CHAPTER 14.4-5

INFERENCE IN BAYESIAN NETWORKS
Outline

- Exact inference by enumeration
- Exact inference by variable elimination
- Approximate inference by stochastic simulation
- Approximate inference by Markov chain Monte Carlo
Inference tasks

- Simple queries: Compute posterior marginal $P(X|\mathcal{E}=\mathbf{e})$.
- Conjunctive queries: $P(X_i, X_j|\mathcal{E}=\mathbf{e}) = P(X_i|\mathcal{E}=\mathbf{e})P(X_j|\mathcal{E}=\mathbf{e})$
- Optimal decisions: Decision networks include utility information; probabilistic inference required for $P(\text{outcome}|\text{action, evidence})$.
- Sensitivity analysis: Which probability values are most critical?
- Value of information: Which evidence to seek next?
- Explanation: Why do I need a new starter motor?
Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing it's explicit representation.

Simple query on the burglary network:

Recursive depth-first enumeration: $O(\frac{1}{w}p)O(n)O$ space, $O(\frac{1}{w}p)O(n)$ time.

Rewrite full joint entries using product of CPT entries:

$\prod_{i=1}^{w} P(B|\mathbf{m}) = \prod_{i=1}^{w} P(B|\mathbf{m}) P(e|m) P(a|e,B,e) P(j|a,B,e) P(m|a)$

Recursivde depth-first enumeration:

$\prod_{i=1}^{w} P(B|\mathbf{m}) P(e|m) P(a|e,B,e) P(j|a,B,e) P(m|a)$

Simple query on the burglary network:

Constructing is explicit representation

Slightly intelligent way to sum out variables from the joint without actually
Function `Enumeration-Ask(X, e, bn)` returns a distribution over `X`, the query variable, observed values for variables `E`, `bn`, a Bayesian network with variables `X`, `e`, `bn`, when inputs: `X`, `e`, `bn`. Returns a distribution over `X`.

```
function Enumeration-Ask(X, e, bn)
returns a distribution over X

where e is extended with
(Enumeration-All(REST-vars(e), e)
× ((X)p | h))p

else return
Enumeration-All(REST-vars(e), e)
× ((X)p | h))p

for each value x of X do
extend e with value x
for X
Normalize
( Enumeration-All(REST-vars(e), e)

return
Normalize
( Enumeration-All(REST-vars(e), e)

Empty?(vars)
then return 1.0

if Y has value y in e
then return P(y | Pa(Y))
else return
P(y | Pa(Y))
Enumerate-All(Rest(vars), e)

end

Return Normalize
( Enumeration-All(REST-vars(e), e)

for each value x of X do
extend e with value x
for X

end

X ∩ E ∩ \{X\} is a Bayesian network with variables E, observed values for variables E, the query variable X.

Input: X, the query variable

Function Enumeration-All(REST-vars(e), e) returns a real number
```
Evaluation tree

For each value of e, compute $P(a|w)P(a|l)P(b)$ repeatedly.

Enumeration is inefficient: repeated computation

$P(j|a) = 0.90$
$P(m|a) = 0.70$
$P(e) = 0.01$

$P(m|a) = 0.05$
$P(j|a) = 0.90$
$P(m|a) = 0.70$
$P(e) = 0.01$

$P(b) = 0.001$
$P(e) = 0.002$
$P(e) = 0.998$

$P(a|b,e) = 0.95$
$P(a|b,e) = 0.06$
$P(a|b,e) = 0.94$

Chapter 14.4 {56
Inference by variable elimination

Variable elimination: carry out summations right-to-left, storing intermediate results (factors) to avoid recomputation.

\[
P_j(B_j|\mathcal{M}) = \sum_a \left( \sum_{A|B;E} P_j(A|B) \sum_{M|A} P(E|A) \right)
\]

Chapter 14.4.5

Inference by variable elimination
Variable elimination: Basic operations

Summing out a variable from a product of factors:

\[(f_1 \cdot f_2) = f_1 + f_2\]

Pointwise product of factors:

\[f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \cdot f_2(y_1, \ldots, y_k, z_1, \ldots, z_l) = f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l)\]

E.g.,

\[f_1(a, b) \cdot f_2(b, c) = f(a, b, c)\]

Summing out a variable from a product of remaining factors:

\[Xf(x_1, \ldots, x_i) = f(x_1, \ldots, x_i, y_1, \ldots, y_k)\]

and up submatrices in pointwise product of remaining factors:

\[\exists f\times \ldots \times f = \exists f\times \ldots \times f + \exists f\times \ldots \times f\]
Variable elimination algorithm

Function Elimination-Ask \((X', e, qn)\) returns a distribution over \(X\)

\(\text{return Normalize(\text{Pointwise-Product}(\text{factors})))\)

\(\text{if var is a hidden variable then factors} \rightarrow \text{Sum-Out(var, factors)}\)

\(\text{foreach var in Vars do}\)

\(\text{factors} \rightarrow \text{Make-Factor(var, e)}\)

\(\text{if var is a hidden variable then factors} \rightarrow \text{Sum-Out(var, factors)}\)

\(\text{Reverse(Vars[\text{reverse}])}\)

Input: \(X\), the query variable

\(q_n\), a belief network specifying joint distribution \(P(X_1, \ldots, X_n)\)

\(e\), evidence specified as an event

\(\text{Elimination-Ask(X, e, qn)}\)

\(\text{Elimination-Ask(X, e, qn)}\)
Consider the query \( P(\text{JohnCalls} | \text{Burglary} = \text{true}) \)

and \( X = \text{JohnCalls}, E = \text{Burglary} \), and

\[ \{X\} = \{X\} \cap \{X\} \]

Sum over \( m \) is identically 1; \( M \) is irrelevant unless \( Y \in \text{Ancestors}(E) \)

Consider the query \( P(\text{JohnCalls} | \text{Burglary} = \text{true}) \)

Irrelevant variables
Irrelevant variables contd.

Defn: moral graph of Bayes net: marry all parents and drop arrows.

Defn: \( A \) is m-separated from \( B \) by \( C \) if m-separated by \( C \) in the moral graph.

Thm 2: For \( P(\text{John Calls | Alarm = true}) \), both Burglary and Earthquake are irrelevant.

Burglary and Earthquake are irrelevant.
Singly connected networks (or polytrees):

- Any two nodes are connected by at most one (undirected) path.
- Time and space cost of variable elimination are $O(n^2 p)$.

Multiply connected networks:

- Equivalent to counting 3SAT models.
- Can reduce 3SAT to exact inference.

Chapter 14.4

Complexity of exact inference

1. $A \land B \land C$
2. $C \land D \land A$
3. $B \land C \land D$

$\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$
Inference by stochastic simulation

Basic idea:
1) Draw $N$ samples from a sampling distribution $S$.
2) Compute an approximate posterior probability $P$.
3) Draw this converges to the true probability $P$.

Outline:
- Markov Chain Monte Carlo (MCMC): sample from a stochastic process.
- Likelihood weighting: use evidence to weight samples.
- Rejection sampling: reject samples disagreeing with evidence.

Chapter 14.4-5
Given the values of parents in $\langle X \rangle_{parents}$

Given the values of parents of

$x \rightarrow X$

for $i = 1$ to $n$
do

$x$ an event with $n$ elements

return $x$

Inputs: $n$, a belief network specifying joint distribution $P(X, Y)$

Function $\text{Prior-Sample}(n)$ returns an event sampled from $P(n)$

Sampling from an empty network
Example
## Example

### Priors

<table>
<thead>
<tr>
<th>P(C)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>0.5</td>
</tr>
<tr>
<td>T</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Conditional Probabilities

| P(W|S,R) |     |
|--------|-----|
| F      | 0.99 |
| T      | 0.99 |

| P(R|C) |     |
|------|-----|
| F    | 0.8 |
| T    | 0.2 |

| P(S|C) |     |
|------|-----|
| F    | 0.1 |
| T    | 0.9 |

### Evidence

- **Wet Grass**
  - P(W|S,R) = 0.99
  - P(W|¬S,¬R) = 0.99

- **Cloudy**
  - P(C) = 0.5

- **Rain**
  - P(R|C) = 0.8
  - P(R|¬C) = 0.2

- **Sprinkler**
  - P(S|C) = 0.1
  - P(S|¬C) = 0.9
Example
Sampling from an empty network contd.

Probability that PriorSample generates a particular event

\[ P_S(x_1 : \cdots : x_n) = \prod_{i=1}^n P(x_i \mid \text{parents}(X_i)) = P(x_1 : \cdots : x_n) \]

\[ \text{Shorthand: } \]

That is, estimates derived from PriorSample are consistent

\[ (u x \cdots 1 x)_d \approx (u x' \cdots 1 x)_d \]

E.g.,

\[ S(0; f; t; t) = 0 \]

Then we have

\[ u x' : \cdots : 1 x \]

Let \( N_{PS}(x_1 : \cdots : x_n) \) be the number of samples generated for event \( x_1 : \cdots : x_n \)

\[ (t', t', f, f)_d = 4.324 \times 8 \times 6 \times 0.324 = 0.5 = (t', t', f, f)_d \]

\[ \text{E.g., the true prior probability} \]

\[ (u x \cdots 1 x)_d = (u x \cdots 1 x)_{SdN}^{\infty \Rightarrow N_{\text{PS}}} = u x \cdots 1 x \]

Probability that PriorSample generates a particular event

Sampling from an empty network contd.
Rejection sampling

\[ P(X|e) \text{ estimated from samples agreeing with } e \]

local variables: \( N, \) a vector of counts over \( X, \) initially zero

function REJECTION-SAMPLING\((X,e,q,n,N)\) returns an estimate of \( P(X|e) \)

\[
\text{return } \left[ X \right] N \\
\text{where } x \text{ is the value of } X \text{ in } x \\
\text{if } x \text{ is consistent with } e \text{ then } \\
\text{for } j = 1 \text{ to } N \text{ do } \\
\text{local variables: } N, \text{ a vector of counts over } X, \text{ initially zero} \\
\text{normalize } N \\
\text{end for} \\
\text{return } \left[ X \right] N \\
\]

E.g. estimate \( P(\text{Rain}|\text{Sprinkler}=true) \)

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

Of those, 8 have \( \text{Rain}=true \) and 19 have \( \text{Rain}=false. \)

27 samples have \( \text{Sprinkler}=true \)

using 100 samples

Similar to a basic real-world empirical estimation procedure.
Analysis of rejection sampling

\[ P(X_j | e) = N_{PS}(X; e) \]

(algorithm defn.)

\[ N_{PS}(X; e) = N_{PS}(e) \]

(normalized by \( N_{PS}(e) \))

(Hence rejection sampling returns consistent posterior estimates)

\[ P(X; e) = P(e) \]

(definition of conditional probability)

\[ P(e) \]

(property of PriorSample)

\[ (\Theta | X) \]

(normalized by \( (\Theta | X) \))

(hopelessly expensive if \( P(e) \) is small)

Problem: hopeless expensive if \( P(e) \) is small

Hence rejection sampling returns consistent posterior estimates

\[ (\Theta | X) \approx \frac{(\Theta)_{PS}(e, X)}{(\Theta)_{PS}(e)} \]

(definition of conditional probability)

\[ \frac{(\Theta)_{PS}(e, X)}{(\Theta)_{PS}(e)} \]

(normalized by \( (\Theta)_{PS}(e) \))

\[ (\Theta)_{PS}(e, X) \]

( algorithm defn.)

\[ (\Theta | X)_{PS} = (\Theta | X) \]
Likelihood-weighting

Idea:

Fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

\[ \text{Likelihood-weighting} \]

\[
((x | \text{parents}) \mid x) \rightarrow \text{a random sample from } \mathbb{P}_x \times m \rightarrow m_
\]

\[
\text{then } \text{if } x \text{ has a value } x^* \text{ in } e \text{ for } i = 1 \text{ to } n \text{ do } x \text{ an event with } n \text{ elements: } x \rightarrow m
\]

\[
\text{function } \text{Weighted-Sample}(q \in e) \text{ returns an event and a weight}
\]

\[
\text{return } x, w
\]

\[
\text{function } \text{Normalize}(W[x]) \text{ returns a vector of weighted counts over } x \text{ where } m + [x]W \rightarrow [x]W
\]

\[
\text{function } \text{Likelihood-Weighting}(X, \text{e}, q, N) \text{ returns an estimate of } P(X | \text{e})
\]

\[
\text{local variables: } W, \text{ a vector of weighted counts over }, \text{initially zero}
\]

\[
\text{for } i = 1 \text{ to } N \text{ do } X
\]

\[
\text{end}
\]
Likelihood weighting example
Likelihood weighting example

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0.10</td>
<td>0.90</td>
</tr>
<tr>
<td>R</td>
<td>0.20</td>
<td>0.99</td>
</tr>
</tbody>
</table>

\[
\omega = 1.0
\]
Likelihood weighting example

\[ m = \frac{1}{0} \]
Likelihood weighting example

\[ m = 1.0 \times 0.1 \]
Likelihood weighting example

\[ w = 1.0 \times 0.1 \]

\[ P(C) \]

\[ P(R|C) \]

\[ P(S|R) \]

\[ P(W|S,R) \]

\[ w = 1.0 \times 0.1 \]

\[ P(C) \]

\[ P(R|C) \]

\[ P(S|R) \]

\[ P(W|S,R) \]

\[ w = 1.0 \times 0.1 \]

\[ P(C) \]

\[ P(R|C) \]

\[ P(S|R) \]

\[ P(W|S,R) \]

\[ w = 1.0 \times 0.1 \]
Likelihood weighting example

Chapter 14.4
Chapter 14.4
Likelihood weighting analysis

Because a few samples have nearly all the total weight, but performance still degrades with many evidence variables. Hence likelihood weighting returns consistent estimates.

\[
S_{WS}(z; e) = \prod_{i=1}^{\text{parents}(z)} \prod_{j=1}^{\text{parents}(e)} P(z_i | z_{\text{parents}(z)}) \cdot P(e_i | e_{\text{parents}(e)}) = P(z, e | z_{\text{parents}(z)})_{\text{SW}}
\]

Weighted sampling probability is somewhere between “prior and posterior” distribution.

Note: pays attention to evidence in ancestors only.

\[
S_{WS}(z; e) = P(z, e | z_{\text{parents}(z)}_{\text{SW}})
\]

Weight for a given sample is

Posteriors distribution

Somewhere between “prior and ancestor” only

Likelihood weighting analysis
Can also choose a variable to sample at random each time

\[
\text{MCMC-Ask}(X, e, bn, N) \quad \text{returns an estimate of } \Pr(X|e)
\]

local variables: \( N[X] \), \( X \) a vector of counts over \( X \), initially zero

for each \( Z \) in \( Z \) do

for \( I = 1 \) to \( N \) do

sample the value of \( Z \) from \( \Pr(Z_i|Z) \)
given the values of \( MB(Z_i) \) in \( x \)
end for

end for

\( x \) the current state of the network, initialized copied from \( e \)
\( N \) the non-evidence variables in \( bn \)

\( \text{Normalize}([X]_N) \) where \( x \) is the value of \( X \) in \( x \)

Given the values of \( Z \) in \( x \) from \( P(e|X) \)

Sample each variable in turn, keeping evidence fixed

Generate next state by sampling one variable given Markov blanket

State of network = current assignment to all variables.

Approximate inference using MCMC
Wander about for a while, average what you see.

With \( \text{sprinkler} = \text{true}, \ \text{WetGrass} = \text{true} \), there are four states:

The Markov chain
MCMC example contd.

Estimate \( P(\text{Rain} | \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true}) \) 

Sample Cloudy or Rain given its Markov blanket; repeat. 

\[ \hat{P}(\text{Rain} | \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true}) \]

Theorem: chain approaches stationary distribution:

Long-run fraction of time spent in each state is exactly proportional to its posterior probability.

E.g., visit 100 states, 31 have Rain = true, 69 have Rain = false.

Chapter 14.4-5
Markov blanket sampling

Markov blanket of Cloudy is Cloudy, Rain, Sprinkler, and WetGrass.

Markov blanket of Rain is Sprinkler and Rain.

Markov blanket of Sprinkler is Cloudy.

Markov blanket of WetGrass is Sprinkler.

Markov blanket of Cloudy is Cloudy.

Probability given the Markov blanket is calculated as follows:

\[
P(X_i | \text{parents}(X_i)) = \frac{\prod_{j \in \text{children}(X_i)} P(Z_j | \text{parents}(Z_j))}{\prod_{j \in \text{parents}(X_i)} P(X_j | \text{parents}(X_j))} 
\]

Easily implemented in message-passing parallel systems, brains

Main computational problems:

1) Difficult to tell if convergence has been achieved
2) Can be wasteful if Markov blanket is large:

\[
P(X_i | \text{mb}(X_i)) \text{ won't change much (law of large numbers)}
\]
Summary

Exact inference by variable elimination:
- Can handle arbitrary combinations of discrete and continuous variables
- Convergence can be very slow with probabilities close to 0 or 1
- LW, MCMC generally insensitive to topology
- LW does poorly when there is lots of (downstream) evidence
- Approximate inference by LW, MCMC:
  - Space = time, very sensitive to topology
  - Polytome on polytrees, NP-hard on general graphs
  - Exact inference by variable elimination: