Partially Observed Data

Shai Kazaz and Tal Gerbi
Structure Learning
Reminder: Structure Learning

• Now we’ll assume we have no prior knowledge of the structure of our model

• We want to learn both the structure and its parameters

• There are 2 reasons to do that:
  – Knowledge Discovery
  – Density Estimation
Reminder: Score and Search

• **Structure score:** assigning some \( \text{Score}(G : D) \) for every structure \( G \), that measures its quality

• **Structure search:** finding some graph \( G \) that maximizes the score function
Structure Learning: Outline

• **Structure Score**
• **Structure Search**
• **Learning Models with Hidden Variables (if time permits)**
Reminder: Structure Score

• Bayesian score:  \( \text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D} | \mathcal{G}) + \log P(\mathcal{G}). \)

• Marginal Likelihood:

\[
P(\mathcal{D} | \mathcal{G}) = \int_{\Theta_g} P(\mathcal{D} | \theta_g, \mathcal{G}) P(\theta_g | \mathcal{G}) d\theta_g.
\]

\[
P(\mathcal{D} | \mathcal{G}) = \prod_i \int_{\Theta_{X_i|Pa_{X_i}}} \prod_m P(x_i[m] | pa_{X_i}[m], \theta_{X_i|Pa_{X_i}}, \mathcal{G}) P(\theta_{X_i|Pa_{X_i}} | \mathcal{G}) d\theta_{X_i|Pa_{X_i}}
\]
Computing the Marginal Likelihood

• In order to evaluate the scoring function, we have to compute the marginal likelihood:

\[ P(\mathcal{D} \mid \mathcal{G}) = \int_{\Theta_\mathcal{G}} P(\mathcal{D} \mid \theta_\mathcal{G}, \mathcal{G})P(\theta_\mathcal{G} \mid \mathcal{G})d\theta_\mathcal{G}. \]

• In case of incomplete data, the computation of the integral includes summation over the unobserved variables.

• It is no longer true that the integral decomposes, as with the fully observed case.

• Can we at least approximate the marginal likelihood?
Method 1: Laplace Approximation

For a function $f(w) = e^{g(w)}$

it holds that

$$\int f(w)dw \approx f(w_0)\left|C\right|^{-\frac{1}{2}}(2\pi)^{\frac{1}{2}\dim(C)}$$

where

$$C = -\left[\frac{\partial^2 g(w)}{\partial x_i \partial x_j}\right]_{w=w_0}$$

and $w_0$ is a the maximum of $g(w)$
Method 1: Laplace Approximation – Cont.

• Taking \( g = \log P(D \mid \theta_G, G) + \log P(\theta_G \mid G) \) and applying Laplace Approximation, we get that:

\[
\log P(D \mid G) \approx \log P(D \mid \tilde{\theta}_G, G) + \frac{\text{dim}(C)}{2} \log(2\pi) - \frac{1}{2} \log |C|
\]

where:

\( \tilde{\theta}_G \) are the MAP parameters for \( G \)

\( C \) is the negative of the hessian of the log-likelihood function

• We can use that fact to estimate the score:

\[
\text{Score}_{\text{Laplace}}(G : D) = \log P(G) + \log P(D \mid \tilde{\theta}_G, G) + \frac{\text{dim}(C)}{2} \log(2\pi) - \frac{1}{2} \log |C|
\]
Method 2: BIC

• Define the BIC score for incomplete data:

\[
\text{score}_{\text{BIC}}(G : D) = \log P(D | \tilde{\theta}_G, G) - \frac{\log M}{2} \text{Dim}[G] + \log P(G) + \log P(\tilde{\theta}_G | G)
\]

where:

\(\tilde{\theta}_G\) are the MAP parameters for \(G\)

\(\text{Dim}[G]\) is the number of independence variables in \(G\)

• It can be shown that:

\[
\text{As } M \to \infty, \text{ we have that:}
\]

\[
\text{score}_{\text{Laplace}}(G : D) = \text{score}_{\text{BIC}}(G : D) + O(1)
\]
Method 3: Cheeseman-Stutz

- We already know how compute the Bayesian score efficiently for fully observed data
- What if we could extend our partial data to a fully observed one, and then use this fact?
- When data is missing, add the statistics given the MAP parameters
Method 3: Cheeseman-Stutz – Cont.

• More formally: extend the data set $D$ to $D_{G,\tilde{G}}^*$, and then approximate $P(D|G)$ with $P(D_{G,\tilde{G}}^*|G)$

• How do we extend the data? Exactly like we extended it in the E-Step of the EM Algorithm

• In other words, for a graph $G$ and a set of parameters $\theta$, we define:

$$M_{D_{G,\theta}^*}[x] = \overline{M}_{P(H/D,G,\theta)}[x]$$
Method 3: Cheeseman-Stutz – Cont.

- So we estimate $P(D / G)$ with $P(D^*_{G,\tilde{\theta}_G} / G)$
- It turns out this is not a very good estimation:

$$P(D | G) = \int \sum_H P(D, H | G)P(\theta | G)d\theta = \sum_H \int P(D, H | G)P(\theta | G)d\theta$$
$$P(D^*_{G,\theta_G} | G) = \int P(D^*_{G,\theta_G} | G)P(\theta | G)d\theta$$

- So estimate the difference:

$$\log P(D | G) - \log P(D^*_{G,\theta_G} | G) \approx \left( \log P(D | \theta_G, G) - \frac{1}{2} \dim(G) \log M \right) -$$
$$\left( \log P(D^*_{G,\theta_G} | \theta_G, G) - \frac{1}{2} \dim(G) \log M \right) = \log P(D | \theta_G, G) - \log P(D^*_{G,\theta_G} | \theta_G, G)$$
Method 3: Cheeseman-Stutz – Cont.

- We obtain that

\[
\log P(D \mid G) = \log P(D_{G, \theta_G}^* \mid G) + \log P(D \mid G) - \log P(D_{G, \theta_G}^* \mid G)
\]

\[
\approx \log P(D_{G, \theta_G}^* \mid G) + \log P(D \mid \theta_G, G) - \log P(D_{G, \theta_G}^* \mid \theta_G, G)
\]

- We can use that fact to estimate the score:

\[
Score_{CS}(G : D) = \log P(D_{G, \theta_G}^* \mid G) + \log P(D \mid \theta_G, G) - \log P(D_{G, \theta_G}^* \mid \theta_G, G)
\]

+ \log P(G)
Method 4: Candidate Method

- We know that:
  \[ P(D, \theta \mid G) = P(D \mid \theta, G) \cdot P(\theta \mid G) \]
  \[ P(D, \theta \mid G) = P(\theta \mid D, G) \cdot P(D \mid G) \]

- Therefore:
  \[ P(D \mid G) = \frac{P(D \mid \theta, G) \cdot P(\theta \mid G)}{P(\theta \mid D, G)} \]

- We can compute all these terms:
  \( P(D / \theta, G) \) can be found by solving an inference problem
  \( P(\theta / G) \) is the prior, which is usually given
  \( P(\theta / D, G) \) can be estimated by using MCMC sampling
Comparison between Scores

![Graphs showing comparison between scores for different models and parameters.]

<table>
<thead>
<tr>
<th>Generating Model</th>
<th>Error in cluster cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Laplace</td>
</tr>
<tr>
<td>n=32, d=4, M=400</td>
<td>0</td>
</tr>
<tr>
<td>n=64, d=4, M=400</td>
<td>0</td>
</tr>
<tr>
<td>n=128, d=4, M=400</td>
<td>0.2</td>
</tr>
<tr>
<td>n=64, d=4, M=400</td>
<td>0</td>
</tr>
<tr>
<td>n=64, d=6, M=400</td>
<td>0.4</td>
</tr>
<tr>
<td>n=64, d=8, M=400</td>
<td>0.2</td>
</tr>
<tr>
<td>n=64, d=4, M=50</td>
<td>0.2</td>
</tr>
<tr>
<td>n=64, d=4, M=100</td>
<td>0</td>
</tr>
<tr>
<td>n=64, d=4, M=200</td>
<td>0</td>
</tr>
</tbody>
</table>
Structure Learning: Outline

• Structure score
• **Structure Search**
• Learning Models with Hidden Variables
Reminder: Structure Search

• Given:
  – Data set: $D$
  – Set of possible network structures: $\mathcal{G}$
  – A scoring function: $\text{Score}$

• Find $\arg\max_{G \in \mathcal{G}} \left( \text{Score}(G : D) \right)$
Reminder: Score Decomposability

• We say that a scoring function $Score$ is decomposable if

$$Score(G : D) = \sum_i FamScore(X_i | Pa^G_{X_i} : D)$$

• In the fully observed case, some of the score functions are decomposable:
  – Likelihood
  – Bayesian (with parameter modularity)
Reminder: Previous Results for Structure Search

• In the case of fully observed data, we know how to:
  – Compute the best tree-structure network
    • For decomposable score functions
  – Compute the best network with a specific order on the nodes and maximum indegree $d$
    • For decomposable score functions
  – Perform heuristics in a search space
Reminder: The Search Space

• Each node in the graph is a network structure
• We connect a network structure $G$ to $G'$ if $G'=o(G)$ for the following operations:
  – Edge addition
  – Edge deletion
  – Edge reversal
Reminder: The Search Space – Cont.

• In the case where the scoring function is decomposable, we define:

\[ \delta(G : o) = \text{score}(o(G) : D) - \text{score}(G : D) \]

• We know that the followings hold:

  If \( o \) is “Add \( X \rightarrow Y \)” and \( X \rightarrow Y \notin G \), then

  \[ \delta(G : o) = \text{FamScore}(Y, Pa_Y^G \cup \{X\} : D) - \text{FamScore}(Y, Pa_Y^G : D). \]

  If \( o \) is “Delete \( X \rightarrow Y \)” and \( X \rightarrow Y \in G \), then

  \[ \delta(G : o) = \text{FamScore}(Y, Pa_Y^G - \{X\} : D) - \text{FamScore}(Y, Pa_Y^G : D). \]

  If \( o \) is “Reverse \( X \rightarrow Y \)” and \( X \rightarrow Y \in G \), then

  \[ \delta(G : o) = \text{FamScore}(X, Pa_X^G \cup \{Y\} : D) + \text{FamScore}(Y, Pa_Y^G - \{X\} : D) - \text{FamScore}(X, Pa_X^G : D) - \text{FamScore}(Y, Pa_Y^G : D). \]

• And also, that:

  If \( o \) is either “Add \( X \rightarrow Y \)” or “Delete \( X \rightarrow Y \)” and \( Pa_Y^G = Pa_Y^{G'} \), then \( \delta(G : o) = \delta(G' : o) \).

  If \( o \) is “Reverse \( X \rightarrow Y \),” \( Pa_Y^G = Pa_Y^{G'} \), and \( Pa_X^G = Pa_X^{G'} \), then \( \delta(G : o) = \delta(G' : o) \).
Indecomposability in Partially Observed Data

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
The algorithms that find the best structures among specific type (tree-structure networks, ordered networks with maximum indegree) rely on the score’s decomposability.

When performing hill climbing in the search space, we can no longer:

- Compute the score of neighbor networks efficiently
- Cache the scores in order to prevent repeated computations

The situation is even worse than in the case for the complete data, because the evaluation of the score functions involve complex computations, such as EM or MCMC procedures.
Heuristics in The Search Space

- The idea: estimate the change in score

- **Option 1**: assume that a step that was good relative to one structure is often probably good also relative to a closely related structure.

- **Option 2**: assume that only the distribution of the changed CPD can be optimized (and use gradient ascent or EM)
  - For some score functions, this provides a lower bound on the score
    - BIC Score
    - Cheeseman-Stutz Score (in practice)
  - We can use this fact to eliminate some of the candidate networks
Structural EM

• It can be shown that

\[
Score_{BIC}(G : D^*_{G_0, \tilde{\theta}_G}) - Score_{BIC}(G_0 : D^*_{G_0, \tilde{\theta}_G}) \leq \]

\[
Score_{BIC}(G : D) - Score_{BIC}(G_0 : D)
\]

Where \( \tilde{\theta}_{G_0} \) are the MAP parameters for \( G_0 \)

• Therefore, if we use the BIC score, and compute

\[
\delta_{D^*_{G_0, \tilde{\theta}_G}}(G : o) = Score_{BIC}(G : D^*_{G_0, \tilde{\theta}_G}) - Score_{BIC}(G_0 : D^*_{G_0, \tilde{\theta}_G})
\]

We get a lower bound on the true change in the BIC score
Algorithm 19.3 The structural EM algorithm for structure learning

Procedure Structural-EM ( 
    $G^0$, // Initial bayesian network structure over $X_1, \ldots, X_n$
    $\theta^0$, // Initial set of parameters for $G^0$
    $D$ // Partially observed data set
)

1  for each $t = 0, 1, \ldots$, until convergence
   // Optional parameter learning step
   $\theta^{t'} \leftarrow$ Expectation-Maximization($G^t, \theta^t, D$)
2  // Run EM to generate expected sufficient statistics for $D_{G^t, \theta^{t'}}^*$
4  $G^{t+1} \leftarrow$ Structure-Learn($D_{G^t, \theta^{t'}}^*$)
5  $\theta^{t+1} \leftarrow$ Estimate-Parameters($D_{G^t, \theta^{t'}, G^{t+1}}^*$)
6  return $G^t, \theta^t$
Structure Learning: Outline

- Structure Score
- Structure Search
- Learning Models with Hidden Variables
So far, we assumed that even a variable is never observed, we still know of its existence and its values.

In reality, we are not always aware of all the hidden variables in our model.

If performed right, introducing hidden variables can significantly simplify our model.

This, however, raises many questions:
- How many hidden variables should we add?
- Where in the network should we connect it?
- How many values should we allow it to have?
Introducing Hidden Variables

• In general, there are 2 type of organizations for hidden variables:
  – Hierarchy organization
  – Overlapping organization
The hidden variables form a treelike hierarchy

Intuition: the distance in the hierarchy should correspond to the degree of dependencies between the variables, so that strongly dependent variables would be closer in the hierarchy.
Overlapping Organization

- Intuition: each hidden variable captures aspects of the instance that several of the observed variables depend on
- Sometime results in a highly intractable structure
Introducing Hidden Variables – Cont.

• We want hidden variables to capture as much information as possible about the variables which they are connected to.

• Therefore, we should look for subset of variables that are highly dependent.

• There are 2 approaches to perform this task:
  1. Learn a structure, find “cliques” in the graph.
  2. Directly consider the empirical distribution – i.e., by finding mutual information between pair of variables.

• After introducing an hidden variable, we can use the structural EM algorithm to “fix mistakes” in the model.
Determining the Cardinality

• Model selection: try multiple cardinalities and choose the best one

• Clustering: solve a clustering problem for H’s Markov blanket, and choose the cardinality whose clustering has the highest explanatory powers
Questions?