Inference in Bayesian networks

Chapter 14.4–5
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 \( P(X_i | E = e) \)
   e.g., \( P(\text{NoGas} | \text{Gauge} = \text{empty}, \text{Lights} = \text{on}, \text{Starts} = \text{false}) \)

Conjunctive queries: \( P(X_i, X_j | E = e) = P(X_i | E = e)P(X_j | X_i, E = e) \)

Optimal decisions: decision networks include utility information;
probabilistic inference required for \( P(\text{outcome} | \text{action}, \text{evidence}) \)

Value of information: which evidence to seek next?

Sensitivity analysis: which probability values are most critical?

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 its explicit representation

Simple query on the burglary network:
\[ P(B|j, m) \]
\[ = \frac{P(B, j, m)}{P(j, m)} \]
\[ = \alpha P(B, j, m) \]
\[ = \alpha \sum_e \sum_a P(B, e, a, j, m) \]

Rewrite full joint entries using product of CPT entries:
\[ P(B|j, m) \]
\[ = \alpha \sum_e \sum_a P(B)P(e)P(a|B, e)P(j|a)P(m|a) \]
\[ = \alpha P(B) \sum_e P(e) \sum_a P(a|B, e)P(j|a)P(m|a) \]

Recursive depth-first enumeration: \( O(n) \) space, \( O(d^n) \) time
function \textsc{Enumeration-Ask}(X, e, bn) returns a distribution over $X$

inputs: $X$, the query variable
---
e, observed values for variables $E$
bn, a Bayesian network with variables $\{X\} \cup E \cup Y$

$Q(X) \leftarrow$ a distribution over $X$, initially empty

for each value $x_i$ of $X$ do
---
extend $e$ with value $x_i$ for $X$
$Q(x_i) \leftarrow \textsc{Enumerate-All}(\text{vars}[bn], e)$

return \textsc{Normalize}(Q(X))

\bigskip

function \textsc{Enumerate-All}(\text{vars}, e) returns a real number

if \textsc{Empty?}(\text{vars}) then return 1.0

$Y \leftarrow \text{First}(\text{vars})$

if $Y$ has value $y$ in $e$
---
then return $P(y \mid Pa(Y)) \times \textsc{Enumerate-All}(\text{Rest}(\text{vars}), e)$
else return $\sum_y P(y \mid Pa(Y)) \times \textsc{Enumerate-All}(\text{Rest}(\text{vars}), e_y)$

where $e_y$ is $e$ extended with $Y = y$
Evulation tree

Enumeration is inefficient: repeated computation
  e.g., computes $P(j|a)P(m|a)$ for each value of $e$
**Inference by variable elimination**

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

\[
\begin{align*}
P(B|j, m) & = \alpha \underbrace{P(B)}_{\overset{\hat{B}}{\sum_e}} \underbrace{P(e)}_{\overset{\hat{E}}{\sum_a}} \underbrace{P(a|B,e)}_{\overset{\hat{A}}{\sum_j}} \underbrace{P(j|a)}_{\overset{\hat{J}}{\sum_M}} P(m|a) \\
& = \alpha P(B) \sum_e P(e) \sum_a P(a|B,e) P(j|a) f_M(a) \\
& = \alpha P(B) \sum_e P(e) \sum_a f_A(a,b,e) f_J(a) f_M(a) \\
& = \alpha P(B) \sum_e P(e) \sum_A f_{\hat{A},J,M}(b,e) \text{(sum out } A) \\
& = \alpha P(B) f_{E,A,J,M}(b) \text{(sum out } E) \\
& = \alpha f_B(b) \times f_{E,A,J,M}(b)
\end{align*}
\]
Variable elimination: Basic operations

**Summing out** a variable from a product of factors:
- move any constant factors outside the summation
- add up submatrices in pointwise product of remaining factors

\[ \Sigma_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \Sigma_x f_{i+1} \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_X \]

assuming \( f_1, \ldots, f_i \) do not depend on \( X \)

**Pointwise product** of factors \( f_1 \) and \( f_2 \):
\[
  f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l)
  = f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l)
\]

E.g., \( f_1(a, b) \times f_2(b, c) = f(a, b, c) \)
Variable elimination algorithm

**function** Elimination-Ask\((X, e, bn)\) **returns** a distribution over \(X\)

**inputs:** \(X\), the query variable
\(e\), evidence specified as an event
\(bn\), a belief network specifying joint distribution \(P(X_1, \ldots, X_n)\)

\(factors \leftarrow []\); \(vars \leftarrow \text{Reverse}(\text{VARS}[bn])\)

**for each** \(var\) **in** \(vars\) **do**

\(factors \leftarrow [\text{Make-Factor}(var, e)|factors]\)

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

**return** Normalize(Pointwise-Product(\(factors\)))
Consider the query $P(\text{JohnCalls}|\text{Burglary} = \text{true})$

$$P(J|b) = \alpha P(b) \sum_{e} P(e) \sum_{a} P(a|b, e) P(J|a) \sum_{m} P(m|a)$$

Sum over $m$ is identically 1; $M$ is irrelevant to the query

Thm 1: $Y$ is irrelevant unless $Y \in \text{Ancestors}(\{X\} \cup E)$

Here, $X = \text{JohnCalls}$, $E = \{\text{Burglary}\}$, and $\text{Ancestors}(\{X\} \cup E) = \{\text{Alarm, Earthquake}\}$

so $\text{MaryCalls}$ is irrelevant

(Compare this to backward chaining from the query in Horn clause KBs)
Irrelevant variables contd.

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

Defn: A is **m-separated** from B by C iff separated by C in the moral graph

Thm 2: Y is irrelevant if m-separated from X by E

For \( P(\text{JohnCalls}|\text{Alarm}=\text{true}) \), both Burglary and Earthquake are irrelevant
Complexity of exact inference

Singly connected networks (or polytrees):
- any two nodes are connected by at most one (undirected) path
- time and space cost of variable elimination are $O(d^k n)$

Multiply connected networks:
- can reduce 3SAT to exact inference $\implies$ NP-hard
- equivalent to counting 3SAT models $\implies$ #P-complete

![Diagram of multiply connected network]

1. $A \lor B \lor C$
2. $C \lor D \lor \neg A$
3. $B \lor C \lor \neg D$
Inference by stochastic simulation

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

Outline:
- Sampling from an empty network
- Rejection sampling: reject samples disagreeing with evidence
- Likelihood weighting: use evidence to weight samples
- Markov chain Monte Carlo (MCMC): sample from a stochastic process
  whose stationary distribution is the true posterior
function \textsc{Prior-Sample}(\textit{bn}) \textbf{returns} an event sampled from \textit{bn} \\
\textbf{inputs:} \textit{bn}, a belief network specifying joint distribution \( P(X_1, \ldots, X_n) \) \\
\textbf{x} ← an event with \textit{n} elements \\
\textbf{for} \textbf{i} = 1 \textbf{to} \textit{n} \textbf{do} \\
\quad \textbf{x}_i ← a random sample from \( P(X_i \mid \text{parents}(X_i)) \) \\
\quad \text{given the values of \textit{Parents}(X_i) in \textbf{x}} \\
\textbf{return} \textbf{x}
Example

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

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

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

Cloudy

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

Sprinkler

Rain

Wet Grass

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

P(C)

.50

P(S|R|C)

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

|     | P(S|C) | P(W|S,R) |
|-----|-------|----------|
| T   | .10   |      .99 |
| F   | .50   |      .90 |
| T   | .80   |      .90 |
| F   | .20   |      .01 |

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

Chapter 14.4–5
Example

\[
P(C) = 0.50
\]

\[
P(S|C)
\]

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

\[
P(R|C)
\]

| C | P(S|C) |
|---|---|
| T | 0.10 |
| F | 0.50 |

| C | P(R|C) |
|---|---|
| T | 0.80 |
| F | 0.20 |

| S | R | P(W|S,R) |
|---|---|---|
| T | T | 0.99 |
| T | F | 0.90 |
| F | T | 0.90 |
| F | F | 0.01 |
Example

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

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

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

Chapter 14.4–5
Example

- Cloudy
- Rain
- Sprinkler
- Wet Grass

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

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

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

Chapter 14.4–5
Example

| C | P(S|C) | P(R|C) | P(W|S,R) |
|---|-------|-------|---------|
| T | .10   | .80   | .99     |
| F | .50   | .20   | .90     |
| S | R     |       |         |
| T | T     | .99   | .90     |
| T | F     | .90   | .90     |
| F | T     | .90   |         |
| F | F     | .01   |         |
Probability that \texttt{PriorSample} generates a particular event 

\[ S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i|\text{parents}(X_i)) = P(x_1 \ldots x_n) \]

i.e., the true prior probability

E.g., \[ S_{PS}(t, f, t, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t, t) \]

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

Then we have

\[
\lim_{N \to \infty} \hat{P}(x_1, \ldots, x_n) = \lim_{N \to \infty} \frac{N_{PS}(x_1, \ldots, x_n)}{N} = S_{PS}(x_1, \ldots, x_n) = P(x_1 \ldots x_n)
\]

That is, estimates derived from \texttt{PriorSample} are \textbf{consistent}

Shorthand: \( \hat{P}(x_1, \ldots, x_n) \approx P(x_1 \ldots x_n) \)
Rejection sampling

\( \hat{P}(X|e) \) estimated from samples agreeing with \( e \)

```
function REJECTION-SAMPLING( X, e, \( bn \), N ) returns an estimate of \( P(X|e) \)
    local variables: \( N \), a vector of counts over \( X \), initially zero
    for \( j = 1 \) to \( N \) do
        \( x \leftarrow \) PRIOR-SAMPLE( \( bn \) )
        if \( x \) is consistent with \( e \) then
            \( N[x] \leftarrow N[x]+1 \) where \( x \) is the value of \( X \) in \( x \)
    return NORMALIZE(\( N[X] \))
```

E.g., estimate \( P(\text{Rain}|\text{Sprinkler} = \text{true}) \) using 100 samples
27 samples have \( \text{Sprinkler} = \text{true} \)
   Of these, 8 have \( \text{Rain} = \text{true} \) and 19 have \( \text{Rain} = \text{false} \).

\( \hat{P}(\text{Rain}|\text{Sprinkler} = \text{true}) = \text{NORMALIZE}(\langle 8, 19 \rangle) = \langle 0.296, 0.704 \rangle \)

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

\[ \hat{P}(X|e) = \alpha N_{PS}(X, e) \quad \text{(algorithm defn.)} \]
\[ = N_{PS}(X, e)/N_{PS}(e) \quad \text{(normalized by } N_{PS}(e)\text{)} \]
\[ \approx P(X, e)/P(e) \quad \text{(property of PRIORSAMPLE)} \]
\[ = P(X|e) \quad \text{(defn. of conditional probability)} \]

Hence rejection sampling returns consistent posterior estimates

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

\( P(e) \) drops off exponentially with number of evidence variables!
Likelihood weighting

Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence.

```
function LIKELIHOOD-WEIGHTING(X, e, bn, N) returns an estimate of P(X|e)
    local variables: W, a vector of weighted counts over X, initially zero
    for j = 1 to N do
        x, w ← WEIGHTED-SAMPLE(bn)
        W[x] ← W[x] + w where x is the value of X in x
    return NORMALIZE(W[X])

function WEIGHTED-SAMPLE(bn, 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
```

Chapter 14.4–5
Likelihood weighting example

\[ w = 1.0 \]
Likelihood weighting example

$w = 1.0$
$w = 1.0$
Likelihood weighting example

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

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

\[
\begin{array}{c|c}
C & P(S|C) \\
\hline
T & .10 \\
F & .50 \\
\end{array}
\]

\[
\begin{array}{c|c}
C & P(R|C) \\
\hline
T & .80 \\
F & .20 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
S & R & P(W|S,R) \\
\hline
T & T & .99 \\
T & F & .90 \\
F & T & .90 \\
F & F & .01 \\
\end{array}
\]

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

$$w = 1.0 \times 0.1 \times 0.99 = 0.099$$
Likelihood weighting analysis

Sampling probability for \texttt{WEIGHTEDSAMPLE} is
\[
S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i|\text{parents}(Z_i))
\]
Note: pays attention to evidence in \texttt{ancestors} only
\[\Rightarrow\] somewhere “in between” prior and posterior distribution

Weight for a given sample \(\mathbf{z}, \mathbf{e}\) is
\[
w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i|\text{parents}(E_i))
\]

Weighted sampling probability is
\[
S_{WS}(\mathbf{z}, \mathbf{e})w(\mathbf{z}, \mathbf{e})
= \prod_{i=1}^{l} P(z_i|\text{parents}(Z_i)) \prod_{i=1}^{m} P(e_i|\text{parents}(E_i))
= P(\mathbf{z}, \mathbf{e}) \text{ (by standard global semantics of network)}
\]

Hence likelihood weighting returns consistent estimates
but performance still degrades with many evidence variables
because a few samples have nearly all the total weight
Approximate inference using MCMC

“State” of network = current assignment to all variables.

Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

function MCMC-Ask\((X, e, bn, N)\) returns an estimate of \(P(X|e)\)

local variables: \(N[X]\), a vector of counts over \(X\), initially zero
- \(Z\), the nonevidence variables in \(bn\)
- \(x\), the current state of the network, initially copied from \(e\)

initialize \(x\) with random values for the variables in \(Y\)
for \(j = 1\) 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 \(x\)
    \(N[x] \leftarrow N[x] + 1\) where \(x\) is the value of \(X\) in \(x\)
  return \(\text{Normalize}(N[X])\)

Can also choose a variable to sample at random each time
The Markov chain

With $Sprinkler = true, WetGrass = true$, there are four states:

Wander about for a while, average what you see
MCMC example contd.

Estimate $P(Rain|Sprinkler = true, WetGrass = true)$

Sample *Cloudy* or *Rain* given its Markov blanket, repeat.
Count number of times *Rain* is true and false in the samples.

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

$\hat{P}(Rain|Sprinkler = true, WetGrass = true) = \text{Normalize}(\langle 31, 69 \rangle) = \langle 0.31, 0.69 \rangle$

Theorem: chain approaches **stationary distribution**:
- long-run fraction of time spent in each state is exactly proportional to its posterior probability
Markov blanket sampling

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

Probability given the Markov blanket is calculated as follows:

\[ P(x'_i | mb(X_i)) = P(x'_i | parents(X_i)) \Pi_{Z_j \in \text{Children}(X_i)} P(z_j | parents(Z_j)) \]

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:
\[ P(X_i | mb(X_i)) \] won’t change much (law of large numbers)
Summary

Exact inference by variable elimination:
- polytime on polytrees, NP-hard on general graphs
- space = time, very sensitive to topology

Approximate inference by LW, MCMC:
- LW does poorly when there is lots of (downstream) evidence
- LW, MCMC generally insensitive to topology
- Convergence can be very slow with probabilities close to 1 or 0
- Can handle arbitrary combinations of discrete and continuous variables