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

8 : Learning in Fully Observed Markov Networks

Lecturer: Eric P. Xing *Scribes: Meng Song, Li Zhou*

1 Why We Need to Learn Undirected Graphical Models

In the previous lectures, we have talked about the structure and parameter learning for the completely observed BNs. In the directed models, the potentials are restricted to conditional distributions which model the dependence of a variable on its parents. However, sometimes an undirected association graph is more informative and natural to do the modeling. For example, in domains such as computer vision, the influences of pixels in an image are intrinsically symmetric. In biology, the gene expressions may be influenced by the unobserved factors where we don't have enough information to know the direction. In this lecture, we will cover the techniques of structural learning and parameter estimation for fully observed MRF.

2 Structural Learning for Completely Observed MRF

2.1 Gaussian Graphical Models

In this section, we will introduce the neighborhood selection for undirected structural learning. To give the background of this method, let's first look at a typical MRF: Gaussian Graphical Model.

As what we have known, when the variables follows the multivariate Gaussian density, it can be expressed as

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

where $\mathbf{x} = [x_1, x_2, \cdots, x_p]^T$. Without loss of generality, let $\mu = 0$ and $Q = \Sigma^{-1}$, then we have

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

Now by observation, we can find that while the lefthand side of the equation is still a joint distribution, the righthand side now describes the structure of a MRF.

$$\frac{1}{Z} exp\{\sum_i \phi(x_i) + \sum_{i<j} \phi(x_i, x_j)\}$$

where $-\frac{1}{2}q_{ii}(x_i)^2$ can be viewed as the node potential $\phi(x_i)$, and $-q_{ij}x_i x_j$ can be viewed as the edge potential $\phi(x_i, x_j)$.

2.2 The Covariance and the Precision Matrices

Given the covariance matrix Σ and the precision matrix Q, we want to investigate the differences of their probability and graphical model interpretation.

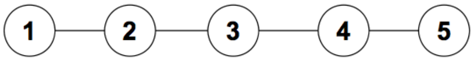
1. For Σ
   With the assumption that $x_1, \cdots, x_p$ follows multivariate Gaussian distribution, x_i and x_j are uncorrelated means they are independent. Therefore, $\Sigma_{i,j} = 0 \Rightarrow x_i \perp x_j$ or $P(x_i, x_j) = P(x_i)P(x_j)$. x_i and x_j are *marginally independent*. Through Σ, we solely concentrate on the relationship between x_i and x_j regardless the other nodes in the graph.

2. For Q
   $Q_{i,j} = 0 \Rightarrow x_i \perp x_j \mid \mathbf{x}_{\text{-}ij}$ or $P(x_i, x_j \mid \mathbf{x}_{\text{-}ij}) = P(x_i \mid \mathbf{x}_{\text{-}ij})P(x_j \mid \mathbf{x}_{\text{-}ij})$ x_i and x_j are *conditionally independent* given the rest of the graph. This is what we need to decide the structure of a graph. Specifically, every non-zero entry in Q corresponds to an edge in the MRF.

   To better understand the conditional independence induced by Q, we can look at $P(x_i, x_j \mid \mathbf{x}_{\text{-}ij}, Q)$. When $q_{ij} = 0$,

$$\begin{aligned} P(x_i, x_j \mid \mathbf{x}_{\text{-}ij}, Q) &= \underbrace{\frac{|Q|^{\frac{1}{2}}}{(2\pi)^{\frac{n}{2}}} exp\{-\frac{1}{2}\sum_{k \neq i,j} q_{kk}(x_k)^2 - \sum_{h,g \neq i,j, h<g} q_{hg}x_h x_g\}}_{constant} exp\{-\frac{1}{2}(q_{ii}(x_i)^2 + q_{jj}(x_j)^2)\} \\ &= P(x_i \mid \mathbf{x}_{\text{-}ij}, Q)P(x_j \mid \mathbf{x}_{\text{-}ij}, Q) \end{aligned}$$

Figure 1 shows a chain graphical model. The Q matrix well captures the structure of the chain, where zero indicates that there is no edge between the two nodes, and non-zero indicates an edge exists. However, if we represent a graph according to the Σ matrix, we will get a clique.



$$\Sigma^{-1} = \begin{pmatrix} 1 & 6 & 0 & 0 & 0 \\ 6 & 2 & 7 & 0 & 0 \\ 0 & 7 & 3 & 8 & 0 \\ 0 & 0 & 8 & 4 & 9 \\ 0 & 0 & 0 & 9 & 5 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 0.10 & 0.15 & -0.13 & -0.08 & 0.15 \\ 0.15 & -0.03 & 0.02 & 0.01 & -0.03 \\ -0.13 & 0.02 & 0.10 & 0.07 & -0.12 \\ -0.08 & 0.01 & 0.07 & -0.04 & 0.07 \\ 0.15 & -0.03 & -0.12 & 0.07 & 0.08 \end{pmatrix}$$

Figure 1: Q and Σ Matrices for a Chain Graphical Model

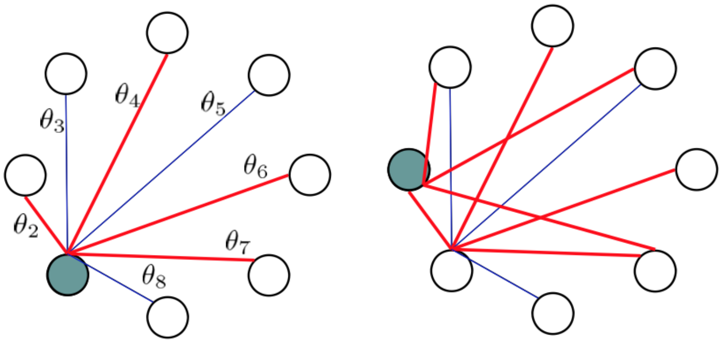
2.3 The Neighborhood Selection Algorithm

Now we know that whether the entries of Q are zero or non-zero can completely encode the structure of the graph. The next questions is how to learn the precision matrix Q. In the ideal situation, Σ is invertible, then we can use MLE to learn Σ. However, in the real world, an common scenario is that $p \gg n$ where p is the number of features, and n is the number of samples. In this case, Σ is not invertible. Thus we are going to learn a sparse GM by finding the non-zero entries in the sparse Q directly from data. The method we used to solve this problem is called *neighborhood selection algorithm* (Figure 2).

Here, we apply LASSO regression to each variable iteratively to find its neighbors. In the *ith* iteration, one variable is indicated as y_i, and all the other variables are represented by vector $\mathbf{x_i}$. We want to compute a vector θ_i in which the non-zero entry θ_i^j indicates an edge between the corresponding variable x_i^j and y_i. And θ_i is just the *ith* row or column in Q. Therefore we need to solve

$$\hat{\theta}_{\mathbf{i}} = \arg\min_{\theta_{\mathbf{i}}} l(\theta_{\mathbf{i}}) + \lambda \|\theta_{\mathbf{i}}\|_1$$

where $l(\theta_{\mathbf{i}}) = logP(y_i \mid \mathbf{x_i}, \theta_{\mathbf{i}})$, and $Y = \theta^T X$.



(a) Pick One Node

(b) Iteration

Figure 2: The Neighborhood Selection Algorithm

Having known the structure of the graph, we put Q back to the equation and perform MLE to estimate the parameters. For the discrete nodes, we can also perform this procedure and simply use L1 regularized logistic regression instead of L1 linear regression. Under finite data and high dimension condition, the graphical regression algorithm is ensured to be consistency.

3 MLE for Decomposable Undirected Graphical Models

Estimating the parameters in UGM is more challenging than in DGM for the reason that the partition function Z involves all parameters log-likelihood, thus log-likelihood is no more decomposable. In some cases, we need to do inferences (i.e. marginalization) to learn parameters even in the fully observed case. In this section, let's begin with a relatively simple case, the decomposable (triangulated) UGM.

3.1 Log Likelihood with Tabular Clique Potentials

To remove Z in the log likelihood, we introduce a notation for counts. The number of times a configuration $\mathbf{x}$ is observed in the dataset D can be represented as

$$m(\mathbf{x}) = \sum_n \delta(\mathbf{x}, \mathbf{x_n})$$

and

$$m(\mathbf{x}_C) = \sum_{\mathbf{x}_{V \setminus C}} m(\mathbf{x})$$

is the count for clique C. In terms of the counts, the likelihood is given by

$$p(D \mid \theta) = \prod_n \prod_{\mathbf{x}} p(\mathbf{x} \mid \theta)^{\delta(\mathbf{x},\mathbf{x_n})}$$

The log likelihood is

$$\begin{aligned} l = \log p(D \mid \theta) &= \sum_n \sum_{\mathbf{x}} \delta(\mathbf{x}, \mathbf{x_n}) \log p(\mathbf{x} \mid \theta) \\ &= \sum_{\mathbf{x}} \sum_n \delta(\mathbf{x}, \mathbf{x_n}) \log p(\mathbf{x} \mid \theta) \\ &= \sum_{\mathbf{x}} m(\mathbf{x}) \log(\frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C)) \\ &= \sum_C \sum_{\mathbf{x}_C} \underbrace{m(\mathbf{x}_C) \log \psi_C(\mathbf{x}_C)}_{l_1} - N \underbrace{\log Z}_{l_2} \end{aligned}$$

We can see that the marginal counts $m(\mathbf{x}_C)$ are the sufficient statistics for our model.

3.2 Maximum Likelihood Estimation

To find MLE, we take the derivatives of the log likelihood with respect to $\psi_C(\mathbf{x}_C)$ and set it to zero. The derivative of the first term can be obtained immediately.

$$\frac{\partial l_1}{\partial \psi_C(\mathbf{x}_C)} = \frac{m(\mathbf{x}_C)}{\psi_C(\mathbf{x}_C)}$$

Then we turn to the second term

$$\begin{aligned} \frac{\partial l_2}{\partial \psi_C(\mathbf{x}_C)} &= \frac{1}{Z} \frac{\partial}{\partial \psi_C(\mathbf{x}_C)} (\sum_{\tilde{x}} \prod_D \psi_D(\tilde{\mathbf{x}}_D)) \\ &= \frac{1}{Z} \sum_{\tilde{\mathbf{x}}} \delta(\tilde{\mathbf{x}}_C, \mathbf{x}_C) \frac{\partial}{\partial \psi_C(\mathbf{x}_C)} (\prod_D \psi_D(\tilde{\mathbf{x}}_D)) \\ &= \frac{1}{Z} \sum_{\tilde{\mathbf{x}}} \delta(\tilde{\mathbf{x}}_C, \mathbf{x}_C) \prod_{D \neq C} \psi_D(\tilde{\mathbf{x}}_D) \\ &= \sum_{\tilde{\mathbf{x}}} \delta(\tilde{\mathbf{x}}_C, \mathbf{x}_C) \frac{1}{\psi_C(\tilde{\mathbf{x}}_C)} \frac{1}{Z} \prod_D \psi_D(\tilde{\mathbf{x}}_D) \\ &= \frac{1}{\psi_C(\mathbf{x}_C)} \sum_{\tilde{\mathbf{x}}} \delta(\tilde{\mathbf{x}}_C, \mathbf{x}_C) p(\tilde{\mathbf{x}}) \\ &= \frac{p(\mathbf{x}_C)}{\psi_C(\mathbf{x}_C)} \end{aligned}$$

Note that when taking derivative of l_2, $\mathbf{x}_C$ is fixed.

Thus,

$$\frac{\partial l}{\partial \psi_C(\mathbf{x}_C)} = \frac{m(\mathbf{x}_C)}{\psi_C(\mathbf{x}_C)} - N\frac{p(\mathbf{x}_C)}{\psi_C(\mathbf{x}_C)} = 0$$

The MLE of the parameters is

$$\hat{p}_{MLE}(\mathbf{x}_C) = \frac{m(\mathbf{x}_C)}{N} = \tilde{p}(\mathbf{x}_C)$$

This tells us an important characterization of maximum likelihood estimates: *for each clique, the model marginals must be equal to the observed marginals (empirical counts)*. However, it doesn't tell us the MLE of the parameters, $\psi_C(\mathbf{x}_C)$ themselves appear implicitly in these equations.

3.3 Decomposable Models

When a model G is decomposable, its joint distribution can be represented as

$$p(\mathbf{x}) = \frac{\prod_C \psi_C(\mathbf{x}_C)}{\prod_S \varphi_S(\mathbf{x}_S)}$$

If G's potentials are defined on maximal cliques, we can find maximum likelihood estimates by inspection. That is, to compute the clique potentials, just set them to the empirical marginals or conditionals, i.e., the separator must be divided into one of its neighbors.

Figure 3 gives us an example of a three node chain model. Its probability can be written as

$$p(x_1, x_2, x_3) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)$$

The maximum likelihood estimates of its clique potentials can be given as

$$\tilde{\psi}_{12,MLE}(x_1, x_2) = \tilde{p}(x_1, x_2)$$
$$\tilde{\psi}_{23,MLE}(x_2, x_3) = \frac{\tilde{p}(x_2, x_3)}{\tilde{p}(x_2)}$$

which also implies that $Z = 1$.

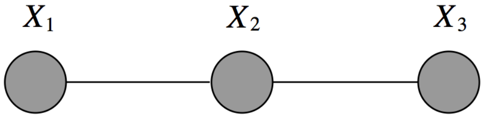


Figure 3: A Three Node Markov Chain

4 MLE for Non-decomposable Undirected Graphical Models

For now we know how to do MLE for decomposable graphical model, however if the clique potential does not directly correspond to the clique marginal, then the graph is non-decomposable. How can we do MLE for non-decomposable graphical model? If the potentials are tabular, we can use Iterative Proportional Fitting (IPF) algorithm and if the potentials are themselves functions of their own parameters, then we can use Generalized Iterative Scaling (GIS) algorithm.

4.1 Iterative Proportional Fitting

Iterative Proportional Fitting (IPF) can be used when the potentials in the undirected graphical model are tabular. It is an iterative algorithm, and hoping that the iterations can converge to a 'fixed point' – the solution for the original implicit equations. One nice thing about IPF is that it is not only a fixed-point algorithm, but also a a coordinate ascent algorithm, so it is guaranteed to converge.

Let's first derive the update rule for each iteration. Set the derivative of likelihood function equal to 0, we can get

$$\frac{m(x_c)}{N\psi_c(x_c)} = \frac{p(x_c)}{\psi_c(x_c)}$$

we can rewrite $\frac{m(x_c)}{N}$ as $\tilde{p}(x_c)$, the empirical marginal of x_c, so

$$\frac{\tilde{p}(x_c)}{\psi_c(x_c)} = \frac{p(x_c)}{\psi_c(x_c)}$$

Note that our goal $\psi_c(x_c)$ is on both side of the equation, so we can not solving it in closed-form from this equation, however we can fix $\psi_c(x_c)$ on the right side and solve for it on the left hand side. At the end we create a rule for iterative update:

$$\psi_c^{(t+1)}(x_c) = \psi_c^{(t)}(x_c)\frac{\tilde{p}(x_c)}{p^{(t)}(x_c)}$$

Also note that in the equation above, $p^{(t)}(x_c)$ is the clique marginal, not the empirical marginal, so we have to do inference for $p^{(t)}(x_c)$ in each iteration. We do the update for all the cliques in each iteration. and it can be proved that it is a coordinate ascent algorithm, where the coordinates are parameters of clique potentials:
First we take the derivative of the log likelihood with respect to the coordinate $\psi_c(x_c)$, for fixed c and varying x_c.

$$\frac{\partial l}{\partial \psi_c(x_c)} = \frac{m(x_c)}{\psi_c(x_c)} - \frac{N}{Z}\sum_{\hat{x}} \delta(\hat{x}_c, x_c) \prod_{D \neq C} \psi_D(\hat{x}_D)$$

To reflect that D is being fixed, we can add an iteration superscript to ψ_D on the right side.

$$\frac{\partial l}{\partial \psi_c(x_c)} = \frac{m(x_c)}{\psi_c^{(t+1)}(x_c)} - \frac{N}{Z^{(t+1)}}\sum_{\hat{x}} \delta(\hat{x}_c, x_c) \prod_{D \neq C} \psi_D^{(t)}(\hat{x}_D)$$

Also, one of IPF's properties is that Z remains constant during the iteration, so $Z^{(t+1)} = Z^{(t)}$, so

$$\begin{aligned}
\frac{\partial l}{\partial \psi_c(x_c)} &= \frac{m(x_c)}{\psi_c^{(t+1)}(x_c)} - \frac{N}{Z^{(t+1)}}\sum_{\hat{x}} \delta(\hat{x}_c, x_c) \prod_{D \neq C} \psi_D^{(t)}(\hat{x}_D) \\
&= \frac{m(x_c)}{\psi_c^{(t+1)}(x_c)} - \frac{N}{Z^{(t)}}\sum_{\hat{x}} \delta(\hat{x}_c, x_c) \prod_{D \neq C} \psi_D^{(t)}(\hat{x}_D) \\
&= \frac{m(x_c)}{\psi_c^{(t+1)}(x_c)} - \frac{N}{\psi^{(t)}(x_c)}\sum_{\hat{x}} \delta(\hat{x}_c, x_c) \frac{1}{Z^{(t)}} \prod_{D} \psi_D^{(t)}(\hat{x}_D) \\
&= \frac{m(x_c)}{\psi_c^{(t+1)}(x_c)} - \frac{N}{\psi^{(t)}(x_c)} p^{(t)}(x_c).
\end{aligned}$$

Now we can see that the IPF update function would set the derivative of log-likelihood above to zero, so the algorithm is coordinate ascent, it will increase the log-likelihood and finally converge to a global maximum.

4.2 Feature Based Model

For most of the graphical model we deal with, the potentials will not be tabular, but will themselves be functions of some parameters. This is because for large cliques, defining tabular potential functions are exponentially costly for inference and would have exponential numbers of parameters to learn. With limited data and time, it is impossible to learn when cliques become very large. Feature-based Clique potentials solve this problem by using a less general parameterization of the clique potentials, that is, for each potential, it defines a set of feature on it, so the parameter space is now the space of features, which can be controlled by ourselves.

For example, consider a clique: three consecutive characters in a string of English text. The full joint clique potential would be $26^3 - 1$, because we have 26 letters in English, and this is a huge potential space. But we can define features based on the three characters, such as whether they are 'ing' or 'ion', and each feature has only 2 possible values as they are binary. So suppose we define n features, we only have to estimate n parameters. Of course we can define more complicated feature than binary feature, such as continuous features.

Let's represent clique potentials as $\psi_c(x_c) = exp(\sum_k \theta_k f_k(x_c))$, we can treat each feature function as a 'micropotential' function. Also note that if the feature functions are indicator function per combination of x_c, we can recover the standard tabular potential. To combine feature into the probability model:

$$\begin{aligned} p(x) &= \frac{1}{Z(\theta)} \prod_c \psi_c(X_c) \\ &= \frac{1}{Z(\theta)} exp \sum_c \sum_k \theta_k f_k(x_c) \end{aligned}$$

we can simplify this form to

$$p(x) = \frac{1}{Z(\theta)} exp \sum_i \theta_i f_i(x_{c_i})$$

This is the form of exponential family model, and features are sufficient statistics. So now our goal is to find MLE under the above form.

4.3 Generalized Iterative Scaling

One method to solve this problem is Generalized Iterative Scaling (GIS) algorithm. It is also a iterative algorithm and try to attack the lower bound of the scaled likelihood function. Scaled likelihood function of UGM can be expressed as:

$$\begin{aligned} \hat{l}(\theta; D) &= l(\theta; D)/N \\ &= \sum_x \hat{p}(x) log p(x|\theta) \\ &= \sum_x \hat{p}(x) \sum_i \theta_i f_i(x) - log Z(\theta) \end{aligned}$$

$Z(\theta)$ is in the log, and we want to get rid of the log, so we can use the linear upper bound of logarithm $logZ(\theta) \leq \mu Z(\theta) - log\mu - 1$. This bound holds for all μ, so we can set $\mu = Z^{-1}(\theta^{(t)})$. Now we have:

$$\hat{l}(\theta; D) \geq \sum_x \hat{p}(x) \sum_i \theta_i f_i(x) - \frac{Z(\theta)}{Z(\theta^{(t)})} - logZ(\theta^{(t)}) + 1$$

We can see that now the first part of the equation is a linear combination of coefficient that we want to learn, and the second part $Z(\theta)$ is now not in logarithm form. We define $\Delta\theta_i^{(t)} = \theta_i - \theta_i^{(t)}$, that is the difference between old and new version of θ. Then we can plug $\Delta\theta_i^{(t)}$ into the lower bound:

$$\begin{aligned}\hat{l}(\theta; D) &\geq \sum_x \hat{p}(x) \sum_i \theta_i f_i(x) - \frac{Z(\theta)}{Z(\theta^{(t)})} - logZ(\theta^{(t)}) + 1 \\ &= \sum_i \theta_i \sum_x \hat{p}(x) f_i(x) - \sum_x p(x|\theta^{(t)}) exp \sum_i \Delta\theta_i^{(t)} f_i(x) - logZ(\theta^{(t)}) + 1\end{aligned}$$

We assume $f_i(x) \geq 0$ and $\sum_i f_i(x) = 1$, then we can have a inequality which has similar form as Jense's inequality: $exp(\sum_i \pi_i x_i) \leq \sum_i \pi_i exp(x_i)$. so we can get the $f_i(x)$ out of the exp in the above equation, so we have:

$$\hat{l}(\theta; D) \geq \sum_i \theta_i \sum_x \hat{p}(x) f_i(x) - \sum_x p(x|\theta^{(t)}) \sum_i f_i(x) exp \Delta\theta_i^{(t)} - logZ(\theta^{(t)}) + 1$$

Take the derivative and set it to zero we can have the closed-form solution of $\Delta\theta$:

$$e^{\Delta\theta_i^{(t)}} = \frac{\sum_x \hat{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)} Z(\theta^{(t)})$$

We also have a relationship between the update function of $\Delta\theta$ and total distribution:

$$\begin{aligned}\theta_i^{(t+1)} &= \theta_i^{(t)} + \Delta\theta_i^{(t)} \\ p^{(t+1)}(x) &= p^{(t)} \prod_i e^{\Delta\theta_i^{(t)} f_i(x)}\end{aligned}$$

so the update rule for GIS is:

$$\begin{aligned}p^{(t+1)}(x) &= p^{(t)}(x) \prod_i \left(\frac{\sum_x \hat{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)}\right)^{(f_i(x))} \\ \theta_i^{(t+1)} &= \theta_i^{(t)} + log\left(\frac{\sum_x \hat{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)}\right)\end{aligned}$$

Similar to IPF, GIS also has to do inference in each iteration, that is to compute the expectation over the feature function $\sum_x p^{(x)}(x) f_i(x)$, so estimate a fully observed MRF can be very difficult.