Probabilistic Foundations of Artificial Intelligence

Exact Inference in Bayesian Networks

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)
Bayesian Networks

- A **Bayesian network structure** is a directed, acyclic graph \( \mathcal{G} \), where each vertex \( s \) of \( \mathcal{G} \) is interpreted as a random variable \( X_s \) (with unspecified distribution)

- A **Bayesian network** \((\mathcal{G}, P)\) consists of
  - A BN structure \( \mathcal{G} \) and ..
  - ..a set of conditional probability distributions (CPTs) \( P(X_s \mid \text{Pa}_{X_s}) \), where \( \text{Pa}_{X_s} \) are the parents of node \( X_s \) such that
  - \((\mathcal{G}, P)\) defines the joint distribution

\[
P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \text{Pa}_{X_i})
\]
Active Trails

- An undirected path in a BN structure $G$ is called active trail for observed variables $O \subseteq \{X_1, \ldots, X_n\}$ if for every consecutive triple of vars $X, Y, Z$ on the path:
  - $X \rightarrow Y \rightarrow Z$ and $Y$ is unobserved ($Y \notin O$)
  - $X \leftarrow Y \leftarrow Z$ and $Y$ is unobserved ($Y \notin O$)
  - $X \leftarrow Y \rightarrow Z$ and $Y$ is unobserved ($Y \notin O$)
  - $X \rightarrow Y \leftarrow Z$ and $Y$ or any of $Y$’s descendants is observed

- Any variables $X_i$ and $X_j$ for which there is no active trail for observations $O$ are called d-separated by $O$. We write $d$-sep($X_i; X_j \mid O$)

- Sets $A$ and $B$ are d-separated given $O$ if $d$-sep($X, Y \mid O$) for all $X$ in $A$, $Y$ in $B$. Write $d$-sep($A; B \mid O$)
Theorem:
\[ \text{d-sep}(X; Y \mid Z) \Rightarrow X \perp Y \mid Z \]

i.e., \( X \) cond. indep. \( Y \) given \( Z \)
if there does not exist any active trail between \( X \) and \( Y \)
for observations \( Z \)

- Converse does not hold in general!
- But for “almost” all distributions
  (except set of measure 0)
Examples

Can we conclude from d-sep. that:

\[ A \perp I \]
\[ A \perp I | F \times \]
\[ A \perp I | F, C \checkmark \]
\[ A \perp I | F, C, E \times \]
\[ A \perp I | F, C, E, G \checkmark \]
Algorithm for d-Separation

- How can we check if $d$-sep$(X; Y \mid Z)$?
  - Idea: Check every possible path connecting $X$ and $Y$ and verify conditions
  - Exponentially many paths!

- Linear time algorithm:
  Find all nodes reachable from $X$
  - 1. Mark $Z$ and its ancestors
  - 2. Do breadth-first search starting from $X$; stop if path is blocked
  - Have to be careful with implementation details (see reading)
Typical Queries: Conditional Distribution

- Compute distribution of some variables given values for others

\[
P(E \mid M = T) = \frac{1}{2} \ P(E, M = T) \\
= \frac{1}{2} \ \sum_{b, a, i} P(E, B = b, A = a, J = j, M = T)
\]
Typical Queries: Maximization

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

  \[
  \arg \max_{e, b, a} \ P(E=e, B=b, A=a \mid J=T, M=f) \]

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

  \[
  \arg \max_{e, b} \ P(E=e, B=b \mid J=T, M=f) = \arg \max \sum_a P(e, b, a \mid J=T, M=f) \]
Hardness of Inference for General BNs

- Computing conditional distributions:
  - Exact solution: \#P-complete
  - NP-hard to obtain any nontrivial approximation

- 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)
Potential for Savings: Variable Elimination!

\[ P(x_1, x_5) \text{ for fixed vals of } x_1, x_5 \]
\[ P(x_1, x_5) = \sum_{x_2 \ldots x_4} P(x_1, \ldots, x_5) = \sum_{x_2 \ldots x_4} P(x_1) P(x_2 | x_1) \ldots P(x_5 | x_4) \]

5 \rightarrow 4
Naive:
\[ 2^{(n-2)} - 1 \]
Improved:
\[ 2^{(n-2)} - 1 \]

Total work: 5 "+"; Naive alg. 8 "+" 5 \ll 8 !!
Variable Elimination in General Graphs

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

\[
P(E|M) = \sum_{b,a,j} P(EM|b,a,j) = \sum_{b,a} P(E)P(b)P(a|E,b)P(d|a)P(M|a)
\]

Intermediate solutions are distributions on fewer variables!
Variable Elimination Algorithm

- Given BN and Query $P(X \mid E=e)$
- Choose an ordering of $X_1, \ldots, X_n$
- Set up initial factors: $f_i = P(X_i \mid \text{Pa}_i)$
- For $i = 1:n$, $X_i \not\in \{X, E\}$
  - Collect and multiply all factors $f$ that include $X_i$
  - Generate new factor by marginalizing out $X_i$
    \[ g = \sum_{x_i} \prod_j f_j \]
  - Add $g$ to set of factors
- Renormalize $P(x, e)$ to get $P(x \mid e)$
# Multiplying Factors

$$g = \sum_{x_i} \prod_{j} f_j$$

Want $f_1 \cdot f_2 = f$!

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$f_1(A,B)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>.3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>.01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$f_2(B,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>.4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>.2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Marginalizing Factors

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

\[ f'' = \exists_{x_a} f' \]

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$f''(A,B)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>.3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>.01</td>
</tr>
</tbody>
</table>
The Order Matters!

- $P(A, B, E, J, M) = P(E)P(B)P(A|E, B)P(J|A)P(M|A)$
- What if we eliminate $A$ first?

\[ f_1, f_2, f_3 \]

`Calculated all factors that depend on A, P(A|E, B), P(J|A), P(M|A)`

\[ f_1 + f_2 + f_3 \]

\[ \sum_{x_A} \text{ depends on A B E J M} \]

\[ g_x(B E J M) \]

Lost all structure
Variable Elimination for Polytrees

A DAG is a polytree iff dropping edge directions results in a tree

1) Pick a root (arbitrary)
2) Orient all edges towards the root
3) Eliminate nodes in topological order

16. Permutation $\pi$ on nodes is topological order if

$$x_i \in \text{Desc}(x_j) \Rightarrow \pi(i) > \pi(j) \forall i, j$$
What About Loops?

- Can do efficient inference on trees.
- What if the graph has loops?
Acknowledgments