Project Report - Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data[1]

Karen Sachs Omar Perez Dana Pe'er Douglas A. Lauffenburger Garry P. Nolan

***Abstract*—This report summarises out implementations, findings, results and conclusion for the project task of finding causal relationships among 11 different phospho-proteins and phospholipids. The dataset used is single cell flow cytometry data, and core probabilistic graphical models' theory is used to generate Bayesian network for the dataset.**

Implementation Environment Details

Data preprocessing was done using MATLAB. Java v1.8 was used for implementation and Eclipse Mars was used as the Integrated Development Environment (IDE). A web based tool named "Graph Online" (graphonline.ru) was used to obtain the structure from the adjacency matrix.

Data Preprocessing

Since the data was following log normal distribution therefore first we took log of the data so that it becomes gaussian in nature. The arity of each data point was changed to 2 and 3. For each data file, the mean (μ) and the standard deviation (σ) was computed. The data points that lied in between μ - σ and μ + σ were changed to 1 denoting high degree of closeness with respect to μ and all other data points were changed to 2 for arity equal to 2. Similar process was followed to discretize the data to arity equal to 3. A MATLAB script (Preprocessing_script.m) was written to discretize the data and is provided along with the report. The data files obtained were fed to the java program to estimate the structure.

An overview of the algorithm

Score based structure learning approach is used to estimate the biological network. The broad idea is to write a scoring function that would measure the quality of the structure in terms of fitting the data. The highest scoring network would then be considered as the estimated structure for the given data set.

After finalising BIC score as the scoring criteria, we considered an edgeless graph/empty graph. We defined the three possible operations to be adding an edge, deleting an edge and reversing the direction of an edge. Operations were performed iteratively and randomly and we kept on evaluating the score. If the score improved then we moved on to the next operation or else we discarded the last operation that degraded the score. This process was repeated for numerous steps and a structure was predicted. This approach is completely in sync with the proposal that we gave in homework 4.

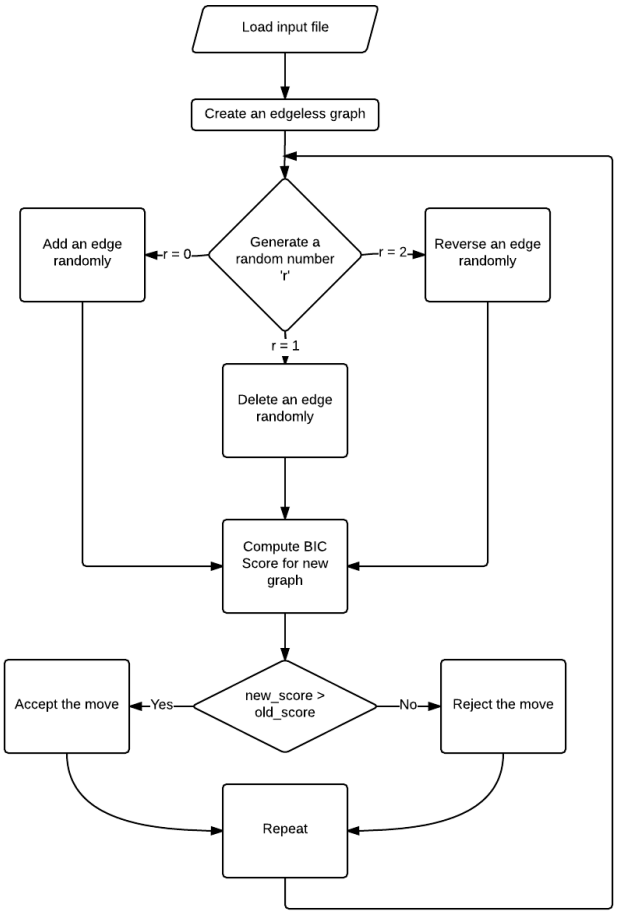


Fig. 1: Flowchart

CODE DESCRIPTION

A short description of the methods written in Structure_learning.java :

- Initialization - This method has no input arguments and it returns an arraylist of string datatype. This method is used to get the list of nodes of the structure to be predicted.
- Create_edgeless_graph - This method takes the list of nodes of the structure to be predicted as the input argument and returns an edgeless graph which is a hash map having string datatype as the key and arraylist of strings as the value which would represent the list of neighbours.
- Add_an_edge_randomly - This method takes the list of nodes, the bayesian network and the data as input and returns a new bayesian network if an edge was added randomly or returns the same old bayesian network without adding an edge if some conditions were violated.
- Delete_an_edge_randomly - This method takes the list of nodes, the bayesian network and the data as input and returns a new bayesian network if an edge was deleted randomly or returns the same old bayesian network without deleting an edge if some conditions were violated.
- Reverse_an_edge_randomly - This method takes the list of nodes, the bayesian network and the data as input and returns a new bayesian network if an edge was reversed randomly or returns the same old bayesian network without reversing an edge if some conditions were violated.
- Calculate_score - This method takes the list of nodes, the bayesian network and the data as input and returns the BIC score of the network.

A precise description of the utility code:

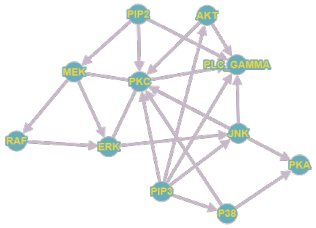
1) List_to_matrix method - This method takes the bayesian network in the form of adjacency list and returns a corresponding adjacency matrix.
2) Read_excel.java - This file is used to read the excel data files and gives the corresponding data array.
3) Check_cycle_matrix.java - This file contains the code for checking for a cycle given an adjacency list using DFS traversal.

RESULTS AND FINDINGS

For each of the data file 10 runs were recorded and adjacency matrix for each run was obtained. All the matrices were used to get a single probability matrix consisting of the probability of presence of an edge in between two nodes. A threshold of 0.5 was used and all the edges with probability greater than 0.5 were marked in the final matrix. This final matrix was then used to obtain the structure of the graph.

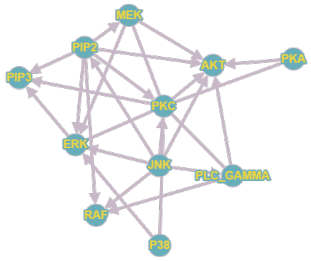
The process was repeated for all the 14 data files which are provided along with this report and the corresponding structures are presented below.

File 1

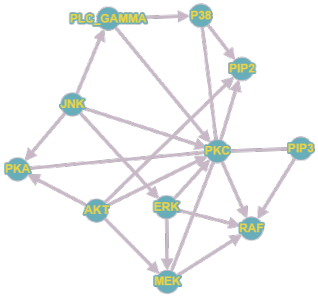


File 2

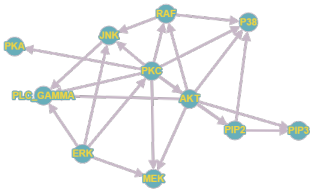
File 3



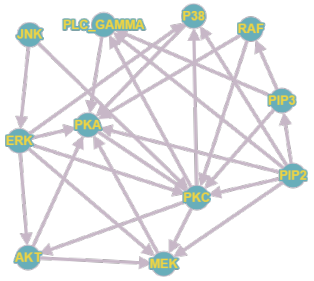
File 4



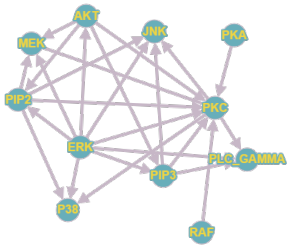
File 5



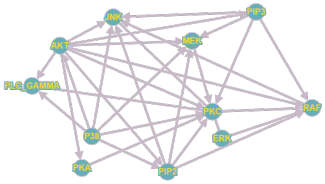
File 6



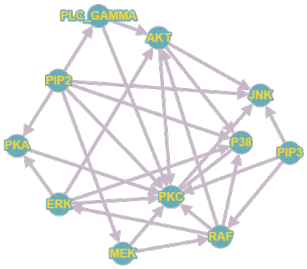
File 7



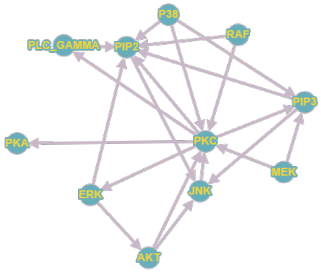
File 8



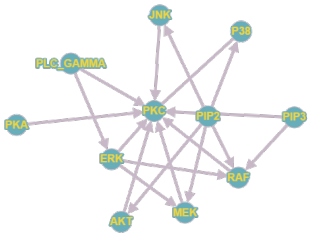
File 9



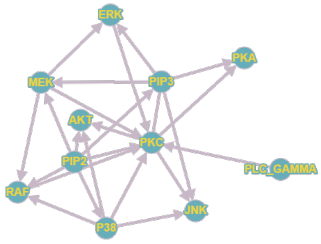
File 10



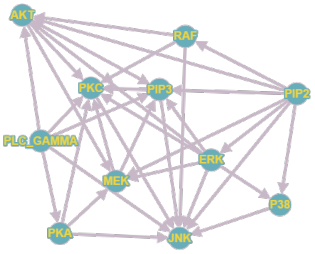
File 11



File 12



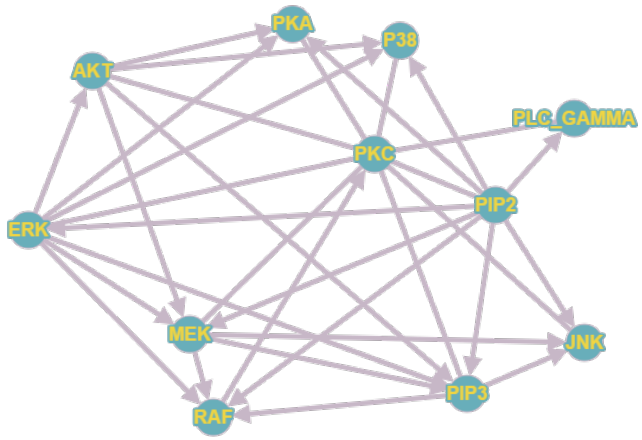
File 13



File 14

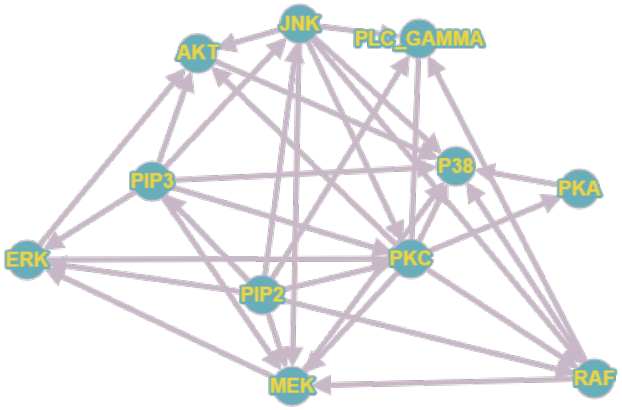
Generated Graphs for individual files with arity 2

After obtaining individual structures, the code was modified to obtain the averaged structure and this was achieved by obtaining adjacency matrix for each file. The adjacency matrices were added for all data files and this task was performed for 10 iterations. The adjacency matrix consisting of sum of all sub-matrices was then averaged to get the probability matrix with threshold 0.5. This was done using Java. This matrix was then transformed into structure and the result is given below.



Averaged out graph generated by java code using threshold probability = 0.5 with arity = 2

MATLAB was then used to dump the entire data into 1 file and it was discretized to high(1) and low(2) levels on the basis of mean and standard deviation. This data was then fed to the program to obtain the sum adjacency matrix which was then averaged for 10 runs to get the probability matrix with threshold 0.5. The probability matrix was then fed to the online tool for generating the graph.

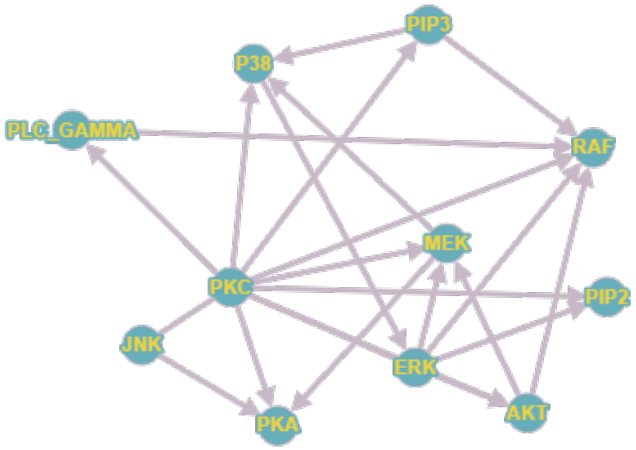


Averaged out graph generated by MATLAB using threshold probability = 0.5 with arity = 2

The above stated process was repeated with the only modification that the arity of data was changed to 3.

Now we had three levels namely low, medium and high. The corresponding structures predicted by using Java and MATLAB are given below and they are followed by the conclusion. The threshold probability of accepting an edge in between two nodes is locked at 0.5.

Averaged out graph generated by java code using threshold probability = 0.5 with arity = 3



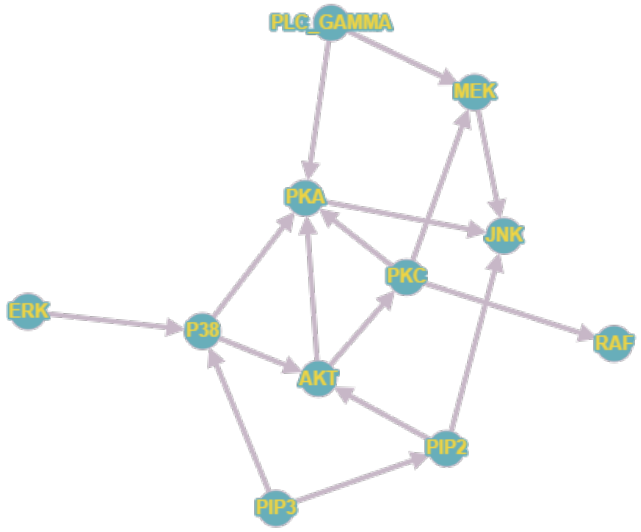
Averaged out graph generated by MATLAB using threshold probability = 0.5 with arity = 3

CONCLUSION

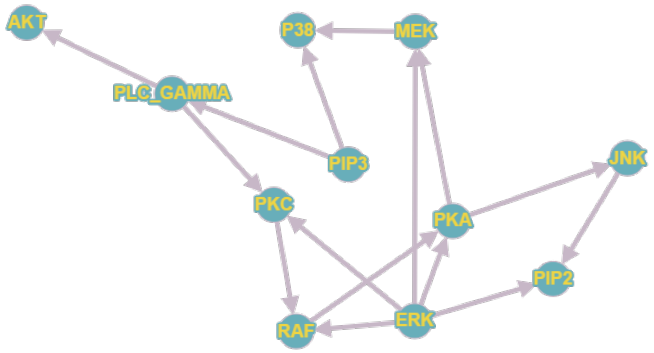
To conclude, we present the following two graphs as results. The first network corresponds to the the dataset discretized at arity 2 and the second network at arity 3. The number of edges in the first network predicted by our algorithm is 14, while it is 16 in the second network.

We also did a one to one analysis of the edges predicted in the paper to the ones obtained by our algorithm. For the first network with arity 2, PKC-PKA, PKC-RAF, PKA-JNK and PIP3-PIP2 edges were captured exactly, while he edge PKA-P38 was observed with revered causality. Also, one of the causality relation between ERK-AKT was seen in our predicted structure as as an indirect causality.

For the second network structure with arity 3, the edges PKC-RAF, PKA-MEK were captured correctly, while the edges ERK-PKA, RAF-PKA, PIP3-PIC-GAMMA were seen to be reversed.



Bayesian network with arity 2



Bayesian network with arity 3

With all the results presented above, we observed that some of the causal relations pertaining in the nature were easily observed in any approach we tried to model. For instance, PKC was seen to play a strong role in almost all the Bayesian Networks generated, be it with the log data discretized into two levels or three levels. Also, we tried with various variations, like using the entire dataset or using it file by file to generate the network.

Also the MAPK motif which is having prime biological importance is observed to be conserved in the averaged out networks.

REFERENCES

[1] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan, "Causal protein-signaling networks derived from multiparameter single-cell data," *Science*, vol. 308, no. 5721, pp. 523–529, 2005.