Chapter 3

Predicting Proteins Functions Using Collaborative Functions

Based on

Hossein Rahmani, Hendrik Blockeel and Andreas Bender, “Collaboration-based function prediction in protein-proein interaction networks”, In: Proceedings of the 10th International Symposium on Advances in Intelligent Data Analysis: 318-327, Lecture Notes in Computer Science 7014 Springer (2011)

3.1 abstract

The cellular metabolism of a living organism is among the most complex systems that man is currently trying to understand. Part of it is described by so-called protein-protein interaction (PPI) networks, and much effort is spent on analyzing these networks. In particular, there has been much interest in predicting certain properties of nodes in the network (in this case, proteins) from the other information in the network. In this paper, we are concerned with predicting a protein's functions. Many approaches to this problem exist. Among the approaches that predict a protein's functions purely from its environment in the network, many are based on the assumption that neighboring proteins tend to have the same functions. In this work we generalize this assumption: we assume that certain neighboring proteins tend to have "collaborative", but not necessarily the same, functions. We propose a few methods that work under this new assumption. These methods yield better results than those previously considered, with improvements in F-measure ranging from 3% to 17%. This shows that the commonly made assumption of homophily in the network (or "guilt by association"), while useful, is not necessarily the best one can make. The assumption of collaborativeness is a useful generalization of it; it is operational (one can easily define methods that rely on it) and can lead to better results.

3.2 Introduction

In recent years, much effort has been invested in the construction of protein-protein interaction (PPI) networks [118]. Much can be learned from the analysis of such networks with respect to the metabolic and signalling processes present in an organism, and the knowledge gained can also be prospectively employed e.g. to predict which proteins are suitable drug targets, according to an analysis of the resulting network [72]. One particular machine learning task that has been considered is predicting the functions of proteins in the network.

A variety of methods have been proposed for predicting the functions of proteins. A large class of them relies on the assumption that interacting proteins tend to have the same functions (this is sometimes called "guilt by association"; it is also related to the notion of homophily, often used in other areas). In this paper we investigate a generalized version of this notion. We rely on the fact that topologically close proteins tend to have *collaborative* functions, not necessarily the same functions. We define collaborative functions as pairs of functions that frequently interface with each other in different interacting proteins. In this way, the assumption becomes somewhat tautological (this definition of collaborative functions implies that the assumption cannot be wrong), but the question remains whether one can, through analysis of PPI networks, correctly identify collaborative functions, and how much gain in predictive accuracy can be obtained by this.

We propose two methods that predict protein functions based on function collaboration. The first method calculates the collaboration value of two functions using an iterative reinforcement strategy; the second method adopts an artificial neural network for this purpose. We perform a comprehensive set of experiments that reveal a significant improvement of F-measure values compared to existing methods.

The rest of paper is organized as follows. Section 2 briefly reviews approaches that have been proposed before to solve this problem. We present the proposed collaboration-based methods in Section 3, and evaluate them in Section 4. Section 5 contains our conclusions.

3.3 Related work

Various approaches have been proposed for determining the protein functions in PPI networks. A first category contains what we could call structure-based methods. These rely on the local or global structure of the PPI network. For instance, Milenkovic et al. [78] describe the local structure around a node by listing for a fixed set of small graph structures ("graphlets") whether the node is part of such a graphlet or not. Rahmani et al. [98] describe nodes by indicating their position in the network relative to specific important proteins in the network, thus introducing information about the global graph structure.

The above methods do not use information about the functions of other nodes to predict the functions of a particular protein. Methods that do use such information form a second category. A prototypical example is the Majority Rule approach [111]. This method simply assigns to a protein the k functions that occur most frequently among its neighbors (with k a parameter). One problem with this approach is that it only considers neighbors of which the function is already known, ignoring all others. This has been alleviated by introducing global optimization-based methods; these try to find global function assignments such that the number of interacting pairs of nodes without any function in common is minimal [121, 119]. Another improvement over the original Majority Rule method consists of taking a wider neighborhood into account [18]. Level k interaction between two proteins means that there is a path of length k between them in the network. Proteins that have both a direct interaction and shared level-2 interaction partners have been found more likely to share functions [18]. Taking this further, one can make the assumption that in dense regions (subgraphs with many edges, relative to the number of nodes) most nodes have similar functions. This has led to Functional Clustering approaches, which cluster the network (with clusters corresponding to dense regions), and subsequently predict the functions of unclassified proteins based on the cluster they belong to [57, 13].

A common drawback of the second category of approaches is that they rely solely on the assumption that neighboring proteins tend to have the same functions. It is not unreasonable to assume that proteins with one particular function tend to interact with proteins with specific other functions. We call such functions "collaborative" functions. Pertinent questions are: can we discover such collaborative functions, and once we know which functions tend to collaborate, can we use this information to obtain better predictions? The methods we propose next, try to do exactly this.

3.4 Two collaboration-based methods

We propose two different methods. Each of them relies on the assumption that interacting proteins tend to have collaborative functions. They try to estimate from the network which functions often collaborate and, next, try to predict unknown functions of proteins using this information.

In the first method, first we extract the collaborative function pairs from the whole network. Then, in order to make prediction for an unclassified protein, we extract the candidate functions based on position of the protein in the network. Finally, we calculate the score of each candidate function. High score candidate functions are those which collaborate more with the neighborhood of unclassified protein. The second method adopts a neural network for modeling the function collaboration in PPI networks.

We use the following notation and terminology. The PPI network is represented by protein set P and interaction set E. Each $e_{pq} \in E$ shows an interaction between two proteins $p \in P$ and $q \in P$. Let F be the set of all the functions that occur in the PPI network. Each classified protein $p \in P$ is annotated with an $|F|$-dimensional vector FS_p that indicates the functions of this protein: $FS_p(f_i)$ is 1 if $f_i \in F$ is a function of protein p, and 0 otherwise. FS_p can also be seen as the set of all functions f_i for which $FS_p(f_i) = 1$. Similarly, the $|F|$-dimensional vector NB_p describes how often each function occurs in the neighborhood of protein p. $NB_p(f_i) = n$ means that among all the proteins in the neighborhood of p, n have function f_i. The neighborhood of p is defined as all proteins that interact with p.

3.4.1 A Reinforcement Based Function Predictor

Consider the Majority Rule method. This method considers as *candidate functions* (functions that might be assigned to a protein) all the functions that occur in its neighborhood, and *ranks* them according to their frequency in that neighborhood (the most frequent ones will eventually be assigned).

Our method differs in two ways. First, we consider extensions of Majority Rule's candidate functions strategy. Instead of only considering functions in the direct neighborhood as candidates, we can also consider functions that occur at a distance at most k from the protein. We consider $k = 1, 2, 3, 4$ and call these strategies First-FL (First function level, this is Majority Rule's original candidate strategy), Second-FL, Third-FL and Fourth-FL. Finally, the All-FL strategy considers all functions as candidate functions.

The second difference is that our method ranks functions according a "function collaboration strength" value, which is computed through iterative reinforcement, as follows. Let $FuncColVal(f_i, f_j)$ denote the strength of collaboration between functions f_i and f_j. We consider each classified protein $p \in P$ in turn. If function f_j occurs in the neighborhood of protein p (i.e., $NB_p(f_j) > 0$) then we increase the collaboration value between f_j and all the functions in FS_p:

$$\forall f_i \in FS_p : FuncColVal(f_i, f_j) += \frac{NB_p(f_j) * R}{support(f_j)}$$

If $NB_p(f_j) = 0$, we decrease the collaboration value between function f_j and all the functions belonging to FS_p:

$$\forall f_i \in FS_p : FuncColVal(f_i, f_j) -= \frac{P}{support(f_j)}$$

$support(f_j)$ is the total number of times that function f_j appears on the side of an edge e_{pq} in the network. R and P are "Reward" and "Punish" coefficients determined by the user.

Formula (3.1) assigns a collaboration score to each candidate function f_c:

$$Score(f_c) = \sum_{\forall f_j \in F} NB_p(f_j) * FuncColVal(f_j, f_c) \tag{3.1}$$

High score candidate function(s) collaborate better with the functions observed in the neighborhood of p and are more likely to be predicted as p's functions.

We call the above method “Reinforcement based collaborative function prediction” (RBCFP), as it is based on reinforcing collaboration values between functions as they are observed.

3.4.2 SOM Based Function Predictor

The second approach presented in this work employs an artificial neural network, and is inspired by self-organizing maps (SOMs). From the PPI network, a SOM is constructed as follows. We make a one-layered network with as many inputs as there are functions in the PPI network, and equally many output neurons. Each input is connected to each output. The network is trained as follows. All weights are initialized to zero. Next, the training procedure iterates multiple times over all proteins in the PPI network. Given a protein p with function vector FS_p and neighborhood vector NB_p, the network's input vector is set to NB_p, and for each j for which $FS_p(f_j) = 1$, the weights of the j'th output neuron are adapted as follows:

$$W_{ij,New} = W_{i,j,Current} + LR * (NB_p(j) - W_{i,j,Current}) \tag{3.2}$$

where W_{ij} is the weight of the connection from input i to output j, and LR (learning rate) is a parameter. Intuitively, this update rule makes the weight vector $W_{.j}$ of output j gradually converge to a vector that is representative for the NB vectors of all proteins that have f_j as one of their functions. Once the network has been trained, predictions will be made by comparing the NB vector of a new protein q to the weight vectors of the outputs corresponding to candidate functions, and predicting the k

functions for which the weight vector is closest to NB_q (using Euclidean distance), with k a parameter determined by the user.

Normally, in a SOM, the weights of the winner neurons (the output neurons whose weights are closest to the input) and that of neurons close to them in the SOM lattice are adjusted towards the input vector. The difference with our method is that our learning method is supervised: we consider as "winner neurons" all output neurons corresponding to the functions of the protein. As usual in SOMs, the magnitude of the weight update decreases with time and with distance from the winner neuron. Here, we take some new parameters into consideration which are $LearningRate(LR)$, $DecreasingLearningRate(DecLR)$ and $TerminateCriteria(TC)$ parameters. LR determines how strongly the weights are pulled toward the input vector, and $DecLR$ determines how much LR decreases with each iteration. TC determines when the training phase of SOM terminates: it indicates the minimum amount of change required in one iteration; when there is less change, the training procedure stops.

Algorithm 3.4.1 summarizes the Training phase of the SOM method.

```
Algorithm 3.4.1: SOM TRAINING PHASE($LR, DecLR, TC$)

procedure SOM-TRAINING($LR, DecLR, TC$)
  $maxChangeInNetworkWeights \leftarrow 0$;
  repeat
    for each $classified\ protein\ p \in P$
      do
        $build\ \ NB_p$
        for each $f_i \in F$
          do $inputNeuron(i) = NB_p(f_i)$
        for each $f_j \in FS_p$
          do
            $apply\ Formula$ (3.2).
            $update\ \ maxChangeInNetworkWeights$
    $LR \leftarrow LR * DecLR$
  until ($maxChangeInNetworkWeights < TC$)
```

3.5 Experiments

3.5.1 Dataset and Annotation Data

We apply our method to three *S. cerevisiae* PPI networks: DIP-Core [25], VonMering [123] and Krogan [59] which contain 4400, 22000 and 14246 interactions among 2388, 1401 and 2708 proteins respectively. The protein function annotation for *S. cerevisiae* PPI networks were obtained from the Yeast Genome Repository [39]. Functions can be described in different levels of detail. For example, two functions 11.02.01 (rRNA synthesis) and 11.02.03 (mRNA synthesis) are considered the same up to the second

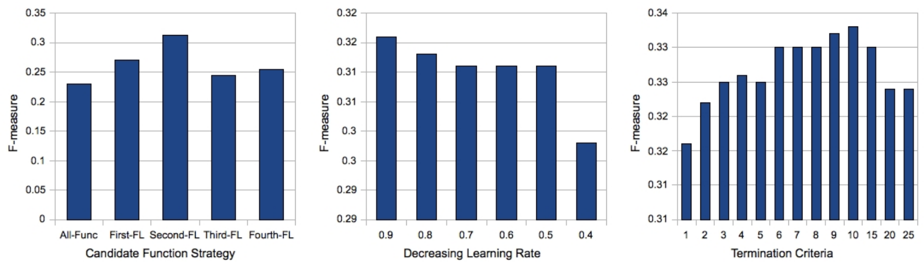

Figure 3.1: Effects of tuning the “Candidate function strategy" parameter, the “Decreasing Learning Rate”, and the “Termination Criteria” of SOM network in Krogan Dataset. Second-FL with DecLR=0.9 and TC=10 produces the best result on Krogan.

function level (i.e., 11.02 = RNA synthesis), but not on deeper levels. The function hierarchy we use contains five different levels, which we will refer to as F-L-i. Thus, for each dataset, five different versions can be produced, one for each function level.

3.5.2 Parameter Tuning

Our methods have parameters for which good values need to be found. Parameters can be tuned by trying out different values, and keeping the one that performed best. Obviously, such tuning carries a risk of overestimating the performance of the tuned method, when it is evaluated on the same dataset for which it was tuned. To counter this effect, we tuned our methods on the Krogan dataset labeled with F-L-1 functions (the most general level of functions in the function hierarchy), and evaluated them with the same parameter values on the other datasets; results for DIP-Core and Von Mering are therefore unbiased. Conversely, for the Krogan dataset, we used parameter settings tuned on DIP-Core. This way, all the results are unbiased.

We tuned the parameters manually, using the following simple and non-exhaustive strategy. Parameters were tuned one at a time. After finding the best value for one parameter, it was fixed and other parameters were tuned using that value. For parameters not yet fixed when tuning a parameter p, we tried multiple settings and chose a value for p that appeared to work well on average. With this approach, the order in which the parameters are tuned can in principle influence the outcome, but we found this influence to be very small in practice.

Fig. 3.1 shows the effects of the consecutive tuning of the different parameters. The best value for “Candidate function strategy" parameter is Second-FL; using this value we found a best DecLR value of 0.9, and using these two values we found an optimum for TC at 10. For LR the default setting of 1 was used.

“Candidate function strategy” = Second-FL, $R = 1$ and $P = 2$ turns out to be the best parameter setting for the RBCFP method.

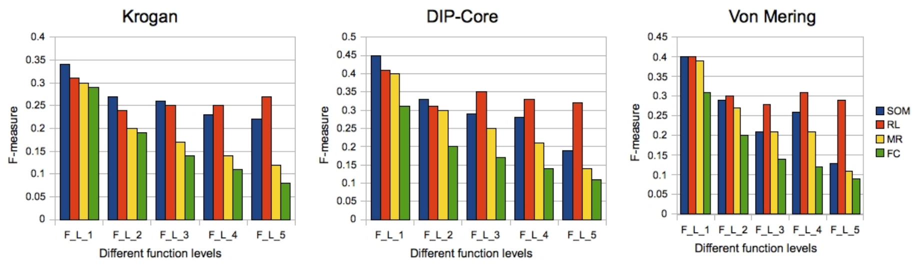

Figure 3.2: F-measures obtained by the new collaboration-based methods (SOM and RBCFP), compared to existing similarity-based methods (MR and FC) at five different function levels, for the Krogan, DIP-Core, and VonMering datasets. On all levels, collaboration based methods predict functions more accurately than similarity based methods.

3.5.3 Comparison to previous methods

In this section, we compare our collaboration-based methods (RBCFP and SOM) with similarity-based methods (Majority Rule and Functional Clustering) on the Krogan, VonMering and DIP-Core datasets, using average F-measure as the evaluation criterion. We perform a leave-on-out cross-validation, leaving out one protein at a time and predicting its functions from the remaining data. For each protein, we predict a fixed number of functions, namely three; this is exactly what was done in the Majority Rule approach we compare to, so our results are maximally comparable. In the proposed methods, we use the parameter values tuned in the previous section. Majority Rule (MR) selects the three most frequently occurring functions in the neighborhood of the protein in the network. Functional clustering (FC) methods differ mainly in their cluster detection technique. Once a cluster is obtained simple methods are usually used for function prediction within the cluster. In our evaluation, we use the clusters from [39] (which were manually constructed by human experts).

Fig. 3.2 compares collaboration-based and similarity-based methods on the Krogan, DIP-Core and VonMering datasets respectively. F-L-i refers to the i'th function level in the function hierarchy. We compare the methods on five different function levels.

In all three datasets, collaboration based methods predict functions more accurately than similarity based methods. As we consider more detailed function levels, the difference between their performance increases. In order to have a general idea about the performance of two method types in different function levels, we take the average of F-measure difference between collaboration based methods and similarity based methods in three datasets. Fig. 3.3 shows the average F-measure difference between two method types. For general function descriptions (first and second function levels), collaboration based methods outperform the similarity based methods with some 4 percent. For more specific function descriptions, for example function level 5,

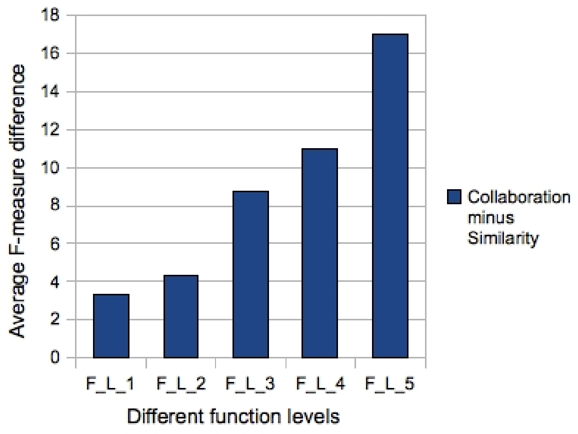

Figure 3.3: Difference in F-measure between the best collaboration-based and the best similarity-based method, averaged over three datasets, for five function levels. The difference increases as we consider more detailed function levels.

the performance difference between two methods increases up to 17 percent.

3.5.4 Extending Majority Rule

We identified the notion of collaboration-based prediction (as supposed to similarity-based prediction) as the main difference between our new methods and the ones we compare with. However, in the comparison with Majority Rule, there is another difference: while Majority Rule assigns only functions from the direct neighborhood to a protein, we found that using candidate functions from a wider neighborhood (including neighbors of neighbors) was advantageous. This raises the question whether majority rule can also be improved by making it consider a wider neighborhood.

We tested this by extending the Majority Rule so that it can consider not only direct neighbors, but also neighbors at distance 2 or 3. We refer to these versions as MR(NB-Li). Fig. 3.4 shows the effect of considering a wider neighborhood in Majority Rule in the three datasets Krogan, VonMering and DIP-Core. There is no improvement in the Krogan and VonMering datasets, and only a small improvement (1%) in DIP-Core, for MR(NB-L2). This confirms that the improved predictions of our methods are due to using the new collaboration-based scores, and not simply to considering functions from a wider neighborhood.

3.6 Conclusion

To our knowledge, this is the first study that considers function collaboration for the task of function prediction in PPI networks. The underlying assumption behind our

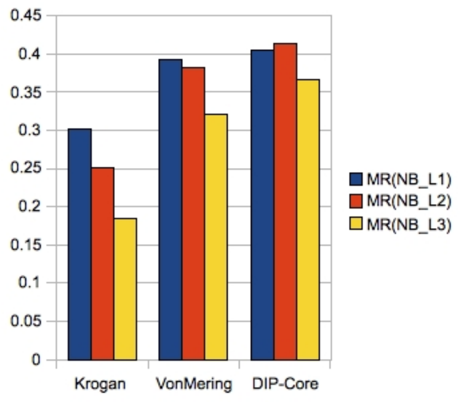

Figure 3.4: Extending Majority Rule by considering other function neighborhood levels. NB-L*i* represents the 1-, 2- or 3-neighborhood of the protein.

approach is that a biological process is a complex aggregation of many individual protein functions, in which topologically close proteins have collaborative, but not necessarily the same, functions. We define collaborative functions as pairs of functions that frequently interface with each other in different interacting proteins.

We have proposed two methods based on this assumption. The first method rewards the collaboration value of two functions if they interface with each other in two sides of one interaction and punishes the collaboration value if just one of the functions occurs on either side of an interaction. At prediction time, this method ranks candidate functions based on how well they collaborate with the neighborhood of unclassified protein. The second method uses a neural network based method for the task of function prediction. The network takes as input the functions occurring in a protein's neighborhood, and outputs information about the protein's functions.

We selected two methods, Majority Rule and Functional Clustering, as representatives of the similarity based approaches. We compared our collaboration based methods with them on three interaction datasets: Krogan, DIP-Core and VonMering. We examined up to five different function levels and we found classification performance according to F-measure values indeed improved, sometimes by up to 17 percent, over the benchmark methods employed. Regarding the relative performance of the proposed methods, their classification performances are similar in the high level function levels but the RBCFP method outperforms the SOM method in more detailed function levels.

Our results confirm that the notion of collaborativeness of functions, rather than similarity, is useful for the task of predicting the functions of proteins. The information about which functions collaborate, can be extracted easily from a PPI network, and using that information leads to improved predictive accuracy.

These results may well apply in other domains, outside PPI networks. The notion of homophily is well-known in network analysis; it states that similar nodes are more likely to be linked together. The notion of collaborativeness, in this context, could also be described as “selective heterophily”. It remains to be seen to what extent this notion may lead to better predictive results in other types of networks.

Acknowledgements

This research is funded by the Dutch Science Foundation (NWO) through a VIDI grant. At the time this research was performed, Andreas Bender was funded by the Dutch Top Institute Pharma, project number: D1-105.