10-708: Probabilistic Graphical Models 10-708, Spring 2014

10 : Modeling Networks, Ising Models and Gaussian Graphical Models

Lecturer: Eric P. Xing *Scribes: Zhiding Yu, Shanghang Zhang*

1 Introduction

This lecture mainly introduced a sparse inverse covariance estimation method named graphical lasso and its many variants. These method can be widely used in the domain of automatic graph structure learning. Conventionally, people used to heuristically model the graph and introduce conditional independence based on domain expertise and empirical observations. Many people propose a variety of different algorithms which can not be quantitatively compared, and claim that their models are optimal. But more recently, people start to look into the problem differently, and think about whether the graph structures can be automatically recovered based on the data itself. This is how graphical lasso is introduced, with the graph structures learned through minimizing a loss function containing sparse promoting penalty terms.

There are potentially many real world applications associated with structure learning. Intuitively, such research can be used to analyze underlying relationships embedded under complex networks. The variants of graphical lasso can also be applied to problems where the network is evolving with the time, such as political election (See Fig. 1) and gene network. Up to today, this is still an open research problem. Some of the topics can be very hard to solve.

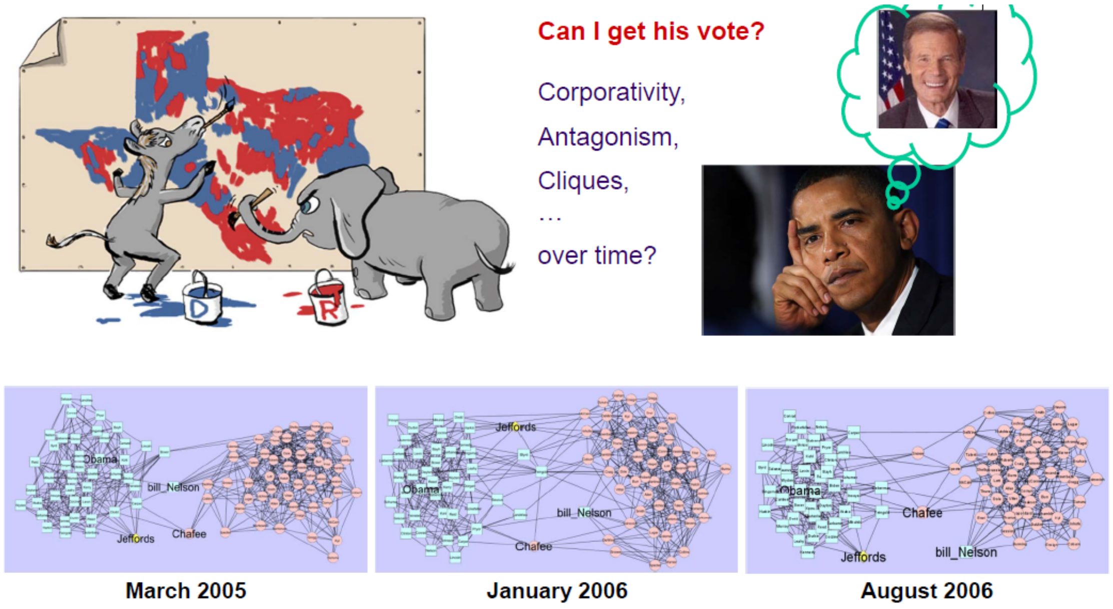

Figure 1: Example of political election modeled as an evolving network.

2 The Basic Variable Selection Problem and Its Intuitions

2.1 From Multivariate Gaussian to Gaussian Graphical Model

To start with the structure learning problem, consider the following multivariate Gaussian distribution:

$$p(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)).$$

The problem here is to estimate the covariance. Without loss of generality, let $\mu = 0$ and $Q = \Sigma^{-1}$ denotes the precision matrix. The probability density function can be expended as:

$$p(x_1, x_2, ..., x_p|\mu = 0, Q) = \frac{|Q|^{1/2}}{(2\pi)^{n/2}} \exp(-\frac{1}{2}\sum_i q_{ii}(x_i)^2 - \sum_{i<j} q_{ij}x_i x_j).$$

The above term can be viewed as a continuous Markov Random Field with potentials defined on every node and edge. The first term $q_{ii}(x_i)^2$ can be viewed as node potential $\phi(x_i)$ while the second term $q_{ij}x_ix_j$ can be regarded as an edge potential $\phi(x_i, x_j)$. The problem now becomes estimating the coefficients q_{ii} and q_{ij} in the precision matrix.

There are many good properties by formulating the problem with precision matrix. The covariance matrix essentially indicates a correlation network and the zero elements $\Sigma_{i,j} = 0$ indicates that X_i and X_j are marginally independent without observing other variables. Intuitively, such independence measure is too brutal and coarse because it indicates the two random variables have no active trails even without observing other nodes. This kind of total independence property is seldom expected to happen in many real world problems.

The precision matrix on the other hand indicates a more sophisticated and subtle relationship between the random variables. If the coefficient is zero then it means the two variables are only conditionally independent given the observations of other nodes. Such conditional independence is expected to be observed a lot in the real problems and therefore it is more important. In addition, this means graphically there is no edge connecting the two nodes, making it much more convenient to be represented, as illustrated in Fig. 2.

$$Q_{i,j} = 0 \quad \Rightarrow \quad X_i \perp X_j | \mathbf{X}_{-ij} \quad \text{or} \quad p(X_i, X_j|\mathbf{X}_{-ij}) = p(X_i|\mathbf{X}_{-ij})p(X_j|\mathbf{X}_{-ij})$$

$$Q = \begin{pmatrix} * & * & * & * & * & 0 \\ * & * & * & * & * & 0 \\ * & * & * & 0 & 0 & 0 \\ * & * & 0 & * & 0 & 0 \\ * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}$$

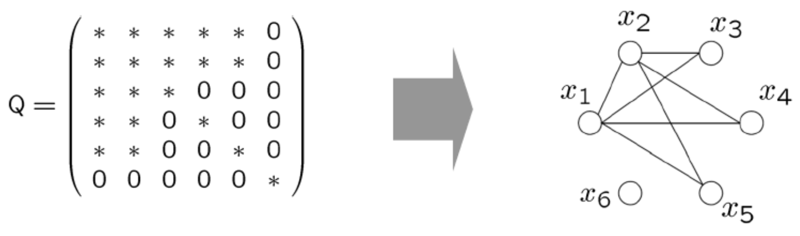

Figure 2: An illustration of the physical meaning of the precision matrix.

2.2 The Sparse Prior and The Meinshausen-Bhlmann(MB) Algorithm

Assuming the network we are interested in can be modeled by Gaussian Graphical Model, the estimation of the precision matrix can be estimated by performing LASSO regression of all nodes to a target node each time and repeat this for each node in the graph (See Fig. 3). Mathematically, the regression problem at each node is formulated as solving the following LASSO problem:

$$\hat{\beta_{\mathbf{1}}} = \arg\min_{\beta_{\mathbf{1}}} ||\mathbf{Y} - \mathbf{X}\beta_{\mathbf{1}}||^2 + \lambda||\beta_{\mathbf{1}}||_1,$$

where $\mathbf{Y}$ is an N dimensional vector with each entry being a node at a specific dimension from the N sets of data observations. Each set of data observation has P nodes, which is illustrated as the P nodes forming a circle in Fig. 3. $\mathbf{X}$ is an $N \times (P-1)$ dimensional matrix with each column being the rest of the nodes in the N sets of observations. $\beta_{\mathbf{1}}$ is a $P-1$ dimensional coefficient weight vector one wants to estimate. The last l_1 term in the equation is the sparse promoting penalty term that enforces sparsity. Notice ideally we want to use $||\beta_{\mathbf{1}}||_0$ which is the l_0 term to minimize the number of supports. However this makes the problem non-convex and difficult to solve. One therefore relaxes this problem to the l_1 case. The above regression is repeated for each node and the zero valued edges are removed. Generally, one is more interested in whether pairwise nodes are conditionally independent or not.

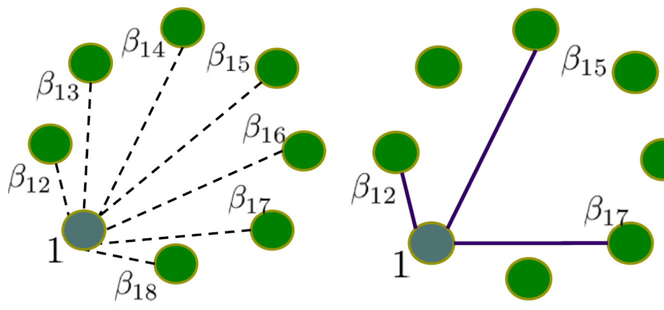

Figure 3: Illustration of LASSO regression repeatedly for each node.

There have been several assumptions associated with the sparse prior.

1. Relevant Covariates are not overly dependent.

2. Large number of irrelevant covariates cannot be too correlated with relevant covariates.

3. Sample quantities converge to expected values quickly.

If the above assumptions are correct, LASSO will be able to recover correct subset of covariates that relevant. And theoretically, there have been proof that graphical lasso can recover the true structure of the graph. Meinshausen et al. proved that if:

$$\lambda_s > C\sqrt{\frac{\log p}{S}},$$

then with high possibility,

$$S(\hat{\beta}) \to S(\beta^*)$$

There are also practical motivations behind this sparse assumption. First of all, in many real applications the precision matrices are indeed sparse. For each node, 50% of connection is already considered to be very dense. Second, for computational reasons people don't want to graph to be too dense because it loses the mathematical tractability and computability. Third, a dense graph also loses the graph interpretation ability.

3 Why Does The Basic MB Algorithm Work?

We have presented the algorithm where the inverse covariance matrix is estimated by repeatedly running LASSO regression on each node. We need to justify why we can do it in this way. For the purpose of the justification, there are some basic facts that one will use.

3.1 Computing Matrix Inverse

Suppose we have:

$$M = \begin{bmatrix} E & F \\ G & H \end{bmatrix},$$

We can diagonalize the matrix as:

$$\begin{bmatrix} I & -FH^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} \begin{bmatrix} I & 0 \\ -H^{-1}G & I \end{bmatrix} = \begin{bmatrix} E - FH^{-1}G & 0 \\ 0 & H \end{bmatrix}.$$

Also denote $M/H = E - FH^{-1}G$. Using the formula $XYZ = W \Rightarrow Y^{-1} = ZW^{-1}X$, we have

$$\begin{aligned} M^{-1} = \begin{bmatrix} E & F \\ G & H \end{bmatrix}^{-1} &= \begin{bmatrix} I & 0 \\ -H^{-1}G & I \end{bmatrix} \begin{bmatrix} (M/H)^{-1} & 0 \\ 0 & H^{-1} \end{bmatrix} \begin{bmatrix} I & -FH^{-1} \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} (M/H)^{-1} & -(M/H)^{-1}FH^{-1} \\ -H^{-1}G(M/H)^{-1} & H^{-1} + H^{-1}G(M/H)^{-1}FH^{-1} \end{bmatrix}. \end{aligned}$$

3.2 Relation between Precision Matrix and Covariance Matrix

Suppose we have:

$$\Sigma = \begin{bmatrix} \sigma_{11} & \vec{\sigma}_1^\top \\ \vec{\sigma}_1 & \Sigma_{-1} \end{bmatrix}$$

Also recall the facts about matrix inverse derived in the previous section, we can have:

$$Q = \begin{bmatrix} q_{11} & -q_{11}\vec{\sigma}_1^\top \Sigma_{-1}^{-1} \\ -q_{11}\Sigma_{-1}^{-1}\vec{\sigma}_1^\top & \Sigma_{-1}^{-1}(I + q_{11})\vec{\sigma}_1\vec{\sigma}_1^\top \Sigma_{-1}^{-1} \end{bmatrix} = \begin{bmatrix} q_{11} & \vec{q}_1^\top \\ \vec{q}_1 & Q_{-1} \end{bmatrix}$$

3.3 Conditional Probabilities in Multivariate Gaussian

Consider the multivariate Gaussian density:

$$p(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x} - \mu)^\top \Sigma^{-1}(\mathbf{x} - \mu)).$$

We can rewrite the joint Gaussian as:

$$p(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} |\mu, \Sigma) = \mathcal{N}(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} | \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix})$$

Using the above notations, we have the following important fact:

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\mathbf{m}_{1|2}, \mathbf{V}_{1|2}),$$

where we have:

$$\mathbf{m}_{1|2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mu_2)$$
$$\mathbf{V}_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

3.4 The justification

With the above three facts, one is ready to justify why the problem can be formulated as a LASSO variable selection problem. Given a Gaussian distribution, the conditional distribution of a single node i given the rest of the nodes can be written as:

$$p(X_i|\mathbf{X}_{-i}) = \mathcal{N}(\mu_i + \Sigma_{X_i\mathbf{X}_{-i}}\Sigma^{-1}_{X_i\mathbf{X}_{-i}}(\mathbf{X}_{-i} - \mu_{\mathbf{X}_{-i}}), \Sigma_{X_iX_i} - \Sigma_{X_i\mathbf{X}_{-i}}\Sigma^{-1}_{\mathbf{X}_{-i}\mathbf{X}_{-i}}\Sigma_{\mathbf{X}_{-i}X_i})$$

Without loss of generality, one has:

$$\begin{aligned} p(X_i|\mathbf{X}_{-i}) &= \mathcal{N}(\Sigma_{X_i\mathbf{X}_{-i}}\Sigma^{-1}_{X_i\mathbf{X}_{-i}}(\mathbf{X}_{-i}), \Sigma_{X_iX_i} - \Sigma_{X_i\mathbf{X}_{-i}}\Sigma^{-1}_{\mathbf{X}_{-i}\mathbf{X}_{-i}}\Sigma_{\mathbf{X}_{-i}X_i}) \\ &= \mathcal{N}(\vec{\sigma}_i^\top\Sigma^{-1}_{-i}\mathbf{X}_{-i}, q_{i|-i}) \\ &= \mathcal{N}(\frac{\vec{q}_i^\top}{-q_{ii}}\mathbf{X}_{-i}, q_{i|-i}) \end{aligned}$$

From here one can already see that the value of a certain node is determined by the linear representation of the other nodes plus a Gaussian noise:

$$X_i \leftarrow \beta_i\mathbf{X}_{-i} + \text{Noise(Gaussain)}.$$

If the assumption about sparse structure is correct, then the structure can be correctly recovered via LASSO. The estimated graph structure can be written as:

$$S_i \equiv \{j : j \neq i, \beta_{ij} \neq 0\}.$$

4 Alternative Methods in Graphical Structure Learning

4.1 The Graphical Lasso Algorithm

There have been more robust and faster algorithms to handle this problem. One of the problem with the basic MB algorithm is that "The loss function is lost". Recall that we have the Multivariate Gaussian density function which can be written as:

$$p(\mathbf{x}|\mu, \Sigma) = \frac{|Q|^{1/2}}{(2\pi)^{n/2}}\exp(-\frac{1}{2}(\mathbf{x}-\mu)^\top Q(\mathbf{x}-\mu)).$$

We want to estimate the precision matrix Q via an l_1-regularized maximum likelihood estimation method such that Q is sparse. Therefore the cost function J is the regularized likelihood. Taking the log of the density function, one has:

$$\begin{aligned} \log p(\mathbf{x}|\mu, \Sigma) &\propto \log\det Q - (\mathbf{x}-\mu)^\top Q(\mathbf{x}-\mu) \\ &= \log\det Q - \text{Tr}((\mathbf{x}-\mu)^\top Q(\mathbf{x}-\mu)) \\ &= \log\det Q - \text{Tr}((\mathbf{x}-\mu)(\mathbf{x}-\mu)^\top Q) \\ &= \log\det Q - \text{Tr}(SQ) \end{aligned}$$

Therefore putting it together with the l_1 regularizer one has:

$$\begin{aligned} J(Q) &= \log\det Q - \text{Tr}(SQ) - \rho||Q||_1 \\ Q^* &= \arg\max_Q(\log\det Q - \text{Tr}(SQ) - \rho||Q||_1) \end{aligned}$$

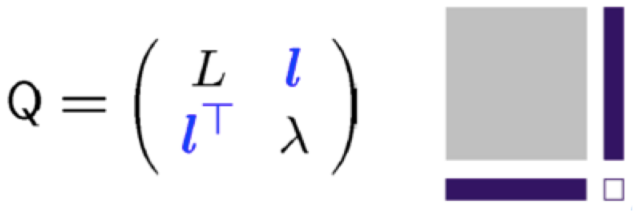
$Q = \begin{pmatrix} L & l \\ l^{\top} & \lambda \end{pmatrix}$

Figure 4: Illustration of updating Q in Graphical LASSO.

Different from MB, the LASSO problems are coupled. Such joint loss function makes the method more stable under noise. The optimization of the above problem is decoupled into updating a row and a column each time in Q. As is illustrated in Fig. 4. Note that the optimization is taken for the blue vectors in the matrix each time. The optimization problem is shown to be LASSO (l_1-regularized quadratic programming). After each optimization, the corresponding row and column are permuted and the new row and column are updated. This process is repeated untill convergence.

4.2 Learning Ising Model for Pairwise MRF

The problem of learning is to deduce the graphical model of a set of random variables given statistics of the random variables. The Ising model is a famous model in statistical physics that has been used as simple model of magnetic phenomena and of phase transitions in complex systems. It is a certain class of binary-variable graphical models with pairwise interactions. We write the P (x|Θ) to denote likelihood of the Ising model, where θii is the vertex parameter and θij is the edge parameter, and xi is the node of the model. The neighborhood selection for this model is built on the logistic regression, which is the L1-regulaerized logistic regression. The theory flow is as follows: Assuming the nodes are discrete (e.g., voting outcome of a person), and edges are weighted, then for a sample x, we have

$$P(\boldsymbol{x}|\Theta) = exp(\sum_{i\in V} \theta^t_{ii} x_i + \sum_{(i,j)\in E} \theta_{ij} x_i \mathrm{x}_j) - A(\Theta))$$

It can be shown the pseudo-conditional likelihood for node k is

$$P_0(x_k|x_{/k}) = logistic(2x_k < \theta_{/k}, x_{/k} >)$$

5 Dealing with Evolving Networks

So far is the theory, the following section will take this forward into the real word and address some interesting application problems. The social network in the congress is a very interesting problem because the graph exists in this network is estimated from the algorithm, instead of the real data. No congress people will tell us who are their friends and who are their enemies, for these are the top secrets, and furthermore, they are changing over time. So how can we do that? The data is kind of strange, which is from the ballet box. The behavior of the congress people can reflect their political opinion. So they cast vote on different bills. A very interesting problem here is what the social network is at every point and on every bill, which is underlying the voting behavior. We can also turn this question into gene expression or Facebook friendship network, and anything where you see the behavior, instead of the network. This is a very generic problem, even in the Facebook, this is still valid because the Facebook has a graph for you, but those graphs are not the real one. People dont actually interact with all the friends on the friends list. People also do not actually know who you are interacting with. So this is an evolving social network problem because the network is changing over time.

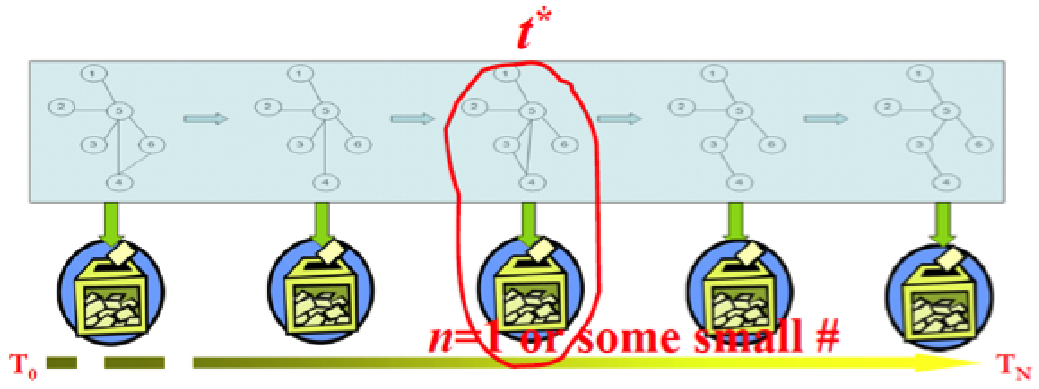

Figure 5: The evolving social network.

The evolving property which increases the additional difficulty in this problem is that you are down to an extreme case of the sample sparsity. You will have an inference problem of p dimensional network from one sample. p will be much larger than 1. This gives rise to a new problem of non-parametric inference of graphical model.

The first inference algorithm is KELLER: Kernel Weighted L1-regularized Logistic Regression. This algorithm still does the neighborhood selection for the network. But a little bit re-engineering of the loss function. The loss function is as follows:

$$l_\omega(\theta_i^t) = \sum_{t'=1}^{T} \omega(x^{t'}; x^t) log P(x_i^{t'}, \theta_i^t)$$

As defined in section 4.2, θi is the vertex parameter and xi is the node of the model. This loss function is just the sum of the regression loss, which is the conditional likelihood of one node given another node. If we only have one sample, we can sum over time. But we have another problem that the network at the next time (t+1) is different from the network at time t. In order to estimate the network at time t, we introduce the kernel function with the assumption that the network has a smooth change. we are going to re-weight every time specific regression conditional likelihood, with the kernel that measure the difference of that point to the time point that estimating the network. This is called the kernel reweighting technique. We can get a virtual data, for which we still sample over time, but the sample at other time will get down weighted. We can thus re-write the neighborhood selection function, which in this case is the logistic regression function with the discrete nodes.

$$P_{\theta^t}(x_i^t | x^t_{\setminus i}) = logistic(2x_i^t < \theta^t_{\setminus i}, x^t_{\setminus i} >)$$

The L-1 norm regression problem is as follows:

$$t^* \in [0, 1]$$

$$\min_{\theta \in \mathbb{R}^{p_n - 1}} \left\{ - \sum_{t \in \mathcal{T}^n} w_t(t^*) \gamma(\theta_i; x^t) + \lambda_1 \|\theta_i\|_1 \right\}$$

where,

$$\gamma(\theta_i^t; x^t) = log P_{\theta_i^t}(x_i^t | x^t_{/i})$$

$$\omega_t(t^*) = \frac{K_{h_n}(t - t^*)}{\sum_{t' \in \mathcal{T}^n} K_{h_n}(t' - t^*)}$$

The only difference is that we have the re-weighting function over the original regression loss. We can use the Gaussian kernel to define the weight, as illustrated as the following graph:

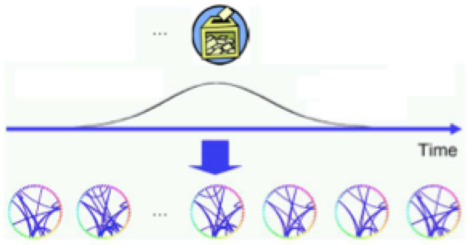

Figure 6: Gaussian kernel to define the weight.

The statistical property of this method are: Dependency Condition; Incoherence Condition; Smoothness Condition and Bounded Kernel.

The second inference algorithm is TESLA: Temporally Smoothed L1-regularized logistic regression. TESLA has the similar idea with KELLER, but using slightly different technique in controlling the coupling between multiple graphs. The problem formulation is as follows:

$$\begin{aligned}\hat{\theta}_i^1, \ldots, \hat{\theta}_i^T &= \arg\min_{\theta_i^1,\ldots,\theta_i^T} \sum_{t=1}^{T} l_{avg}(\theta_i^t) \\ &+\lambda_1 \sum_{t=1}^{T} \| \theta_{-i}^t \|_1 \\ &+\lambda_2 \sum_{t=2}^{T} \| \theta_i^t - \theta_i^{t-1} \|_q^q,\end{aligned}$$

where,

$$l_{avg}(\theta_{\mathbf{i}}^{\mathbf{t}}) = \frac{1}{N^t} \sum_{d=1}^{N^t} \log\ P\left(x_{d,i}^t | \mathbf{x}_{\mathbf{d},-\mathbf{i}}^{\mathbf{t}}, \theta_{\mathbf{i}}^{\mathbf{t}}\right).$$

There are two steps for the estimation procedure: 1) estimate block partition on which the coefficient functions are constant, and 2) estimate the coefficient functions on each block of the partition.

For the last part of the lecture, Professor gave us two stories regarding the finding of such technique: 1) Senate network, and 2) Progression and Reversion of Breast Cancer cells. To justify the graphical estimation is very difficult. We prefer to run on some data set where we at least know if we can justify the efficacy or not. For instance, we all know that there are two major parties in American politics. If the algorithm fails to reflect that, it does not make sense.

6 Summary

This Lecture has three major parts: Gaussian Graphical Model; Neighborhood selection and Time-varying Markov networks. We have learnt Gaussian Graphical Model to capture graph topology in multivariable domain; we have learnt the neighborhood selection to perform that kind of estimation, we have also learnt the KELLER and TESLA methods to do the time variant network estimation.