Last Time

- Score-based structure learning
  - Candidate structures; Score function; Search for the high-scoring structure

- Scoring functions
  - Maximum likelihood score
    - \(\text{Score}_\ell(G; D) = \log P(D | G, 0'_G)\) where \(0'_G\) is MLE for \(G\)
    - Prone to overfitting
  - Bayesian score
Bayesian Score

- Main principle of the Bayesian approach
  - Whenever we have uncertainty over anything, place a distribution over it.
  - What uncertainty? \((G, \Theta)\)

\[
P(G \mid D) = \frac{P(D \mid G)P(G)}{P(D)}
\]

- Marginal likelihood
- Prior over structures
- Marginal probability of Data

\[
P(D) \text{ does not depend on the network}
\]

Bayesian Score: 
\[
Score_B(G : D) = \log P(D \mid G) + \log P(G)
\]

Marginal Likelihood of Data Given G

Bayesian Score: 
\[
Score_B(G : D) = \log P(D \mid G) + \log P(G)
\]

\[
P(D \mid G) = \int_{\theta_G} P(D \mid G, \theta_G)P(\theta_G \mid G)d\theta_G
\]

Marginal likelihood
- Likelihood
- Prior over parameters

Note similarity to maximum likelihood score, but with the key difference that ML finds maximum of likelihood and here we compute average of the terms over parameter space
Marginal Likelihood: Binomial Case

- Assume a sequence of $m$ coin tosses
- By the chain rule for probabilities

$$P(x[1],...,x[m]) = P(x[1]) \cdot \ldots \cdot P(x[m] \mid x[1],...,x[m-1])$$

$$P(D \mid G) = \int P(D \mid G, \theta_G) P(\theta_G \mid G) d\theta_G$$

- Recall that for Dirichlet priors

$$P(x[m+1] = H \mid x[1],...,x[m]) = \frac{M^m_H + \alpha_H}{m + \alpha_H + \alpha_f}$$

- Where $M^m_H$ is number of heads in first $m$ examples

$$P(x[1],...,x[m]) = \frac{[\alpha_H \cdot \ldots \cdot (\alpha_H + M_H - 1)] [\alpha_f \cdot \ldots \cdot (\alpha_f + M_f - 1)]}{\alpha \cdot \ldots \cdot (\alpha + M - 1)}$$
**Marginal Likelihood: Binomial Case**

\[
P(x[1],...,x[m]) = \frac{\prod_{i=1}^{m} (\alpha_{x_i} + M_{x_i} - 1) \prod_{i=1}^{r} (\alpha_{y_i} + M_{y_i} - 1)}{\prod_{i=1}^{K} (\alpha + M - 1)}
\]

Simplify using \( \Gamma(x+1) = x\Gamma(x) \)

\[
(\alpha)(\alpha+1) \cdots (\alpha+M-1) = \frac{\Gamma(\alpha+M)}{\Gamma(\alpha)}
\]

\[
P(x[1],...,x[m]) = \frac{\Gamma(\alpha)}{\Gamma(\alpha+M)} \cdot \frac{\Gamma(\alpha_{x_1} + M_{x_1})}{\Gamma(\alpha_{y_1})} \cdot \frac{\Gamma(\alpha_{y_1} + M_{y_1})}{\Gamma(\alpha_{x_1})} \cdots \frac{\Gamma(\alpha_{x_r} + M_{x_r})}{\Gamma(\alpha_{y_r})}
\]

For multinomials with Dirichlet prior

\[
P(x[1],...,x[m]) = \frac{\Gamma(\alpha)}{\Gamma(\alpha+M)} \cdot \prod_{i=1}^{K} \frac{\Gamma(\alpha_{x_i} + M[x_i'])}{\Gamma(\alpha_{y_i})}
\]

---

**Marginal Likelihood: BayesNets**

- **Network structure** determines form of marginal likelihood \( P(D|G) \)

<table>
<thead>
<tr>
<th>( D )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>( Y )</td>
<td>H</td>
<td>T</td>
<td>H</td>
<td>H</td>
<td>T</td>
<td>T</td>
<td>H</td>
</tr>
</tbody>
</table>

**Network 1: Two Dirichlet marginal likelihoods**

- \( P(X[1],...,X[7]) \)
- \( P(Y[1],...,Y[7]) \)

**Network G0**

- \( X \)
- \( Y \)
Network structure determines form of marginal likelihood $P(D|G)$

Network 2: Three Dirichlet marginal likelihoods

- $P(X[1], \ldots, X[7])$

As we get more data, the Bayesian score prefers $G_1$ where $X$ and $Y$ are dependent.
Marginal Likelihood: BayesNets

The marginal likelihood has the form:

\[
P(D \mid G) = \prod_{i} \prod_{pa_i} \frac{\Gamma(\alpha_{pa_i})}{\Gamma(\alpha_{pa_i} + M[pa_i])} \prod_{i} \frac{\Gamma(\alpha_{x_i,pa_i} + M[x_i, pa_i])}{\Gamma(\alpha_{x_i,pa_i})}
\]

where

- \(M(\ldots)\) are the counts from the data
- \(\alpha(\ldots)\) are hyperparameters for each family

Bayesian Score: Asymptotic Behavior

- For \(M \to \infty\), a network \(G\) with Dirichlet priors satisfies

\[
\log P(D \mid G) = l(\hat{\theta}_G : D) - \frac{\log M}{2} \text{Dim}(G) + O(1)
\]

\text{Dim}(G): \text{number of independent parameters in } G

- Approximation is called **BIC score**

\[
\text{Score}_{\text{BIC}}(G : D) = l(\hat{\theta}_G : D) - \frac{\log M}{2} \text{Dim}(G)
\]

- Score exhibits tradeoff between fit to data and complexity
- Mutual information grows linearly with \(M\) while complexity grows logarithmically with \(M\)
  - As \(M\) grows, more emphasis is given to the fit to the data
Bayesian Score: Asymptotic Behavior

For $M \to \infty$, a network $G$ with Dirichlet priors satisfies

$$
\log P(D \mid G) = I(\hat{\theta}_G : D) - \frac{\log M}{2} \text{Dim}(G) + O(1)
$$

$$
= M \sum_{i=1}^{n} I_{p}(X_i, Pa_{X_i}) - M \sum_{i=1}^{n} H_{p}(X_i) - \frac{\log M}{2} \text{Dim}(G) + O(1)
$$

Bayesian score is consistent

As $M \to \infty$, the true structure $G^*$ maximizes the score

- Spurious edges will not contribute to likelihood and will be penalized
- Required edges will be added due to linear growth of likelihood term relative to $M$ compared to logarithmic growth of model complexity

Priors

Bayesian Score: $Score_B(G : D) = \log P(D \mid G) + \log P(G)$

- Structure prior $P(G)$
  - Uniform prior: $P(G) \propto \text{constant}$
  - Prior penalizing number of edges: $P(G) \propto c^{|G|}$ ($0 < c < 1$)
  - Normalizing constant across networks is similar and can thus be ignored
**Priors**

*Bayesian Score:* \( \text{Score}_B(G : D) = \log P(D \mid G) + \log P(G) \)

- Parameter prior \( P(\theta \mid G) \)
  - BDe prior
    - \( M_0 \): equivalent sample size
    - \( B_0 \): prior network representing the prior probability of events
    - Set \( \alpha(x_i, pa_i^G) = M_0 P(x_i, pa_i^G \mid B_0) \)
      - Note: \( pa_i^G \) may not the same as parents of \( X_i \) in \( B_0 \)
      - Compute \( P(x_i, pa_i^G \mid B_0) \) using standard inference in \( B_0 \)
  - BDe requires assessing prior network \( B_0 \)
    - Can naturally incorporate prior knowledge
  - BDe is consistent and asymptotically equivalent (up to a constant) to BIC

**Summary: Network Scores**

- **Decomposability**
  - Likelihood, BIC, (log) BDe have the form
    \[
    \text{Score}(G : D) = \sum_i \text{Score}(X_i \mid Pa_i^G : D)
    \]
  - All are **score-equivalent**
    - \( G \) I-equivalent to \( G' \) \( \Rightarrow \) \( \text{Score}(G) = \text{Score}(G') \)
So far, we discussed scores for evaluating the quality of different candidate BN structures... Let’s now examine how to find a structure with a high score.

**STRUCTURE SEARCH**

---

**Optimization Problem**

**Input:**
- Training data $D = \{X[1],...,X[M]\}$
- Scoring function (including priors, if needed)
- Set of possible structures (search space)
  - Including prior knowledge about structure

**Output:**
- A network (or networks) that maximize the score

**Key Property:**
- Decomposability: the score of a network is a sum of terms.

$$Score(G : D) = \sum_{i} Score(X_i | Pa^G_i : D)$$
Learning Trees

- **Trees**
  - At most one parent per variable

- **Why trees?**
  - Elegant math
    ⇒ we can solve the optimization problem efficiently (with a greedy algorithm)
  - Sparse parameterization
    ⇒ avoid overfitting while adapting to the data

---

**Learning Trees**

- Let $p(i)$ denote parent of $X_i$, or 0 if $X_i$ has no parent
- We can write the score as

$$\text{Score}(G : D) = \sum_i \text{Score}(X_i : Pa_i)$$

$$= \sum_{i : p(i) \neq 0} \text{Score}(X_i : X_{p(i)}) + \sum_{i : p(i) = 0} \text{Score}(X_i)$$

$$= \sum_{i : p(i) \neq 0} (\text{Score}(X_i : X_{p(i)}) - \text{Score}(X_i)) + \sum_i \text{Score}(X_i)$$

- Score = sum of edge scores + constant
Learning Trees

- **Algorithm**
  - Construct graph with vertices: 1,...,n
  - For all \((i,j)\), set edge score \(w(i \rightarrow j) = \text{Score}(X_j | X_i) - \text{Score}(X_j)\)
  - If the score satisfies score equivalence, \(w(i \rightarrow j) = w(j \rightarrow i)\)
  - Structure learning problem: Find the tree structure with maximum sum of weights.
    - Solve an undirected spanning tree (forest) problem and determine directions of edges afterwards.
    - This can be done using standard algorithms in low-order polynomial time by building a tree in a greedy fashion (e.g. Kruskal’s maximum spanning tree algorithm)

- **Theorem:** Procedure finds the tree with maximal score (sum of \(w(i \rightarrow j)\) for all edges \(i \rightarrow j\))

- When score is likelihood, then \(w(i \rightarrow j)\) is proportional to \(I(X_i; X_j)\). This is known as the Chow & Liu method.

Learning Trees: Example

Tree learned from data
D sampled from the ICU-Alarm network
\(D = \{X[1],...,X[M]\}\)

Not every edge in tree is in the original network
Tree direction is arbitrary --- we can’t learn about arc direction
Beyond Trees

- Problem is not easy for more complex networks
  - Example: Allowing two parents, greedy algorithm is no longer guaranteed to find the optimal network

- Theorem:
  - Finding maximal scoring network structure with at most \( k \) parents for each variable is NP-hard for \( k>1 \)

- In fact, no efficient algorithm exists

Fixed Ordering

- For any decomposable scoring function \( \text{Score}(G:D) \)

\[
\text{Score}(G:D) = \sum \text{Score}(X_j \mid Pa_i^G : D)
\]

and ordering \( \alpha \) the maximal scoring network has:

\[
Pa_i^G = \arg \max_{U \subseteq \{X_j: X_j < X_i\}} \text{Score}(X_i \mid U_i : D)
\]

(since choice at \( X_i \) does not constrain other choices)

→ For fixed ordering, the structure learning problem becomes a set of independent problems of finding parents of \( X_i \).

- If we bound the in-degree per variable by \( d \), then complexity is exponential in \( d \)
Heuristic Search

We address the problem by using heuristic search

- Define a search space:
  - nodes are possible structures
  - edges denote adjacency of structures

- Traverse this space looking for high-scoring structures

- Search techniques:
  - Greedy hill-climbing
  - Best first search
  - Simulated Annealing
  - ...

Typical operations:

- Add B → D
- Delete B → C
- Reverse C → B

Search space
Exploiting Decomposability

- **Decomposability:** \( \text{Score}(G : D) = \sum_i \text{Score}(X_i | Pa^G_i : D) \)

- **Caching:** To update the score after a local change, we only need to re-score the families that were changed.

Greedy Hill Climbing

- **Simplest heuristic local search**
  - Start with a given network
    - empty network
    - best tree (tree learning)
    - a random network
  - At each iteration
    - Evaluate all possible changes
    - Apply change that leads to best improvement in score
    - Reiterate
  - Stop when no modification improves score
- Each step requires evaluating \( O(n^2) \) new changes
Greedy Hill Climbing Pitfalls

- Greedy Hill-Climbing can get stuck in:
  - **Local Maxima**
    - All one-edge changes reduce the score
  - **Plateaus**
    - Some one-edge changes leave the score unchanged
    - Happens because I-equivalent networks received the same score and are neighbors in the search space
  - Both occur during structure search
  - Standard heuristics can escape from both
    - Randomization and restart
    - TABU search: Keep a list of recent operators we applied, and in each step, we do not consider operators that reverse the effect of recently applied operators.

Model Selection

- So far, we focused on single model
  - Given \(D = \{X[1], \ldots, X[M]\}\), find best scoring model
    \[
    \hat{G} = \arg \max_G P(G \mid D)
    \]
  - Use it to predict next example
    \[
    P(X[M + 1] \mid D) = P(X[M + 1] \mid D, \hat{G})
    \]
- Implicit assumption
  - Making predictions based on the Bayesian estimation rule:
    \[
    P(X[M + 1] \mid D) = \sum_G P(X[M + 1] \mid D, G) P(G \mid D)
    \]
  - Best scoring model dominates the weighted sum
    - Valid with many data instances (very large \(M\))
- Pros:
  - We get a single structure
  - Allows for efficient use in our tasks
- Cons:
  - We are committing to the independencies of a particular structure
  - Other structures might be as probable given the data
Announcements

- Solution for PS #1 uploaded.

- Typo in Q5 of PS #2
  - Let $C_i$ be some clique such that $\text{Scope}[\phi']$...
  - 1 free late day for PS #2 (due 5/3 at noon; CSE536)

- PS #3 is ready (please pick it up).

Acknowledgement

- These lecture notes were generated based on the slides from Prof Eran Segal.