Machine Learning

Learning Graphical Models

Eric Xing

Lecture 12, August 14, 2010

Reading:
Inference and Learning

- A BN $M$ describes a unique probability distribution $P$

- Typical tasks:
  - Task 1: How do we answer queries about $P$?
    - We use inference as a name for the process of computing answers to such queries
    - So far we have learned several algorithms for exact and approx. inference
  
  - Task 2: How do we estimate a plausible model $M$ from data $D$?
    i. We use learning as a name for the process of obtaining point estimate of $M$.
    ii. But for Bayesian, they seek $p(M|D)$, which is actually an inference problem.
    iii. When not all variables are observable, even computing point estimate of $M$ need to do inference to impute the missing data.
Learning Graphical Models

The goal:

Given set of independent samples (assignments of random variables), find the best (the most likely?) graphical model (both the graph and the CPDs)

\[(B,E,A,C,R) = (T,F,F,T,F)\]
\[(B,E,A,C,R) = (T,F,T,T,F)\]
\[\ldots\]
\[(B,E,A,C,R) = (F,T,T,T,F)\]

<table>
<thead>
<tr>
<th>E</th>
<th>B</th>
<th>(P(A \mid E,B))</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>b</td>
<td>0.9</td>
</tr>
<tr>
<td>e</td>
<td>b</td>
<td>0.2</td>
</tr>
<tr>
<td>e</td>
<td>b</td>
<td>0.9</td>
</tr>
<tr>
<td>e</td>
<td>b</td>
<td>0.01</td>
</tr>
</tbody>
</table>

\(E, B\) \(A\) \(R\) \(C\)
Learning Graphical Models

● Scenarios:
  ● completely observed GMs
    ● directed ✓
    ● undirected ✓
  ● partially observed GMs
    ● directed ✓
    ● undirected (an open research topic)

● Estimation principles:
  ● Maximal likelihood estimation (MLE) ✓
  ● Bayesian estimation
  ● Maximal conditional likelihood
  ● Maximal "Margin"

● We use learning as a name for the process of estimating the parameters, and in some cases, the topology of the network, from data.
ML Parameter Est. for completely observed GMs of given structure

- The data:

\[ \{(z^{(1)}, x^{(1)}), (z^{(2)}, x^{(2)}), (z^{(3)}, x^{(3)}), \ldots (z^{(N)}, x^{(N)})\} \]
The basic idea underlying MLE

- Likelihood
  (for now let's assume that the structure is given):

\[
L(\theta \mid X) = p(X \mid \theta) = p(X_1 \mid \theta_1)p(X_2 \mid \theta_2)p(X_3 \mid X_3, X_3; \theta_3)
\]

- Log-Likelihood:

\[
l(\theta \mid X) = \log p(X \mid \theta) = \log p(X_1 \mid \theta_1) + \log p(X_2 \mid \theta_2) + \log p(X_3 \mid X_3, X_3, \theta_3)
\]

- Data log-likelihood

\[
l(\theta \mid DATA) = \log \prod_{n} p(X_n \mid \theta) = \sum_{n} \log p(X_{n,1} \mid \theta_1) + \sum_{n} \log p(X_{n,2} \mid \theta_2) + \sum_{n} \log p(X_{n,3} \mid X_{n,1}X_{n,2}, \theta_3)
\]

- MLE

\[
\{\theta_1, \theta_2, \theta_3\}_{MLE} = \arg \max l(\theta \mid DATA)
\]

\[
\theta_1^* = \arg \max \sum_n \log p(X_{n,1} \mid \theta_1), \quad \theta_2^* = \arg \max \sum_n \log p(X_{n,2} \mid \theta_2), \quad \theta_3^* = \arg \max \sum_n \log p(X_{n,3} \mid X_{n,1}X_{n,2}, \theta_3)
\]
Example 1: conditional Gaussian

- The completely observed model:
  - \( Z \) is a class indicator vector
    \[
    Z = \begin{bmatrix}
    Z^1 \\
    Z^2 \\
    \vdots \\
    Z^M
    \end{bmatrix},
    \]
    where \( Z^m = [0,1] \), and \( \sum Z^m = 1 \)
    and a datum is in class \( m \) w.p. \( \pi_i \)
    \[
    p(z^i = 1 | \pi) = \pi_i = \pi_1^z \times \pi_2^z \times \ldots \times \pi_M^z
    \]
    \[
    p(Z) = \prod_m \pi_m^z
    \]
  - \( X \) is a conditional Gaussian variable with a class-specific mean
    \[
    p(x | z^m = 1, \mu, \sigma) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ \frac{1}{2\sigma^2} (x - \mu_m)^2 \right\}
    \]
    \[
    p(x | z, \mu, \sigma) = \prod_m N(x | \mu_m, \sigma)^z
    \]
Example 1: conditional Gaussian

- **Data log-likelihood**

\[
l(\theta \mid D) = \log \prod_n p(z_n, x_n) = \log \prod_n p(z_n \mid \pi)p(x_n \mid z_n, \mu, \sigma)
\]

\[
= \sum_n \log p(z_n \mid \pi) + \sum_n \log p(x_n \mid z_n, \mu, \sigma)
\]

\[
= \sum_n \sum_m \pi_m^z \log \pi_m + \sum_n \sum_m N(x_n \mid \mu_m, \sigma) \frac{1}{2\sigma^2}(x_n - \mu_m)^2 + C
\]

- **MLE**

\[
\pi_m^* = \arg \max l(\theta \mid D), \quad \Rightarrow \frac{\partial}{\partial \pi_m} l(\theta \mid D) = 0, \forall m, \quad \text{s.t.} \sum_m \pi_m = 1
\]

\[
\Rightarrow \pi_m^* = \frac{\sum_n z_n^m}{N} = \frac{n_m}{N}
\]

\[
\text{the fraction of samples of class } m
\]

\[
\mu_m^* = \arg \max l(\theta \mid D), \quad \Rightarrow \mu_m^* = \frac{\sum_n z_n^m x_n}{\sum_n z_n^m} = \frac{\sum_n z_n^m x_n}{n_m}
\]

\[
\text{the average of samples of class } m
\]
Example 2: HMM: two scenarios

- **Supervised learning**: estimation when the “right answer” is known
  - **Examples**:
    - GIVEN: a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands
    - GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

- **Unsupervised learning**: estimation when the “right answer” is unknown
  - **Examples**:
    - GIVEN: the porcupine genome; we don’t know how frequent are the CpG islands there, neither do we know their composition
    - GIVEN: 10,000 rolls of the casino player, but we don’t see when he changes dice

- **QUESTION**: Update the parameters $\theta$ of the model to maximize $P(x|\theta)$ --- Maximal likelihood (ML) estimation
Recall definition of HMM

- Transition probabilities between any two states
  \[ p(y_t^i = 1 \mid y_{t-1}^i = 1) = a_{i,j}, \]
  or
  \[ p(y_t \mid y_{t-1}^i = 1) \sim \text{Multinomial}(a_{i,1}, a_{i,2}, \ldots, a_{i,M}), \forall i \in \mathbb{I}. \]

- Start probabilities
  \[ p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M). \]

- Emission probabilities associated with each state
  \[ p(x_t \mid y_t^i = 1) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, \ldots, b_{i,K}), \forall i \in \mathbb{I}. \]
  or in general:
  \[ p(x_t \mid y_t^i = 1) \sim f(\cdot \mid \theta_i), \forall i \in \mathbb{I}. \]
Supervised ML estimation

- Given \( x = x_1 \ldots x_N \) for which the true state path \( y = y_1 \ldots y_N \) is known,

\[
\ell(\theta; x, y) = \log p(x, y) = \log \prod_n \left( \prod_{t=2}^T p(y_{n,t} | y_{n,t-1}) \prod_{t=1}^T p(x_{n,t} | x_{n,t}) \right)
\]

- Define:
  \( A_{ij} = \text{# times state transition } i \rightarrow j \text{ occurs in } y \)
  \( B_{ik} = \text{# times state } i \text{ in } y \text{ emits } k \text{ in } x \)

- We can show that the maximum likelihood parameters \( \theta \) are:

\[
\begin{align*}
\alpha_{ij}^{ML} &= \frac{\#(i \rightarrow j)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=1}^T y_{n,t}^i y_{n,t}^j}{\sum_n \sum_{t=2}^T y_{n,t-1}^i} = \frac{A_{ij}}{\sum_j A_{ij}} \\
\beta_{ik}^{ML} &= \frac{\#(i \rightarrow k)}{\#(i \rightarrow \bullet)} = \frac{\sum_n \sum_{t=1}^T y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^T y_{n,t}^i} = \frac{B_{ik}}{\sum_k B_{ik}}
\end{align*}
\]

- If \( y \) is continuous, we can treat \( \{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\} \) as \( N \times T \) observations of, e.g., a Gaussian, and apply learning rules for Gaussian …
Supervised ML estimation, ctd.

- **Intuition:**
  - When we know the underlying states, the best estimate of $\theta$ is the average frequency of transitions & emissions that occur in the training data

- **Drawback:**
  - Given little data, there may be **overfitting**:
    - $P(x|\theta)$ is maximized, but $\theta$ is unreasonable
      - 0 probabilities – VERY BAD

- **Example:**
  - Given 10 casino rolls, we observe
    \[
    x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3 \\
    \]
  - Then:
    \[
    a_{FF} = 1; \quad a_{FL} = 0 \\
    b_{F1} = b_{F3} = .2; \\
    b_{F2} = .3; b_{F4} = 0; b_{F5} = b_{F6} = .1
    \]
Pseudocounts

- Solution for small training sets:
  - Add pseudocounts
    \[ A_{ij} = \# \text{ times state transition } i \rightarrow j \text{ occurs in } y + R_{ij} \]
    \[ B_{ik} = \# \text{ times state } i \text{ in } y \text{ emits } k \text{ in } x + S_{ik} \]
  - \( R_{ij}, S_{ij} \) are pseudocounts representing our prior belief
  - Total pseudocounts: \( R_i = \sum_j R_{ij}, S_i = \sum_k S_{ik} \)
    - --- "strength" of prior belief,
    - --- total number of imaginary instances in the prior

- Larger total pseudocounts \( \Rightarrow \) strong prior belief

- Small total pseudocounts: just to avoid 0 probabilities --- smoothing

- This is equivalent to Bayesian est. under a uniform prior with "parameter strength" equals to the pseudocounts
MLE for general BN parameters

- If we assume the parameters for each CPD are globally independent, and all nodes are fully observed, then the log-likelihood function decomposes into a sum of local terms, one per node:

$$\ell(\theta; D) = \log p(D | \theta) = \log \prod_{X_i} \left( \prod_{x_i} p(x_{n,i} | x_{n,\pi_i}, \theta_i) \right) = \sum_i \left( \sum_n \log p(x_{n,i} | x_{n,\pi_i}, \theta_i) \right)$$
Example: decomposable likelihood of a directed model

- Consider the distribution defined by the directed acyclic GM:

\[ p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_1) p(x_3 \mid x_1, \theta_3) p(x_4 \mid x_2, x_3, \theta_1) \]

- This is exactly like learning four separate small BNs, each of which consists of a node and its parents.
E.g.: MLE for BNs with tabular CPDs

- Assume each CPD is represented as a table (multinomial) where

\[ \theta_{ijk} \overset{\text{def}}{=} p(X_i = j \mid X_{\pi_i} = k) \]

- Note that in case of multiple parents, \( X_{\pi_i} \) will have a composite state, and the CPD will be a high-dimensional table

- The sufficient statistics are counts of family configurations

\[ n_{ijk} \overset{\text{def}}{=} \sum_n x_{n,i}^j x_{n,\pi_i}^k \]

- The log-likelihood is

\[ \ell(\theta; D) = \log \prod_{i,j,k} \theta_{ijk}^{n_{ijk}} = \sum_{i,j,k} n_{ijk} \log \theta_{ijk} \]

- Using a Lagrange multiplier to enforce \( \sum_j \theta_{ijk} = 1 \), we get:

\[ \theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{i,j',k} n_{ij'k}} \]
Learning partially observed GMs

- The data:
  \( \{(x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(N)})\} \)
What if some nodes are not observed?

- Consider the distribution defined by the directed acyclic GM:

\[ p(x \mid \theta) = p(x_1 \mid \theta_1) p(x_2 \mid x_1, \theta_1) p(x_3 \mid x_1, \theta_3) p(x_4 \mid x_2, x_3, \theta_1) \]

- Need to compute \( p(x_H \mid x_V) \) → inference
Recall: EM Algorithm

- A way of maximizing likelihood function for latent variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some “missing” or “unobserved” data from observed data and current parameters.
  2. Using this “complete” data, find the maximum likelihood parameter estimates.

- Alternate between filling in the latent variables using the best guess (posterior) and updating the parameters based on this guess:

  - E-step: \( q^{t+1} = \arg\max_q F(q, \theta^t) \)
  - M-step: \( \theta^{t+1} = \arg\max_\theta F(q^{t+1}, \theta^t) \)

- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.
EM for general BNs

while not converged

% E-step
for each node $i$

$$ESS_i = 0$$ % reset expected sufficient statistics
for each data sample $n$
do inference with $X_{n,H}$
for each node $i$

$$ESS_i += \langle SS_i(x_{n,i}, x_{n,\pi_i}) \rangle_{p(x_{n,H}|x_{n,-H})}$$

% M-step
for each node $i$

$$\theta_i := \text{MLE}(ESS_i)$$
Example: HMM

- **Supervised learning**: estimation when the “right answer” is known
  - **Examples**:
    - **GIVEN**: a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands
    - **GIVEN**: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

- **Unsupervised learning**: estimation when the “right answer” is unknown
  - **Examples**:
    - **GIVEN**: the porcupine genome; we don’t know how frequent are the CpG islands there, neither do we know their composition
    - **GIVEN**: 10,000 rolls of the casino player, but we don’t see when he changes dice

- **QUESTION**: Update the parameters $\theta$ of the model to maximize $P(x|\theta)$ -- Maximal likelihood (ML) estimation
The Baum Welch algorithm

- The complete log likelihood

\[ \ell_c(\theta; x, y) = \log p(x, y) = \log \sum_n \left( \prod_{t=2}^{T} p(y_{n,t} | y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} | x_{n,t}) \right) \prod_{n} p(y_{n,1} | x_{n}) \]

- The expected complete log likelihood

\[ \langle \ell_c(\theta; x, y) \rangle = \sum_n \left( \langle y_{n,1}^i \rangle p(y_{n,1}^i | x_n) \log \pi_i \right) + \sum_{n,t=2}^{T} \left( \langle y_{n,t-1}^i y_{n,t}^j \rangle p(y_{n,t-1}^i, y_{n,t}^j | x_n) \log a_{i,j} \right) + \sum_{n,t=1}^{T} \left( x_{n,t}^k \langle y_{n,t}^i \rangle p(y_{n,t}^i | x_n) \log b_{i,k} \right) \]

- EM
  - The E step
    \[ \gamma_{n,t}^i = \langle y_{n,t}^i \rangle = p(y_{n,t}^i = 1 | x_n) \]
    \[ \xi_{n,t}^{i,j} = \langle y_{n,t-1}^i y_{n,t}^j \rangle = p(y_{n,t-1}^i = 1, y_{n,t}^j = 1 | x_n) \]
  - The M step ("symbolically" identical to MLE)
    \[ \pi_i^{ML} = \frac{\sum_{n} \gamma_{n,t}^i}{N} \]
    \[ a_{i,j}^{ML} = \frac{\sum_{n} \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_{n} \sum_{t=1}^{T-1} \gamma_{n,t}^i} \]
    \[ b_{i,k}^{ML} = \frac{\sum_{n} \sum_{t=1}^{T} \gamma_{n,t}^i x_{n,t}^k}{\sum_{n} \sum_{t=1}^{T-1} \gamma_{n,t}^i} \]
Unsupervised ML estimation

- Given $x = x_1...x_N$ for which the true state path $y = y_1...y_N$ is unknown,

- **EXPECTATION MAXIMIZATION**

0. Starting with our best guess of a model $M$, parameters $\theta$:

1. Estimate $A_{ij}, B_{ik}$ in the training data
   - How? $A_{ij} = \sum_{n,t} \langle y_{n,t-1}^i y_{n,t}^j \rangle$, $B_{ik} = \sum_{n,t} \langle y_{n,t}^i \rangle x_{n,t}^k$,

2. Update $\theta$ according to $A_{ij}, B_{ik}$
   - Now a "supervised learning" problem

3. Repeat 1 & 2, until convergence

This is called the Baum-Welch Algorithm

We can get to a provably more (or equally) likely parameter set $\theta$ each iteration
ML Structural Learning for completely observed GMs

Data

\((x_1^{(1)}, \ldots, x_n^{(1)})\)
\((x_1^{(2)}, \ldots, x_n^{(2)})\)
\(\ldots\)
\((x_1^{(M)}, \ldots, x_n^{(M)})\)
\[ \ell(G, D) = \log p(D \mid \theta_G, G) \]

\[ = \log \prod_n \left( \prod_i p(x_{n,i} \mid x_{n,\pi_i(G)}, \theta_{i|\pi_i(G)}) \right) \]

\[ = \sum_i \left( \sum_n \log p(x_{n,i} \mid x_{n,\pi_i(G)}, \theta_{i|\pi_i(G)}) \right) \]

\[ = M \sum_i \left( \sum_{x_i, x_{\pi_i(G)}} \frac{\text{count}(x_i, x_{\pi_i(G)})}{M} \log p(x_i \mid x_{\pi_i(G)}, \theta_{i|\pi_i(G)}) \right) \]

\[ = M \sum_i \left( \sum_{x_i, x_{\pi_i(G)}} \hat{p}(x_i, x_{\pi_i(G)}) \log p(x_i \mid x_{\pi_i(G)}, \theta_{i|\pi_i(G)}) \right) \]

From sum over data points to sum over count of variable states
Information Theoretic Interpretation of ML (con'd)

\[ \ell(\theta_G, G; D) = \log \hat{p}(D | \theta_G, G) \]

\[ = M \sum_i \left( \sum_{x_i, \pi_i(G)} \hat{p}(x_i, x_{\pi_i(G)}) \log \hat{p}(x_i, \pi_i(G), \theta_{i|\pi_i(G)}) \right) \hat{p}(x_i) \]

\[ = M \sum_i \left( \sum_{x_i, \pi_i(G)} \hat{p}(x_i, x_{\pi_i(G)}) \log \frac{\hat{p}(x_i, x_{\pi_i(G)}, \theta_{i|\pi_i(G)})}{\hat{p}(x_{\pi_i(G)}) \hat{p}(x_i)} \right) \hat{p}(x_i) \]

\[ = M \sum_i \left( \sum_{x_i, \pi_i(G)} \hat{p}(x_i, x_{\pi_i(G)}) \log \frac{\hat{p}(x_i, x_{\pi_i(G)}, \theta_{i|\pi_i(G)})}{\hat{p}(x_{\pi_i(G)}) \hat{p}(x_i)} \right) \hat{p}(x_i) \]

\[ = M \sum_i \hat{I}(x_i, x_{\pi_i(G)}) - M \sum_i \hat{H}(x_i) \]

Decomposable score and a function of the graph structure
Structural Search

- How many graphs over $n$ nodes? $O(2^{n^2})$

- How many trees over $n$ nodes? $O(n!)$

- But it turns out that we can find exact solution of an optimal tree (under MLE)!
  - Trick: in a tree each node has only one parent!
  - Chow-liu algorithm
Chow-Liu tree learning algorithm

- Objection function:

$$
\ell(\theta_G, G; D) = \log \hat{p}(D | \theta_G, G) = M \sum_i \hat{I}(x_i, x_{\pi_i(G)}) - M \sum_i \hat{H}(x_i) \quad \Rightarrow \quad C(G) = M \sum_i \hat{I}(x_i, x_{\pi_i(G)})
$$

- Chow-Liu:
  - For each pair of variable $x_i$ and $x_j$
    - Compute empirical distribution:
      $$\hat{p}(X_i, X_j) = \frac{\text{count}(x_i, x_j)}{M}$$
    - Compute mutual information:
      $$\hat{I}(X_i, X_j) = \sum_{x_i, x_j} \hat{p}(x_i, x_j) \log \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i) \hat{p}(x_j)}$$
  - Define a graph with node $x_1, \ldots, x_n$
    - Edge $(i, j)$ gets weight $\hat{I}(X_i, X_j)$
Chow-Liu algorithm (con'd)

- Objection function:

\[
\ell(\theta_G, G; D) = \log \hat{p}(D \mid \theta_G, G) \\
= M \sum_i \hat{I}(x_i, x_{\pi_i(G)}) - M \sum_i \hat{H}(x_i)
\]

\[
C(G) = M \sum_i \hat{I}(x_i, x_{\pi_i(G)})
\]

- Chow-Liu:
  
  Optimal tree BN
  
  - Compute maximum weight spanning tree
  
  - Direction in BN: pick any node as root, do breadth-first-search to define directions
  
  - I-equivalence:

\[
C(G) = I(A, B) + I(A, C) + I(C, D) + I(C, E)
\]
Structure Learning for general graphs

- **Theorem:**
  - The problem of learning a BN structure with at most $d$ parents is NP-hard for any (fixed) $d \geq 2$

- **Most structure learning approaches use heuristics**
  - Exploit score decomposition
  - Two heuristics that exploit decomposition in different ways
    - Greedy search through space of node-orders
    - Local search of graph structures
Inferring gene regulatory networks

- Network of cis-regulatory pathways

- Success stories in sea urchin, fruit fly, etc, from decades of experimental research
- Statistical modeling and automated learning just started
Gene Expression Profiling by Microarrays

[Diagram showing interactions between Receptors A and B, Kinases C, D, and E, Trans. Factor F, and Genes G and H.]
Structure Learning Algorithms

- **Structural EM (Friedman 1998)**
  - The original algorithm

- **Sparse Candidate Algorithm (Friedman et al.)**
  - Discretizing array signals
  - Hill-climbing search using local operators: add/delete/swap of a single edge
  - Feature extraction: Markov relations, order relations
  - Re-assemble high-confidence sub-networks from features

- **Module network learning (Segal et al.)**
  - Heuristic search of structure in a "module graph"
  - Module assignment
  - Parameter sharing
  - Prior knowledge: possible regulators (TF genes)
Learning GM structure

- Learning of best CPDs *given* DAG is easy
  - collect statistics of values of each node given specific assignment to its parents

- Learning of the graph topology (structure) is **NP-hard**
  - heuristic search must be applied, generally leads to a *locally* optimal network

- **Overfitting**
  - It turns out, that richer structures give higher likelihood $P(D|G)$ to the data (adding an edge is always preferable)

$$P(C | A) \leq P(C | A, B)$$

- more parameters to fit => more freedom => always exist more "optimal" CPD(C)

- **We prefer** *simpler* (more explanatory) networks
  - Practical scores **regularize** the likelihood improvement complex networks.
Learning Graphical Model Structure via Neighborhood Selection
Undirected Graphical Models

Why?

Sometimes an UNDIRECTED association graph makes more sense and/or is more informative

- gene expressions may be influenced by unobserved factor that are post-transcriptionally regulated

- The unavailability of the state of B results in a constrain over A and C
Gaussian Graphical Models

- Multivariate Gaussian density:
  \[ p(x \mid \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

- WOLG: let \( \mu = 0 \) \( Q = \Sigma^{-1} \)

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

- We can view this as a continuous Markov Random Field with potentials defined on every node and edge:
The covariance and the precision matrices

- **Covariance matrix** \( \Sigma \)

  \[ \Sigma_{i,j} = 0 \quad \Rightarrow \quad X_i \perp X_j \quad \text{or} \quad p(X_i, X_j) = p(X_i)p(X_j) \]

  - Graphical model interpretation?

- **Precision matrix** \( Q = \Sigma^{-1} \)

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

  - Graphical model interpretation?
Sparse precision vs. sparse covariance in GGM

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

\[ \Sigma^{-1}_{15} = 0 \iff X_1 \perp X_5 | X_{nbrs(1) \text{ or } nbrs(5)} \]

\[ \iff \]

\[ X_1 \perp X_5 \iff \Sigma_{15} = 0 \]
Another example

How to estimate this MRF?

What if $p \gg n$

- MLE does not exist in general!
- What about only learning a “sparse” graphical model?
  - This is possible when $s=o(n)$
  - Very often it is the structure of the GM that is more interesting …
The conditional dist. of a single node $i$ given the rest of the nodes can be written as:

$$p(X_i|X_{-i}) = \mathcal{N}(\mu_i + \Sigma_{X_iX_{-i}}\Sigma_{X_{-i}X_{-i}}^{-1}(X_{-i} - \mu_{X_{-i}}), \\
\Sigma_{X_iX_i} - \Sigma_{X_iX_{-i}}\Sigma_{X_{-i}X_{-i}}^{-1}\Sigma_{X_{-i}X_i})$$

- WOLG: let $\mu = 0$

$$p(X_i|X_{-i}) = \mathcal{N}(\Sigma_{X_iX_{-i}}\Sigma_{X_{-i}X_{-i}}^{-1}X_{-i}, \Sigma_{X_iX_i} - \Sigma_{X_iX_{-i}}\Sigma_{X_{-i}X_{-i}}^{-1}\Sigma_{X_{-i}X_i})$$

$$= \mathcal{N}(\hat{\sigma}_i^{-1}X_{-i}, q_{ii})$$

$$= \mathcal{N}(\frac{q_i^T}{-q_{ii}}, q_{ii})$$

We can write the following conditional auto-regression function for each node:
Conditional independence

- From

\[ p(X_i|X_{-i}) = \mathcal{N}(\frac{q_i^T}{-q_{ii}}X_{-i}, q_{ii}) \]

- Let:

\[ S_i \equiv \{ j : j \neq i, \theta_{ij} \neq 0 \} \]

- Given an estimate of the neighborhood \( s_i \), we have:

\[ p(X_i|X_{-i}) = p(X_i|X_s) \]

- Thus the neighborhood \( s_i \) defines the Markov blanket of node \( i \)
Recall lasso

\[ \hat{\theta}_i = \arg \min_{\theta_i} l(\theta_i) + \lambda_1 \| \theta_i \|_1 \]

where \[ l(\theta_i) = \log P(y_i|x_i, \theta_i). \]
Graph Regression

Neighborhood selection

Lasso:

\[
\hat{\theta} = \arg \min_{\theta} \sum_{t=1}^{T} l(\theta) + \lambda_1 \| \theta \|_1
\]
Graph Regression
It can be shown that:
given iid samples, and under several technical conditions (e.g., "irrepresentable"),
the recovered structured is "sparsistent" even when $p >> n$
Consistency

- **Theorem**: for the graphical regression algorithm, under certain verifiable conditions (omitted here for simplicity):

\[
P \left[ \hat{G}(\lambda_n) \neq G \right] = \mathcal{O}(\exp(-Cn^\epsilon)) \rightarrow 0
\]

Note the from this theorem one should see that the regularizer is not actually used to introduce an “artificial” sparsity bias, but a devise to ensure consistency under finite data and high dimension condition.
Learning (sparse) GGM

- Multivariate Gaussian over all continuous expressions

\[ p([x_1, \ldots, x_n]) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2} (\tilde{x} - \mu)^T \Sigma^{-1} (\tilde{x} - \mu) \right\} \]

- The precision matrix \( Q = \Sigma^{-1} \) reveals the topology of the (undirected) network

- Learning Algorithm: Covariance selection
  - Want a sparse matrix \( Q \)
  - As shown in the previous slides, we can use L_1 regularized linear regression to obtain a sparse estimate of the neighborhood of each variable
Recent trends in GGM:

- **Covariance selection (classical method)**
  - Dempster [1972]:
    - Sequentially pruning smallest elements in precision matrix
  - Drton and Perlman [2008]:
    - Improved statistical tests for pruning

- **L₁-regularization based method (hot !)**
  - Meinshausen and Bühlmann [Ann. Stat. 06]:
    - Used LASSO regression for neighborhood selection
  - Banerjee [JMLR 08]:
    - Block sub-gradient algorithm for finding precision matrix
  - Friedman et al. [Biostatistics 08]:
    - Efficient fixed-point equations based on a sub-gradient algorithm

Serious limitations in practice: breaks down when covariance matrix is not invertible

Structure learning is possible even when # variables > # samples
Learning Ising Model (i.e. pairwise MRF)

- Assuming the nodes are discrete, and edges are weighted, then for a sample \( x_d \), we have

\[
P(x_d|\Theta) = \exp \left( \sum_{i \in V} \theta^t_{i,i} x_{d,i} + \sum_{(i,j) \in E} \theta_{ij} x_{d,i} x_{d,j} - A(\Theta) \right)
\]

- It can be shown following the same logic that we can use L_1 regularized logistic regression to obtain a sparse estimate of the neighborhood of each variable in the discrete case.