10-708: Probabilistic Graphical Models 10-708, Spring 2016

10 : Gaussian graphical models and Ising models: modeling networks

Lecturer: Eric P. Xing *Scribes: Xiongtao Ruan, Kirthevasan Kandasamy*

1 Introduction

1.1 Networks in real world

This lecture introduced methods for modeling networks. Gaussian graphical models and Ising models were introduced in the lecture. These methods are popular in learning the structure of networks. In real word, networks are important and interested to researchers. Networks come from lots of areas, such as the Jesus network which represents relationships of characters in Jeses, social networks that represent assiciation among users; Internet networks that represents the connection between different nodes; gene regulatory networks and so on.

Network can also evolves through time. Figure 1 shows a network of genes in the development of an embryo. In different time, the gene regulatory networks are different in corresponding to the activation and regulation of different developmental genes.

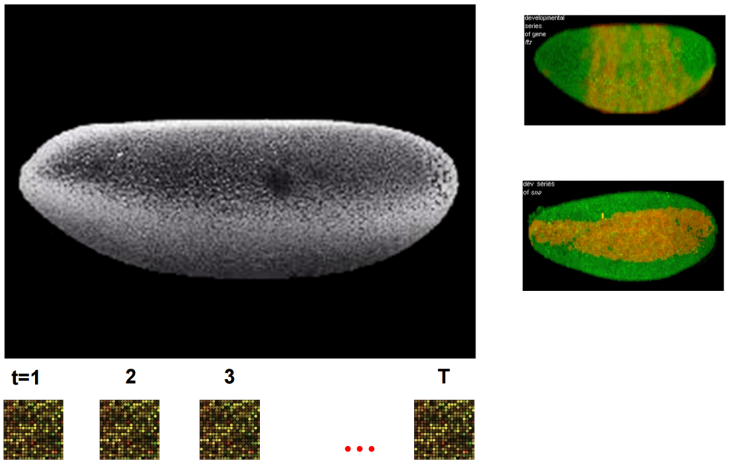


Figure 1: Evolving of Gene regulatory networks in embryo development.

In sum, in real world there are lots of networks and some may evolves through time. It is necessary to develop some statistical methods to study them.

1.2 Two optimal approaches for structure learning

An important question in modeling networks is to decide the structures of networks. The general idea is to sort to 'optimal' approaches, that is, utilize algorithms that guarantee to return a structure that maximizes the objectives (usually likelihood). For structure learning, two approaches that is suitable for different kind of networks have been learned:

- The Chow-Liu algorithm: this algorithm is appropriate for tree-structured graph;
- Pairwise Markov random fields: it is used for undirected graphs

2 Pairwise Markov Random Fields

The basic idea for pairwise Markov random fields is to model the edges as parameters, where if there is connection between two nodes, then there is non-zero parameters for the pair. In other words, we use a matrix of parameters to encode the graph structure. As is shown in Figure 2, θ_{ij} represents the parameter for edge (i, j). The structure of the network can be inferred through the distribution below the network.

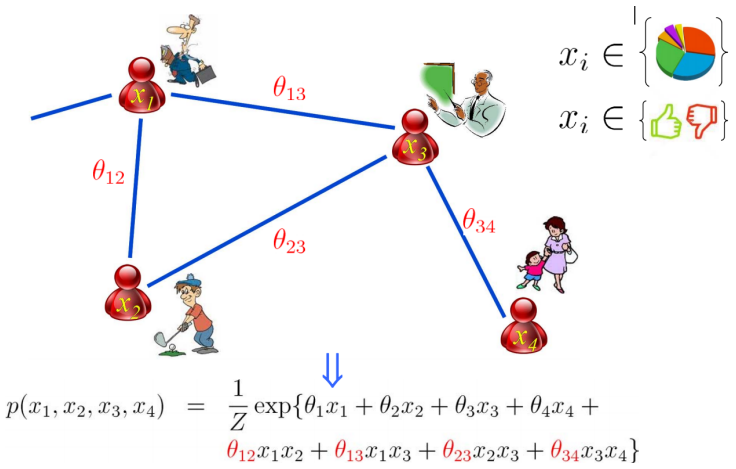


Figure 2: Relationship network and the method for structure learning

The states of nodes can be either discrete, which is called Ising /Potts model, or continuous, which is called Gaussian graphical model, or even heterogeneous.

2.1 Ising Models

Assuming an exponential family distribution, the joint probability of a pairwise MRF, parametrised by Θ is given by,

$$p(x|\Theta) = \exp\left(\sum_{i \in V} \theta_{ii}^{\top} x_i \; + \sum_{(i,j) \in E} \theta_{ij} x_i x_j - A(\Theta)\right).$$

If the variables are binary, it is straightforward to argue that the conditional probability of x_k given x_{-k} is given by a logistic regression model. Concretely,

$$p_\Theta(x_k|x_{-k}) = \text{logistic}\left(2x_k\langle\Theta_{-k}, x_{-k}\rangle\right).$$

This suggests that we could use an L_1 regularised logistic regression approach to learn the neighbors of x_k. The approach also extends to vector valued nodes provided we use an appropriate group lasso penalty on elements of Θ that correspond to one variable.

2.2 Gaussian Graphical Model

As is mentioned above, gaussian graphical models (GGMs) are continuous form of pairwise MRFs. The basic assumption for GGMs is that the variables in the network follows multivariate Gaussian distribution. The distribution for GGMs is

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

where μ is the mean, Σ is the covariance matrix, n is the dimension of data (number of variables).

Here we can show how to convert a GGM to the form of pairwise MRF. WLOG, let $\mu = 0$, precision matrix $Q = \Sigma^{-1}$, then the above distribution can be written as:

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

This is the distribution of a continuous Markov Random Field with potentials defined on each node i as $\exp\{-\frac{1}{2}q_{ii}(x_i)^2\}$ and on each edge (i, j) as $\exp\{-q_{ij}x_ix_j\}$. Moreover, the edges in GGMs corresponds to non-zero elements in the precision matrix.

2.3 Markov versus Correlation Network

Though in GGM, the edge information in the network is encoded in the covariance matrix, there is a distinct difference between Markov and correlation network, which is also based on covariance matrix. In correlation network, if $\Sigma_{i,j} = 0$, then X_i ard X_j are assumed to be independent. While a GGM is Markov network based on precision matrix, if $Q_{ij} = 0$, then X_i and X_j are conditionally independent given all other variables, that is, they have pairwise Markovity. Compared with correlation network, this kind conditional independence or partial correlation coefficients are a more sophisticated dependence measure. Figure 3 below illustrates the network with corresponding precision matrix.

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

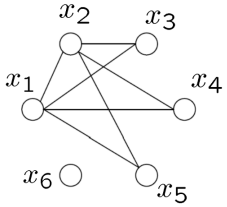
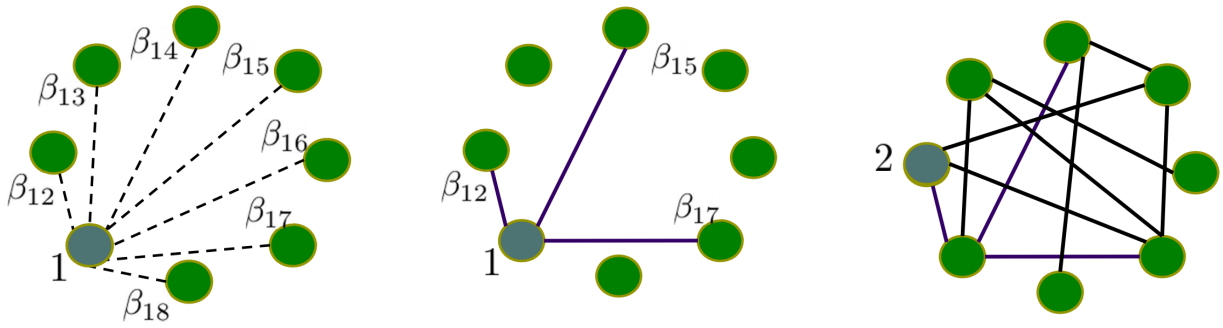


Figure 3: Conversion of precision matrix to the corresponding network



(a) LASSO regression for node 1 (b) Edge selection of node 1 (c) LASSO for all nodes

Figure 4: Neighborhood selection with LASSO

2.4 Network Learning with the LASSO (Meinshausen-Bühlmann algorithm)

In network structural learning, a common assumption is the sparsity of a network, that is, the edges are few in the network and the parameter matrix is sparse. Sparsity assumption usually makes empirical sense, as in lots of networks, the interaction of one node is limited to a few other nodes, such as in gene regulatory networks, a gene interacts with small groups of other genes. On the other hand, sparse networks are now feasible to infer in spite of high dimension of variables.

Network learning with LASSO can be used to perform neighborhood selection for the network. In this method, for each node, LASSO regression with the objective function below is performed for all nodes to this node (node 1), as is show in Figure 4a. β_{ij} is defined as the parameter for edge (i, j), and $\beta_{\mathbf{i}} := (\beta_{i,1}, ..., \beta_{i,n})$

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

where $Y \in \mathbf{R}^T$ is the vector of independent observations of T times for node 1, $X \in \mathbf{R}^{T\times(n-1)}$ is the matrix of observations for all other nodes, and λ is the regularization parameter that controls the sparsity of β_1. By this L_1 penalty, only few edges are kept for this node as is shown in Figure 4b. Then the same procedure is repeated for all other nodes, as is shown in Figure 4c. And combining the selections for all nodes, the edge set for the whole network is expressed as:

$$\hat{\mathcal{E}} = \{(u, v) : \max(|\hat{\beta}_{uv}|, |\hat{\beta}_{vu}|) > 0\}$$

There are three assumptions for LASSO:

- **Dependency Condition:** Relevant Covariates are not overly dependent
- **Incoherence Condition:** Large number of irrelevant covariates cannot be too correlated with relevant covariates
- **Strong concentration bounds:** Sample quantities converge to expected values quickly

If the above assumptions are met, LASSO will asymptotically recover correct subset of covariates that relevant. In papers Meinshausen and Buhlmann 2006, and Wainwright 2009, the conclusion above were proved. The mathematical form is: if $\lambda_s > C\sqrt{\frac{\log p}{S}}$, then with high probability, $S(\hat{\beta}) \to S(\beta^*)$

Neighborhood selection with LASSO is intuitively simple and has theoretical guarantee. However, it it is only suitable for iid-sampled networks. For non-iid sampled networks or time evolving networks, other kinds of algorithms are needed.

3 Gaussian Graphical models

3.1 Variables Dependencies in GGMs

The conditional dependencies in GGMs are basis for the algorithms for network structure inference. Here we will show the joint Gaussian distribution and related conditional distributions. If $\mathbf{x_1}$ and $\mathbf{x_2}$ follows

$$p(\begin{bmatrix}\mathbf{x_1}\\ \mathbf{x_2}\end{bmatrix}|\mu,\Sigma) = \mathcal{N}(\begin{bmatrix}\mathbf{x_1}\\ \mathbf{x_2}\end{bmatrix}|\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix},\begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix})$$

Obviously the marginal distributions for $\mathbf{x_1}$ and $\mathbf{x_2}$ are also Gaussian distributions, and the expressions are as follows respectively.

$$\begin{aligned} p(\mathbf{x_1}) &= \mathcal{N}(\mathbf{x_1}|\mathbf{m}_1^m, \mathbf{V}_1^m) \\ \mathbf{m}_1^m &= \mu_1 \\ \mathbf{V}_1^m &= \Sigma_{11} \end{aligned} \qquad \begin{aligned} p(\mathbf{x_2}) &= \mathcal{N}(\mathbf{x_2}|\mathbf{m}_2^m, \mathbf{V}_2^m) \\ \mathbf{m}_2^m &= \mu_2 \\ \mathbf{V}_2^m &= \Sigma_{22} \end{aligned}$$

The conditional distribution of $\mathbf{x_1}$ given $\mathbf{x_2}$ or $\mathbf{x_2}$ given $\mathbf{x_1}$ are also Gaussian distributions. The expressions of the distributions are as follows:

$$\begin{aligned} p(\mathbf{x_1}|\mathbf{x_2}) &= \mathcal{N}(\mathbf{x_1}|\mathbf{m}_{1|2}, \mathbf{V}_{1|2}) \\ \mathbf{m}_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x_2} - \mu_2) \\ \mathbf{V}_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \end{aligned} \qquad \begin{aligned} p(\mathbf{x_2}|\mathbf{x_1}) &= \mathcal{N}(\mathbf{x_2}|\mathbf{m}_{2|1}, \mathbf{V}_{2|1}) \\ \mathbf{m}_{2|1} &= \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(\mathbf{x_1} - \mu_1) \\ \mathbf{V}_{2|1} &= \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} \end{aligned}$$

3.1.1 The matrix inverse lemma

Here we present the trick for matrix inverse, which is useful for the prove of conditional Gaussian distribution. For a block-partitioned matrix:

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

First, we diagonalize M:

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

To simplify the expression, a value called Schur complement is defined as:

$$M/H = E - FH^{-1}G$$

Then, we inverse the matrix, using this formula:

$$XYZ = W \quad \Rightarrow \quad Y^{-1} = ZW^{-1}X$$

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

By above derivation, we have the matrix inverse lemma:

$$(E - FH^{-1}G)^{-1} = E^{-1} + E^{-1}F(H - GE^{-1}F)^{-1}GE^{-1}$$

3.1.2 Conversion between covariance and the precision matrices

For covariance matrix, we do the partition for the first variable with all other variables like the following expression.

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

So by matrix inverse procedure above, we can get the partitioned precision matrix as:

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

where $q_{11} = {\sigma_{11}}^{-1}$.

3.1.3 Single-node Conditional

Based on above section, we can write the conditional distribution of a single node i given the rest of node as:

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

Without loss of generality, let $\mu = 0$, we have:

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

From this equation, for each node we can write the following conditional auto-regression function:

$$X_i = \frac{\vec{q}_i^T}{-q_{ii}}\mathbf{X}_{-i} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, q_{i|-i})$$

This could be interpreted as a node can be expressed as the linear combination of all other nodes with a Gaussian noise. And we can estimate the neighborhood for each node based on the auto-regression coefficient. The neighborhood of node i is defined as:

$$S_i = \{j :, j \neq i, \theta_{ij} \neq 0\}$$

S_i defines the Markov blanket of node i, we have

$$p(X_i|\mathbf{X}_s) = p(X_i|\mathbf{X}_{-i})$$

3.2 Some Recent Trends in Gaussian Graphical Models

One of the most classical methods for estimating structure in a GGM is due to Arther P. Dempster who sequentially prunes the smallest elements in a precision matrix. Drton & Perlman 2007 refines this approach by providing improved statistical tests for pruning. However, this approach has serious limitations in practice, particularly when the covariance matrix is not invertible. Recently, there has been a flurry of activity in L_1 regularised methods for structure learning in GGMs. Meinshausen & Bühlmann 2006 estimates the neighborhood of a variable in a Gaussian model via Lasso regression. Friedman et al. 2008 adopts a maximum likelihood approach subject to an L_1 penalty on the coefficients of the precision matrix. Banerjee et al. 2008 uses a block sub-gradient algorithm for finding the precision matrix. Below, we review them in more detail.

3.2.1 Graphical Lasso

Let the sample covariance constructed using the data be $S = \frac{1}{n}\sum_{i=1}^{n}(X_i - (\overline{X}))(X_i - (\overline{X}))^\top$. Here $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ is the sample mean. Then, the Gaussian log likelihood of the inverse covariance Q can be shown to be equal to $\log\det Q - \mathrm{tr}(SQ)$. In the Graphical Lasso, we maximise this likelihood subject to an element-wise L_1 norm penalty on Q. Precisely, we solve

$$\hat{Q} = \arg\min_{Q} \log\det Q - \mathrm{tr}(SQ) - \rho\|Q\|_{1,1}.$$

The estimated neighborhood is then the non-zero elements of $\hat{Q}$.

3.2.2 Coupled Lasso

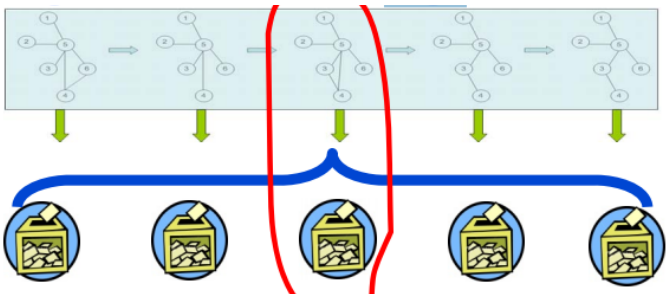
Here, we focus on just one row and column at each step. First we write

$$Q = \begin{bmatrix} Q & \ell \\ \ell^\top & \lambda \end{bmatrix}.$$

At each step we solve for ℓ. The difference with the MB algorithm is essentially that the resulting Lasso problems are coupled since we solve them iteratively. This coupling is essential for stability under noise.

4 Time Varying Networks

Consider the time varying network below. At each time step we have a process which is characterised by a network whose structure changes with time.



Our goal is to infer the structure at each time step using just one datum (or a limited amount of data) for each time step. However, the change in structure from one time step to another is assumed to be smooth so we wish to use data from other time steps too when estimating structure at one step. We review a series of methods in time varying networks.

4.1 KELLER

KELLER, for Kernel Weighted L_1-regularised Logistic Regression, solves the following optimisation problem over $\theta_i^t \in \mathbb{R}^{p-1}$ to estimate the neighborhood of i at time step t,

$$\hat{\theta}_i^t = \arg\min_{\theta_i^t} \ell_w(\theta_i^t) + \lambda \|\theta_i^t\|_1.$$

Here $\ell_w(\cdot)$ is the time weighted conditional log likelihood, $\ell_w(\theta) = \sum_{s=1}^T w(x_t, x_s) \log p(x_i^s | x_{-i}^s; \theta)$. The conditional likelihood is given by a logistic regression model. The weighting w is chosen so that neighboring time steps to t have higher weight than time steps far away from t. For instance, if each time step corresponds to an actual time instant t', the authors recommend using the following weighting scheme.

$$w_t(s) = \frac{K_{h_T}(t' - s')}{\sum_{t=1}^T K_{h_T}(t' - s')}.$$

Here, t', s' are the times corresponding to the time steps t, s and K_h is a smoothing kernel with bandwidth h. The dependence of h on T is made explicit since we might want to adjust the bandwidth depending on how regularly we observe the data. The authors also show that under certain regularity conditions on the problem, we also have a consistent estimator for the structure at time t. Precisely,

$$\mathbb{P}\left(\widehat{G}(\lambda, h_T, t) \neq G^t\right) \in \mathcal{O}\left(\exp\left(-\frac{Cnh_T}{s_T^3} + C' \log p\right)\right),$$

where C, C' are constants. The right hand side goes to zero as we have data from more time steps, i.e. as T increases.

4.2 TESLA

TESLA, or Temporally Smoothed L_1-regularised Logistic Regression, solves the following optimisation problem for estimating time-varying networks. It optimises over the parameters of one node over all times steps jointly. Precisely, it solves the following optimisation problem.

$$\hat{\theta}_i^1, \ldots, \hat{\theta}_i^T = \operatorname*{argmin}_{\hat{\theta}_i^1, \ldots, \hat{\theta}_i^T} \sum_{t=1}^T \ell_{avg}(\theta_i^t) + \lambda_1 \sum_{i=1}^T \|\theta_i^t\|_1 + \lambda_2 \sum_{i=1}^T \|\theta_i^t - \theta_i^{t-1}\|_q^q.$$

Here, $\ell_{avg}(\theta_i^t) = \frac{1}{N^t}\sum_{d=1}^{N^t} \log p(x_{d,i}^t | x_{d,-i}^t, \theta_i^t)$ is the conditional log likelihood. The $\sum_{i=1}^{T} \|\theta_i^t\|_1$ penalty term encourages sparse solutions while $\sum_{i=1}^{T} \|\theta_i^t - \theta_i^{t-1}\|_q^q$ encourages smoothness across different time steps. It is easy to see that the above problem can be cast as a constrained convex optimisation problem. However, the authors also recommend the following alternative estimation scheme. (i) First estimate the block partition on which the coefficient functions are constant via the following,

$$\min_{\beta} \sum_{i=1}^{T} (Y_i - X_i\beta(t_i))^2 + 2\lambda_2 \sum_{k=1}^{p} \|\beta_k\|_{TV}.$$

(ii) Then, estimate the coefficient functions on each block of the partition,

$$\min_{\gamma \in \mathbb{R}^p} \sum_{i \in nbd(j)} (Y_i - X_i\gamma)^2 + 2\lambda_1 \|\gamma\|_1.$$

The advantage of the two step procedure is that choosing the parameters λ_1, λ_2 is now easier and the optimisation problem is also faster. The authors show that structure estimation in TESLA using the alternative procedure is consistent under certain regularity conditions. In contrast to KELLER, it does not need any smoothness assumptions and can accommodate abrupt changes.

4.3 Other Graph estimation Scenarios

Recently, there has been a lot of interest in structure estimation in more complex scenarios. Some examples include time varying Bayesian networks (Song et al. 2009), estimation with missing data (Kolar & Xing 2012)and multi-attribute data (Kolar et al. 2013)