Completion of Biological Networks

with Output Kernel Trees

Seminar on computational systems biology
Department of Computer Science
University of Helsinki

Cheng Lu
28.2.2007

1. Introduction

Elucidating biological networks between proteins appears nowadays as one of the most important challenge in systems biology. Computational approaches to this problem are important complement experimental technologies and to help biologists in designing new experiments. [1]

We know some interactions in a group of proteins, and we put an edge between these two proteins if there is a interaction between them. In this way we can form a protein-protein interaction network.

The problem we are facing is how to predicate whether two new proteins will interact with each other from the protein-protein interaction network we know. Fig-0 is a simple protein-protein interaction network, where points denote proteins and edges denote interaction between those two proteins. From the network, we want to predicate whether protein 4 and 5 will interact with each other.

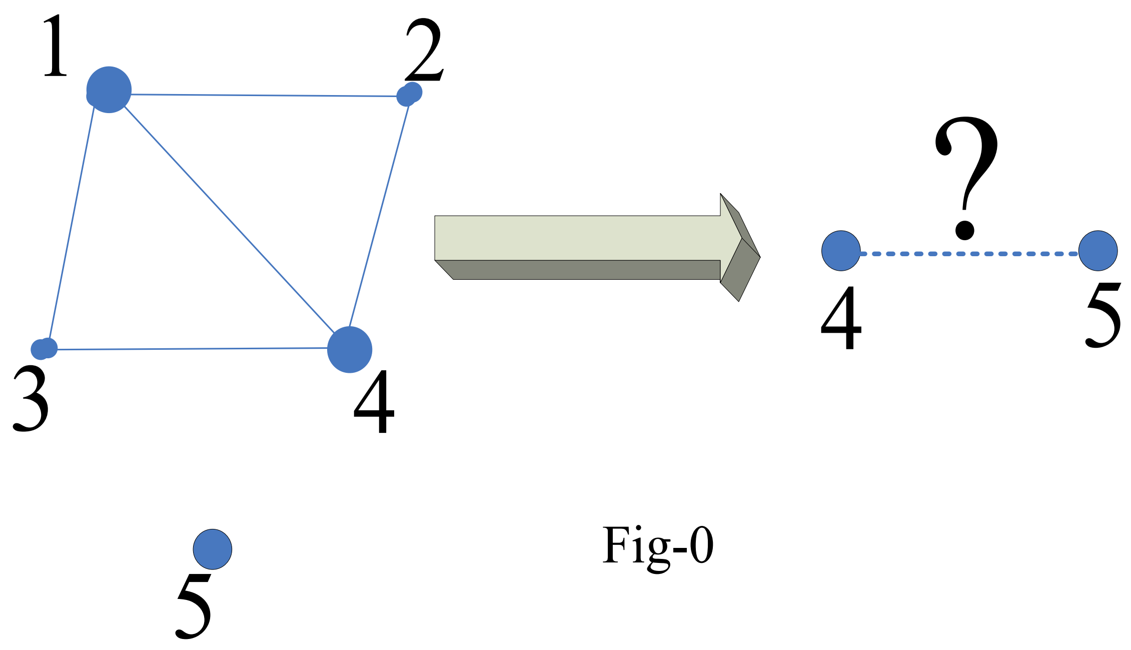

Fig-0

Here I will introduce using output kernel tree method [2] to perform an experiment on a network of protein-protein interactions.

2. Supervised network inference

In this section we will introduce some basic things about protein-protein network, such as symbolization of the network, concept of kernel and diffusion kernel. With these knowledge, it is possible for us to find some function to predicate the possibility of an interaction between two proteins.

Let G = (V,E) be an undirected graph with vertices V and edges $E \subset V \times V$. $|V| = m$ is the number of nodes in the graph. Each vertex means a protein and an edge means there is a interaction between those two proteins.

Let x(v) denotes the feature vector for the vertex in some feature space. So x(v) tell us all the features of protein v.

The goal of the graph inference is to determine from the knowledge of G a function $e(x(v), x(v')) : V \times V \rightarrow \{0,1\}$, ideally such that $e(x(v), x(v')) = 1 \Leftrightarrow (v, v') \in E$, where *v* and *v’* are two arbitrary vertices.

To solve this problem let us know something about kernel. A kernel is defined as a function $k : V \times V \rightarrow R$ which induces some feature map ϕ into some Hilbert Space H, i.e. some function space, such that $k(v, v') = <\phi(v), \phi(v')>$, where $<,>$ means inner product.

To find the function e(x(v),x(v’)) we define a kernel *k*(v, v’) such that adjacent vertices lead to high values of *k* and non-adjacent ones lead to smaller ones. Here we use the diffusion kernel proposed in [3].

Diffusion kernel is $K = \exp(-\beta L)$, where $L = D - A$ is the Laplacian matrix of the graph, with *D* the diagonal matrix of node connectivity and *A* the adjacency matrix, and β is a user-defined parameter that controls the degree of smoothness of the kernel. Then we need to normalize the kernel matrix in order to make each kernel value in the matrix vary between 0 and 1.

Here is a simple example for calculating the diffusion kernel in the network in Fig-0.

$$\underset{D}{\begin{bmatrix} 3 & & & \\ & 2 & & \\ & & 2 & \\ & & & 4 \end{bmatrix}} - \underset{A}{\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}} = \underset{L}{\begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ -1 & -1 & -1 & 4 \end{bmatrix}}$$

Then we can use *L* to compute *K*, noticing that we should use matrix exponentiation.

Now if we want to predicate whether protein *i* will interact with a new protein *j*, we need to find an approximation of the kernel values between them by exploring their feature vector values.

Once we got the kernel value between two proteins, we can begin our prediction. If it is greater than some threshold, we predicate that these two protein will interact with each other, else we think they will not interact.

Fig-1 is based on the network of Fig-0. Assuming we manage to calculate the kernel value between protein 4 and 5, it says that if the kernel value is greater than the threshold we set, they will interact with each other.

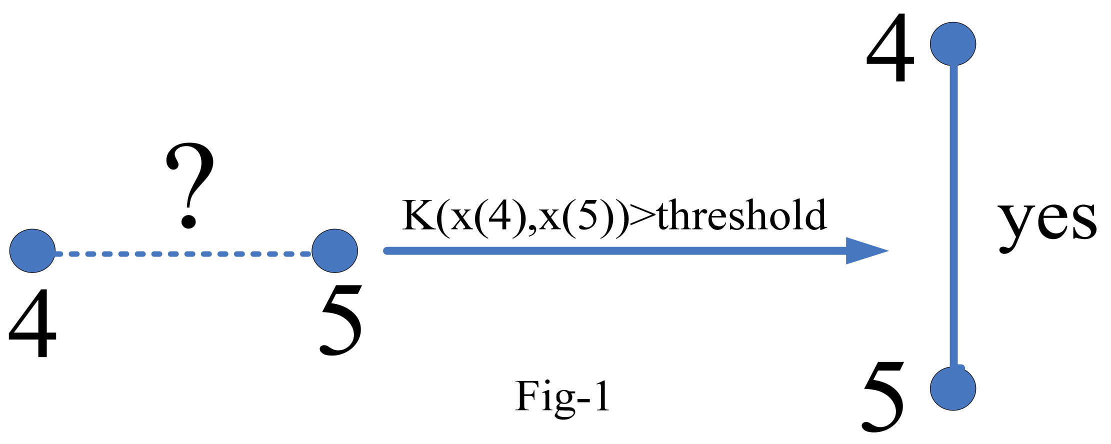

Fig-1

3. Output kernel trees

Output kernel trees (OK3, [4]) are a transformation of standard classification and classification and regression trees [5] that can handle any output space over which a kernel may be defined.

The idea of output kernel trees is to use the known proteins to construct a classification and regression tree. Using this tree, we can find the group for a new protein. Therefore we can use this group to estimate the behaviors of the new protein for the reason that they are similar.

3.1 Learning stage

We use CART algorithm [5] to construct the classification and regression tree.

The basic idea of CART is to recursively split the learning sample with binary tests on some feature of the protein, trying at each split to reduce as much as possible the variance of the output feature vector in the left and right subsamples of the learning cases corresponding to that split. The output feature vector is the function $\phi(v)$ mentioned in section 2, here we use the output y to represent it.
More specifically, we can formulate it as follows:

$$Score(T,S) = \mathrm{var}\{y \mid S\} - \frac{N_l}{N}\mathrm{var}\{y \mid S_l\} - \frac{N_r}{N}\mathrm{var}\{y \mid S_r\} \quad (3\text{-}1)$$

Where T is the split based on some feature of the proteins, S is the local learning sample of size N at the node to split, S_l and S_r is are its left and right child of size N_l and N_r after the split. $\mathrm{var}\{y \mid S\}$ denotes the empirical variance of the output feature vector in the subset S:

$$\mathrm{var}\{y \mid S\} = \frac{1}{N}\sum_{i=1}^{n}(y_i - \frac{1}{N}\sum_{i=1}^{n} y_i)^2 \qquad (3\text{-}2)$$

So at each step, we can calculate a score for each feature test. For example, if it is a boolean variable, we choose the feature test by check whether this variable is true; if it is a numerical variable, we choose all possible feature tests by varying the threshold. Then we choose the feature test which leads to the highest score. In this way we split the left subsample and right subsample until some stopping criteria is met. (e.g. the size of the local sample is lower than some threshold or all possible splits cause no significant variance reduction.

Once we managed to construct the tree, each leaf is labeled with a prediction $\hat{y}_L$ computed as:

$$\hat{y}_L = \frac{1}{N_L}\sum_{i=1}^{N_L} y_i \qquad (3\text{-}3)$$

Where N_L is the number of proteins that reach this leaf.

Here is a simple classification and regression tree:

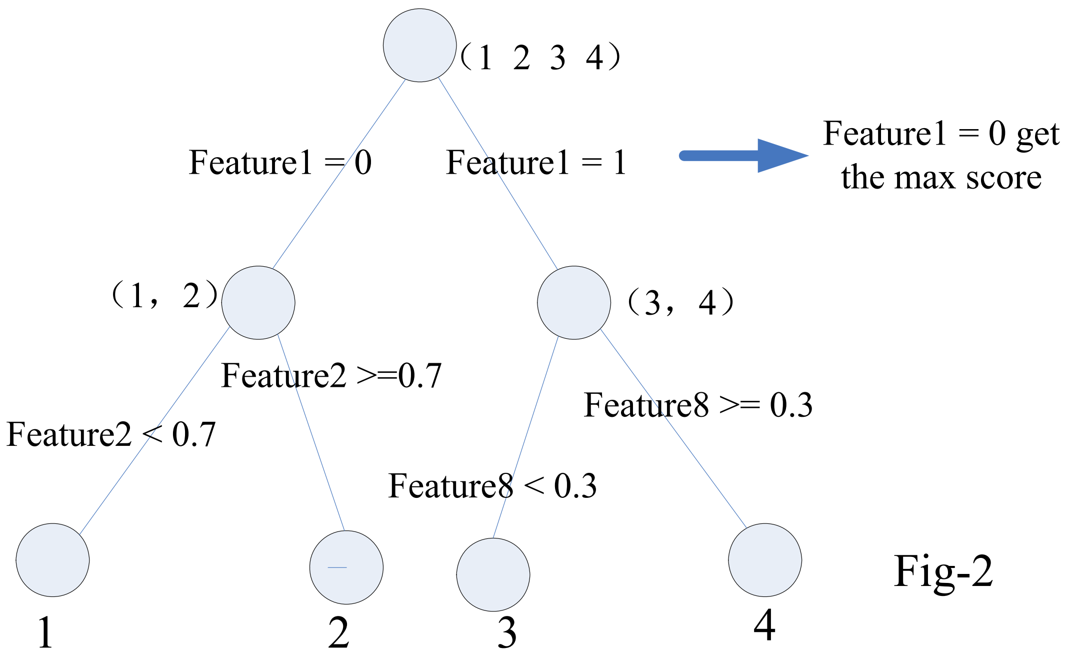

Fig-2

In fact we can not use the Equations (3-1), (3-2) and (3-3) because we do not know what the function ϕ is. However we can use kernel values to compute Equation (3-2), which makes us be able to construct the tree but fail to make the predicate like Equation (3-3). To make the variance more general, we rewrite Equation (3-2) in n-dimensional space:

$$\operatorname{var}\{y\,|\,S\} = \frac{1}{N}\sum_{i=1}^{n} \| y_i - \frac{1}{N}\sum_{i=1}^{n} y_i \|^2 \qquad (3\text{-}4)$$

We can use kernel trick, i.e., some kind of mathematical transformation, to compute Equation (3-4) as follows:

$$\operatorname{var}\{y\,|\,S\} = \frac{1}{N}\sum_{i=1}^{N} \| y_i - \frac{1}{N}\sum_{i=1}^{N} y_i \|^2 = \frac{1}{N}\sum_{i=1}^{N} k(v_i, v_i) - \frac{1}{N^2}\sum_{i,j=1}^{N} k(v_i, v_j) \qquad (3\text{-}5)$$

3.2 Prediction stage

As we have mentioned in the former section, we can not simply use Equation (3-3) to make a prediction for we do not know what the output feature function is.

However, our final goal is to calculate the kernel value between two proteins which are described by their feature vectors $x(v)$ and $x(v')$ separately.

Let us assume protein v_1 reaches leaf L_1, which contains vertices $\{v_1^1,\ldots\ldots,v_{N_1}^1\}$, and protein v_2 reaches leaf L_2, which contains vertices $\{v_1^2,\ldots\ldots,v_{N_2}^2\}$. Again we will use this kernel trick to calculate the kernel value between v_1 and v_2 as follows:

$$\hat{k}(v,v') = <\hat{\phi}_1,\hat{\phi}_2> = \frac{1}{N_1N_2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}<\phi(v_i^1),\phi(v_j^2)> = \frac{1}{N_1N_2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}k(v_i^1,v_j^2) \quad (3\text{-}6)$$

As we can see from (3-6), we are able to compute $\hat{k}(v,v')$ only using the kernel values between those proteins we know. And this kernel value prediction is what we want to find in section 2.

Here is the example for the network in Fig-0. Suppose 5 belongs to the leaf (1,2), 4 reaches the leaf (3,4), then we compute the kernel value between 5 and 4 as follows:

$$\begin{tikzcd} 5 \arrow[r] & (1,2) \arrow[dr] & \\ & & \frac{1}{2}\times\frac{1}{2}\times(k(1,3)+k(1,4)+k(2,3)+k(2,4)) \\ 4 \arrow[r] & (3,4) \arrow[ur] & \end{tikzcd}$$

Fig-3

4 results and discussion

4.1 Data

I carry out experiments on a protein-protein interaction network in the yeast *S.erevisiae,* using the method introduced in the former sections. This network is borrowed from [6] that consists of the high confidence interactions highlighted in [7]. There are 984 proteins and 2438 interactions in the network.

This network is smoothed by a diffusion kernel, where we set the parameter β of the diffusion kernel to 3.0.

4.2 Input features

Expression data (expr): 1×77 numerical vector from [8] for each protein.
Phylogenetic profiles (phy): 1×145 boolean vector from [9] for each protein.
Localization data (loc): 1×23 Boolean vector from [10].

4.3 ROC analysis

ROC, Receiver Operating Characteristic, is a graphical plot of the sensitivity vs. (1 - specificity) for a binary classifier system as its discrimination threshold is varied. The ROC can also be represented equivalently by plotting the fraction of true positives (TP) vs. the fraction of false positives (FP). [11]

We can gain a simple impression from Fig-4 about ROC. TP means the number of predictions which are true and positive; FP means false and positive, FN means false and negative, TN means true and negative.

$$P(TP)=\frac{TP}{TP+FN} \qquad P(FP)=\frac{FP}{FP+TN}$$

We can use AUC to judge our prediction. AUC is the area under the ROC curve. The bigger AUC is, the better our prediction is. For example, if AUC=1, it means all our prediction are correct.

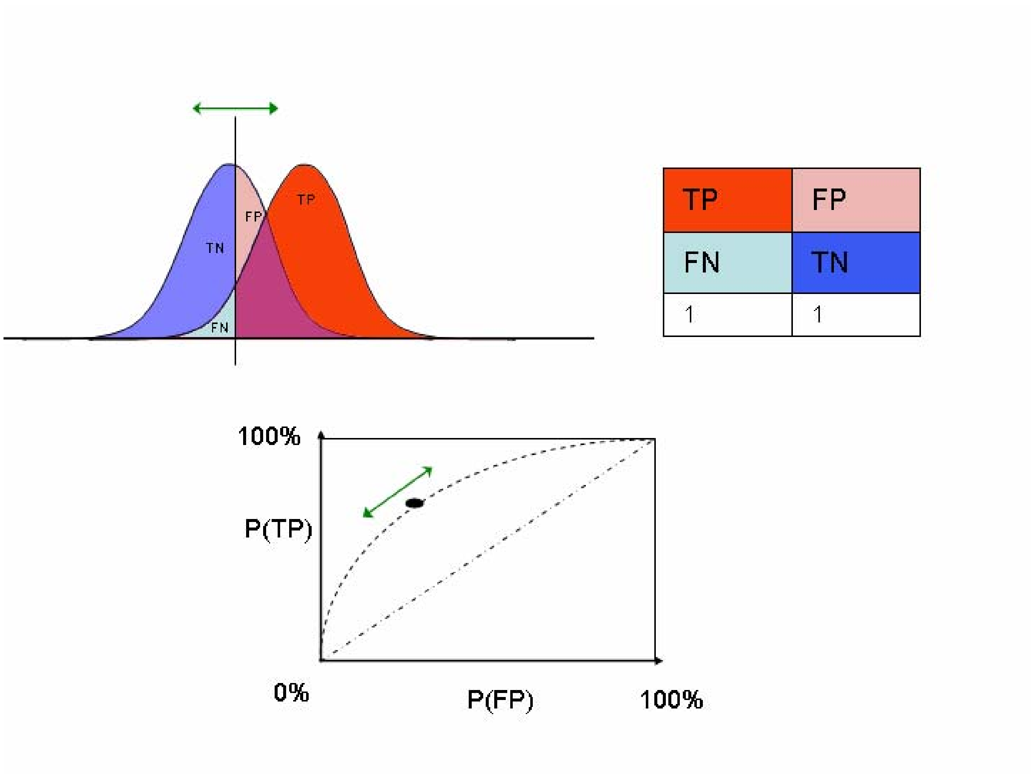

Fig-4

4.4 Results

Here we use ten-fold cross-validation. That is to say, split the whole sample in 10 fold, use 9 fold as training dataset and the remaining one as test dataset, and we do it 10 times by choosing different fold as test dataset.

Fig-5 are two ROC curves I produced, one using fold-4 and the other using fold-9.

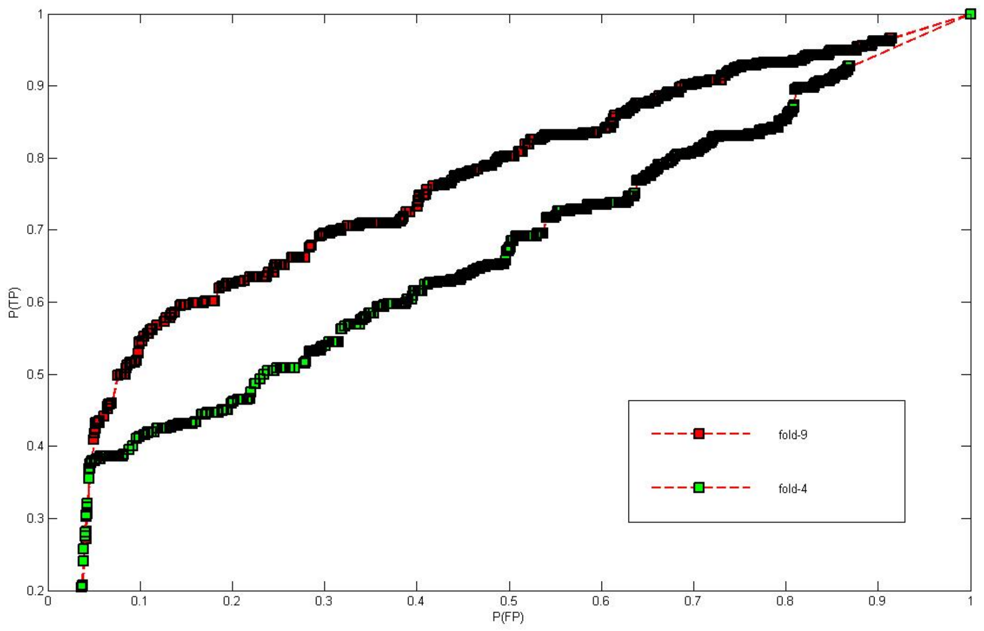

Fig-5

As we can see from the Fig-4, the result of fold-9 tell us our prediction is good in some sense, but the result of fold-4 gives us little information for half of the predictions are correct and half are wrong.

We can find that in the beginning and ending parts of the curves there are not so many points. That is because there are too many kernel values are 0. Imagine an extreme case where we have 9999 zeros and only 1 one in the adjacent matrix. The consequence is that even after diffusion a large number of points still have the same value in the diffusion kernel.

4.4 Further development

The prediction based on a single tree did not give very good result. A better method is to use ensemble trees to make the prediction. Then we need to construct many trees and assign a weight to each tree we use. For more information, please consult [1].

In this method each tree can be constructed using OK3 described in section 3, while randomizing the choice of the split at each node. To tree weight assigning, we can simply treat each tree with the same weight. More information about this method can be found in [12].

Reference:

[1] Pierre Geurts, Nizar Touleimat, Marie Dutreix, Florence d'Alche-Buc. Inferring biological networks with output kernel trees. *BMC Bioinformatics, PMSB06 special issue*, 2007 [also http://www.montefiore.ulg.ac.be/services/stochastic/pubs/2007/GTDD07/ 14.03.2007]

[2] Geurts, Pierre and Wehenkel, Louis and d'Alche-Buc, Florence. Kernelizing the Output of Tree-Based Methods. *Proceedings of the 23rd International Conference on Machine Learning*, pages 345—352 [also http://www.icml2006.org/icml_documents/camera-ready/044_Kernelizing_the_Outp.pdf 14.03.2007]

[3] Kondor R, Lafferty J: Diffusion kernels on graphs and other discrete input spaces. In *Proc. of the 19th International Conference on Machine Learning* 2002:315–322.

[4] Geurts P, Wehenkel L, d'Alch´e–Buc F: Kernelizing the output of tree-based methods. In *Proceedings of the 23rd International Conference on Machine Learning*. Edited by Cohen W, Moore A, ACM 2006:345–352.

[5] Breiman L, Friedman J, Olsen R, Stone C: *Classification and Regression Trees*. Wadsworth International 1984.

[6] Kato T, Tsuda K, Kiyoshi A: Selective integration of multiple biological data for supervised network inference. *Bioinformatics* 2005, 21(10):2488–2495.

[7] von Mering C, Krause R, Snel B, Cornell M, Oliver S, S F, P B: Comparative assessment of large-scale data sets of protein-protein interactions. *Nature* 2002, 417(6887):399–403

[8] Eisen M, Spellman P, Patrick O, Botstein D: Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci.* 1998, 95:14863–14868.
[9] Yamanishi Y, Vert JP: Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics* 2005, 21:i468–i477.

[10] Huh W, Falvo J, Gerke C, Carroll A, Howson R, Weissman J, O'Shea E: Global analysis of protein localization in budding yeast. Nature 2003, 425:686–691.

[11] ROC [http://en.wikipedia.org/wiki/Receiver_Operating_Characteristic 14.03.2007]

[12] Geurts P, Ernst D, Wehenkel L: Extremely randomized trees. *Machine Learning* 2006, 36:3–42.