10-708: Probabilistic Graphical Models, Spring 2020

16: Modeling networks: Gaussian graphical models and Ising models

Lecturer: Eric P. Xing *Scribe: Yu-Hsuan Wang, Ta Chung Chi, Ayush Raina, Yijia Sun*

1 Structure learning

1.1 Introduction

In previous lectures, we assume we have enough domain knowledge to design the structure of our models which could be illustrated with the following examples:

1. We know there are multiple clusters in the data and we know each cluster could be represented with a Gaussian distribution ⇒ We use Gaussian mixture model to model our data.
2. We know the relation between different time steps of data and time step t provides sufficient information for us to model time step $t+1$ ⇒ We use hidden Markov model to model our data.
3. We know the dependency between concepts and encode the dependency with a knowledge graph or expert system like Figure 1.

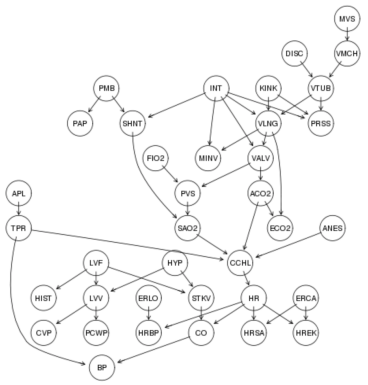


Figure 1

In this lecture, we are doing the other way around which is called *structure learning*. We are going to learn the dependency, correlations or the causal structures of entities from our data. There are different categories of structure learning:

- Tree v.s. Non-Tree: The learned structure could be constrained to be a tree or not. If we constrain the learned structure to be a tree, there would be a nice mathematical guarantee that we could achieve it in polynomial time [1]. On the other hand, without such constraint the problem becomes NP-hard while there are many heuristics and approximations could be applied to solve it.

- Directed v.s. Undirected: The learned structures could also be categorized by whether they are directed or undirected. If the learned structure is a directed structure, we could apply causal discovery approach to solve it. The approach would be introduced in the following lectures. On the other hand, in this lecture, we are going to introduce how to learn an undirected structure in the following sections.

1.2 Network learning

There are many problems that require undirected network learning. For example, the dependency between different users in a social network or the dependency between web pages on the internet. However, the problem could be more complicated with the fact that the network is dynamic. For example, Figure 2 shows the relations between members of congress over time. The dynamic nature increase the difficulty of learning.

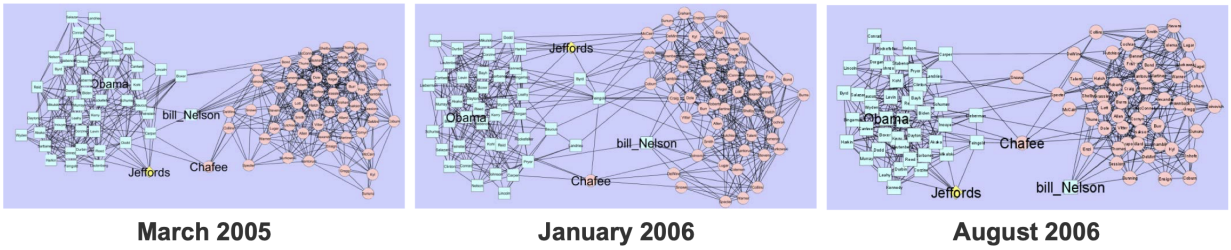


Figure 2

In the good old days, structure learning is ad hoc. For example, when we are trying to capture the relations between figures in the bible, we would use the co-occurrence of their names within a sentence. However, we could also use the co-occurrence within passage or different window and all of them would give us very different results. As a result, we need a more standarized approach to perform structure learning. In the following sections, we are going to introduce these approaches.

2 Network inference as parameter estimation

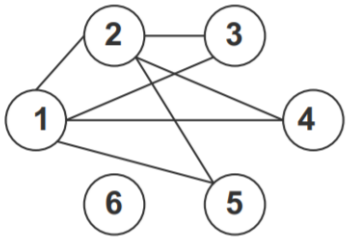


Figure 3

Markov random field is used to model data set with discrete, continuous, or heterogeneous nodal values by defining potential functions on cliques in a graph. Every pair-wise weight is attribute to an edge that connects the pair-wise random variables of interest. Intuitively, if the weight has zero value, there does not exist an edge that connect the pair-wise random variables. Consider a graph as shown in Figure 3 and the corresponding joint probability distribution in Eq.(1)

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z} \exp\{\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_{12} x_1 x_2 + \theta_{13} x_1 x_3 + \theta_{23} x_2 x_3 + \theta_{34} x_3 x_4\} \quad (1)$$

We are able to obtain a parameter matrix which encodes the graph structure as shown in 4 The matrix

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

Figure 4

creates a bridge between parameter learning and structure learning.

One example of Markov Random Filed model is a multivariate Gaussian with has a probability density function as shown in Eq.(2)

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

Without loss of generality, we assume $\mu = 0, Q = \Sigma^{-1}$.

$$p(x_1, x_2, ...x_n|\mu = 0, Q) = \frac{|Q|^{1/2}}{(2\pi)^{n/2}} \exp\{-\frac{1}{2} \sum_i q_{ii}(x_i)^2 - \sum_{i<j} q_{ij} x_i x_j\} \quad (3)$$

The values in precision matrix Q corresponds to the weights of the singleton and pair-wise potentials in a continuous Markov Random Field. Therefore, Gaussian graphical model $\mathbf{X}^{(n)} \sim \mathcal{N}(\mathbf{0}, \Sigma^{(n)})$ is pair-wise *Markov Network* where the precision matrix carries structural information of the graph. The conditional independence and partial correlation coefficients are a more sophisticated dependence measure. For a given precision matrix Q, recall that

$$Q_{i,j} = 0 \implies X_i \perp\!\!\!\perp X_j | \mathbf{X}_{-ij}$$

or

$$p(X_i, X_j|\mathbf{X}_{-ij}) = p(X_i|\mathbf{X}_{-ij}) p(X_J|\mathbf{X}_{-ij})$$

We need to distinguish *Markov Network* from *Correlation network* which is based on the covariance matrix

$$\Sigma_{i,j} = 0 \implies X_i \perp\!\!\!\perp X_j, p(X_i, X_j) = p(X_i) p(X_j) \quad (4)$$

The former represents conditional independence among the variables, while the latter represents marginal independence. However, the conversion between correlation network and markov network is nontrivial ,i.e. sometimes the precision matrix Q cannot be obtained by just inverting the covariance matrix. Therefore, the goal is to bypass inverting the covariance matrix and to learn the preciion matrix directly. One common assumption is that the precision matrix is structurally sparse. The sparsity assumption makes empirical sense. For example, genes are only assumed to interface with small groups of other genes. The assumption also make statistical sense as learning is now feasible in high dimensions with small sample size.

2.1 Network learning with the LASSO

For a linear regression problem $y = \beta \mathbf{X} + \epsilon, \epsilon \sim \mathcal{N}(\mu, \sigma^2)$, LASSO regression has the form

$$\min_{\beta_0, \beta} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 \text{ s.t. } \sum_{j=1}^{p} |\beta_j| \leq t \quad (5)$$

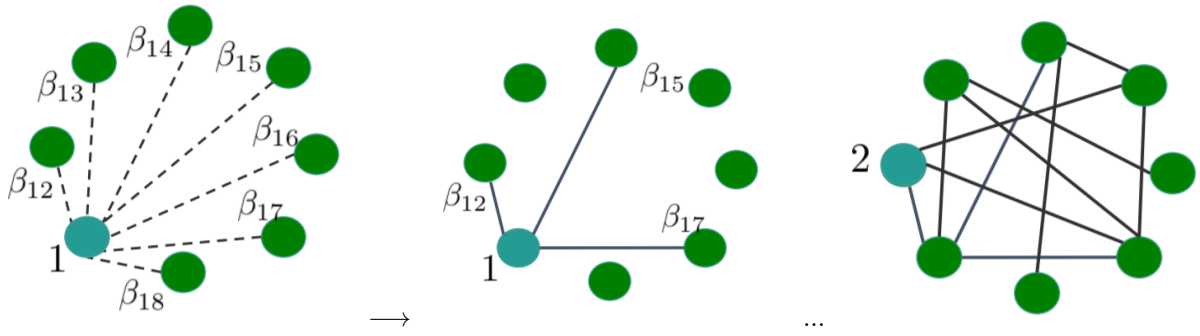
L1 regularization in the LASSO guarantees sparsity under the following assumptions

- 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 we project linear regression to a graphical representation and perform LASSO regression of all nodes to a target node, LASSO can select the neighborhood of each node by solving

$$\hat{\beta}_i = \text{argmin}_{\beta_i} ||\mathbf{Y} - \mathbf{X}\beta_i||^2 + \lambda||\beta_i||_1$$

Repeat the LASSO regression for every node will form the total edge set



$$\hat{\varepsilon} = (u, v) : \max(|\hat{\beta_{uv}}|, |\hat{\beta_{vu}}|) > 0 \tag{6}$$

This algorithm is called the *Graphical LASSO* where LASSO is performed on every node with respect to all the other nodes in the whole collection. It is been proved in [2, 3] that Graphical LASSO has consistent structure recovery, i.e. this procedure has high probability of recovering the true structure of the graph. If $\lambda_s > C\sqrt{\frac{\log p}{S}}$, then with high probability $S(\hat{\beta}) \to S(\beta^*)$.

2.2 Graphical LASSO regression

Equation 2 for a bi-variate Gaussian distribution case can be written as follows:

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

The conditional probabilities hence can be written as:

$$p(\mathbf{x_2}) = \mathcal{N}(\mathbf{x_2}|m_2^m, V_2^m) \text{ where } m_2^m = \mu_2 \text{ and } V_2^m = \Sigma_{22} \tag{7}$$

$$p(\mathbf{x_1}|\mathbf{x_2}) = \mathcal{N}(\mathbf{x_1}|m_{1|2}, V_{1|2}) \text{ where } m_{1|2} = \mu_1\Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x_2} - \mu_2) \text{ and } V_{1|2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{8}$$

Expanding the covariance and precision matrices:

$$\Sigma = \begin{bmatrix}\sigma_{11} & \sigma_1^T\\ \sigma_1 & \Sigma_{-1}\end{bmatrix}$$

Note that here σ_{11} is the variance of the first variable and Σ_{-1} is the covariance matrix for the rest $n-1$ variables.

The precision matrix can be represented as below following the matrix inversion lemma:

$$\mathcal{Q} = \Sigma^{-1} = \begin{bmatrix} q_{11} & -q_{11}\sigma_1^T\Sigma_{-1}^{-1} \\ -q_{11}\Sigma_{-1}^{-1}\sigma_1 & \Sigma_{-1}^{-1}\left(\mathbf{I} + q_{11}\sigma_1\sigma_1^T\Sigma_{-1}^{-1}\right) \end{bmatrix} = \begin{bmatrix} q_{11} & -q_1^T \\ q_1 & \mathcal{Q}_{-1} \end{bmatrix} \tag{9}$$

This is used to simplify the conditional distributions of a particular random variable in the multi-variate gaussian from Eq 7-8.

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

Plugging values from Eq 9 the expression is simplified as a single-variate gaussian for conditional auto-regression of the random variable:

$$p(X_i|\mathbf{X_{-i}}) = \mathcal{N}\left(\frac{\vec{q_i}^T}{-q_{ii}}\mathbf{X_{-i}}, q_{i|-i}\right)$$

Similar expressions can be written for each node in the graph. X comes from a gaussian distribution and can be represented as $X = \frac{\vec{q_i}^T}{-q_{ii}}\mathbf{X_{ii}} + \epsilon$ this is similar to a standard regression formulation where $X_i = \beta_i X_{-i} + \epsilon$. Where β_i represents the coefficients of the regression. Here i are the different columns of the matrix Q. For every value of i a LASSO regression is solved and the values are learned. The beta coefficients are dependent on the zero and non-zero values of the Q matrix given if a particular variable lies in the neighbourhood (is connected) of the others. Hence an estimate of the neighborhood s_i, we have:

$$p(X_i|\mathbf{X_{-i}}) = p(X_i|\mathbf{X_s}) \text{ where } s_i \text{ defines the Markov blanket of node } i$$

Iterating this process over all random variables gives the Meinshausen-Buhlmann(MB) algorithm for structure learning. An alternate method is called L_1 -regularized maximum likelihood learning and it directly learns the value of the Q matrix. First step is to compute the sample covariance matrix from data and an optimization problem is formed based on the maximum likelihood estimator as shown below:

$$\mathcal{Q}^* = argmax_Q \left\{\log det\mathcal{Q} - tr(SQ) - \rho||Q||_1\right\}$$

The algorithm iterates over the different columns of the $\mathcal{Q}$ matrix and minimize the above loss function. This leads to a more stable solution as there is a single loss function and the solution is achieved iteratively converging to a final solution.

3 Learning Ising Model

If the node x_i of graph we intend to learn is discrete, for example, voting outcome of a person, we have:

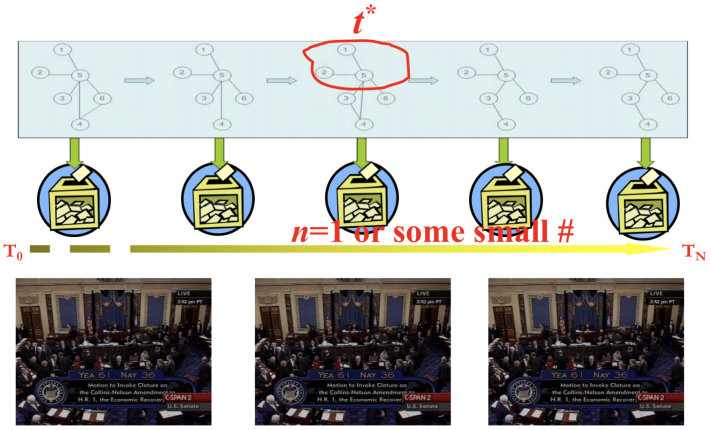
$$P(x|\theta) = \exp\left(\sum_{i\in v}\theta_{ii}^T x_i + \sum_{(i,j)\in E}\theta_{ij}x_i x_j - A(\Theta)\right) \tag{10}$$

Note that this is a pairwise MRF, where θ_{ij} reveals the graph structure. However, we cannot use simple linear regression to do graphical lasso due to the discrete nature of the random variable. Therefore, the conditional is no longer gaussian but logistic:

$$P_\theta(x_k|x_{-k}) = logistic(2x_k \cdot {\theta_{-k}}^T x_{-k}) \tag{11}$$

If each node is a variable length vector instead, partial correlation computing may be useful for us to draw connections between different dimensional nodes.

4 Evolving Social Networks



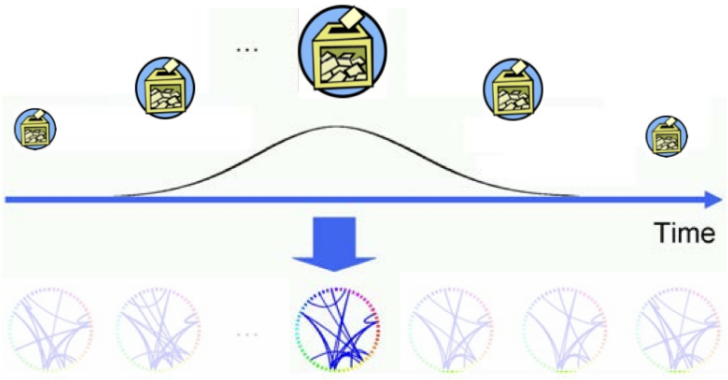
Evolving social graphs are interesting and hard to estimate because in real world, graphs are evolving and are not i.i.d generated. The difficulty is that we may have only one sample at each particular time step, which is not good for statistical estimation. However, the idea is that graphs are highly dependent across time steps, so we can come up with an algorithm to estimate the graph structure using all the samples over time.

4.1 kernel weighted L1 regularized Logistic Regression

The formal equation is given by:

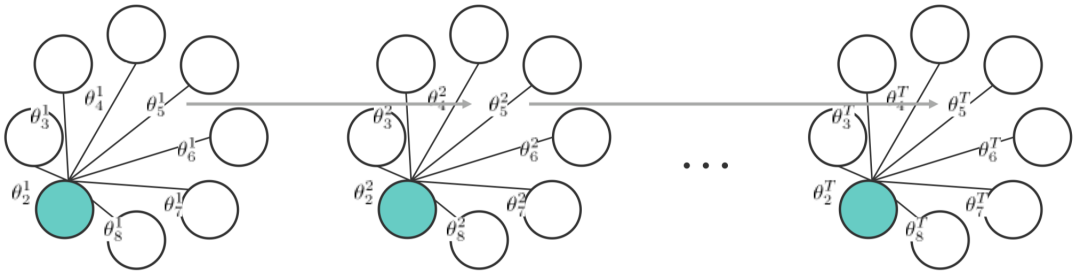
$$\hat{\theta}_i^t = \arg\min_{\theta_i^t} l_w(\theta_i^t) + \lambda_1 \|\theta_i^t\|_1, \tag{12}$$

where $l_w(\theta_i^t) = \sum_{t'=1}^{T} w(x^{t'}, x^t) \log p(x_i^{t'} | x_{-i}^{t'}, \theta_i^t)$.



Note that we introduce a weight function, which is defined for two time steps that measures the distance of graphs of two time steps. This operation is done over all time steps. The closer it is to the time of interest, the higher the weight should be. The good thing is we can utilize all the samples instead of only that for the specific time step to perform structure estimation.

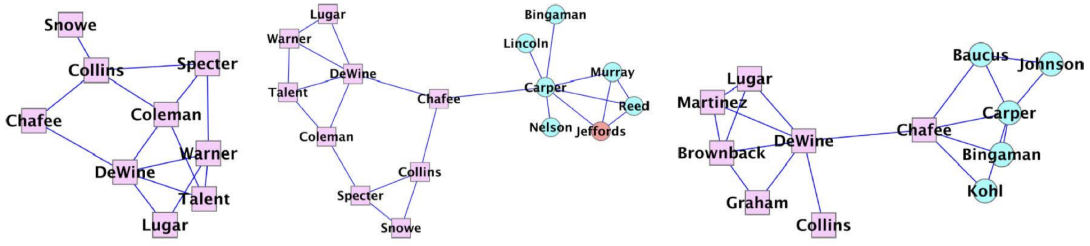
4.2 TESLA: Temporally Smoothed L_1 regularized logistic regression



$$\hat{\theta}_i^1, \cdots, \hat{\theta}_i^T = \arg\min_{\theta_i^1, \cdots, \theta_i^T} \sum_{+} t = 1^T l_{avg}(\theta_i^t) + \lambda_1 \sum_{t=1}^{T} \|\theta_{-i}^t\|_1 + \lambda_2 \sum_{t=2}^{T} \|\theta_i^t - \theta_i^{t-1}\|_q^q, \tag{13}$$

where $l_{avg}(\theta_i^t) = \frac{1}{N^t} \sum_{d=1}^{N^t} \log p(x_{d,i}^t, \theta_i^t)$. Note that we are estimating all the graphs together, therefore, the samples that define the conditional likelihood are not going to be re-weighted. Further, we are able to measure the difference of all graphs. The high level intuition of this objective is that the evolving is smooth and gentle over time. For a particular edge, we hypothesize that the difference should be small. In other words, if an edge is presented in a graph previously, it should be presented in the next graph with high chance.

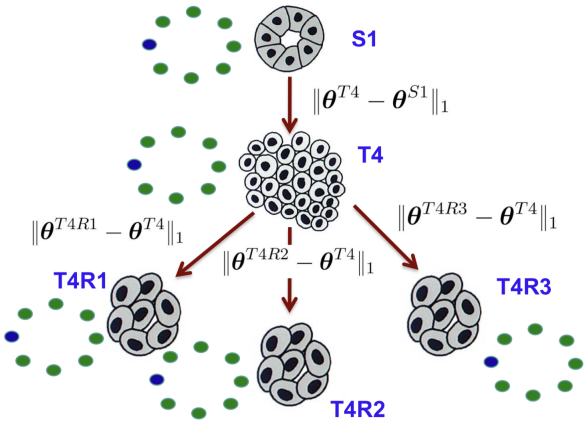
4.3 Senate Network



(a) t = 0.1 (b) t = 0.4 (c) t = 0.8

We use the data that consists of voting records on 542 bills of 100 senators, where each vote is a binary outcome. We can see that senator Chafee gradually changed his preference over time.

4.4 Breast Cancer Cells



The breast cancer cell develops over different stages such that scientists can observe different structures over time. The goal is to determine if the cell is normal given its structure. Specifically, we want to penalize the total variation of the graphs that are adjacent in the network.

References

[1] C Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[2] N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Ann. Statist.*, 34(3):1436–1462, 06 2006.

[3] M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1 -constrained quadratic programming (lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202, May 2009.