10-708: Probabilistic Graphical Models, Spring 2015

10 : Gaussian graphical models and Ising models: modeling networks

Lecturer: Eric P. Xing *Scribes: Min Hyung Lee, Yan Xia*

1 Overview

There are two ways of exploring a network. The first is a global approach, in which a statistical model is fit on the whole graph. For instance, one can observe that the degree of each node follow a power law distribution, or explore how clusters or cliques form in the network. This approach is not really a graphical model approach, and is not useful in the **actionable** point of view. The graphical model approach or the microscopic approach tend to be directed to the individuals. This lecture is mainly on two things: learning the graph structure of a model, specifically a Gaussian graphical model, and learning how they evolve through time.

2 Structural Learning for Completely Observed Graphical Models

2.1 Gaussian Graphical Models

Before exploring ways of learning the structure of a Gaussian graphical model (GGM), we first look at some nice properties that they have. GGM is based on the multivariate Gaussian distribution, which has the following density function.

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

Here, without loss of generality, we assume $\boldsymbol{\mu} = \mathbf{0}$ and define $\Sigma^{-1} = Q = [q_{ij}]$. Then the pdf is as follows:

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

Now we can look at the distribution as a form of a Markov Random Field (MRF), where the potential for minimal cliques (edges and vertices) are $\psi_i(x_i) = exp\{-\frac{1}{2}q_{ii}(x_i)^2\}, \psi_{ij}(x_i, x_j) = exp\{q_{ij}x_ix_j\}$. It can now be observed that the q_{ij} have a one to one correspondence with the topology of the graph i.e. $q_{ij} = 0$ if and only if there is no edge between node i and node j. Thus, as shown in Figure 1, the precision matrix can be easily converted to a MRF. Moreover, by the pairwise markov assumtion of the MRF, the following property holds.

$$q_{ij} = 0 \Leftrightarrow (i, j) \notin E \Leftrightarrow X_i \perp X_j | \boldsymbol{X}_{-i,j}$$

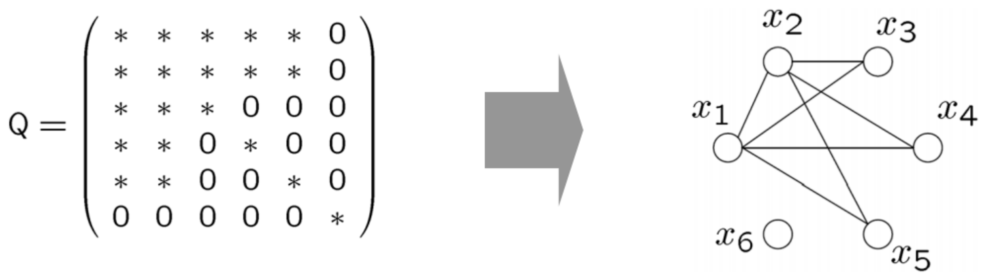
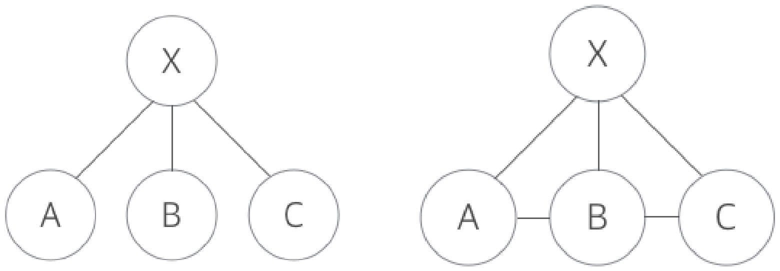


Figure 1: Converting precision matrix to Markov random field



(a) Markov random field (b) Correlation network

Figure 2: Contrasting Markov random field and correlation netork

In contrast to the GGM, the correlation network, which is based on the covariance matrix, was often used in the past to learn the interdependencies of random variables. In this network, there is an edge between node i and node j if and only if $\Sigma_{i,j} \neq 0$, which the two random variables are dependent. However, consider the markov random field in Figure 2a. It can easily be observed that nodes A, B, and C are conditionally independent given X. The correlation network in Figure 2b does not show this fact, since nodes A, B, and C are dependent without observing X. In fact, the correlation network shows no interesting fact about the relationship between the models. Thus, we focus on GGM.

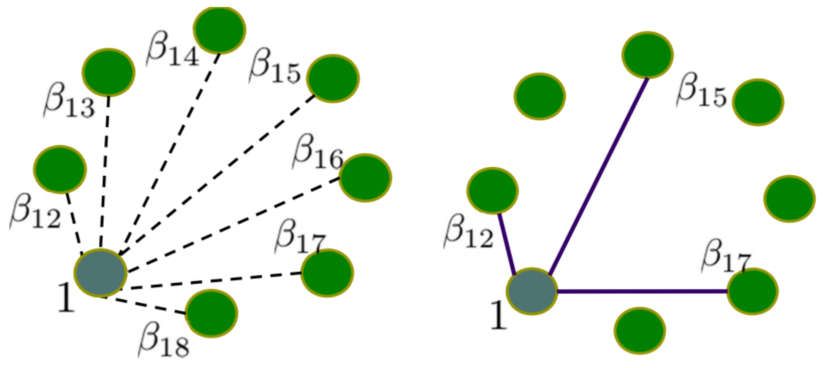
2.2 Learning Gaussian Graphical Models (GGM)

2.2.1 Neighborhood selection algorithm (Meinshausen-Bühlmann algorithm)

Learning a GGM corresponds to estimating the precision matrix Q. The most intuitive solution would be to invert the empirical covariance matrix, but when $n < d$, the empirical covariance matrix is underdetermined and cannot be inverted. Nevertheless, we can still estimate Q by imposing sparsity (regularization), since we know that most of its entries are 0.

As such, we can construct a neighborhood selection algorithm with LASSO regularization as follows, in which for each node, we predict among all other nodes, which ones are neighbors. For a node i, define weights β_{ij} between node i and node j, as in Figure 3a. We can construct a L1 regularized linear regression to learn the weights as follows:

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{i}} = \arg\min_{\boldsymbol{\beta}_{\boldsymbol{i}}} ||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}_{\boldsymbol{i}}||^2 + \lambda||\boldsymbol{\beta}_{\boldsymbol{i}}||_1$$



(a) Definition of weights of node 1 (b) Learned weights of node 1

Figure 3: Constructing GGM using linear regression

L1 regularization compared to L2 regularization enforces sparsity in the weights, setting most βs to 0. Specifically, if the problem satisfies three conditions: dependency, incoherence, and strong concentration bounds conditions, LASSO will asymptotically recover the correct subset of weights that are relevant. As such, we can consider only the β_{ij} that are larger than 0, to select the neighbors of node i, as shown in Figure 3a. Combining the estimated results of all nodes, the total edge set can be constructed as follows:

$$\hat{\boldsymbol{\varepsilon}} = \{(u, v) : |\hat{\boldsymbol{\beta}}_{\boldsymbol{uv}}| > 0 \text{ or } |\hat{\boldsymbol{\beta}}_{\boldsymbol{vu}}| > 0\}$$

Although the process seems to be a heuristic way of finding the edges, it can be proved that the process is consistent. Formally, if $\lambda > C\sqrt{\frac{\log p}{S}}$, then with high probability, $S(\hat{\boldsymbol{\beta}}) \to S(\boldsymbol{\beta}^*)$.

Now we prove that the graph constructed by the neighborhood selection algorithm have the properties of a normal MRF. By using the matrix inverse lemma, we can formulate the conditional probability of a node given all other nodes as follows:

$$p(X_i|\boldsymbol{X}_{-i}) = N(\mu_i + \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}(\boldsymbol{X}_{-i} - \boldsymbol{\mu}_{-i}), \Sigma_{ii} - \Sigma_{i,-i}\Sigma_{-i,-i}^{-1}\Sigma_{-i,i})$$

Assuming $\boldsymbol{\mu} = \mathbf{0}$, the above can be simplified as follows:

$$p(X_i|\boldsymbol{X}_{-i}) = N(\frac{\boldsymbol{q}_i^\top}{-q_{ii}}\boldsymbol{X}_{-i}, q_{i|-i})$$
$$\boldsymbol{q}_i = Q_{i,-i}$$

From this formulation, the auto-regression function can be constructed as follows:

$$X_i = \frac{\boldsymbol{q}_i^\top}{-q_{ii}}\boldsymbol{X}_{-i} + \varepsilon, \varepsilon \sim N(0, q)$$

Now, define $\boldsymbol{\beta}_i = \frac{\boldsymbol{q}_i^\top}{-q_{ii}}$ and we obtain the linear regressor used in the neighborhood selection process. Thus, by setting $\beta_{ij} = 0$, we set the corresponding q_{ij} to 0.

Define the neighborhood of node i as follows:

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

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

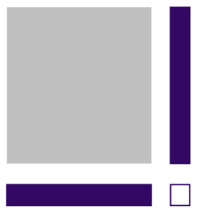


Figure 4: L_1-regularized maximum likelihood learning

Since $p(X_i|\boldsymbol{X}_{-1}) = N(\frac{\boldsymbol{q}_i^\top}{-q_{ii}}\boldsymbol{X}_{-i}, q_{i|-i}) = N(\boldsymbol{\beta_i}\boldsymbol{X}_{-i}, \varepsilon)$, $p(X_i|\boldsymbol{X}_{-1}) = p(X_i|\boldsymbol{x}_{S_i})$. Thus, the neighborhood S_i defines the Markov blanket of node i.

2.2.2 Graph LASSO algorithm (Block optimization method)

Graph LASSO algorithm seeks to directly maximize the log likelihood. This algorithm takes the sample covariance matrix S as input, where

$$S_{i,j} = \frac{1}{N}\sum_{n=1}^{N} x_i^{(n)} x_j^{(n)}$$

and outputs the sparse precision matrix Q that maximize the log likelihood.

$$Q^\star = \arg\max_{Q} \left\{ \underbrace{\ln|Q| - \mathrm{tr}(SQ)}_{\text{Log likelihood}} - \underbrace{\rho\|Q\|_1}_{\text{Regularizer}} \right\}$$

Q is a sparse matrix that can be thought of as an inverse of S, which is not directly invertible. The trace term can be expanded to:

$$\mathrm{tr}(SQ) = \sum_{i,j} x_i x_j q_{ij} + \sum_i q_{ii} x_i^2$$

and it can be shown that the log likelihood term

$$\ln|Q| - \mathrm{tr}(SQ) = \ln \prod_{i=1}^{N} \mathcal{N}\left(\boldsymbol{x}^{(i)} \mid 0, Q^{-1}\right)$$

This optimization problem can be solved by coordinate ascent. As shown in Figure 4, in each iteration we focuses only on one row/column (blue vectors), keeping the others constant (gray block). The optimization for blue vectors can be shown to be LASSO (L_1-regularized quadratic programming). This algorithm is more accurate than the MB algorithm, and the main difference of this algorithm is that the LASSO problem are coupled, and this coupling is essential for stability under noise.

There is an interesting analogy between Gaussian Graphical Model and Ising Model. The Ising model can be thought of as a discrete analogue of the Gaussian graphical model.

$$P(\boldsymbol{x} \mid \Theta) = \exp\left(\sum_{i\in V} \theta_{ii} x_i + \sum_{(i,j)\in E} \theta_{ij} x_i x_j - A(\Theta)\right)$$

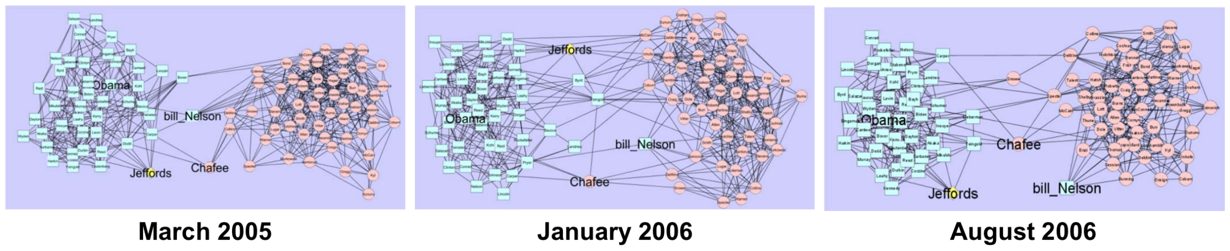


Figure 5: Example: learning the relationship among senators over time

It can be shown that the pseudo-conditional likelihood for node k is given by a logistic regression model.

$$P_\theta(x_k \mid x_{-k}) = \text{logit}(2x_k \langle \theta_{-k}, x_{-k} \rangle)$$

This provides a way of learning the Ising model. Other methods introduced in this lecture will not apply here because the term $A(\Theta)$ is unknown and needs to be solved as well.

3 Evolving Networks

This lecture also discussed briefly on the algorithms for learning dynamic, time-varying networks. Many real world problems requires the modeling of dynamic networks that are rewiring over time. For example, using the senate voting records dataset, which contains the vote records of 100 senators on 542 bills, we wish to discover how the relationship between senators changes over time (Figure 5). In these problems, both the data-generating process and latent relational structure change over time in these networks. The challenge for this type of problem is the unavailability of serial snapshots of the time-varying networks, because at each time point the number of examples are usually very small. Two algorithms were introduced in the lecture to learn the time-varying networks.

1. *Kernel Weighted L_1-regularized logistic regression*(KELLER) tries to solve the data scarcity at individual time points using the following estimation:

   $$\hat{\theta}_i^t = \arg\min_{\theta_i^t} l_w(\theta_i^t) + \lambda_1 \|\theta_i^t\|_1 \quad \forall t$$

   where

   $$l_w(\theta_i^t) = \sum_{t'=1}^{T} w(x^{t'}, x^t) \log P(x_i^{t'} \mid \boldsymbol{x}_{-i}^{t'}, \theta_i^t)$$

   This problem is relatively easy to solve, because it is essentially a LASSO problem. Also, this formulation tries to solve the data scarcity problem by giving the examples at different time points weights, and using all of them in the estimation. The weights for each example is generated using kernel functions, which captures similarity between the data at a time of interest and all other time points. This way, we can estimate the time-specific networks one by one, based on "virtual iid" samples.

   Specifically, we can apply Ising model to the dataset, and the conditional likelihood follows a logistic regression model.

   $$P_\theta(x_i^t \mid x_{-i}^t) = \text{logit}(2x_i^t \langle \theta_{-i}^t, x_{-i}^t \rangle)$$

   This way, the structure of the network can be learned using neigborhood selection: $S(x_i) = \{j \mid \theta_{i,j} \neq 0\}$. Thus, the overall algorithm for time-specific graph regression are as follows:

Estimate at $t^\star \in [0,1]$

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

where

$$\begin{aligned} \gamma(\theta_i; x^t) &= \log P(x_i^{t'} \mid \boldsymbol{x}_{-i}^{t'} \\ w_t(t^\star) &= \frac{K_{h_n}(t - t^\star)}{\sum_{t' \in \mathcal{T}^n} K_{h_n}(t - t^\star)} \end{aligned}$$

KELLER is structural consistent assuming four conditions, namely dependency condtion, incoherence condition, smoothness condition and bounded kernel condition. With some additional assumptions, it can be shown that

$$\mathbb{P}(\hat{G}(\lambda_1, h_n, t\star) \neq G^{t^\star}) = \mathcal{O}\left(\exp\left(-C \frac{n h_n}{s_n^3} + C' \log p \right) \right) \to 0$$

2. *Temporally Smoothed L_1-regularized logistic regression* (TESLA) tries to force the structure at near time points to be similar. It enforces this constraint by including total variance penalties.

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