# Learning Bayesian Networks in Presence of Missing Data

Narges Bani Asadi

**Motivation: Signal Transduction Networks:** The study of Signal Transduction Networks is one of the major subjects of interest in Systems Biology. Signal Transduction pathways are means of regulating numerous cellular functions in response to changes in the cell's chemical or physical environment. Signal transduction often involves a sequence of biochemical reactions inside the cell, which are carried out by proteins. The network of interacting proteins and phosphates relay information as well as amplifying them. Their interaction is by activation or inhibition of the next molecule in the chain. There are usually more than one pathways functioning as well as interacting in a cell at the same moment. The target proteins that can be gene regulatory proteins, ion channels, or components of a metabolic pathway have important affect on the cell behavior.

In this project we try to show how Bayesian network learning methods are helpful in inferring the causal interactions between the molecules. To study the signal transduction network one needs to have a way of measuring proteins involved in the process and should also have a way to silence or force each of them or groups of them to a specific value in order to produce intervention data that is necessary to distinguish correlation from causality. Although Microarray technology has had a lot of progress and people are able to measure the expression of all the human genes in parallel, measuring the proteins that are active in cell is still a very challenging task. Current technology is capable of measuring only a few (around 20) proteins simultaneously in a single cell. Therefore it is of great value to develop methods that can extract the information about the structure of network as much as possible from the partially observed variables. This fact motivated me to work especially in learning the Bayesian network in presence of missing data.

The data I have used to test my algorithms is from flow cytometry experiments (FACS). In FACS experiments they put fluorescent markers inside the cells that bind to proteins floating around inside. Then they put them inside a tiny tube with a fluid flowing through it that causes the cells to pass one by one through a number of lasers that can detect the quantity of proteins with markers bound to them, giving one an idea of the amount of each protein in the cell. What the scientists have done is to apply a number of activators and inhibitors to cell samples and then checking 11 different proteins in this manner. Each experiment has between 650 and 1000 cells in it and there were 9 different experiments. This data is published in [1].

**Bayesian Networks:** Bayesian networks are useful models in representing and learning complex stochastic relationships between interacting variables and their probabilistic nature is capable of modeling the noise that is inherent in biological data. A Bayesian network is a DAG consisted of two parts: 1.The structure or the directed edges that encode the causal relations and conditional independencies between the variables. 2. The local parameters or the distribution function and parameters that encode the distribution of a child value given its parents (CPDs). Bayesian networks can include continuous and discrete variables as vertices [2]. In this project I have focused on the discrete value case. One of the most challenging tasks relating to Bayesian Networks is to infer the graph structure from experimental data. Finding the most probable structure is a NP-hard

problem in the complete data case. In-complete data imposes both foundational problems as well as computational complexities in the already challenging task of structure learning. We assume that the data is missing at random (MAR assumption) to simplify the missing data likelihood function. In this case the likelihood function is composed of different likelihood functions each for one completion of the missing values. We have an exponential number of completing these values and in the worst case each of them will contribute a different mode to the overall likelihood.

$$l(\theta : D) = \sum_{H} P(D, H \mid \theta)$$

As a result we lose the unimodality of the likelihood function, the closed form representation and its decomposition to product of parameter likelihoods. [3]
That is why we try to convert the missing data problem to a complete data problem first by guessing the missing values using the observed variables.
The most referenced algorithm for structure learning with missing data is the structural EM (SEM) proposed by Nir Friedman. In SEM we start with a random structure and its parameters. We run parametric EM on it to impute the missing values and update its parameters. Then we use this completed data to score the next possible graphs that can be reached with local changes to the current graph and for each of the candidates we use the ML parameter assignment. After several updates to the structure we run EM again to re-calculate the missing values. One problem with SEM apart from its slow convergence is its stickiness. Since we update the graph based on one possible imputation of the data it can easily get trapped in local minima. One might try different initializations and multiple restarts but it becomes intractable when we have a large number of missing variables.
Also the greedy search in the graph space is not the best one can do especially as the number of graph vertices grows. We try to develop an algorithm that finds the best possible structures based on multiple completions of the data to escape local minima. Also unless one has a huge number of completed data cases it is impossible to learn a one best scoring graph that is why we will infer a data base of probable graphs in each iteration and use all of them to infer the missing values. So I will discuss the algorithm in two steps: 1. learning the probable graphs given the multiple versions of the complete data. 2. Imputing the missing values given the graph data base.

**Learning the Probable Graphs:** Learning is usually done by scoring the candidate graphs. Different scoring metrics have been developed such as BIC, ML and Bayesian score. These different metrics all prefer the more probable graphs given the data and usually penalize the more complex graph to avoid overfitting.
I used Bayesian Score in the implementation:

$$P(G \mid D) \propto P(D \mid G)P(G)$$
$$P(D \mid G) = \int P(D \mid G, \theta)P(\theta \mid G)d\theta \quad (1)$$

The likelihood integral in (1) has a closed form if we choose the parameters distribution from the exponential family with conjugate priors. Since we are using discrete variables we choose the multinomial distribution with Dirichlet priors:

(2)
$$P(\theta \mid G) = \prod_{i=1}^{n}\prod_{j=1}^{q}\Gamma(\alpha_{ij})\prod_{k=1}^{r}\frac{\theta_{ijk}^{\alpha_{ijk}-1}}{\Gamma(\alpha_{ijk})}$$
$$P(G \mid D) \propto \prod_{i=1}^{n}\prod_{j=1}^{q}\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})}\prod_{k=1}^{r}\frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}$$

We use the BDe metric for the alpha parameters to have score-equivalence. The *Nijk'*s in (2) are sufficient statistics that can be calculated by going through all the data cases and counting the number of times node *i* has value *k* while its parents are in the state *j*.
The space of possible graphs is a super exponential function of number of variables in the network hence the exhaustive search in this space is computationally intractable if we have more than say 5 variables. You can see the growth of the search space by number of variables in Table 1.
Therefore people usually use heuristic search methods to explore the space.
Markov Chain Monte Carlo is a general optimization method that performs a random walk in the space of networks and by applying Metropolis Hastings theorem will ultimately converge to the posterior probability, i.e. in the mixed MCMC chain each network will be sampled with the frequency proportionate to its probability.

| Vertices | Graphs |
|---|---|
| 4 | 453 |
| 5 | 29281 |
| 10 | $4.17\times10^{18}$ |
| 11 | $3.16\times10^{22}$ |
| 20 | $2.34\times10^{72}$ |
| 30 | $2.71\times10^{158}$ |
| 40 | $1.12\times10^{276}$ |

**Table 1: Growth of Search Space**

Using Metropolis-Hastings rule we accept the proposed graph G' with Probability A(G->G')

(3) $$A(G->G') = \frac{P(G' \mid D)T(G->G')}{P(G \mid D)T(G'->G)}$$ In which T(G->G') is the transition probability. But instead of running MCMC in the graph space we decided to use the order space. By order we mean a topological order of the variables of a graph such that each node is placed after its parents. It has been shown by Cooper et al that finding the probable graphs is no longer NP-hard if we know the correct ordering of the nodes [5]. But finding the correct ordering is still a difficult task unless one has enough domain knowledge. So we score the orders in a similar way we score graphs and try to find the more probable orders given the data. The MCMC in the order space has been studied by Friedman et al in [4]. Using MCMC in the order space rather than the graph space has several advantages: 1. The order space is a much smaller space and hence easier to explore $2^{o(n\log n)}$ vs. $2^{\Omega(n^2)}$. 2. We can have more efficient local moves such as swapping the place of two nodes. 3. The transition probabilities need to be calculated in the graph space based on all the possible DAGs that can be reached by applying the local change operators to the current and proposed graphs but one can assume the transition probabilities to be equal in the order space and avoid the expensive acyclicity checks 4.

We can use the domain knowledge on ordering of the variables to define priors on the orders.
Using the Dynamic Programming technique we can compute the order score efficiently [4]:

$$(4) \quad P(\prec | D) = \sum_{G \in \prec} P(G | D) \propto \sum_{G \in \prec} \prod_{i=1}^{n} score(X_i, Pa_i | D) = \prod_{i=1}^{n} \sum_{Pa_i \in Pa_\prec} score(X_i, Pa_i | D)$$

To compute this score efficiently one can make a database of pre-computed local scores indexed by the parent set for each node [6]. In this data base we will list the score of all possible families for the nodes. In order to score a particular order we only accumulate the entries that have a compatible parent set. And finally we multiply the nodes total scores to have the order score. In case of missing data and having multiple imputations we can score the order given each of the completed data bases separately and then take the average of these scores as the final order score.
As the number of graph vertices grows it gets intractable to list all possible parent sets for nodes. That is why people usually limit the nodes in-degree by k: a small number like 3 or 4 as the data-base grows polynomially with the k. In order to have a more informative way to list the parent sets and also allow some larger parent sets for some nodes of the graph I used the regression tree idea. We can use the regression tree modeling technique to find the most informative nodes for each node and list all subsets of those nodes as a possible parent set. I ran the regression tree algorithm using the stop criteria of having less than 30 data points assigned to each node trying to minimize the error $E = \sum_{c \in leaves} \sum_{i \in C} (y_i - y^*)^2$ where $y^* = \frac{1}{c} \sum_{i=1}^{c} y_i$ in a greedy way for a data base of 5400 data cases in the network studied in Sachs et al paper [1]. You can see the results in Table 2 that shows regression tree has captured most of the relevant parents for each node in the network in figure 1. The numbers in the table indicate at what depth of the tree the nodes branch out and the lower numbers indicate more informative nodes. This is not the perfect method to find the parent set as it does not capture the causal relations but can be useful along with the naïve way of listing all possible size k parent sets.
To help mixing of the MCMC chain I used a more complex MCMC method called parallel-tempering. In parallel tempering we have several MCMC chains each running at a different "Temperature". The higher Temperature chains accept the "bad-moves" easier and hence explore the space faster. Each chain can get a proposed sample from its neighbor chain and accept that using the Metropolis-Hastings rules. I implemented the parallel tempering algorithm for a simple 5 node network for which I generated observational and perturbation data. I used the Cooper-Yoo[7] method for calculating the scores in case of perturbation. And I was able to capture the high scoring orders.
**Graph Sampler**: In the second stage of the algorithm assuming we have captured the high scoring orders we will sample graphs by sampling the parents of each node based on the pre-computed scores given these orders.

$$P(Pa_i \mid x_i) = \frac{P(x_i \mid Pa_i)P(Pa_i)}{\sum_{Pa_i \in \prec} P(x_i \mid Pa_i)P(Pa_i)}$$

The orders do not partition the graph space and each graph is compatible with different number of orders. This will introduce bias in the graph sampling process [6]. To solve this problem we will maintain a database of the unique graphs that cover most of the probability space of the high probable orders. We will sample graphs from each order and store them in the data base until we cover 95% probability of that order and then we will go to the next order but we will not restore the repeating graphs. This way we will end up with a data base of probable graphs that can be used in the next step of the algorithm that is the missing data imputation.

**Missing Data Imputation:** We will use Gibbs sampling method to infer the missing values. In Gibbs sampling we iteratively fix the values of all data cases instead of one data case. We use the fixed data cases to learn the parameters of the graph and then we use the partial observed case to run Bayesian network message passing algorithm [8] and complete that partially observer data case. Since we are having a data base of graphs instead of one graph we need to run Gibbs Sampler in all of them and infer the missing data using the probability that is the average probability of all the graphs. Then we pick another case and re-assign its values and we repeat this until convergence. We will run the Gibbs sampler multiple times on the data to have multiple imputations and will use all of those to compute an average score in the order scoring phase in the MCMC order sampler. Figure 2 shows the steps of the algorithm.

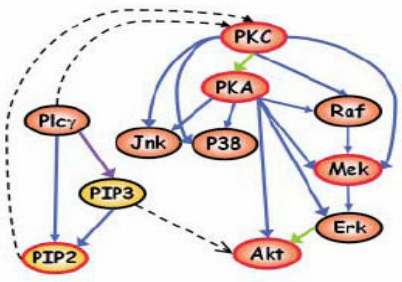


**Figure 1: Network of 11 proteins in T-cell**

| | RAF | MEK | PLCg | PIP2 | PIP3 | ERK | AKT | PKA | PKC | P38 | JNK |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RAF | | 4 | | | | | 2 | 1 | 1 | 2 | |
| MEK | 1 | | 2 | | | | | | | | 1 |
| PLCg | | | | 1 | 2 | | 3 | | | | 4 |
| PIP2 | | | 1 | | 1 | | 3 | | | | |
| PIP3 | | | | 3 | | | 2 | | 4 | | |
| ERK | | | | | | | 1 | | | | |
| AKT | | 4 | 3 | 2 | | 1 | | | 3 | | |
| PKA | 2 | 3 | | | 4 | 2 | | | | 3 | 3 |
| PKC | 3 | 2 | 3 | 4 | 3 | 3 | | 3 | | 1 | 2 |
| P38 | 3 | | | | | | 2 | 3 | 2 | | 4 |
| JNK | | 1 | | | | 3 | | 2 | | 3 | |

Predictors

**Table 2 : List of Candidate Parent Sets**

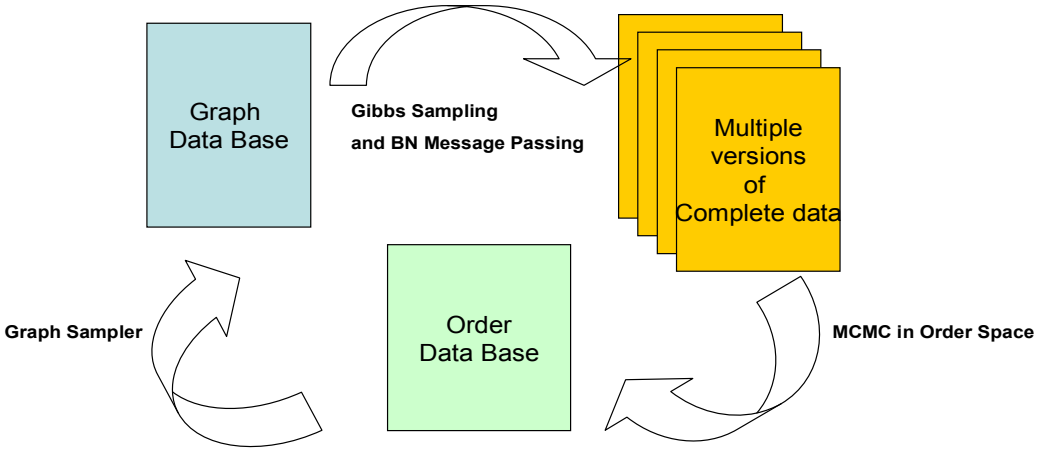


**Figure 2: Algorithm Steps**

**Acknowledgement**
I have started to work on this project this quarter under supervision of my advisor Professor Wing H. Wong.

**References:**

1. Karen Sachs, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, Garry P. Nolan2 "*Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data"*, Science vol 308, 2005
2. David Heckerman, "*A Tutorial on Learning with Bayesian Networks"*, 1995
3. Daphne Koller, Nir Friedman, "*Probabilistic Models in Artificial Intelligence"*, 2005
4. Nir Friedman, Daphne Koller "*Being Bayesian About Network Structures*", *Proceedings of the 16th Annual Conference on Uncertainty in AI (UAI)* (pp. 201-210). 2000
5. Gregory E Cooper, Edward Herskovits, *"A Bayesian Method for the Induction of Probabilistic Networks from Data"*, Machine Learning 9, 309-374 -1992
6. Byron Ellis, Wing H. Wong, *"Learning Bayesian Network Structures from Experimental Data"*, to be published 2006
7. Gregory E Cooper, Changwon Yoo *"Causal Discovery from a Mixture of Experimental and Observational Data",* Proceedings of Uncertainty in Artificial Intelligence, p116-125, 1999
8. J. Pearl, *"Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference"*, Morgan Kaufmann, San Mateo, CA, 1988