Conditional Random Fields for Protein Function Prediction

Thies Gehrmann[1], Marco Loog[2], Marcel J.T. Reinders[1,3,4], and Dick de Ridder[1,3,4]

[1] Delft Bioinformatics Lab, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
[2] Pattern Recognition Lab, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
[3] Netherlands Bioinformatics Centre, P.O. Box 9101, 6500 HB Nijmegen, The Netherlands
[4] Kluyver Centre for Genomics of Industrial Fermentation, P.O. Box 5057, 2600 GA Delft, The Netherlands

Abstract. Markov Random Fields (MRF) have been shown to be good predictors of functional annotation, using protein-protein interaction data. Many other sources of data can also be used in this prediction task, but they are typically not integrated. In this study, we extend a method using MRFs in order to allow the use of additional data.

A conditional random field (CRF) model is proposed as an alternative to an MRF model in order to remove the requirement of modeling relationships between the sources of data. We observe that a substantial performance improvement is possible using additional data, such as genetic interaction networks. The improvement gained from each source of evidence is not the same for each protein function, indicating that each source supplies different information. We demonstrate that CRFs can be used to efficiently integrate various sources of data to predict functional annotations.

1 Introduction

The annotation of genomes is of the utmost importance and the value a deeper understanding of biological processes has to the future of humanity can not be overstated. Proteins found on these genomes are assigned functional annotations based on biological experimentation. This is no easy task, since proteins may be involved in several kinds of functions, and testing every protein's involvement in every function is infeasible due to the number of proteins and the complexity of their interactions. Therefore, the functional annotation of proteins remains largely incomplete, even for the most well studied organisms. By predicting which proteins are most likely to have a specific function, we can reduce the expenses and work required to get a more complete genomic annotation. This is the reason why *in-silico* prediction of protein function is important.

Algorithms that predict functional annotations differ not only in their underlying method, but also in the data they operate upon. For a recent review,

A. Ngom et al. (Eds.): PRIB 2013, LNBI 7986, pp. 184–195, 2013.

see [1]. Some take a machine learning approach by extracting features from sequences to predict a functional assignment. Profile methods take advantage of patterns such as conserved regions or structural qualities found in the proteins themselves and compare them to annotated proteins.

Network models use biological network data to predict protein function. Primitive network models determine the function of an individual protein based upon the known function of proteins in its immediate neighborhood. Graph theoretic models model the entire network simultaneously, and diffuse information between the proteins according to the edges defined in the graph. Probabilistic network models also model the entire network, but assign labels with an associated probability.

Markov Random Fields (MRFs), one kind of probabilistic network model, have often been used in predicting protein functions from network data [2,3,4]. Deng *et. al.* [2] defined an MRF model for predicting protein functional annotations, and laid the basic framework. They define an MRF over a protein-protein interaction (PPI) network, where pairwise interactions between proteins are modeled by factors in the MRF. Kourmpetis *et. al.* [3] extended this method by improving parameter estimation through multiple parameter estimation steps. These MRF models mostly use protein-protein interaction data, but there are many other biological network sources that can suggest functional similarity such as genetic interaction networks. Here, we show that the use of additional network data, integrated with a conditional random field (CRF) model, can give increased performance over the previous methods.

Perhaps the most similar approach to ours used MRFs and included GI networks [5,6]. These methods have some limitations; their models do not make use of continuous data, assume independence between network sources, and use a single parameter estimation step. A follow-up paper uses a more sophisticated parameter estimation scheme [7]. In this paper, we take into account many more sources of evidence in a CRF model which corrects these flaws.

2 Method

Previous models using MRFs need to either model the relationships between the input data, which can become complicated, and is essentially unnecessary, or assume independence [5,6,7], which is often wrong. Conditional random fields are the discriminative version of MRFs which model the dependence of the output on the input rather than the full joint distribution of the input and output. Our contribution to the field - to extend the previous framework in [2] and [3] to a CRF model with multiple sources of data - is described here.

2.1 Conditional Random Fields

A conditional random field (CRF) is a discriminative graphical model which splits the variables into two sets; input variables X, and output variables Y. We are not interested in modeling the relationships between variables within

X; these may not be related to the problem we wish to solve, and can even be very difficult to model. By conditioning over X, we assume a dependence upon X, but make no assumptions upon the distributions of variables within X. This allows us to model more complex relationships between the variables in Y and X. A CRF is represented as a factor graph, in which the random variables are represented as nodes, and factors describe the dependencies between them. The CRF distribution is defined in terms of its factors $f \in F$, conditioned over X:

$$p(Y|\theta, X) = \frac{1}{Z(X)} \prod_{f \in F} \psi_f(Y_f, X) \tag{1}$$

$$Z(\theta, X) = \sum_{y'_1} ... \sum_{y'_n} \prod_{f \in F} \psi_f(Y'_f, X) \ , \tag{2}$$

where Z is a normalization function dependent upon X, and Y_f are all the variables in Y involved in factor f.

2.2 The Model

We therefore, for the problem of protein function prediction, represent the protein labels and sources of evidence between them (e.g. physical interactions, co-expression) as nodes in a factor graph, where factors describe the relationships between them. The CRF model has to integrate multiple sources of evidence which each describe in some form the functional relationships between proteins. We consider one function at a time. For each protein p_i, we introduce a variable y_i, which describes its label (1 if the protein has the function, 0 otherwise). Networks which describe interactions between proteins define their context with respect to their functional labels. An edge from network σ which describes an interaction between proteins p_i and p_j, is represented by the random variable $ev_\sigma(i.j)$. We group all ev variables in the set X. Figure 1 shows the basic outline of the method.

The probability of a particular labeling for the graph is defined in terms of the factorization of the model. Figure 2 illustrates how a simple 5-protein network can be factorized with two kinds of factor nodes, pairwise factors, ψ_p and singular factors ψ_s. These are defined in terms of their respective energy functions, U_p and U_s:

$$\psi_p(y_i, y_j; \theta, X) = \exp\{U_p(y_i, y_j; \theta, X)\}$$
$$\psi_s(y_i; \theta, X) = \exp\{U_s(y_i; \theta, X)\} \ .$$

U_s is defined as α if the node has the label, and 0 otherwise:

$$U_s(y_i; \theta, X) = \alpha y_i \ , \tag{3}$$

where α is a parameter. U_p is a function which depends on whether (a) both nodes have the label, (b) one node has the label, or (c) neither node has the

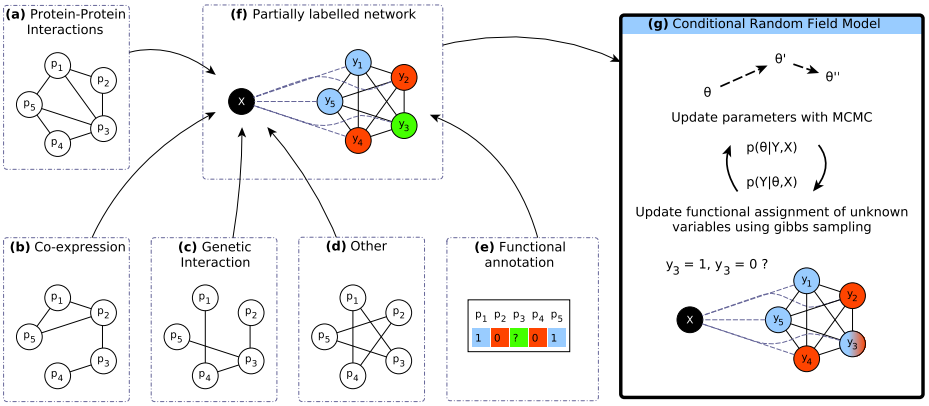


Fig. 1. Conditional random field analysis for protein function prediction. **a-d**: Different sources of evidence between proteins define the functional relationships between proteins. **e**: Variables that describe the functional annotations of proteins are introduced. **f**: Dependencies between the variables are described by the graphical model. **g**: Missing labels are inferred in an iterative scheme.

label. It connects two protein label variables, and the evidence of interactions between them. For a single source of evidence σ this can be expressed as::

$$\begin{aligned} U_\sigma(y_i, y_j; \theta, X) = & \beta_{\sigma,11} y_i y_j ev_\sigma(i,j) + & \text{(a)} \\ & \beta_{\sigma,10}[(1-y_i)y_j + y_i(1-y_j)] ev_\sigma(i,j) + & \text{(b)} \\ & \beta_{\sigma,00}(1-y_i)(1-y_j) ev_\sigma(i,j)\ , & \text{(c)} \end{aligned} \qquad (4)$$

where $(\beta_{\sigma,11}, \beta_{\sigma,01}, \beta_{\sigma,00})$ are parameters. All sources of evidence are combined to form the general pairwise energy function U_p:

$$U_p(y_i, y_j; \theta, X) = \sum_\sigma U_\sigma(y_i, y_j; \theta, X)\ . \qquad (5)$$

When there is only one source of evidence, the model is equivalent to that of [3]. The structure of the label network is no longer explicitly defined by any single biological network, so in order to ensure that nodes are able to use all relationships available to them in X, the graph is by default fully connected between all the variables $y_1 \cdots y_n$; Each pair $(y_i, y_j) \in Y \times Y$ are connected by a pairwise factor. In the event where there is no evidence between two proteins, the relevant potential becomes equal to 1, and therefore does not influence the statistical distribution (1) ($\exp\{0\} = 1$).

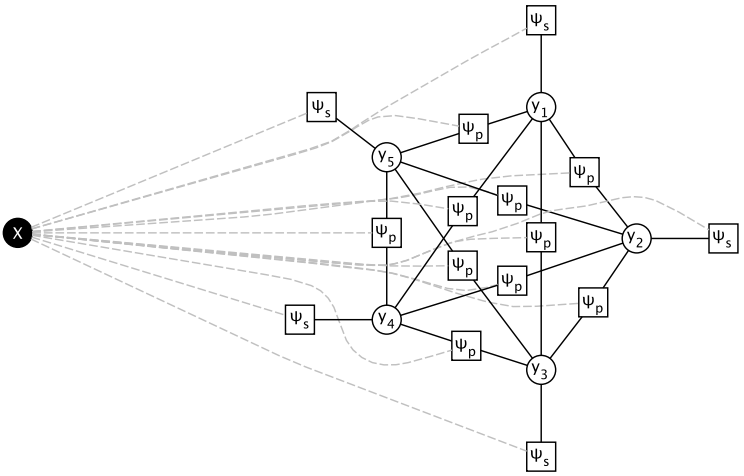


Fig. 2. CRF factorization of the model: Each factor ψ_s and ψ_p in the graph is dependent upon the additional data X

2.3 Conditional Probability

We can express the conditional probability of an individual node being positive in terms of the logistic function, as in [2]:

$$p(y_i = 1|Y_{-i}, \theta, X) = \frac{\psi_s(y_i) \prod_{(y_i,y_j)\in Y\times Y} \psi_p(y_i, y_j, X)}{\sum_{y'_i} \psi_s(y'_i) \prod_{(y'_i,y'_j)\in Y\times Y} \psi_p(y'_i, y'_j, X)} , \tag{6}$$

$$= \frac{\exp\{\ell\}}{1+\exp\{\ell\}} , \tag{7}$$

where ℓ in this case is defined as the log-odds of the probability of the labels at the node, and Y_{-i} refers to all nodes in Y with the exception of i.

$$\ell = \log \frac{p(y_i = 1|Y_{-i}, \theta, X)}{1 - p(y_i = 1|Y_{-i}, \theta, X)} \tag{8}$$

$$= \alpha + \sum_{\sigma} \sum_{(y_i,y_j)\in Y\times Y} [\delta_\sigma y_j ev_\sigma(i,j) + \epsilon_\sigma(1-y_j) ev_\sigma(i,j)] .$$

For each data source σ, two parameters, δ_σ and ϵ_σ are introduced, which replace the β parameters. The derivation is given in [8].

2.4 Inference

We wish to maximize (1), which is normally done with belief propagation. However, because of the size of the network and the number of factors, this can be

intractable. Instead, a Gibbs sampling algorithm is used to predict the labels. The Gibbs sampler operates by sampling a new label for each node from the conditional distribution at that node (7), and using the updated labels to sample new labels for the remaining proteins. The new labels will be used in the following iteration.

2.5 Parameter Estimation

For general graphs as these, exact parameter estimation is intractable [9]. Instead, parameters $\theta = (\alpha, \delta_\sigma, \epsilon_\sigma)$ are found by maximizing the pseudo-likelihood function (PLF), which has been described as a good approximation to the likelihood function. Kourmpetis [3] improved the parameter estimation by re-estimating them iteratively. This measure assumes that the density factorizes into the conditional distributions at each node:

$$PLF(Y|\theta, X) = \prod_{y_i \in Y} p(y_i|Y_{-i}, \theta, X) \; . \tag{9}$$

Each conditional (7) reminds us of the logistic function; logistic regression is thus used to update the parameters θ.

At each iteration, we re-estimate the parameters using the entire set of proteins, including the unknown ones for which new labels were just predicted. The new parameters θ^* are accepted over the previous parameters θ with a Metropolis step, (i.e., with probability):

$$A(\theta^*, \theta) = \min\left(1, \frac{PLF(Y|\theta^*, X)}{PLF(Y|\theta, X)}\right) \; , \tag{10}$$

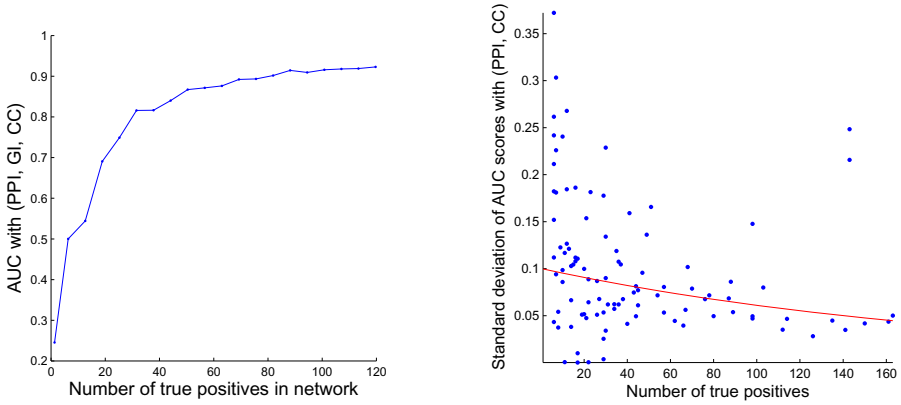
i.e. the new parameters will be accepted with probability $A(\theta^*, \theta)$ from a standard uniform distribution.

3 Experimental Setup

The yeast (*Saccharomyces cerevisiae*) genome is well studied and comes with a plethora of data, making it an excellent organism to test on. In earlier work [3], a PPI dataset from [10] was used. In order to gain additional information sources, different datasets were used. Functional annotations for yeast were taken from the Gene Ontology website. This gave functional annotations to 6383 proteins for 4631 GO terms [11]. An outline of the initialization of the algorithm is given in [8].

3.1 Dataset

We collected a dataset of the following sources:



(a) (b)

Fig. 3. Performance depends on the class distribution in the training set: **a**: The learning curve when using co-citation and genetic interaction data. **b**: As the number of true positives increases, we observe a lower standard deviation in AUC scores. Here co-citation data is used as additional data. The line indicates a best fit for an exponential decay function, with $y = 0.1 \exp(-0.005x)$.

(**PPI**) Protein-Protein Interaction: Collected from BioGRID [12].
(**KI**) Kinase Interaction: Collected from PhosphoGRID [13].
(**GI**) Genetic Interaction: Collected from BioGRID.
(**CX**) Co-expression: Collected from MegaYeast [14].
(**CC**) Co-citation: Collected from STRINGdb [15].

The PPI and KI networks were combined into the same network. The co-citation scores from STRINGdb was mapped onto a logistic curve between 0 and 1. (It is not clear how STRINGdb calculated the original co-citation scores).

3.2 Performance Evaluation

100 functions were randomly selected from the *Biological Processes* and *Molecular Function* ontologies in the Gene Ontology (GO). For each GO term, we select a test set of 300 proteins to mark as unknown; this constitutes the testing set in a cross validation procedure. To ensure that there is positive data in the test set, it is constructed such that it contains approximately 20% of all positive labels for that particular GO term. For each function, the model was trained 10 times using different test sets, and the Area Under the Receiver Operator Characteristic Curve (AUC) score is calculated. For more information, see [8].

4 Results and Discussion

Before discussing any results on the data, we report how the model behaves in training and predicting. i.e. whether parameter estimates converge, and predictions are reliable.

4.1 Model Behavior

Iterative parameter estimation improves on the initial prediction performance. Parameter estimates converge very quickly, usually within one or two steps (data not shown); In the remaining iterations the values only oscillate a little, as in [3]. Despite having good parameter estimates, we cannot stop iterating after just a few steps due to the changing labels after each Gibbs step; The Gibbs sampler needs time to build up a good average of the label probabilities. Like in [3], we observe that the intercept parameter, α is estimated well in the first step (Deng *et. al.*'s estimate), while the δ_σ and ϵ_σ parameters usually are not. Since the intercept parameter is less sensitive to the individual labelings of the nodes we can imagine that it is easier to estimate.

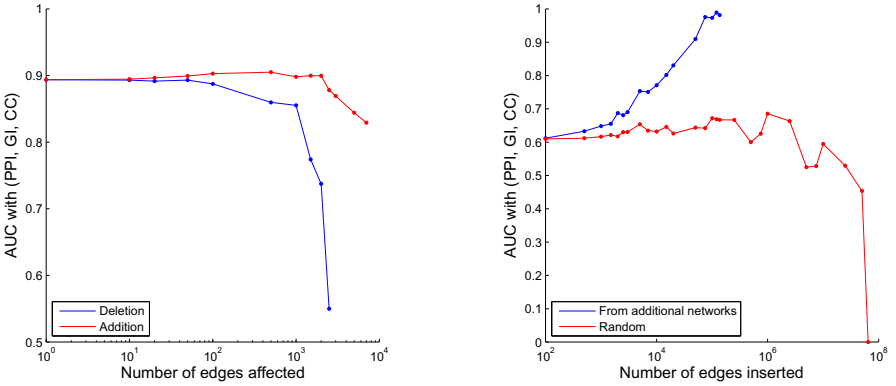
Performance depends on class balance in the training set. The number of proteins annotated with a function (the number of true positives) influences the performance of the method. By removing functional annotations from a function which has many true positives[1], we can see how the performance improves as the number of true positives increases. Figure 3a shows a learning curve which illustrates that as we add more true positives, we are able to predict more accurately.

Illustrated in figure 3b is how the standard deviation of predictions for each function varies depending upon the number of true positives. With more true positives (i.e. more training data), we have a lower standard deviation. An exponential decay function is fit to the data to demonstrate the trend of the data; The standard deviation actually decreases. Unfortunately, a function rarely has that many true positives [8].

The precision of predictions with our method is comparable to that of [3]. This is important to us; if using more data were to give us (on average) a better prediction but with a large variance, it would be useless in practice. Functions for which the performance is bad, generally have a high standard deviation [8].

The model is robust to noise. Rather surprisingly, the model is quite insensitive to random noise. We test the model by independently deleting and adding edges to the relevant 'unknown' proteins. These tests were run under optimal conditions; A specific function[1] was selected for which there were many true positives (many proteins have been annotated with it), and for which the model already performs well. Figure 4a indicates that the model is more sensitive to edge deletions, however, this is due to the fact that there are a limited number of edges that can be deleted before none are left.

[1] *GO:0004672* 'Protamine kinase activity', from the molecular function ontology.



(a) (b)

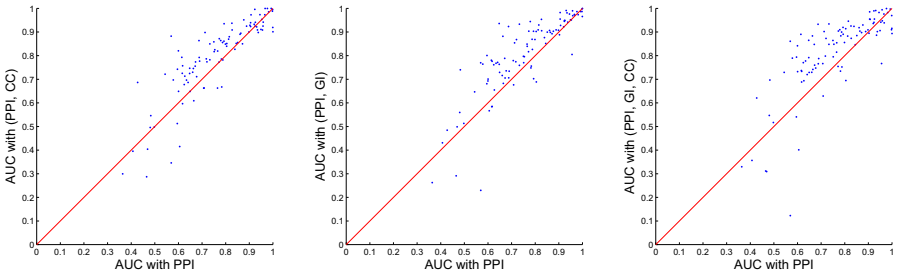
Fig. 4. Model robustness: **a**: The resilience of the method when deleting edges from, or adding edges into the network. **b**: When adding random edges to the network, we do not get a significant performance increase.

A little noise acts as a regularization. Adding random edges can provide a kind of regularization, possibly by adding edges smoothing the network (i.e. making neighborhoods more similar to each other), making the classification task easier. Evident from figure 4a and studied in figure 4b, is that adding a few random edges gives a slightly better performance. For a given function[1], we create two distinct subnetworks from all the sources of evidence, (a) containing only edges also present in the PPI network and (b) all other edges. We compare adding edges to (a) either randomly, or from (b). Adding useful edges from (b) improves performance drastically, in contrast, adding random edges helps only a little, until the graph becomes saturated with edges and no relationships are distinguishable anymore.

4.2 Some Additional Sources Improve Prediction

Adding data sources gives better performance in some cases, and this is reflected in Figures 5a-c. These figures plot the performance of the baseline ([3], using PPI data), against the performance of our method. Any point above the diagonal line means that that particular function has a better performance.

On average, CC data gives us a large improvement over the previous methods, GI data a slightly larger improvement, and the combination of the two an even larger improvement. Figure 5c shows the results from combination of CC and GI networks, which gives the largest improvement on average. The increase in performance is due to the additional data, which provides new information on relationships between proteins. Note the fact that CC data may be good is because of possible bias (two proteins may be co-cited because they *already*



(a) (b) (c)

Fig. 5. The effect of using additional data: We compare the performance per function against that of [3]. On the x-axis are the AUC scores of the algorithm per function, using only protein-protein interaction data (equivalent to [3]), and on the y-axis the AUC scores of the algorithm using additional data. **a**: Using CC data. **b**: Using GI data. **c**: Using CC *a*nd GI data.

have the same functional annotation). CX data invariably has a detrimental effect upon performance [8].

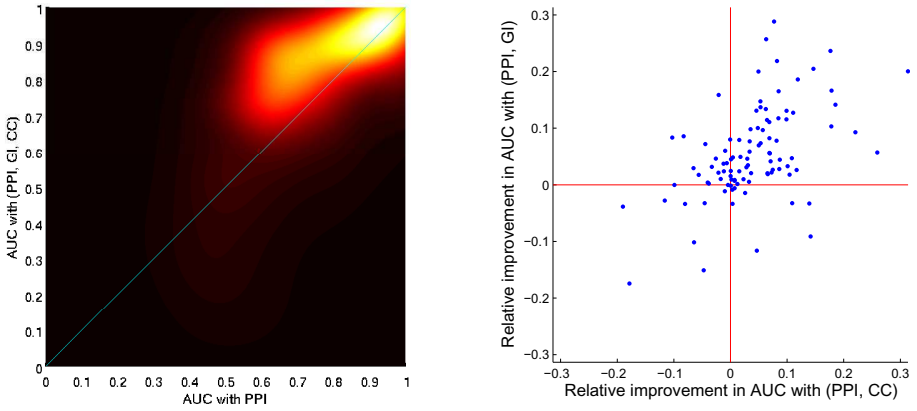
When there are very few true positives for a function, the additional parameters make it difficult to train the model properly, and performance may suffer. In such cases it may be advisable to revert to a single source of evidence.

When we consider each performance pair (only PPI, additional sources) the center of a normal distribution and take into account their standard deviations, we can sum the distributions to get an idea of the performance over all functions. This plot is seen in figure 6a. Most of the density is above the diagonal (over 85%), indicating that the predictions with our method are expected to be better than those in [3].

Data sources complement each other. Figure 6b plots the improvement relative to the Kourmpetis *et. al.* model when using different data sources. It shows that even though there is often a common improvement, correlation is not very high. A function for which one data source helps does not necessarily benefit from another data source. This means that different sources of evidence supply information valuable to predict different functions; The sources of evidence complement each other and are not interchangeable.

5 Discussion

Here we present, for the first time, a CRF model that can be used to easily and effectively predict protein function. The ability to accurately predict which proteins are involved in a function is of great importance to biologists. Whereas MRF models have been used before, our CRF model demonstrates that additional information helps improve prediction. GI and CC networks provide the most useful information. Data sources which have a continuous value would be



(a) (b)

Fig. 6. **a**: A density plot reveals the general trend of model performance. **b**: Different sources of evidence supply different information to the model. We calculate the improvement in prediction but taking the difference in AUC scores when running the model with additional data, and when running it without. They are not highly correlated ($\rho_{GI,CC} \approx 0.57$).

able to describe more subtle relationships between proteins, rather than just strong ones, but such sources are hard to find.

We describe thoroughly the construction of the model and analyze its performance. A more complex factorization could be constructed to describe more complicated relationships between proteins. Furthermore, a nonlinear combination of sources of evidence could give rise to a richer description of the requirements for functional similarity (e.g. proteins should interact *and* be co-expressed).

Despite any improvements, this method is still stochastic, as evident in the variance experienced over multiple runs. Consequently, in practice the model would have to be run multiple times to ascertain exactly which proteins are consistently the most highly ranked.

Acknowledgments. We thank Marc Hulsman for providing some valuable insights and the data sources.

References

1. Radivojac, P., Clark, W.T., Oron, T.R., Schnoes, A.M., Wittkop, T., Sokolov, A., Graim, K., Funk, E.A.: A large-scale evaluation of computational protein function prediction. Nature Methods 10(3) (January 2013)
2. Deng, M., Zhang, K., Mehta, S., Chen, T., Sun, F.: Prediction of protein function using protein-protein interaction data. Journal of Computational Biology 10(6), 947–960 (2003)

3. Kourmpetis, Y.A.I., van Dijk, A.D.J., Bink, M.C.A.M., van Ham, R.C.H.J., ter Braak, C.J.F.: Bayesian Markov random field analysis for protein function prediction based on network data. PLoS ONE 5(2), 9293 (2010)
4. Letovsky, S., Kasif, S.: Predicting protein function from protein/protein interaction data: a probabilistic approach. Bioinformatics 19(suppl. 1), 197–204 (2003)
5. Deng, M., Chen, T., Sun, F.: An integrated probabilistic model for functional prediction of proteins. Journal of Computational Biology 11(2-3), 463–475 (2004)
6. Deng, M., Tu, Z., Sun, F., Chen, T.: Mapping Gene Ontology to proteins based on protein-protein interaction data. Bioinformatics 20(6), 895–902 (2004)
7. Kourmpetis, Y.A.I., van Dijk, A.D.J., van Ham, R.C.H.J., ter Braak, C.J.F.: Genome-wide computational function prediction of *Arabidopsis* proteins by integration of multiple data sources. Plant Physiology 155(1), 271–281 (2011)
8. Gehrmann, T.: Conditional random fields for protein function prediction. M.sc. thesis, Delft University of Technology, Delft (2012)
9. Sutton, C., McCallum, A.: An Introduction to Conditional Random Fields (November 2010)
10. Collins, S.R., Kemmeren, P., Zhao, X.C., Greenblatt, J.F., Spencer, F., Holstege, F.C.P., Weissman, J.S., Krogan, N.J.: Toward a comprehensive atlas of the physical interactome of *Saccharomyces cerevisiae*. Molecular & Cellular Proteomics 6(3), 439–450 (2007)
11. Michael Ashburner, C.A.: Creating the gene ontology resource: design and implementation. Genome Research 11(8), 1425–1433 (2001)
12. Stark, C., Breitkreutz, B.J., Reguly, T., Boucher, L., Breitkreutz, A., Tyers, M.: BioGRID: a general repository for interaction datasets. Nucleic Acids Research 34(suppl. 1), D535–D539 (2006)
13. Stark, C., Su, T.C., Breitkreutz, A., Lourenco, P., Dahabieh, M., Breitkreutz, B.J., Tyers, M., Sadowski, I.: PhosphoGRID: a database of experimentally verified in vivo protein phosphorylation sites from the budding yeast Saccharomyces cerevisiae. Database 2010 (January 2010)
14. Gasch, A.: Megayeast expression dataset (August 2012), `http://gasch.genetics.wisc.edu/datasets.html`
15. Szklarczyk, D., Franceschini, A., Kuhn, M., Simonovic, M., Roth, A., Minguez, P., Doerks, T., Stark, M., Muller, J., Bork, P., Jensen, L.J., von Mering, C.: The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored. Nucleic Acids Research 39(database issue), D561–D568 (2011)