Modeling with Bayesian networks

Using Bayesian networks for real-world problems requires two steps:
- constructing an appropriate Bayesian network
- solve the problems by applying one of the possible queries

How to construct a Bayesian network?
1. define network variables and their values
   - distinguish between query, evidence, and intermediary variables
2. define network structure
   - for each var X, answer the question: what set of var’s are direct causes of X?
3. define network parameters (CPTs)
   - difficulty and objectivity depend on problem

Reasoning with Bayesian networks

Four general types of queries one can pose:
- probability of evidence: how likely is a variable instantiation \( e \rightarrow Pr(e)=? \)
- prior and posterior marginals: how probable is an instantiation of a limited set of variables \( Pr(x_1, ..., x_m|=?) \) or \( Pr(x_1, ..., x_m|e)=? \)
- most probable explanation (MPE): what is the most probable instantiation of all network var’s given some evidence \( e \rightarrow x \) with \( Pr(x_1, ..., x_n|e)=max \)?
- maximum a posteriori hypothesis (MAP): what is the most probable instantiation of a subset of var’s given some evidence \( e \rightarrow x \) with \( Pr(x_1, ..., x_m|e)=max \)?

All these queries can be computed from a Bayesian network.
But, how?

Inference algorithms

- variable elimination (see last lecture)
- jointree algorithm
- recursive conditioning
- belief propagation
- Monte Carlo Markov Chain
Variable elimination

Variable elimination:
- given a distribution \( Pr(A,B,C,D,E) \), variable \( A \) with values \( a_i \) can be summed out by
\[
Pr(B, C, D, E) = \sum_{a_i} Pr(a_i, B, C, D, E)
\]

Definition: factor \( f \) over var's \( X \) is a function that maps each instantiation \( x \) of \( X \) to a number \( f(x) \geq 0 \)
- can represent any marginal or conditional distribution
- Summing out a variable from a factor:
  - marginalizing \( X \), projecting on \( Y \)
- Multiplying factors:
  \[
  (f_1 f_2)(z) := f_1(x) f_2(y) \quad \text{with} \quad x \sim z, y \sim z
  \]

Example: compute prior marginal \( Pr(C) \) by eliminating first \( A \), then \( B \)
\[
Pr(C) = \sum_B \Theta_{C,B} \sum_A \Theta_{A,B} \Theta_{B,A}
\]

Order of elimination irrelevant for result, but not for computational costs!!

Other possibility: first \( B \) then \( A \)
\[
Pr(C) = \sum_A \Theta_{A} \sum_B \Theta_{B,A} \Theta_{C,B}
\]

largest factor with 2 variables

Best order: smallest width = number of var's in the largest factor constructed
- can be computed offline, but NP-hard
- heuristics

Example: compute prior marginal \( Pr(C) \) by eliminating first \( A \), then \( B \)

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( \Theta_{A,B} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>.54</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>.5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>.44</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( \Theta_{B,A} \Theta_{C,B} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>.34</td>
<td>( \Theta_{B,A} )</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>.190</td>
<td>( \Theta_{C,B} )</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>.190</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>.634</td>
<td></td>
</tr>
</tbody>
</table>

\[
\sum \Theta_{B,A} \Theta_{C,B} = .34 + .190 = .534
\]

\[
Pr(C) = .534 \times .54 = .288
\]

Use for computing prior marginals:
- express joint distribution as factor multiplication, viewing CPTs as factors, e.g. \( Pr(a,b,c,d,e) \):
\[
Pr(a, b, c, d, e) = \Theta_{E|C} \Theta_{D|B,C} \Theta_{C|A} \Theta_{B|A} \Theta_{A}
\]
- compute marginal distribution by summing out variables from these factors, e.g. \( Pr(D,E) \):
\[
Pr(D, E) = \sum_{A,B,C} \Theta_{E|C} \Theta_{D|B,C} \Theta_{C|A} \Theta_{B|A} \Theta_{A}
\]

Can (and should) be simplified according to:
\[
\sum f_1 f_2 = f_1 \sum f_2 \quad \text{if} \quad X \text{appears only in} \quad f_1
\]

Use for computing posterior marginals:
- need to compute the factor \( Pr(Q|e) \)
Better to compute joint marginal \( Pr(Q,e) \) and normalize to get \( Pr(Q|e) \)
- also gives \( Pr(e) \) for free as \( Pr(e) = \sum_q Pr(Q,e) \)

Example:

<table>
<thead>
<tr>
<th>( D )</th>
<th>( E )</th>
<th>( Pr(Q,e) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>.448</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>.192</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>.112</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>.248</td>
</tr>
</tbody>
</table>

\[
\sum \frac{Pr(Q,e)}{Pr(e)} = .11904 / .48 = .48
\]
Variable elimination

Use elimination for computing joint marginals:
- zero out all rows that are not compatible with evidence \( e \)

**Definition:** reduction of factor \( f(X) \) given evidence \( e \) is another factor over \( X \) denoted by \( f^*(X) \) defined by
\[
f^*(x) := \begin{cases} f(x) & \text{if } x \sim e \\ 0 & \text{otherwise} \end{cases}
\]
- distributivity with factor multiplication: \( (f_1 f_2)^* = f_1^* f_2^* \)

Joint marginal \( \Pr(Q,e) \) can hence be computed as follows:
- **Example:** \( Q = \{D,E\} \)
\[
\Pr(Q,e) = \sum_{A,B,C} \Theta^e_{E|C} \Theta^e_{D|BC} \Theta^e_{C|A} \Theta^e_{B|A} \Theta^e_A
\]

**Example:** compute posterior marginal \( \Pr(Q=\{C\},e,A=true) \) by eliminating first \( A \), then \( B \)
\[
\begin{align*}
\Pr(C=true,A=true) &= 0.192 \\
\Pr(C=false,A=true) &= 0.408 \\
\Pr(A=true) &= 0.6 \\
\Pr(C=true|A=true) &= 0.192/0.6 = 0.32
\end{align*}
\]

**Factor elimination**

Generalization of variable elimination to factor elimination, i.e. elimination of sets of variables (Lauritzen & Spiegelhalter 1988)
- elimination order \( \rightarrow \) elimination trees

**Definition:** An elimination tree \((T, \Theta)\) for a set of factors \( S \) is a tree \( T \), in which each node \( \Theta_i \) is assigned exactly one factor in \( S \)
- factors are the CPTs in the Bayesian network
- different tree structures are possible:
Factor elimination

**Factor elimination in elimination trees:**
- eliminate a node (factor) if all its neighbors, except the one closer to the root, have been eliminated
- when a node \( i \) is about to be eliminated, it will have a single neighbor \( j \) and \( i \)'s factor is projected and multiplied into factor of \( j \)
  - viewed as passing a „message“ from \( i \) to \( j \)

**Using factor elimination for computing marginal over \( Q \):**
- pick one node \( r \) with \( Q \subseteq \text{vars}(r) \) as root node
- push messages toward the root
- when all messages are available in root, multiply with factor \( r \) and project to \( Q \)

Jointtree algorithm

**Definition:** A jointtree \((T, C)\) for a DAG \( G \) is a tree \( T \) in which each node has a label \( C_i \) (called cluster), satisfying the properties:
- each cluster is a set of nodes from \( G \)
- each family* in \( G \) appears in some cluster
- if a node appears in two clusters \( C_i, C_j \), it must appear in every cluster on the path connecting nodes \( i \) and \( j \) in the jointtree

the separator of edge \( ij \) is defined as \( S_{ij} := C_i \cap C_j \)

*family = a node along with its parents
Also known as junction trees, clique trees, Markov trees, hypertrees

An evidence indicator is a factor over variable \( X \) that captures the value of \( X \) in evidence \( e \): \( \lambda_X(x) = 1 \) if \( x \) consistent with \( e \), 0 otherwise

Recursive conditioning

**Idea:** simplify a problem by solving a number of cases and combining the results to a solution to the original problem (**case analysis**)

\[
\Pr(x) = \sum_e \Pr(x, e)
\]

**Approach:** reduce query on a network into a queries on simpler networks
- if var \( E \) given as evidence, the network can be pruned (see above)
- in general: any query \( \Pr(q, e) \) leads to decomposition into networks \( N_e^r \) and \( N_e^l \) such that

\[
\Pr(q) = \sum_e \Pr(q, e)
= \sum_e \Pr_e^l(q^l, e^l) \Pr_e^r(q^r, e^r)
\]

*cutset* \( e \) to \( \{B, C\} \)
Recursive conditioning

Recursive conditioning algorithm:

- Decompose network in a divide-and-conquer fashion, following an appropriate cutset
- When at leaf node, look up the conditioned CPT
- Propagate value back according to $\sum_e Pr_f(q', e')Pr_f(q', e')$

Question (again): what is an appropriate cutset (order)?

Answer: All are valid, some lead to less work

1. Need to minimize total number of considered cases
2. Use decomposition trees: full binary trees, leaves are CPTs in the network
3. Useful to employ caching techniques (Darwiche, chap. 8)

Belief propagation

Belief propagation algorithm for computing joint marginals $Pr(X, e)$:

- Identical to jointtree algorithm for jointtrees that coincide with the polytree network structure
- Example:

$$Pr(BCD, e) = \Theta_D|BC\pi_D(B)\pi_D(C)\lambda_E(D)\lambda_F(D)$$

Problem: leads to „deadlocks“ in some network, when each message is dependent on any other

Solution: Iterative belief propagation

- Assume initial values to each message in the network
- Propagate beliefs and re-iterate
- Converge to a „fixed point“ solution
- May generally have multiple fixed points on a given network
- May oscillate on some networks, loop forever

Belief propagation

Approximative inference algorithms

- Originally for exact inference in polytree networks, then generalized to approximative solution for arbitrary networks
- Spectrum of approximations: trade-off quality with computational costs

Belief propagation algorithm for computing joint marginals $Pr(X, e)$:

- Identical to jointtree algorithm for jointtrees that coincide with the polytree network structure
- Example:

*polytree = network with only one path between any two nodes*
**Stochastic sampling**

*Idea:* simulate an event according to some probability of occurrence, estimate the prob. of this event from its frequency in these simulations

**Simulating a Bayesian network:**
A Bayesian network induces a distribution \( Pr(X) \)
- visit each node in topological order
- generate value for each node according to \( Pr(x|u) \)
- end with a sample \( \{x^1, \ldots, x^n\} \) of \( n \) events
- estimate probability \( ^\wedge Pr(x) \) of value \( x \) from its frequency in this sample
- show that \( ^\wedge Pr(x) \) converges against \( Pr(x) \) with increasing \( n \)

**Example**

<table>
<thead>
<tr>
<th>C</th>
<th>P(SIC)</th>
<th>C</th>
<th>P(RIC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>.10</td>
<td>T</td>
<td>.80</td>
</tr>
<tr>
<td>F</td>
<td>.50</td>
<td>F</td>
<td>.20</td>
</tr>
</tbody>
</table>

| S   | R | P(W|S,R) |
|-----|---|---------|
| T   | T | .99     |
| T   | F | .90     |
| F   | T | .90     |
| F   | F | .01     |

<table>
<thead>
<tr>
<th>C</th>
<th>P(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>.50</td>
</tr>
<tr>
<td>F</td>
<td>.50</td>
</tr>
</tbody>
</table>

| C   | P(R|C) | C   | P(R|C) |
|-----|-------|-----|-------|
| T   | .80   | T   | .80   |
| F   | .20   | F   | .20   |

| C   | P(S|C) | C   | P(S|C) |
|-----|-------|-----|-------|
| T   | .10   | T   | .10   |
| F   | .50   | F   | .50   |

| S   | R | P(W|S,R) |
|-----|---|---------|
| T   | T | .99     |
| T   | F | .90     |
| F   | T | .90     |
| F   | F | .01     |

<table>
<thead>
<tr>
<th>C</th>
<th>P(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>.50</td>
</tr>
<tr>
<td>F</td>
<td>.50</td>
</tr>
</tbody>
</table>
Stochastic sampling

Sampling relies on taking probability as expectation about a function
- expectation value of a function \( f(\mathbf{X}) \):
  \[
  \mu = \mathbb{E}[f(\mathbf{X})] := \sum_x f(x) \cdot Pr(x)
  \]
- variance of a function \( f(\mathbf{X}) \):
  \[
  \sigma^2 = \text{Var}(f) := \sum_x (f(x) - \mu)^2 \cdot Pr(x)
  \]

Direct sampling function:
- let \( \alpha(x) := 1 \) if \( \alpha \) true at \( x \), 0 otherwise
- then:
  \[
  \mathbb{E}[\alpha] = Pr(\alpha)
  \]
  \[
  \text{Var}(\alpha) = Pr(\alpha)Pr(\neg\alpha) = Pr(\alpha) - Pr(\alpha)^2
  \]

That is, approximating \( Pr \) boils down to estimating the expectation

How?

Monte Carlo simulation

Principle:
- simulate random sample \( \mathbf{x}^1, \ldots, \mathbf{x}^n \) from sampling distribution \( Pr(\mathbf{X}) \)
- evaluate function at each instantiation \( f(\mathbf{x}^1), \ldots, f(\mathbf{x}^n) \)
- compute arithmetic average of attained values: sample mean
  \[
  \hat{\alpha}(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}^i)
  \]
- works because of law of large numbers: for function \( f \) with expectation \( \mu \)
  and every \( \epsilon > 0 \):
  \[
  \lim_{n \to \infty} Pr(|\hat{\alpha}(\mathbf{x}) - \mu| < \epsilon) = 1
  \]

Monte Carlo simulation using \( \hat{\alpha}(\mathbf{x}) \) gives direct sampling:
- simulate sample \( \mathbf{x}^1, \ldots, \mathbf{x}^2 \) from Bayesian network
- compute values \( \hat{\alpha}(\mathbf{x}^1), \ldots, \hat{\alpha}(\mathbf{x}^n) \)
- estimate \( Pr(\alpha) \) using sample mean \( \hat{\alpha}(\mathbf{x}) \)

Rejection sampling

Goal: Calculate conditional prob. \( Pr(\alpha|b) \) with \( Pr(\alpha) \) induced by network

Approach:
- calculate estimate for \( Pr(\alpha \land b) \) and \( Pr(b) \): \( \hat{\alpha}_n(\gamma), \hat{\beta}_n(\beta) \) with \( \gamma = \alpha \land \beta \)
- take ratio as estimate for \( Pr(\alpha|b) \):
  \[
  \hat{\alpha}_n(\gamma)/\hat{\beta}_n(\beta)
  \]
  - \( c_1 = \#\text{samples with } a \land b \text{ true}, c_2 = \#\text{samples with } b \text{ true} \rightarrow (c_1/n)/(c_2/n) = c_1/c_2 \)
- reject all samples in which \( b \) is false: rejection sampling

Example: estimate \( Pr(\text{Rain}|\text{Sprinkler}=\text{true}) \) from 100 samples; 27 have \( \text{Sprinkler}=\text{true} \), of these 8 have \( \text{Rain}=\text{true} \), 19 have \( \text{Rain}=\text{false} \)

\[
Pr(\text{Rain}|\text{Sprinkler} = \text{true}) = \text{Normalize}(8, 19) = 0.296, 0.704
\]

True answer: \( <0.3, 0.7> \)

Importance sampling

Idea: reduce variance due to rare events by sampling from an importance distribution \( Pr' \) emphasizing instantiations consistent with rare event

Monte Carlo simulation using the importance sampling function:
\[
\hat{\alpha}(\mathbf{x}) = Pr(x)/Pr'(x) \text{ if } a \text{ true at instantiation } x, 0 \text{ otherwise}
\]

Improves on direct sampling only when \( Pr' \) emphasizes important events no less than \( Pr \)

Finding ideal distribution generally not feasible, but some other weaker conditions can be ensured easier and still improve on variance
Importance sampling

Likelihood weighting: given evidence e, what is Pr(x|e)?
- generate only samples that are consistent with e
- fix evidence variables, sample non-evidence var’s and weight sample by likelihood it accords the evidence
- consistent estimate, but performance gets worse with growing evidence because few samples have ~all total weight

function Weighted-Sample(n, e) returns an event and a weight
  x ← an event with n elements; w ← 1
  for i = 1 to n do
    if X_i has a value x_i in e
      then w ← w × P(X_i = x_i | parents(X_i))
    else x_i ← a random sample from P(X_i | parents(X_i))
  return x, w

Example

Query: P(Rain | Sprinkler=true, WetGrass=true) = ??

w = 1.0
\[ w = 1.0 \times 0.1 \]

\[ w = 1.0 \times 0.1 \times 0.99 = 0.099 \text{ = weight for event } \langle t, t, t, t \rangle \]
Markov Chain Monte Carlo (MCMC)

Network is in a state = current assignment to variables
next state: sample non-evidence variable X given its Markov blanket
= variables that, when known, make other variables irrelevant to X
  - Markov blanket of Cloudy is Sprinkler and Rain
  - Markov blanket of Rain is Sprinkler, Cloudy, WetGrass

```
function MCMC-Ask(X, e, ln, N) returns an estimate of P(X|e)
local variables: N[X], a vector of counts over X, initially zero
  Z, the non-evidence variables in ln
  x, the current state of the network, initially copied from e
initialize x with random values for the variables in Y
for i = 1 to N do
  for each Z_i in Z do
    sample the value of Z_i in x from P(Z_i|mb(Z_i))
    given the values of MB(Z_i) in x
    N[x] = N[x] + 1 where x is the value of X in x
return NORMALIZE(N[X])
```

MCMC - Markov blanket sampling

Because of the transition probabilities, sampling runs into an “equilibrium” in which time spent in each state is proportional to its posterior probability

Transition probability (given the Markov blanket) is:
\[ P(z_i|\text{mb}(X_i)) = P(z_i|\text{parents}(X_i)) \Pi_{j \in \text{Children}(X_i)} P(z_j|\text{parents}(Z_j)) \]
- easily implemented in parallel systems

Main difficulties:
- difficult to tell if and when convergence has been achieved
- can be wasteful if Markov blanket large, prob doesn’t change much

Bayes nets inference algorithms - summary

Exact algorithms
- Variable Elimination and Factor Elimination
- Jointree algorithm
- Recursive conditioning

Approximative algorithms
- Belief propagation
- Stochastic sampling (Monte Carlo simulation)
  - direct sampling
  - importance sampling, likelihood weighting
- Monte Carlo Markov Chain