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., $P(\text{NoGas}|\text{Gauge = empty, Lights = on, Starts = false})$
- 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 $P(\text{outcome}|\text{action, evidence})$
- 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:

$$P(B_i | j, m) = \frac{P(B_i, j, m)}{P(j, m)} = \frac{\alpha \sum_{c} \sum_{a} P(B, c, a, j, m)}{\alpha \sum_{c} \sum_{a} P(B, c, a, j, m)}$$

Rewrite full joint entries using product of CPT entries:

$$P(B_i | j, m) = \frac{\alpha \sum_{c} \sum_{a} P(B_i)P(c)P(a | B, c)P(j | a)P(m | a)}{\alpha \sum_{c} \sum_{a} P(c)P(a | B, c)P(j | a)P(m | a)}$$

Recursive depth-first enumeration: $O(n)$ space, $O(d^n)$ time

Evaluation tree

| $P(b)$ | $P(c)$ | $P(d | c)$ |
|--------|--------|----------|
| .001   | .002   | .998     |

| $P(a | b, c)$ | $P(a | b, \neg c)$ |
|--------------|------------------|
| .95          | .05              |

| $P(a | \neg b, c)$ | $P(a | \neg b, \neg c)$ |
|------------------|-----------------------|
| .94              | .06                   |

| $P(m | a)$ | $P(m | \neg a)$ |
|-----------|-------------|
| .90       | .10         |

| $P(\neg m | a)$ | $P(\neg m | \neg a)$ |
|----------------|-------------------|
| .70            | .30               |

| $P(\neg m | \neg a)$ | $P(\neg m | a)$ |
|--------------------|---------------|
| .01                | .99           |

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

\[ P(B|m) = \frac{\alpha P(B) \sum_{x} P(c) \sum_{y} P(a|B,c) P(j|a) P(m|a)}{\sum_{x} P(c) \sum_{y} P(a|B,c) P(j|a) f_M(a)} \]

\[ = \frac{\alpha P(B) \sum_{x} P(c) \sum_{y} P(a|B,c) P(j|a) f_M(a)}{\sum_{x} P(c) \sum_{y} P(a|B,c) P(j|a) f_M(a)} \]

\[ = \frac{\alpha P(B) \sum_{x} P(c) \sum_{y} P(a|B,c) f_M(a)}{\sum_{x} P(c) \sum_{y} f_M(a)} \]

\[ = \alpha f_B(b) \times f_{E|JM}(b) \]

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

\[ P(J|b) = \alpha P(b) \sum_{x} P(c) \sum_{y} P(a|b,c) P(j|a) P(m|a) \]

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

so \( \text{MaryCalls} \) is irrelevant

(Compare this to backward chaining from the query in Horn clause KBs)

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

\[ \sum_{x} f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_{i+1} \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_{i+1} \times \cdots \times f_k \]

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

Pointwise product of factors \( f_i \) and \( f_j \):

\[ f_{1}(x_1, \ldots, x_i, y_1, \ldots, y_i) \times f_{2}(y_1, \ldots, y_i, z_1, \ldots, z_i) = f_{1}(x_1, \ldots, x_i, y_1, \ldots, y_i, z_1, \ldots, z_i) \]

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

Irrelevant variables

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 \( \text{Burglary} \) and \( \text{Earthquake} \) are irrelevant

Variable elimination algorithm

```
function ELIMINATION-ASSOC(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(X1, ..., Xn)

factors := []
for each var in vars do
    factors := [MAKE-FACTOR(var, e)[factors]
if var is a hidden variable then factors := SUM-OUT(var, factors)
return NORMALIZE(POINTWISE-PRODUCT(factors))
```

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 \( \text{Burglary} \) and \( \text{Earthquake} \) 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 \( O(d^n) \)

Multiply connected networks:
- can reduce 3SAT to exact inference \( \Rightarrow \) NP-hard
- equivalent to counting 3SAT models \( \Rightarrow \#P \)-complete
Inference by stochastic simulation

Basic idea:
1) Draw $N$ samples from a sampling distribution $S$
2) Compute an approximate posterior probability $P$
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

**Sampling from an empty network**

```python
def Prior-Sample(bn):
    inputs: bn, a belief network specifying joint distribution $P(X_1, \ldots, X_n)$
    x -- an event with $n$ elements
    for $i = 1$ to $n$
        $x_i$ -- a random sample from $P(X_i | \text{parents}(X_i))$
given the values of $\text{Parents}(X_i)$ in $x$
    return x
```

Example

```
Cloudy
Rain
Sprinkler
Wet Grass

P(C)
T .50
F .50

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

P(S|C)
T .10
F .90

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

Chapter 14.4

Example

```
Cloudy
Rain
Sprinkler
Wet Grass

P(C)
T .50
F .50

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

P(S|C)
T .10
F .90

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

Chapter 14.4
Sampling from an empty network contd.

Probability that \( \text{PriorSample} \) 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., \( S_{PS}(t, f, t) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324 = P(t, f, t) \)

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

Then we have

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

That is, estimates derived from \( \text{PriorSample} \) are consistent

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

Rejection sampling

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

function \( \text{Rejection-Sampling}(X, e, b, n) \) returns an estimate of \( P(X | e) \)

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

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

\( x \leftarrow \text{Prior-Sample}(e) \)

if \( x \) is consistent with \( e \) then

\( N[x] \leftarrow N[x] + 1 \) where \( x \) is the value of \( X \) in \( x \)

end if

end for

return \( \text{Normalize}(N[X]) \)

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 \( \text{Rain} = \text{false} \).

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

Similar to a basic real-world empirical estimation procedure

Analysis of rejection sampling

\( P(X | e) = aN_{PS}(X, e) \) (algorithm defn.)

\( = \frac{N_{PS}(X, e)}{N_{PS}(e)} \) (normalized by \( N_{PS}(e) \))

\( \approx \frac{P(X, e)}{P(e)} \) (property of \( \text{PriorSample} \))

\( = P(X | e) \) (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.

**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 \(N\)
  - \(x, w \leftarrow \text{Weighted-Sample}(bn)\)
  - \(W[x] = W[x] + w\) where \(x\) is the value of \(X\) in \(x\)
- Return \(\text{Normalize}(W[X])\)

**Weighted-Sample** \((bn, e)\) returns an event and a weight

- \(x\) an event with \(n\) elements; \(w\) 1
- For \(i = 1\) to \(n\)
  - If \(X_i\) has a value \(x_i\) in \(e\), then \(w \leftarrow w \times P(X_i = x_i | \text{parents}(X_i))\)
  - Else \(x_i \leftarrow \text{a random sample from } P(X_i | \text{parents}(X_i))\)
- Return \(x, w\)

**Likelihood weighting example**

<table>
<thead>
<tr>
<th>Cloudy</th>
<th>Rain</th>
<th>Sprinkler</th>
<th>Wet Grass</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>.80</td>
<td>.20</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>.10</td>
<td>.50</td>
</tr>
</tbody>
</table>

\(P(R|C)\)

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>W(S,R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>.99</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>.90</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>.90</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>.01</td>
</tr>
</tbody>
</table>

\(P(S|C)\)

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>W(S,R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>.90</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>.90</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>.90</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>.01</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>W(S,R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>.90</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>.90</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>.90</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>.01</td>
</tr>
</tbody>
</table>

\(P(C)\)

<table>
<thead>
<tr>
<th>C</th>
<th>T</th>
<th>P(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>.50</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>.01</td>
</tr>
</tbody>
</table>

\(w = 1\)

\(w = 1 \times 0.1\)

\(w = 1 \times 0.1\)
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, ln, 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 ln
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, in Z do
    sample the value of Z, in x from P(Z|mb(Z))
given the values of MB(Z) in x
N[z] = N[z] + 1 where z is the value of X in x
return 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
P(Rain|Sprinkler = true, WetGrass = true) = Normalize(31, 69) = (0.31, 0.69)

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

Likelihood weighting example

\[ w = 1.0 \times 0.1 \]

Likelihood weighting analysis

Sampling probability for WeightedSample is

\[ S(u, e) = \prod_{i=1}^{m} P(e_i|parents(E_i)) \]

Note: pays attention to evidence in ancestors only
somewhere “in between” prior and posterior distribution.

Weight for a given sample u, e is

\[ w(u, e) = \prod_{i=1}^{m} P(e_i|parents(E_i)) \]

Weighted sampling probability is

\[ S(u, e)w(u, e) = \prod_{i=1}^{m} P(z_i|parents(Z_i)) \prod_{i=1}^{m} P(e_i|parents(E_i)) \]

\[ = P(u, e) \] (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.

Likelihood weighting example

\[ w = 1.0 \times 0.1 \times 0.99 = 0.099 \]
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:

\[ P(x_i|mb(X_i)) = P(x_i|\text{parents}(X_i)) \prod_{j \in \text{Children}(X_i)} P(z_j|\text{parents}(Z_j)) \]

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