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 $P(X_i|E=e)$
   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
Enumetation algorithm

```plaintext
function 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 \text{a distribution over } X, \text{ initially empty}
    for each value } x_i \text{ of } X \text{ do}
        extend e with value } x_i \text{ for } X
        Q(x_i) \leftarrow ENUMERATE-ALL(Ordered-Vars[bn], e)
    return \text{NORMALIZE}(Q(X))

function ENUMERATE-ALL(vars, e) returns a real number
    if EMPTY?(vars) then return 1.0
    Y \leftarrow FIRST(vars)
    if \ Y \text{ has value } y \text{ in } e
        then return } P(y | Pa(Y)) \times ENUMERATE-ALL(REST(vars), e)
    else return } \sum_y P(y | Pa(Y)) \times ENUMERATE-ALL(REST(vars), e_y)
        where } e_y \text{ is } e \text{ extended with } Y = y
```
Enumeration is inefficient: repeated computation
e.g., computes $P(j|a)P(m|a)$ for each value of $e$
Inference by variable elimination

Store intermediate results, a.k.a. factors, to avoid recomputation.

Variable elimination: carry out computations right-to-left

\[
P(B|j, m) = \alpha \underbrace{P(B)}_B \underbrace{\sum_e P(e)}_E \underbrace{\sum_a P(a|B, e)}_A \underbrace{P(j|a)}_J \underbrace{P(m|a)}_M
\]

\[
= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) P(j|a) f_M(a)
\]

\[
= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) f_J(a) f_M(a)
\]

\[
= \alpha P(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a)
\]

\[
= \alpha P(B) \sum_e P(e) f_{EJA}(b, e) \text{ (sum out } A)\]

\[
= \alpha P(B) f_{EJA}(b) \text{ (sum out } E)\]

\[
= \alpha f_B(b) \times f_{EJA}(b)
\]

\[
f_M(A) = \begin{pmatrix} P(m|a) \\ P(m|\neg a) \end{pmatrix}
\]

\[
f_J(A) = \begin{pmatrix} P(j|a) \\ P(j|\neg a) \end{pmatrix}
\]

\[
f_M(A) = \begin{pmatrix} P(j|a) P(m|a) \\ P(j|\neg a) P(m|\neg a) \end{pmatrix}
\]

\[
f_A(A, B, E) = \begin{pmatrix} P(\neg a|\neg b, \neg e) & P(\neg a|\neg b, e) / P(a|\neg b, \neg e) & P(a|\neg b, e) \\ P(\neg a|b, \neg e) & P(\neg a|b, e) / P(a|b, \neg e) & P(a|b, e) \end{pmatrix}
\]
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

\[ \sum_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \sum_x f_{i+1} \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_X \]
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 \(P(X_1, \ldots, X_n)\)

\begin{align*}
\text{factors} &\leftarrow []; \text{vars} \leftarrow \text{REVERSE(VARS[bn])} \\
\text{for each } \text{var in vars do} \\
&\quad \text{factors} \leftarrow [\text{MAKE-FACTOR(var, e)}]|\text{factors} \\
&\quad \text{if var is a hidden variable then factors} \leftarrow \text{SUM-OUT(var, factors)} \\
\text{return } \text{NORMALIZE(POINTWISE-PRODUCT(factors))}
\end{align*}
Irrelevant variables

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 $m = 1$.

Therefore, $\text{MaryCalls}$ is irrelevant to the query.

Theorem: $Y$ 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
Another example

Consider the query $P(A|m, \neg e)$

![Diagram with nodes B, E, A, J, and M connected by edges]
Irrelevant variables contd.

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

Defn: Y is m-separated from X by E iff separated by E in the moral graph

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

For $P(JohnCalls|Alarm=true)$, 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 $O(d^kn)$

Multiply connected networks
- NP-hard (actual #P-complete)
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
**Sampling from an empty network**

```latex
function PRIOR-SAMPLE(bn) returns an event sampled from bn
inputs: bn, a belief network specifying joint distribution \( P(X_1, \ldots, X_n) \)

\( x \leftarrow \text{an event with } n \text{ elements} \)
\[ \text{for } i = 1 \text{ to } n \text{ do} \]
\[ x_i \leftarrow \text{a random sample from } P(X_i \mid \text{Parents}(X_i)) \]
\text{return } x
```
Example

P(C)
.50

| 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**

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

**Rain**

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

**Wet Grass**

P(C) = .50
Example

Cloudy

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

Sprinkler

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

Rain

Wet Grass

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

- **P(C)**: 0.50
- **P(S|C)**:
  - T: 0.10
  - F: 0.50
- **P(R|C)**:
  - T: 0.80
  - F: 0.20
- **P(W|S,R)**:
  - TT: 0.99
  - TF: 0.90
  - FT: 0.90
  - FF: 0.01
Example

- Cloudy
  - P(C) = 0.50
  - P(S|C)
    - T: 0.10
    - F: 0.50
  - P(W|S,R)
    - T T: 0.99
    - T F: 0.90
    - F T: 0.90
    - F F: 0.01

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

- Wet Grass
Example

Cloudy

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

P(C) = .50

Sprinkler

Rain

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

Wet Grass

| S | R | P(W|S,R) |
|---|---|---------|
| T | T | .99     |
| T | F | .90     |
| F | T | .90     |
| F | F | .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     |
Sampling from an empty network contd.

Probability that `PriorSample` generates a particular event

\[ S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i|Parents(X_i)) = P(x_1 \ldots x_n) \]
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 `PriorSample` are **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 \( e \)

```plaintext
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
        \( x \) ← 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 \( P(Rain|Sprinkler = true) \) using 100 samples
- 27 samples have \( Sprinkler = true \)
  - Of these, 8 have \( Rain = true \) and 19 have \( Rain = false \).

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

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

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

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

\begin{verbatim}
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
\end{verbatim}
### Likelihood weighting example

- **P(C)**
  - .50

- **Cloudy**
  - **P(S|R)**
    - T T: .99
    - T F: .90
    - F T: .90
    - F F: .01

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

- **Wet Grass**
  - **P(W|S,R)**
    - T T: .99
    - T F: .90
    - F T: .90
    - F F: .01

- **w = 1.0**
Likelihood weighting example

\[ P(C) = 0.50 \]

\[ \begin{array}{c|c}
C & P(S|C) \\
T & 0.10 \\
F & 0.50 \\
\end{array} \]

\[ \begin{array}{c|c|c|c}
S & R & P(W|S,R) \\
T & T & 0.99 \\
T & F & 0.90 \\
F & T & 0.90 \\
F & F & 0.01 \\
\end{array} \]

\[ w = 1.0 \]
Likelihood weighting example

\[ w = 1.0 \]
Likelihood weighting example

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

\[
\begin{array}{c|c|c|c|c|c|c}
\text{C} & \text{P(S|C)} & \text{P(W|S,R)} \\
\hline
\text{T} & .10 & \text{T} & .99 \\
\text{F} & .50 & \text{T} & .90 \\
\text{T} & .10 & \text{F} & .90 \\
\text{F} & .50 & \text{F} & .01 \\
\end{array}
\]

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

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

\[
P(C) = 0.50
\]

\[
P(S|C)
\]

\[
\begin{array}{c|c}
C & P(S|C) \\
\hline
T & 0.10 \\
F & 0.50 \\
\end{array}
\]

\[
P(R|C)
\]

\[
\begin{array}{c|c}
C & P(R|C) \\
\hline
T & 0.80 \\
F & 0.20 \\
\end{array}
\]

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

\[
\begin{array}{c|c|c}
S & R & P(W|S,R) \\
\hline
T & T & 0.99 \\
T & F & 0.90 \\
F & T & 0.90 \\
F & F & 0.01 \\
\end{array}
\]

\[w = 1.0 \times 0.1 \times 0.99 = 0.099\]
Likelihood weighting analysis

Sampling probability for `WEIGHTEDSAMPLE` is

\[ S_{WS}(z, e) = \prod_{i=1}^{l} P(z_i | Parents(Z_i)) \]

Note: pays attention to evidence in **ancestors** only

\[ \Rightarrow \text{ somewhere “in between” prior and posterior distribution} \]

Weight for a given sample \(z, e\) is

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

Weighted sampling probability is

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

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

Start by assigning all variables in network random values.

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

Generate sequence of states by sampling one variable at a time, and choosing new value for variable given its Markov blanket.

Variables can be sampled sequentially or at random, keeping evidence fixed.
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**: over a long run, fraction of time spent in each state is exactly posterior probability
**MCMC Algorithm**

```
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
        N[x] ← N[x] + 1 where x is the value of X in x
        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
    return Normalize(N[X])
```
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:

\[
P(x_i' | MB(X_i)) = P(x_i' | Parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j | Parents(Z_j))
\]

Main computational problems:

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

\[
P(X_i | MB(X_i)) \text{ won’t change much}
\]

Sampling with Markov blanket called *Gibbs sampler*.

Other sampling schemes are used in AI (e.g., simulated annealing, Metropolis).
Summary

Exact inference by variable elimination:
- polytime on polytrees
- NP-hard on general graphs
- 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