein pairs. Then a classification algorithm uses the computed similarities to classify pairs as interacting or not. We have applied our algorithm to the task of classifying protein pairs in yeast. As we have shown, our algorithm outperforms previous methods suggested for this task and can also derive meaningful biological results for known pathways.

In this paper we have used random forest to learn a similarity function between protein pairs. Recently, a number of methods have been suggested for learning distance matrices [22]. We would like to test some of these methods and see if they can improve our classification accuracy. Specifically, it will be challenging to apply these methods to datasets with missing values, as is the case here.

Interestingly, many of the features determined to be important using our method are indirect measurements. This opens the possibility of extending this method to determine interacting pairs in organisms where little high throughput direct information is available, such as humans.

Acknowledgements This research work is supported by National Science Foundation Information Technology Research grant number 0225656.

References

1. Von Mering C, et al., Comparative assessment of large-scale data sets of protein-protein interactions. *Nature* 417:399-403, 2002
2. Bader GD, Hogue CWV. Analyzing yeast protein-protein interaction data obtained from different sources. *Nature Biotechnology* 20:991-997, 2003
3. Uetz P, et al., A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature.* 403(6770):623-7, 2000
4. Ito T, et al., A comprehensive two-hybrid analysis to explore the yeast protein interactome., *Proc Natl Acad Sci,* 10;98(8):4569-74, 2001
5. Gavin AC, et al., Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*. 415(6868):141-7, 2002
6. Ho Y, et al., Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. *Nature* 415(6868), 2002
7. Enright AJ, et al., Protein interaction maps for complete genomes based on gene fusion events. *Nature*. 402(6757):86-90, 1999
8. Huh WK, et al, Global analysis of protein localization in budding yeast. *Nature.* 425(6959):686-91, 2003
9. Ghaemmaghami S, et al., Global analysis of protein expression in yeast. *Nature.* 425(6959):737-41, 2003
10. Tong A.H.Y. et al. Global Mapping of the Yeast Genetic Interaction Network. *Science.* 303: 808-813, 2004
11. R Jansen, et al., A Bayesian networks approach for predicting protein-protein interactions from genomic data, *Science* 302: 449-53, 2003
12. Lan V. Zhang, et al., Predicting co-complexed protein pairs using genomic and proteomic data integration, *BMC Bioinformatics.* 5 (1): 38, 2004
13. Gilchrist MA, et al. A statistical framework for combining and interpreting proteomic datasets. *Bioinformatics.* 20(5):689-700, 2004
14. Deng M, et al., Inferring domain-domain interactions from protein-protein interactions. *Genome Res*. 12(10):1540-8, 2002
15. Lee et al., Transcriptional Regulatory Networks in Saccharomyces cerevisiae, *Science* 298:799-804, 2002
16. Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium (2000), *Nature Genet.* 25: 25-29, Dec. 2003
17. Xenarios I, et al., DIP: The Database of Interacting Proteins: 2001 update, *Nucleic Acids Res.* 29(1):239-41, 2001
18. Bar-Joseph Z, et al., Computational discovery of gene modules and regulatory networks, *Nat Biotechnol.* (11):1337-42, 2003
19. Breiman, L.Random Forests, *Machine Learning,* 45, 5-32, 2001
20. Elion, E.A. Ste5: a meeting place for MAP kinases and their associates, *Trends Cell Biol.* 5, 322-7, 1995
21. Supporting and supplementary information for this paper is available at http://www.cs.cmu.edu/~qyj/psb05_PPI.html
22. E.P. Xing, et al. Distance Metric Learning, with application to Clustering with side-information, *NIPS*, 2002