Gene Regulatory Network from Cancer Datasets

by

Ankani Chattoraj and Aparajita Khan

Project Adviser : Professor Mita Nasipuri

Computer Science and Engineering Department
Jadavpur University

Motivation

Gene Regulatory Networks (*GRNs*) are the most important organizational level in the cell where signals from the cell state and the outside environment are integrated in terms of activation and inhibition of genes. Revolution in molecular biology research has led to availability of complete genome sequences and DNA microarray experiments can simultaneously report gene expression levels at a genome-wide scale. With such an outburst of biological information the target now is to reverse engineer [1] and elucidate the system structure by reasoning backwards from expression level observations.

Cancer is a disease not of single genes, but rather of genomes and/or networks of molecular interaction and control. Constructing *GRN* (*Gene Regulatory Network*) in healthy and diseased tissues is critical to understand cancer phenotypes, devising effective therapeutics and prioritizing drug targets [2].

Work Overview

Target :

In this work we are trying to obtain a Gene Regulatory Network from Gene Expression Data of subjects affected with Cancer.

Step 1 :

Each instance of gene expression for various genes, corresponding to a particular subject in the dataset, takes real values. So we first set expression levels from the available data for each instance [3]. The values within 25^{th} *percentile* of the data are set as *level* 1 and those taking values from 25^{th} *percentile* to 50^{th} *percentile* are set as *level* 2 , values from 50^{th} *percentile* to 75^{th} *percentile* are given *level* 3 and values greater than 75^{th} *percentile* are set *level* 4. We then replace the real values by these *levels* in the data instances.

Step 2 :

We then apply the *Sparse Candidate Algorithm* [4] on the modified dataset and also incorporate certain innovative manipulations of our own in the algorithm mentioned. Each gene is treated as a node (*Discrete Random Variable*) which can take *four values* as per the levels defined.

This algorithm in its initial step uses *mutual information* between each pair of nodes to select k *Candidate Parents*, where k is any fixed natural number deciding the maximum number of parents would we prefer each node to have. So, for each node X_i we separately compute the *mutual information* value with other nodes i.e $MI(X_i\,,\,X_j)$ *where* $i \neq j$, arrange them in descending order and select k nodes giving the highest MI values with respect to node X_i. These k nodes are the *Candidate Parents* for node X_i. Repeating this process for each node we generate a directed graph which is expected to have many cycles.

Step 3 :

Now we remove cycles from the network as our aim is to obtain a *Bayesian Network* and hence the graph structure should be a *DAG*.

To remove cycles we start with a random node and as soon as a cycle is detected, say $X_l \rightarrow X_m \rightarrow X_n \rightarrow X_l$ we check the parent of each node involved in the cycle and rank their positions in terms of previous MI values computed. For example, in our cycle,

$$Parent(X_l) = X_n \ , \ Parent(X_m) = X_l \text{ and } Parent(X_n) = X_m.$$

So, now we rank the position of the parents, say, when MI values with respect to node X_l were arranged in decreasing order, the value $MI(X_l\,,\,X_n)$ took 5^{th} position in terms of ranking. Similarly, for node X_m the value $MI(X_m\,,\,X_l)$ takes 9^{th} position and for node X_n say the value $MI(X_n\,,\,X_m)$ takes 3^{rd} position. Then we say that the edge $X_l \rightarrow X_m$ had the least probability of being added among all other edges. This is because, if we had selected k to be less than 9 then this edge would not be added at all.

Finally removing all cycles in this manner we obtain a *DAG*.

Step 4 :

Next we define a scoring function in terms of a *Bayesian Network* and the *Dataset D* which tells us how good the network/graph matches or fits the data D. For this we propose to use *Bayesian Scoring Method* which would suit our purpose the most.

Step 5 :

Next the algorithm suggests to find the highest scoring graph from the set of graphs given by $\{G \mid \forall nodes\, X_i\, of\, G,\, Parent^G(X_i) \subseteq Candidate\ Parents(X_i)\}$. For this a local search procedure is applied as otherwise searching from the huge set of graphs would be *NP hard*. The *Greedy Hill Climbing* algorithm is used but with a small change. In the algorithm propose not to omit a graph if the operations of addition or reversal of edge results in a cycle. Rather we propose to remove the cycle in the method stated previously and compute the score to see how well it matches the data. Making this change however does not change the *order* of the *worst case complexity*.

Step 6 :

Once the local optimum is obtained in the *Greedy Hill Climbing* [5] algorithm we are done with our first iteration and have a *Bayesian Network* say, B^1. Having only one iteration can make the network biased towards initial choice of candidate parents. Therefore it is suggested to continue the iteration using the basic procedure to choose better candidates for the next iterations respectively.

From the next iteration (i.e, from second iteration till convergence) however instead of the MI values we use $Kullback - Leibler\ Divergence$ to measure the *discrepency* between our estimate of the n^{th} iteration $P_{B^n}(X\,,\,Y)$ and the empirical estimate $P(X\,,\,Y)$ obtained from the data. For computing $P_{B^n}(X\,,\,Y)$ we obtain the parameters with the help of *Bayesian Parameter Estimation* technique selecting a *Dirichlet Prior* [6] for each parameter to be estimated. We, then *marginalise* over the *joint distribution* to obtain $P(X\,,\,Y)$ for each pair of nodes such that $X \neq Y$ and $Y is not an element of the set Parent(X)$ in B^n.

Step 7 :

As done previously we again arrange the D_{KL} values for each node in descending order and update our *Candidate Parent* set with respect to each node as for node X_i the new $Candidate\ Parent = Parent(X_i) \cup \{X_{rank1}, \ldots, X_{rank(k-l)}\}$, where l is $\mid Parent(X_i) \mid$ in B^n.

Step 8 :

The network is rebuilt using the updated *Candidate Parent* sets for each node. It is again expected to contain cycles which is removes by the procedure mentioned in **Step 3** and the remaining steps i.e **Step 4, 5, 6, 7** follows.

Step 9 :

Iteration continues till convergence of *score function* that is $Score(B^{n-1}) = Score(B^n)$. At each iteration a structure is returned that scores at least as well as the network in its previous iteration step with respect to the data. So, $Score(B^{n+1} \mid D) \geq Score(B^n \mid D)$ and also it is bounded by the best scoring network. So, the *score* is a monotonically increasing bounded function and hence the algorithm it is guaranteed to stop with the *score matching stopping criterion*.

Step 10 :

We finally expect to have a *Bayesian Network* to be more specific a *Gene Regulatory Network* in our context that shows which genes regulate the others and how. We expect to validate it with *Biological Information* and also draw important conclusions from the network that are relevant in this regard.

Current Work Status

We are almost done with coding the above steps in *MatLab*. We are now trying to see how well our algorithm works on real life datasets, especially with the changes in the algorithm, we are curious to see the extent of improvement from previously done work by researchers in this field.

We are also trying to collaborate with people working on Biotechnology so that we can validate our work biologically.

Main References

[1] *Reverse Engineering Gene Regulatory Networks by Alexander J Hartemink.*

[2] *Gene regulatory network inference* :
evaluation and application to ovarian cancer allows the prioritization of drug target Piyush B Madhamshettiwar et al.

[3] *Using Bayesian Networks to analyze Expression Data by Nir Friedman et al.*

[4] *Learning Bayesian Network Structure from Massive Datasets* :
The "Sparse Candidate" Algorithm by Nir Friedman et al.

[5] *Probabilistic Graphical Models* :
Principles and Techniques by Daphne Koller and Nir Friedman.

[6] *Learning Bayesian Networks the combination of knowledge and statistical data by David Heckerman et al.*

[7] *Other relevant papers and reading materials.*