Announcements

- Recitations: Every Tuesday 4-5:30 in 243 Annenberg
- Homework 1 due in class Wed Oct 21
- Project proposals due tonight (Monday Oct 19)
Structure learning

Two main classes of approaches:

- Constraint based
  - Search for P-map (if one exists):
    - Identify PDAG
  - Turn PDAG into BN (using algorithm in reading)
  - **Key problem**: Perform independence tests

- Optimization based
  - Define scoring function (e.g., likelihood of data)
  - Think about structure as parameters
  - More common; can solve simple cases exactly
Finding the optimal MLE structure

- Optimal solution for MLE is always the fully connected graph!!! 😞
  - Non-compact representation; Overfitting!!

Solutions:
- Priors over parameters / structures (later)
- Constraint optimization (e.g., bound #parents)
Bayesian learning

- Make prior assumptions about parameters $P(\theta)$
- Compute posterior

\[
P(\theta | D) = \frac{P(\theta)P(D|\theta)}{P(D)} \propto P(\theta)P(D|\theta)
\]

Given data $D$ want to predict

\[
P(x | D) = \int P(\theta | D)P(x | \theta) \, d\theta
\]

In MLE

\[
P(x | D) \approx P(x | \hat{\theta}) \quad \hat{\theta} = \operatorname{argmax} \quad P(D | \theta)
\]
Conjugate priors

Consider parametric families of prior distributions:
- \( P(\theta) = f(\theta; \alpha) \)
  - \( \alpha \) is called “hyperparameters” of prior

A prior \( P(\theta) = f(\theta; \alpha) \) is called \textit{conjugate} for a likelihood function \( P(D | \theta) \) if \( P(\theta | D) = f(\theta; \alpha') \)
  - Posterior has same parametric form
  - Hyperparameters are updated based on data \( D \)

Obvious questions (answered later):
- How to choose hyperparameters??
- Why limit ourselves to conjugate priors??
Posterior for Beta prior

- **Beta distribution**

  \[ P(\theta) = \text{Beta}(\theta; \alpha_H, \alpha_T) = \frac{\theta^{\alpha_H-1}(1 - \theta)^{\alpha_T-1}}{B(\alpha_H, \alpha_T)} \]

- **Likelihood:**

  \[ P(D \mid \theta) = \theta^{m_H} (1 - \theta)^{m_T} \]

- **Posterior:**

  \[
P(\theta \mid D) \propto P(\theta) P(D \mid \theta) \propto \theta^{\alpha_H+m_H-1} (1 - \theta)^{\alpha_T+m_T-1}
  \]

  \[ P(\theta \mid D) = \text{Beta}(\theta \mid \alpha_H+m_H, \alpha_T+m_T) \]
Why do priors help avoid overfitting?

\[ P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \theta_\mathcal{G}) dP(\theta_\mathcal{G} \mid \mathcal{G}) \]

- This Bayesian Score is tricky to analyze. Instead use:

\[ \log P(\mathcal{D} \mid \mathcal{G}) \approx \log P(\mathcal{D} \mid \mathcal{G}, \hat{\theta}_\mathcal{G}) - \frac{\log m}{2} \text{Dim}(\mathcal{G}) \]

- Why??

**Theorem**: For Dirichlet priors, and for \( m \to \infty \):

\[ \log P(\mathcal{D} \mid \mathcal{G}) \to \log P(\mathcal{D} \mid \mathcal{G}, \hat{\theta}_\mathcal{G}) - \frac{\log m}{2} \text{Dim}(\mathcal{G}) + \mathcal{O}(1) \]
This approximation is known as **Bayesian Information Criterion** (related to Minimum Description Length).

- Trades goodness-of-fit and structure complexity!
- Decomposes along families (computational efficiency!)
- Independent of hyperparameters! (Why??)
Consistency of BIC

- Suppose true distribution has P-map G*
- A scoring function Score(G ; D) is called **consistent**, if, as $m \rightarrow \infty$ and probability $\rightarrow 1$ over D:
  - G* maximizes the score
  - All non-I-equivalent structures have strictly lower score

**Theorem**: BIC Score is consistent!

- Consistency requires $m \rightarrow \infty$. For finite samples, priors matter!
Parameter priors

- How should we choose priors for discrete CPDs?
- Dirichlet (computational reasons). But how do we specify hyperparameters??

- K2 prior:
  - Fix $\alpha$
  - $P(\theta_X | \text{pa}_X) = \text{Dir}(\alpha, \ldots, \alpha)$

- Is this a good choice?

\[
P(\theta_Y) = \text{Dir}(\alpha, \alpha) \\
\implies \text{Equivalent sample size} \quad 2\alpha
\]

\[
P(\theta_Y | X = h) = \text{Dir}(\alpha, \alpha) \\
P(\theta_Y | X = t) = \text{Dir}(\alpha, \alpha) \\
\implies \text{Equivalent sample size} \quad 4\alpha
\]
BDe prior

Want to ensure “equivalent sample size” $m'$ is constant

Idea:

- Define $P'(X_1,...,X_n)$
  
  For example: $P'(X_1,...,X_n) = \prod_i \text{Uniform}(\text{Val}(X_i))$

- Choose equivalent sample size $m'$

- Set $\alpha_{x_i | pai} = m' P'(x_i, pai)$

$$
\alpha_y = m' P'(y) = m' \sum_x P(x,y) = \sum_x \alpha_{y|x}
$$
Score consistency

- A scoring function is called score-consistent, if all I-equivalent structures have the same score.

\[ G \quad \text{Score consistency:} \quad l(G) = l(G') \Rightarrow \text{Score}(G : D) = \text{Score}(G' : D) \]

- K2 prior is inconsistent!
- BDe prior is consistent
- In fact, Bayesian score is consistent $\Leftrightarrow$ BDe prior on CPTs!!
Score decomposability

- Proposition: Suppose we have
  - Parameter independence
  - Parameter modularity: if X has same parents in G, G’, then same prior.
  - Structure modularity: \( P(G) \) is product over factors defined over families (e.g.: \( P(G) = \exp(-c|G|) \))

- Then Score(D : G) decomposes over the graph:
  \[
  \text{Score}(G ; D) = \sum_i \text{FamScore}(X_i \mid \text{Pa}_i ; D)
  \]

- If G’ results from G by modifying a single edge, only need to recompute the score of the affected families!!
Bayesian structure search

- Given consistent scoring function \( \text{Score}(G : D) \), want to find graph \( G^* \) that maximizes the score.

- Finding the optimal structure is **NP-hard** in most interesting cases (details in reading).

- Can find optimal tree/forest efficiently (Chow-Liu).

- Want practical algorithm for learning structure of more general graphs.
Local search algorithms

- Start with empty graph (better: Chow-Liu tree)
- Iteratively modify graph by
  - Edge addition
  - Edge removal
  - Edge reversal
- Need to guarantee acyclicity (can be checked efficiently)
- Be careful with I-equivalence (can search over equivalence classes directly!)
- May want to use simulated annealing to avoid local maxima
Efficient local search

If Score is decomposable, only need to recompute affected families!
Alternative: Fixed order search

- Suppose we fix order $X_1, ..., X_n$ of variables.
- Want to find optimal structure s.t. for all $X_i$:
  \[ \text{Pa}_i \subseteq \{X_1, ..., X_{i-1}\} \]

\begin{align*}
\text{For } i &= 1 \text{ to } n \\
\quad \text{For each } A \subseteq \{X_i, ..., X_{i-1}\} \\
\quad \text{Compute } \text{Fan Score}(X_i | A) \\
\quad A^* &= \arg\max_A \text{Fan Score}(X_i | A) \\
\quad \text{Pa}_i &= A^* \\
\text{Score}(G; D) &= \sum_{i=1}^{n} \text{Fan Score}(X_i | \text{Pa}_i) \\
\end{align*}

\[
\Rightarrow \text{Find optimal structure!}
\]
Fixed order for d parents

- Fix ordering

- For each variable $X_i$
  - For each subset $A \subseteq \{X_1, \ldots, X_{i-1}\}$, $|A| \leq d$
    - compute $\text{FamScore}(X_i \mid A)$
  - Set $\text{Pa}_i = \arg\max_A \text{FamScore}(X_i \mid A)$

- If score is decomposiable $\Rightarrow$ optimal solution!!

- Can find best structure by searching over all orderings!
Searching structures vs orderings?

- **Ordering search**
  - Find optimal BN for fixed order
  - Space of orderings “much smaller” than space of graphs..
    - \( n! \) orderings vs \( 2^{n^2} \) directed graphs (counting DAGs more complicated)

- **Structure search**
  - Can have arbitrary number of parents
  - Cheaper per iteration
  - More control over possible graph modifications
What you need to know

- Conjugate priors
  - Beta / Dirichlet
  - Predictions, updating of hyperparameters
- Meta-BN encoding parameters as variables
- Choice of hyperparameters
  - BDe prior \( \Rightarrow \) score consistency
- Decomposability of scores and implications
- Local search
  - On graphs
  - On orderings (optimal for fixed order)
Key questions

- How do we specify distributions that satisfy particular independence properties?
  - Representation

- How can we identify independence properties present in data?
  - Learning

- How can we exploit independence properties for efficient computation?
  - Inference
Bayesian network inference

- Compact representation of distributions over large number of variables
- (Often) allows efficient **exact inference** (computing marginals, etc.)

**HailFinder**
- 56 vars
- ~ 3 states each
- \( \Rightarrow \sim 10^{26} \) terms
- > 10,000 years on Top supercomputers
Typical queries: Conditional distribution

- Compute distribution of some variables given values for others

\[
P(E=T | M=T) = \frac{P(E=T, M=T)}{P(M=T)}
\]

\[
P(E=T, M=T) = \sum_{b} \sum_{a} \sum_{j} \frac{P(E=T, M=T, B=b, A=a, J=j)}{P(E)P(B)P(A|B)P(J|A)P(M)}
\]

Naive approach exponential in \# vars...
Typical queries: Maximization

- **MPE (Most probable explanation):** Given values for some vars, compute most likely assignment to all remaining vars

  \[
  \text{Given } J = \bar{F}, M = T, \text{ find } \\
  (e^*, b^*, a^*) = \arg\max_{e, b, a} P(J = \bar{F}, M = T, e, b, a) \\
  \]

- **MAP (Maximum a posteriori):** Compute most likely assignment to some variables

  \[
  e^* = \arg\max_{e} P(J = \bar{F}, M = T, E = e) = \\
  \text{arg} \max_{e} \sum_{a, b} P(J = \bar{F}, M = T, e, a, b) \\
  \]

\[\text{MPE and MAP don't necessarily give same answers} \ldots\]
Hardness of computing conditional prob.

- Computing $P(X=x \mid E=e)$ is NP-hard

**Proof:** by reduction from 3SAT

Given boolean formula $\Phi = (x_1 \lor x_2 \lor x_3) \land (\neg x_3 \lor x_4 \lor x_5) \lor \ldots$

does there exist a sat. assignment to $x_1, \ldots, x_n$

![Diagram](image)

$P(X_i) = \text{Bern}(0.5)$

$C_i = C_{i-1} \land \text{Truth val for Clause}_i$

If sat. $\Rightarrow P(C_m = \top) > 0$
Hardness of computing cond. prob.

- In fact, it’s even worse: $P(X=x \mid E=e)$ is $\#P$ complete

$$P(C_m = T) = \sum_{X_1 \ldots X_m} P(X_1 \ldots X_m) - \frac{P(C_m = T \land X_1 \ldots X_m)}{2^m}$$

$$P(C_m = \overline{T}) = \frac{\# \text{ sat assignments}}{2^m}$$

$\Rightarrow \#P$ hardness
Hardness of inference for general BNs

- Computing conditional distributions:
  - Exact solution: #P-complete
  - Approximate solution:
    - Absolute approx: Finding \( |P(x) - P'(x)| < \epsilon \) \( \text{NP-hard} \) for \( \epsilon = \frac{1}{2} \)
    - Relative approx: \( 1 - \epsilon < \frac{P(x)}{P'(x)} < 1 + \epsilon \) \( \text{NP-hard for } \epsilon > 0 \)

- Maximization:
  - MPE: NP-complete
  - MAP: \( \text{NP}^{\text{PP}} \)-complete

- Inference in general BNs is really hard 😞

- Is all hope lost?
Inference

- Can exploit structure (conditional independence) to efficiently perform **exact inference** in many practical situations.

- For BNs where exact inference is not possible, can use algorithms for **approximate inference** (later this term).
Computing conditional distributions

Query: \( P(X | E=e) \)

\[
P(X | e) = \frac{P(X,e)}{P(e)} \propto P(X,e)
\]

\( \equiv \) Renormalize (over \( X \)) to get
\[
P(X | e)
\]
Inference example

\[ P(E \mid m) \propto P(E, m) \]

\[ = \sum_{a, i, j, b} P(E, m, a, i, j, b) \]

\[ = \sum_{a, i, j, b} P(E) P(b) P(a \mid E, b) P(G_i \mid k) P(m \mid a) \]

*in general, exponentially many terms*
Potential for savings: Variable elimination!

\[ P(X_5, x_i) = \sum_{x_2} \sum_{x_3} \sum_{x_4} P(x_i) P(x_2 | x_i) P(x_3 | x_2) P(x_4 | x_3) P(x_5 | x_4) \]

distributively:
\[ P(x_i) \sum_{x_2} P(x_2 | x_i) \sum_{x_3} P(x_3 | x_2) \sum_{x_4} P(x_4 | x_3) P(x_5 | x_4) \]

How many additions: 3

\[ X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_5 \]

\[ \mathcal{G}_2(x_i, x_5) \]

Intermediate solutions are distributions on fewer variables!
Variable elimination in general graphs

- Push sums through product as far as possible
- Create new factor by summing out variables

\[ P(E, m) = \sum_{b, a, i, j} P(E) P(b) P(a | E, b) P(j | a) P(m | i, a) \]

\[ = P(E) \sum_b P(b) \sum_a P(a | E, b) P(m | i, a) \sum_{j, a} P(j | a) \]

\[ = f_A(E, i, s, m) \]

\[ f_B(E, m) \]
Removing irrelevant variables

Delete nodes not on active trail between query vars.
Variable elimination algorithm

- Given BN and Query $P(X \mid E=e)$
- Remove irrelevant variables for $\{X,e\}$
- Choose an ordering of $X_1,\ldots,X_n$
- Set up initial factors: $f_i = P(X_i \mid Pa_i)$
- For $i = 1:n$, $X_i \notin \{X,E\}$
  - Collect all factors $f$ that include $X_i$
  - Generate new factor by marginalizing out $X_i$
    
    $g = \sum_{x_i} \prod_j f_{ij}$
  - Add $g$ to set of factors
- Renormalize $P(x,e)$ to get $P(x \mid e)$
Multiplying factors

\[ g = \sum_{x_i} \Pi_{j} f_j \]

\[ f_1(A, B), f_2(B, C) \]

\[ f' = f_1 \cdot f_2 \]

\[ f'(A, B, C) \]

\[ f'(A=T, B=F, C=F) = f_1(A=T, B=F) \cdot f_2(B=F, C=F) \]
Marginalizing factors

\[ g = \sum_{x_i} \prod_{j} f_j \]

\[ \Rightarrow g = \sum_{A} \ell'(A, B) \]

\[ \begin{array}{c|c|c|c|c|c}
A & B & g'(A, B) & \ell(A, B) \\
\hline
T & T & & & & \\
F & T & & & & \\
\hline
T & F & & & & \\
F & F & & & & \\
\end{array} \]

\[ g(B) = \ell(A=T, B) + \ell'(A=F, B) \]
Tasks

- Read Koller & Friedman Chapter 17.4, 18.3-5, 19.1-3

- Homework 1 due in class Wednesday Oct 21