Inference in Bayesian Networks

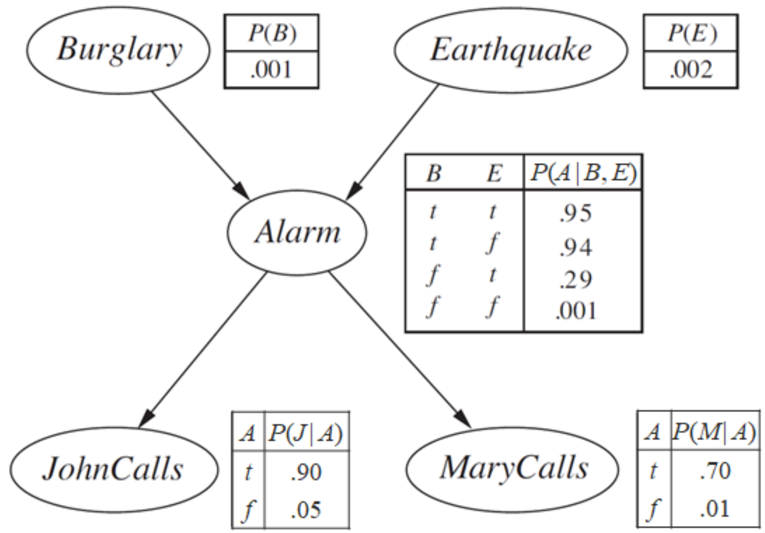
Contents

- Exact Inference by Enumeration
- Variable Elimination Algorithm
  - Operations on Factors
  - Variable Ordering and Variable Relevance
  - Complexity of Exact Inference
- Approximate Inference by Stochastic Simulation
  - Direct Sampling Methods
  - Inference by Markov Chain Simulation

Exact Inference by Enumeration

- Simple query on the burglary network:

$$\mathbf{P}(B \mid j, m)$$
$$= \mathbf{P}(B, j, m) / P(j, m)$$
$$= \alpha \mathbf{P}(B, j, m)$$
$$= \alpha \sum_e \sum_a \mathbf{P}(B, e, a, j, m)$$



- Rewrite full joint entries using product of CPT entries

For $Burglary = true$:

$$P(b \mid j, m)$$
$$= \alpha \sum_e \sum_a P(b)\, P(e)\, P(a \mid b, e)\, P(j \mid a)\, P(m \mid a)$$
$$= \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e)\, P(j \mid a)\, P(m \mid a)$$

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

Exact Inference by Enumeration

function ENUMERATION-ASK(X, $\mathbf{e}$, bn) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x of X **do**
    $\mathbf{Q}(x) \leftarrow$ ENUMERATE-ALL(bn.VARS, $\mathbf{e}_x$)
      where $\mathbf{e}_x$ is $\mathbf{e}$ extended with $X = x$
  return NORMALIZE($\mathbf{Q}(X)$)

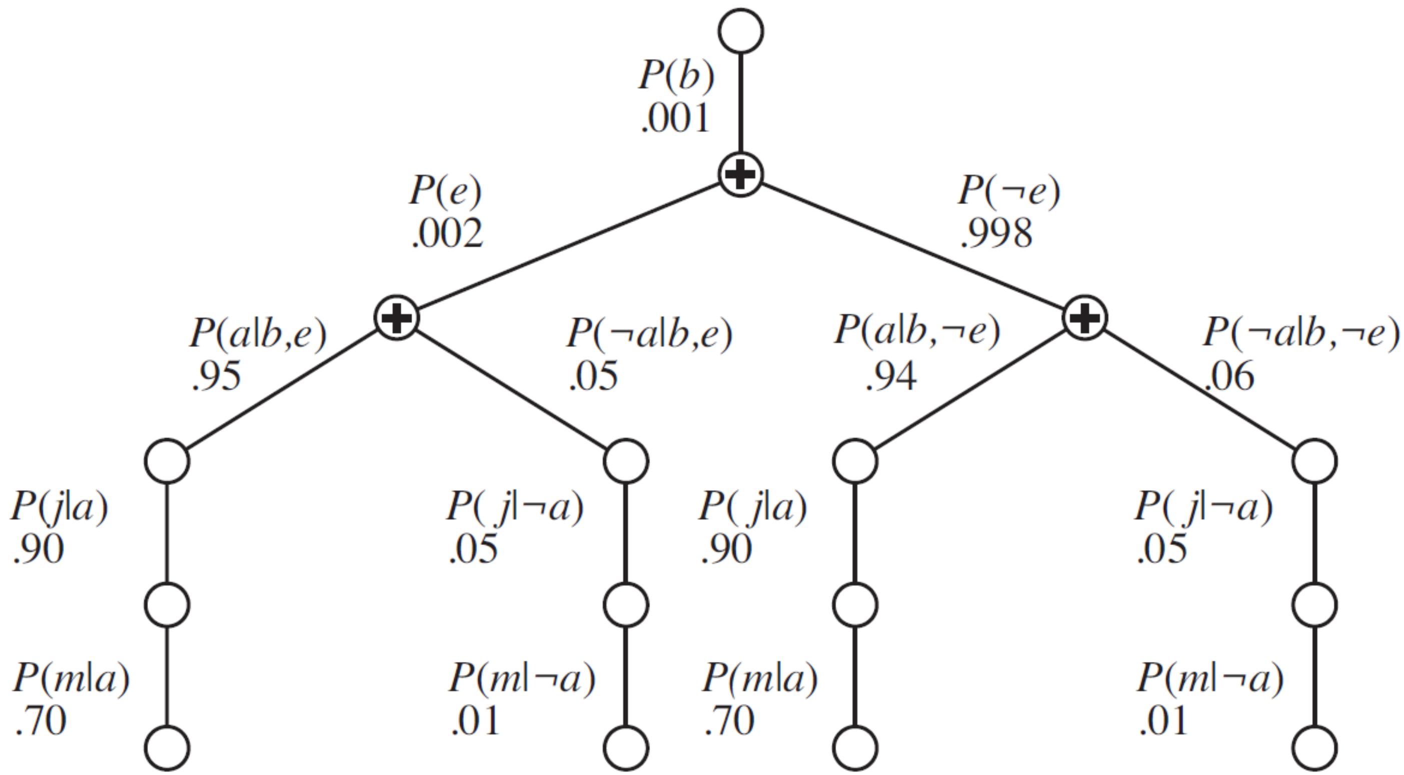
$$\boxed{\mathbf{P}(B \mid j, m) = \alpha \textstyle\sum_e \sum_a \mathbf{P}(B, e, a, j, m)}$$

---

function ENUMERATE-ALL($vars$, $\mathbf{e}$) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in $\mathbf{e}$
    then return $P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
    else return $\sum_y P(y \mid parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is $\mathbf{e}$ extended with $Y = y$

$$\boxed{P(b \mid j, m) = \alpha P(b) \textstyle\sum_e P(e) \sum_a P(a \mid b, e) P(j \mid a) P(m \mid a)}$$

Exact Inference by Enumeration



$$P(b \mid j, m) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e) P(j \mid a) P(m \mid a)$$

- Enumeration is inefficient: repeated computation
  - e.g., computes $P(j \mid a) P(m \mid a)$ for each value of e

Variable Elimination Algorithm

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

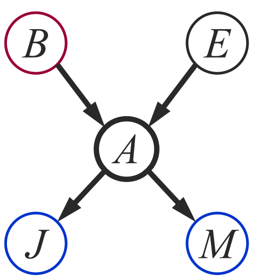
$$\mathbf{P}(B \mid j,m) = \alpha \underbrace{\mathbf{P}(B)}_{\mathbf{f}_1(B)} \sum_e \underbrace{P(e)}_{\mathbf{f}_2(E)} \sum_a \underbrace{\mathbf{P}(a \mid B,e)}_{\mathbf{f}_3(A,B,E)} \underbrace{P(j \mid a)}_{\mathbf{f}_4(A)} \underbrace{P(m \mid a)}_{\mathbf{f}_5(A)}$$

- $\mathbf{f}_5(A)$ and $\mathbf{f}_4(A)$ are 2-element vectors that just depend on A because J and M are fixed by the query:

$$\mathbf{f}_5(A) = \begin{pmatrix} f_5(a) \\ f_5(\neg a) \end{pmatrix} = \begin{pmatrix} P(m \mid a) \\ P(m \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0.70 \\ 0.01 \end{pmatrix}$$

$$\mathbf{f}_4(A) = \begin{pmatrix} P(j \mid a) \\ P(j \mid \neg a) \end{pmatrix} = \begin{pmatrix} 0.90 \\ 0.05 \end{pmatrix}$$

$$\mathbf{f}_4(A) \times \mathbf{f}_4(A) = \begin{pmatrix} 0.90 \times 0.70 \\ 0.05 \times 0.01 \end{pmatrix}$$



Variable Elimination Algorithm

- $\mathbf{f}_3(A, B, E)$ is an 8-element vector:

$$\begin{array}{r} \\ \mathbf{f}_3(a, B, E) \rightarrow \\ \\ \\ \mathbf{f}_3(A,B,E) = \\ \\ \\ \mathbf{f}_3(\neg a, B, E) \rightarrow \\ \\ \end{array} \begin{pmatrix} P(a \mid b, e) \\ P(a \mid b, \neg e) \\ P(a \mid \neg b, e) \\ P(a \mid \neg b, \neg e) \\ P(\neg a \mid b, e) \\ P(\neg a \mid b, \neg e) \\ P(\neg a \mid \neg b, e) \\ P(\neg a \mid \neg b, \neg e) \end{pmatrix} = \begin{pmatrix} 0.95 \\ 0.94 \\ 0.29 \\ 0.001 \\ 0.05 \\ 0.06 \\ 0.71 \\ 0.999 \end{pmatrix}$$

- In terms of factors, the query expression is written as

$$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \sum_a \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$$

Variable Elimination Algorithm

- The × operator is not ordinary matrix multiplication but instead the pointwise product operation (See page 12)

- The process of evaluating a query expression is the process of summing out variables (right to left) from pointwise products of factors to produce new factors, eventually yielding the posterior distribution over the query variable

$$\mathbf{P}(B \mid j,m) = \alpha\, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \sum_a \mathbf{f}_3(A,B,E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$$

$$\mathbf{P}(B \mid j,m) = \alpha\, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B,E)$$

$$\mathbf{P}(B \mid j,m) = \alpha\, \mathbf{f}_1(B) \times \mathbf{f}_7(B)$$

Variable Elimination Algorithm

- First, we sum out A: $\mathbf{P}(B \mid j,m) = \alpha \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \sum_a \mathbf{f}_3(A,B,E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$

$$\mathbf{f}_6(B,E) = \sum_a \mathbf{f}_3(A,B,E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$$

$$= \mathbf{f}_3(a,B,E) \times f_4(a) \times f_5(a) + \mathbf{f}_3(\neg a,B,E) \times f_4(\neg a) \times f_5(\neg a)$$

$$\begin{pmatrix} 0.95 \\ 0.94 \\ 0.29 \\ 0.001 \end{pmatrix} \times 0.9 \times 0.7 + \begin{pmatrix} 0.05 \\ 0.06 \\ 0.71 \\ 0.999 \end{pmatrix} \times 0.05 \times 0.01 = \begin{pmatrix} 0.5985250 \\ 0.5922300 \\ 0.1830550 \\ 0.0011255 \end{pmatrix}$$

Now, we are left with the expression

$$\mathbf{P}(B \mid j,m) = \alpha \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B,E)$$

Variable Elimination Algorithm

- Next, we sum out E:

$$\mathbf{f}_7(B) = \sum_e \mathbf{f}_2(E) \times \mathbf{f}_6(B, E)$$
$$= f_2(e) \times \mathbf{f}_6(B, e) + f_2(\neg e) \times \mathbf{f}_6(B, \neg e)$$

where $\mathbf{f}_2(E) = \begin{pmatrix} f_2(e) \\ f_2(\neg e) \end{pmatrix} = \begin{pmatrix} P(e) \\ P(\neg e) \end{pmatrix} = \begin{pmatrix} 0.002 \\ 0.998 \end{pmatrix}$

$$0.002 \times \begin{pmatrix} 0.5985250 \\ 0.1830550 \end{pmatrix} + 0.998 \times \begin{pmatrix} 0.5922300 \\ 0.0011255 \end{pmatrix} = \begin{pmatrix} 0.592242590 \\ 0.001489359 \end{pmatrix}$$

This leaves the expression

$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \mathbf{f}_7(B)$, where $\mathbf{f}_1(B) = \begin{pmatrix} P(b) \\ P(\neg b) \end{pmatrix} = \begin{pmatrix} 0.001 \\ 0.999 \end{pmatrix}$

Variable Elimination Algorithm

- Finally, $\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \mathbf{f}_7(B)$ can be evaluated by taking the pointwise product and normalizing the result

$$\alpha \times \underbrace{\begin{pmatrix} 0.001 \\ 0.999 \end{pmatrix} \times \begin{pmatrix} 0.592242590 \\ 0.001489359 \end{pmatrix}}_{\text{pointwise product}} = \alpha \times \underbrace{\begin{pmatrix} 0.000592242590 \\ 0.001487869641 \end{pmatrix}}_{\text{resulting factor}} = \begin{pmatrix} 0.2847 \\ 0.7153 \end{pmatrix}$$

Operations on Factors

- The pointwise product $\mathbf{f}_1(A, B) \times \mathbf{f}_2(B, C) = \mathbf{f}_3(A, B, C)$

  Variables of $\mathbf{f}_3$ $\cdots$ union of the variables in $\mathbf{f}_1$ and $\mathbf{f}_2$

  Elements of $\mathbf{f}_3$ $\cdots$ product of the corresponding elements in $\mathbf{f}_1$ and $\mathbf{f}_2$

A	B	$\mathbf{f}_1(A, B)$	B	C	$\mathbf{f}_2(B, C)$	A	B	C	$\mathbf{f}_3(A, B, C)$
T	T	.3	T	T	.2	T	T	T	.3×.2
T	F	.7	T	F	.8	T	T	F	.3 ×.8
F	T	.9	F	T	.6	T	F	T	.7 ×.6
F	F	.1	F	F	.4	T	F	F	.7 ×.4
						F	T	T	.9 ×.2
						F	T	F	.9 ×.8
						F	F	T	.1 ×.6
						F	F	F	.1 ×.4

Operations on Factors

- To sum out A from $\mathbf{f}_3(A, B, C)$ we calculate

$$\mathbf{f}(B,C) = \sum_a \mathbf{f}_3(A,B,C) = \mathbf{f}_3(a,B,C) + \mathbf{f}_3(\neg a,B,C)$$

$$= \begin{pmatrix} .06 & .24 \\ .42 & .28 \end{pmatrix} + \begin{pmatrix} .18 & .72 \\ .06 & .04 \end{pmatrix} = \begin{pmatrix} .24 & .96 \\ .48 & .32 \end{pmatrix}$$

A	B	$\mathbf{f}_1(A, B)$	B	C	$\mathbf{f}_2(B, C)$	A	B	C	$\mathbf{f}_3(A, B, C)$
T	T	.3	T	T	.2	T	T	T	.06
T	F	.7	T	F	.8	T	T	F	.24
F	T	.9	F	T	.6	T	F	T	.42
F	F	.1	F	F	.4	T	F	F	.28
						F	T	T	.18
						F	T	F	.72
						F	F	T	.06
						F	F	F	.04

- Any factor that does not depend on the variable to be summed out can be moved outside the summation

$$\sum_e \mathbf{f}_2(E) \times \mathbf{f}_3(A,B,E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$$

$$= \mathbf{f}_4(A) \times \mathbf{f}_5(A) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_3(A,B,E)$$

Variable Ordering and Variable Relevance

- The variable elimination algorithm (next page) includes an ORDER function
  - Different orderings $\rightarrow$ different intermediate factors $\rightarrow$ different time and space complexities
  - E.g., if we eliminate E before A the calculation becomes

$$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \sum_a \mathbf{f}_4(A) \times \mathbf{f}_5(A) \times \sum_e \mathbf{f}_2(E) \times \mathbf{f}_3(A, B, E)$$

$$\mathbf{P}(B \mid j, m) = \alpha\, \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(E) \times \sum_a \mathbf{f}_3(A, B, E) \times \mathbf{f}_4(A) \times \mathbf{f}_5(A)$$

$$\begin{aligned} P(b \mid j, m) &= \alpha \textstyle\sum_a \sum_e P(b)\, P(e)\, P(a \mid b, e)\, P(j \mid a)\, P(m \mid a) \\ &= \alpha P(b) \textstyle\sum_a \sum_e P(e)\, P(a \mid b, e)\, P(j \mid a)\, P(m \mid a) \\ &= \alpha P(b) \textstyle\sum_a P(j \mid a)\, P(m \mid a) \sum_e P(e)\, P(a \mid b, e) \end{aligned}$$

  - Intractable to determine the optimal ordering

Variable Ordering and Variable Relevance

function ELIMINATION-ASK(X, **e**, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $factors \leftarrow [\,]$
  foreach var **in** ORDER(bn.VARS) **do**
    $factors \leftarrow$ [MAKE-FACTOR(var, **e**) | $factors$]
    if var is a hidden variable **then** $factors \leftarrow$ SUM-OUT(var, $factors$)
  return NORMALIZE(POINTWISE-PRODUCT($factors$))

* Notice that factors are not multiplied until a variable is summed out from the accumulated product

The variable elimination algorithm for inference in Bayesian networks

$$\mathbf{P}(B \mid j, m) = \alpha \underbrace{\mathbf{P}(B)}_{\mathbf{f}_1(B)} \sum_e \underbrace{P(e)}_{\mathbf{f}_2(E)} \sum_a \underbrace{\mathbf{P}(a \mid B, e)}_{\mathbf{f}_3(A,B,E)} \underbrace{P(j \mid a)}_{\mathbf{f}_4(A)} \underbrace{P(m \mid a)}_{\mathbf{f}_5(A)}$$

$\longleftarrow$ ORDER (best?)

Variable Ordering and Variable Relevance

- Irrelevant variables:
  - Consider the query $\mathbf{P}(JohnCalls \mid Burglary = true)$

    $$\mathbf{P}(J \mid b) = \alpha P(b) \sum_e P(e) \sum_a P(a \mid b, e)\, \mathbf{P}(J \mid a) \sum_m P(m \mid a)$$

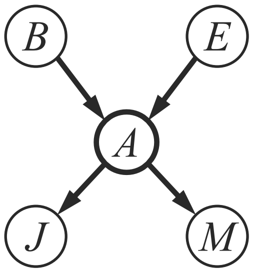
    Sum over m is identically 1; M is irrelevant to the query

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

Here, $X = JohnCalls$, $\mathbf{E} = \{Burglary\}$, and

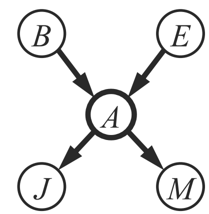
$Ancestors(\{X\} \cup \mathbf{E}) = \{Alarm, Earthquake\}$

so $MaryCalls$ is irrelevant



Complexity of Exact Inference

- **Singly connected** networks (or **polytrees**):
  - Any two nodes are connected by at most one (undirected) path
  - Time and space complexities of variable elimination are linear in the number of CPT entries

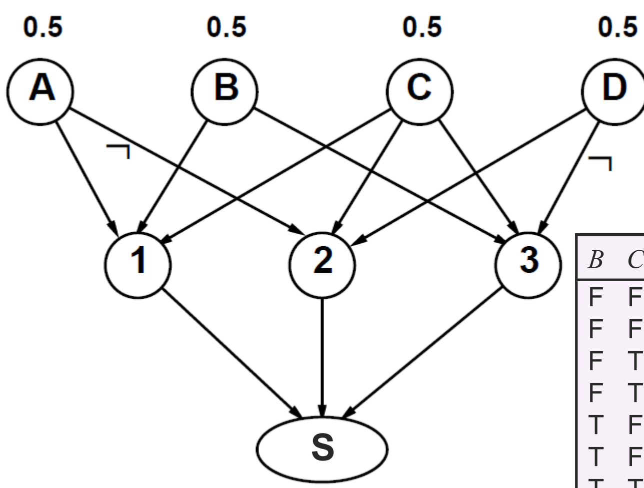


- **Multiply connected** networks:
  - CNF-SAT is reducible to exact inference

$$S = (A \vee B \vee C) \wedge (\neg A \vee C \vee D) \wedge (B \vee C \vee \neg D)$$

SAT is satisfiable iff $P(S) > 0$

$\rightarrow$ NP-hard

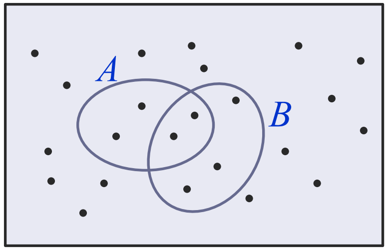


B	C	D	$P(3 \mid B, C, D)$
F	F	F	1
F	F	T	0
F	T	F	1
F	T	T	1
T	F	F	1
T	F	T	1
T	T	F	1
T	T	T	1

Approximate 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

$P(A|B)$?

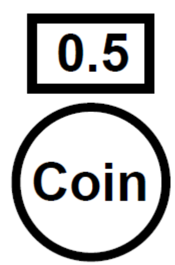


- Outline:
  - Direct sampling methods:
    - Sampling from an empty network
    - Rejection sampling: reject samples disagreeing with evidence
    - Likelihood weighting: use evidence to weight samples
  - Markov Chain Monte Carlo (MCMC): sample from a stochastic process whose stationary distribution is the true posterior

* Markov chain: a stochastic process in which future states are independent of past states given the present state

Direct Sampling Methods

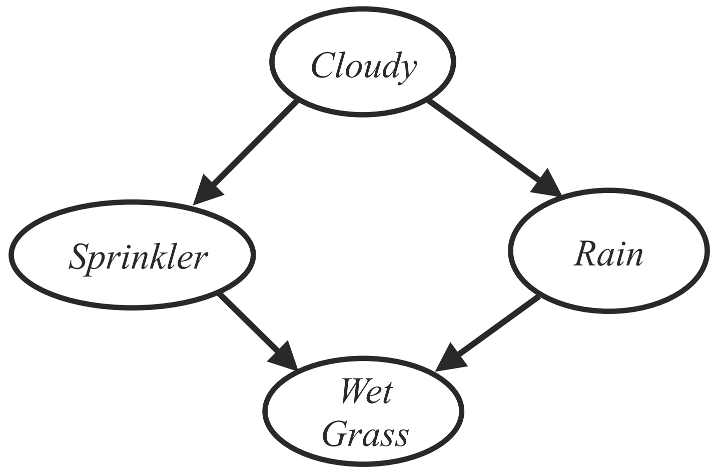
```
function PRIOR-SAMPLE(bn) returns an event sampled from the prior specified by bn
   inputs: bn, a Bayesian network specifying joint distribution P(X1, . . . ,Xn)

   x ← an (empty) event with n elements
   foreach variable Xi in X1, . . . , Xn do     /* sample variables in topological order */
      x[i] ← a random sample from P(Xi | parents(Xi))
                 given the values of Parents(Xi) in x
   return x
```



Sampling from an Empty Network

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

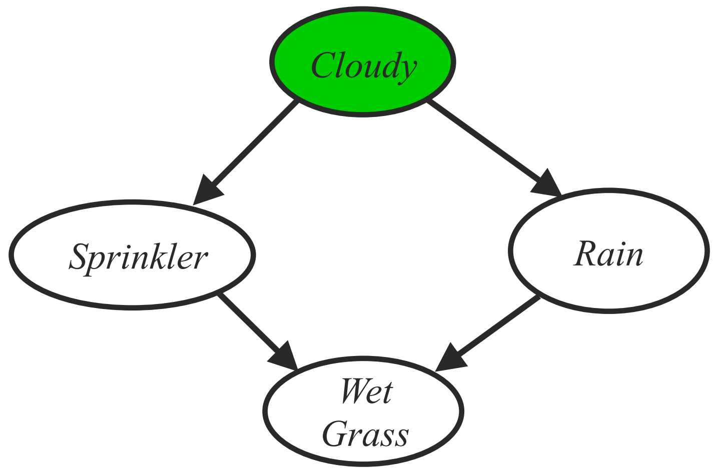
C	$P(R \mid C)$
T	.80
F	.20

S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sampling from an Empty Network

Example:

P(C)
.50



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

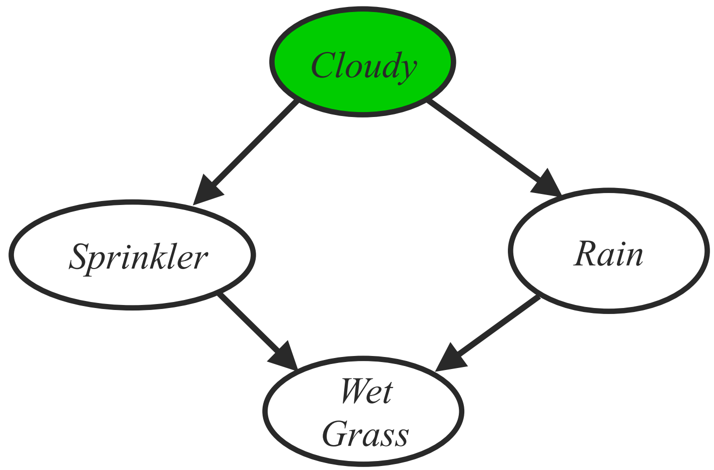
C	P(R | C)
T	.80
F	.20

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

Sampling from an Empty Network

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

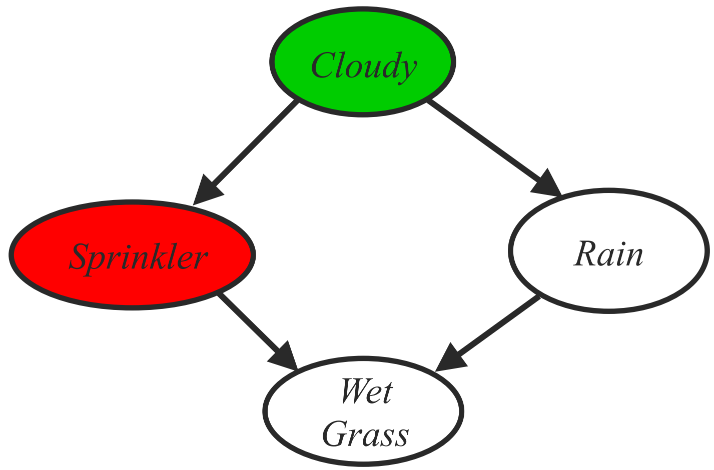
C	$P(R \mid C)$
T	.80
F	.20

S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sampling from an Empty Network

Example:

P(C)
.50



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

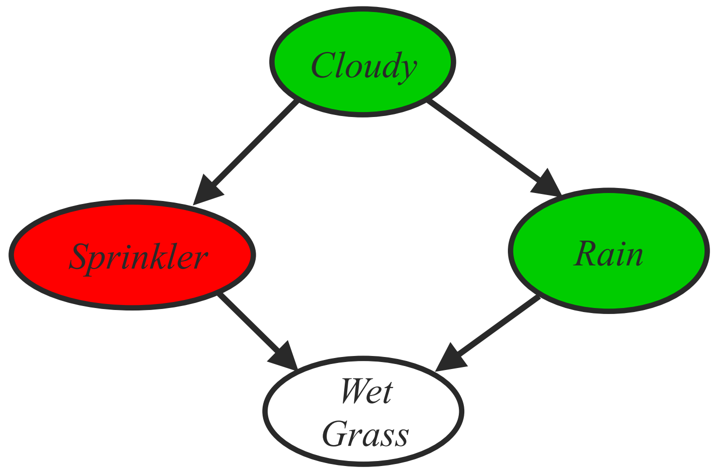
C	P(R | C)
T	.80
F	.20

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

Sampling from an Empty Network

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

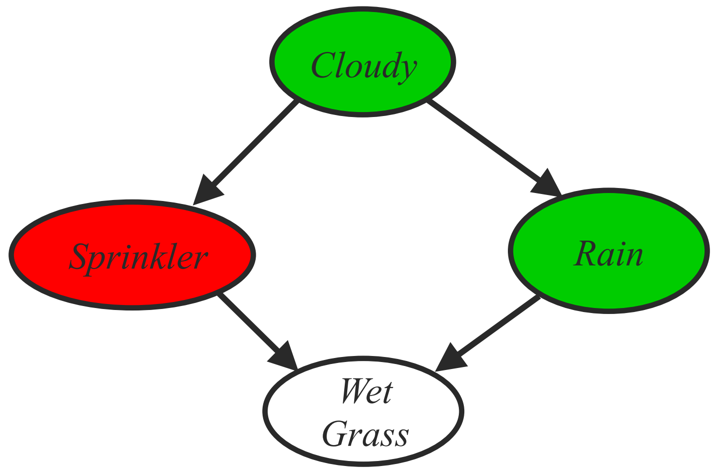
C	$P(R \mid C)$
T	.80
F	.20

S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sampling from an Empty Network

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

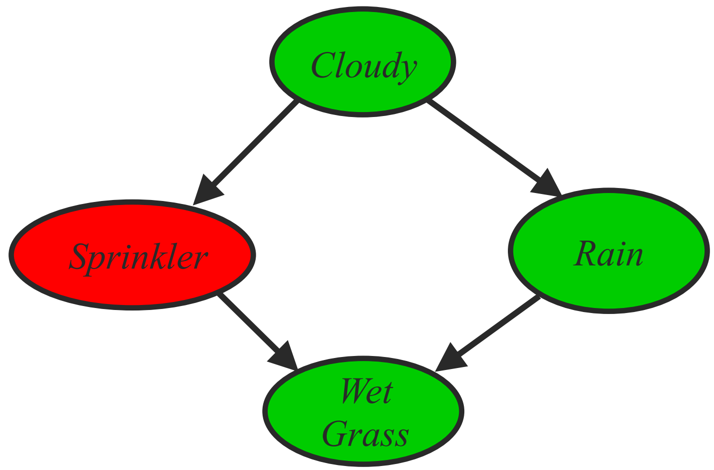
C	$P(R \mid C)$
T	.80
F	.20

S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sampling from an Empty Network

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

C	$P(R \mid C)$
T	.80
F	.20

S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

PRIOR-SAMPLE returns the event [*true*, *false*, *true*, *true*]

- Probability that PRIOR-SAMPLE generates a particular event

$$S_{PS}(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid parents(X_i)) = P(x_1,\ldots,x_n)$$

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 $x_1, \ldots, x_n$ out of a total of N samples. Then we have

$$\lim_{N\to\infty} \hat{P}(x_1,\ldots,x_n) = \lim_{N\to\infty} N_{PS}(x_1,\ldots,x_n)/N$$

Since this is the probability that PRIOR-SAMPLE generates the event $x_1, \ldots, x_n$,

$$= S_{PS}(x_1,\ldots,x_n) = P(x_1,\ldots,x_n)$$

i.e., estimates derived from PRIOR-SAMPLE are consistent

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

Rejection Sampling

- $\hat{\mathbf{P}}(X \mid \mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

```
function REJECTION-SAMPLING(X, e, bn, N) returns an estimate of P(X | e)
   local variables: N, a vector of counts for each value of X, initially zero

   for j = 1 to N do
      x ← PRIOR-SAMPLE(bn)
      if x is consistent with e then
         N[x] ← N[x] + 1 where x is the value of X in x
   return NORMALIZE(N[X])
```

- E.g., estimate $\mathbf{P}(Rain \mid Sprinkler = true)$ using 100 samples

   27 samples have $Sprinkler = true$

   Of these, 8 have $Rain = true$ and 19 have $Rain = false$

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

   - Similar to a basic real-world empirical estimation procedure

Rejection Sampling

- Analysis of rejection sampling:

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e}) \qquad \text{(algorithm definition)}$$

$$= \mathbf{N}_{PS}(X, \mathbf{e}) / N_{PS}(\mathbf{e}) \qquad \text{(normalized by } N_{PS}(\mathbf{e})\text{)}$$

$$= (\mathbf{N}_{PS}(X, \mathbf{e}) / N) / (N_{PS}(\mathbf{e}) / N)$$

$$= \mathbf{P}(X, \mathbf{e}) / P(\mathbf{e}) \qquad \text{(property of } \textsc{Prior-Sample}\text{)}$$

$$= \mathbf{P}(X \mid \mathbf{e}) \qquad \text{(definition of conditional probability)}$$

- Hence rejection sampling returns consistent posterior estimates
- Problem: hopelessly expensive if $P(\mathbf{e})$ is small
  - $P(\mathbf{e})$ 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 $\mathbf{P}(X \mid \mathbf{e})$
  local variables: **W**, a vector of weighted counts for each value of X, initially zero

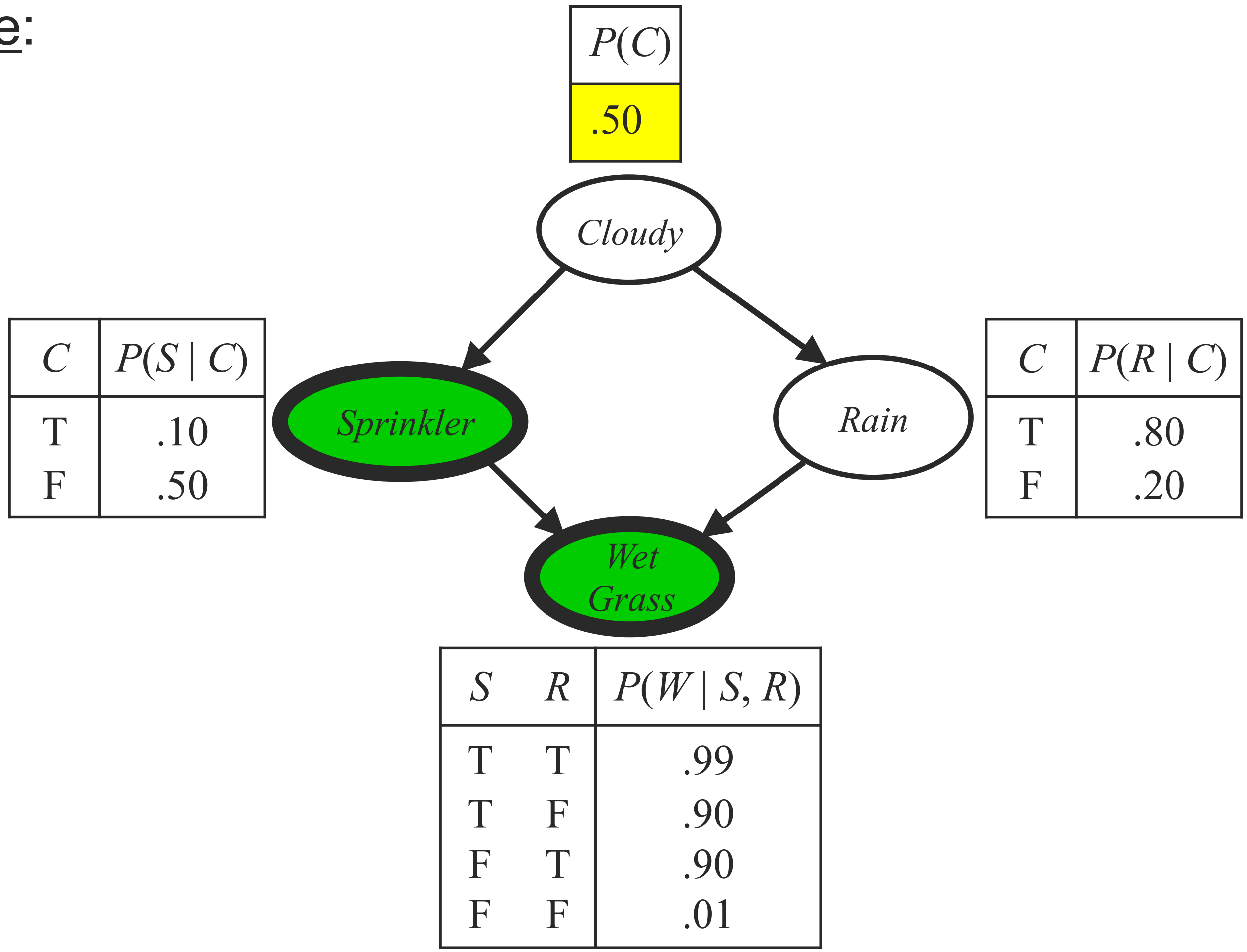
  for j = 1 to N **do**
    x, $w \leftarrow$ WEIGHTED-SAMPLE(bn, **e**)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in **x**
  return NORMALIZE($\mathbf{W}[X]$)

---

function WEIGHTED-SAMPLE(bn, **e**) **returns** an event and a weight

  $w \leftarrow 1$; $\mathbf{x} \leftarrow$ an event with n elements initialized from **e**
  foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    if X_i is an evidence variable with value x_i in **e**
      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      else $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return **x**, w

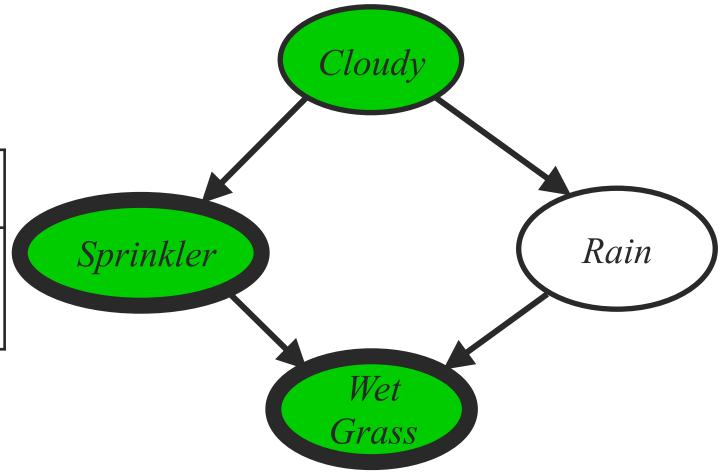
Example:


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

$w = 1.0$

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

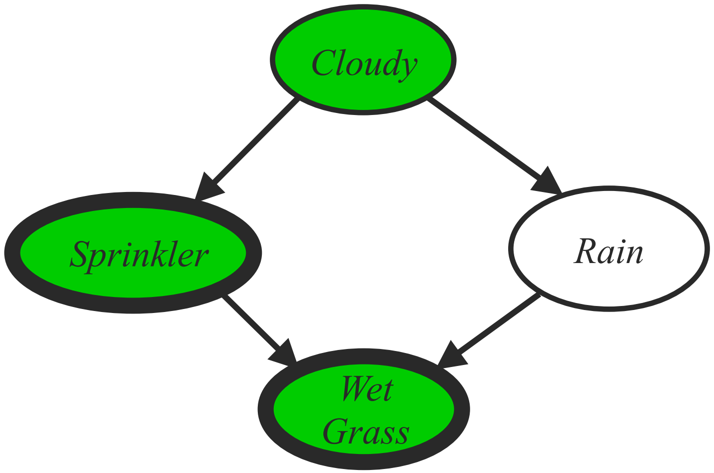
C	$P(R \mid C)$
T	.80
F	.20

S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

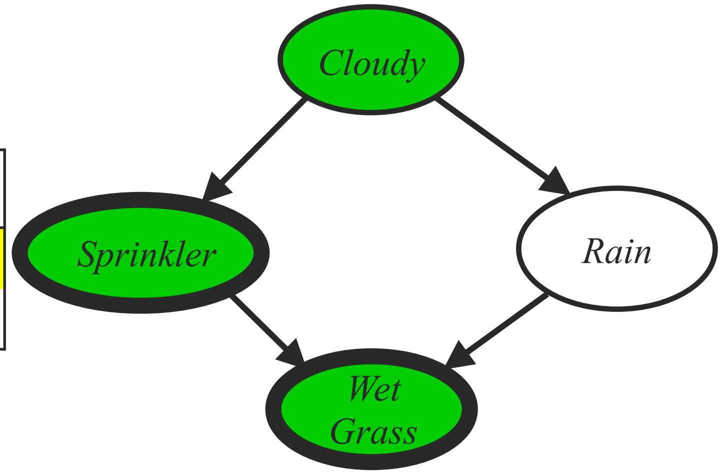
C	$P(R \mid C)$
T	.80
F	.20

S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0$

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

C	$P(R \mid C)$
T	.80
F	.20

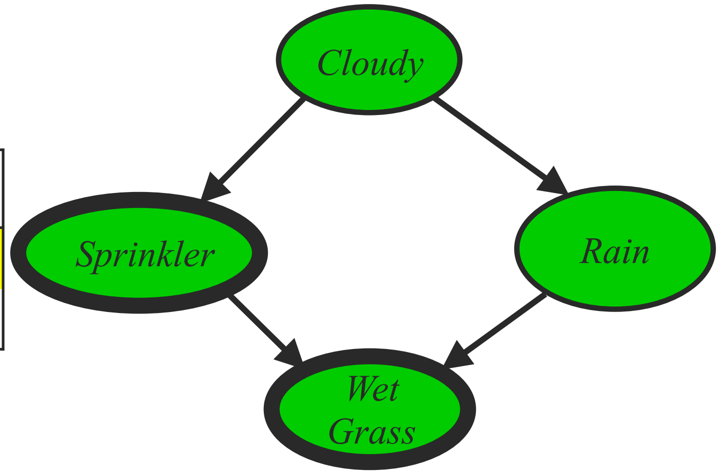
S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0 \times 0.1$

Likelihood Weighting

Example:

P(C)
.50



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

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

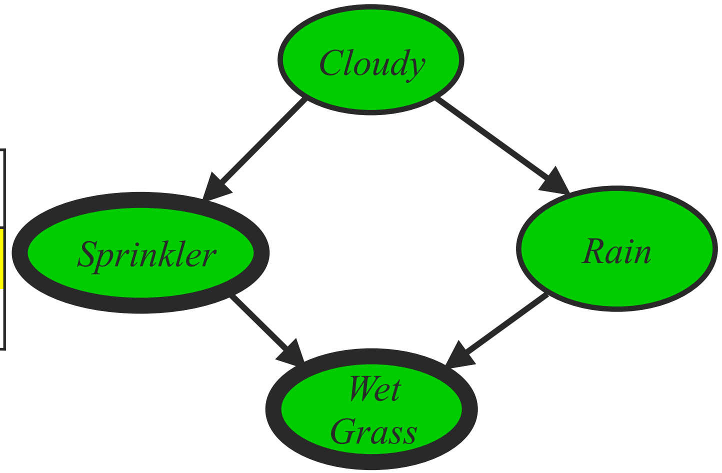
S	R	P(W | S, R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0 \times 0.1$

Likelihood Weighting

Example:

$P(C)$
.50



C	$P(S \mid C)$
T	.10
F	.50

C	$P(R \mid C)$
T	.80
F	.20

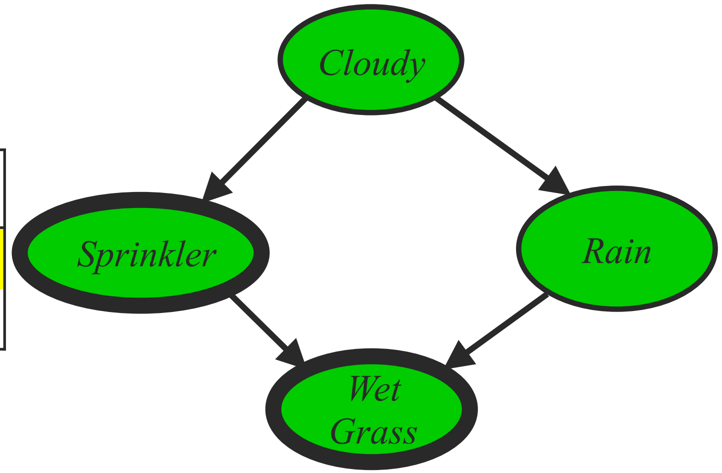
S	R	$P(W \mid S, R)$
T	T	.99
T	F	.90
F	T	.90
F	F	.01

$w = 1.0 \times 0.1$

Likelihood Weighting

Example:

P(C)
.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

$w = 1.0 \times 0.1 \times 0.99 = 0.099$

$\mathbf{x} = (true, true, true, true)$

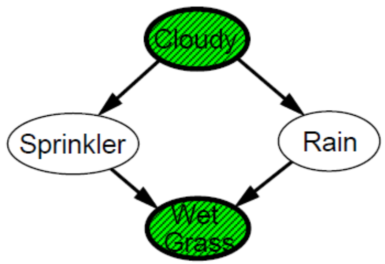
Likelihood Weighting

- Sampling probability for WEIGHTED-SAMPLE is

$$S_{WS}(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{l} P(z_i \mid parents(Z_i)) \quad (l\text{: \# of non-evidence variables})$$

Note: pays attention to evidence in ancestors only

$\Rightarrow$ somewhere "in between" prior and posterior distribution



- Weight for a given sample **z**, **e** is

$$w(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{m} P(e_i \mid parents(E_i)) \quad (m\text{: \# of evidence variables})$$

- Weighted sampling probability is

$$S_{WS}(\mathbf{z},\mathbf{e})w(\mathbf{z},\mathbf{e}) = \prod_{i=1}^{l} P(z_i \mid parents(Z_i)) \prod_{i=1}^{m} P(e_i \mid parents(E_i))$$
$$= P(\mathbf{z},\mathbf{e})$$

by standard global semantics of network

Likelihood Weighting

- For a query variable X,

$$\hat{\mathbf{P}}(X \mid \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{N}_{WS}(X, \mathbf{y}, \mathbf{e}) \mathbf{w}(X, \mathbf{y}, \mathbf{e}) \quad \text{(from LIKELIHOOD-WEIGHTING)}$$

$$= \alpha' \sum_{\mathbf{y}} \frac{\mathbf{N}_{WS}(X, \mathbf{y}, \mathbf{e})}{N} \mathbf{w}(X, \mathbf{y}, \mathbf{e})$$

$$\approx \alpha' \sum_{\mathbf{y}} \mathbf{S}_{WS}(X, \mathbf{y}, \mathbf{e}) \mathbf{w}(X, \mathbf{y}, \mathbf{e}) \quad \text{(for large } N\text{)}$$

$$= \alpha' \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{y}, \mathbf{e})$$

$$= \alpha' \mathbf{P}(X, \mathbf{e}) = \mathbf{P}(X \mid \mathbf{e})$$

- Hence likelihood weighting returns consistent estimates
- Performance still degrades with many evidence variables because a few samples have nearly all the total weight (especially when the evidence variables appear late in the ordering)

Inference by Markov Chain Simulation

- MCMC (Markov Chain Monte Carlo) generates each sample by making a random change to the preceding sample
  - Can be thought as being in a particular current state specifying a value for every variable and generating a next state by making random changes to the current state (state = sample)
- Gibbs sampling is a form of MCMC especially well suited for Bayesian networks
  - State of network = current assignment to all variables
  - Generates next state by sampling one variable given all other variables (or equivalently given Markov blanket)
  - Samples each variable in turn, or at random, with evidence variables fixed

Inference by Markov Chain Simulation

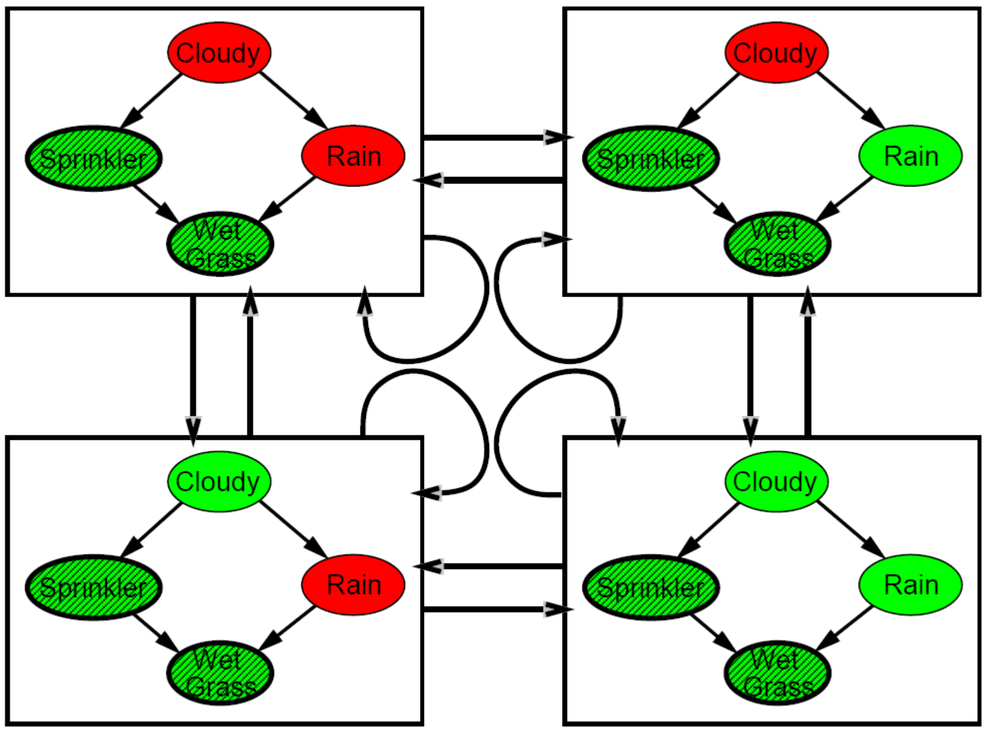
function GIBBS-SAMPLING(X, $\mathbf{e}$, bn, N) **returns** an estimate of $\mathbf{P}(X \mid \mathbf{e})$
    local variables: $\mathbf{N}$, a vector of counts for each value of X, initially zero
                            $\mathbf{Z}$, the nonevidence variable in bn
                            $\mathbf{x}$, the current state of the network, initially copied from $\mathbf{e}$

    initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Z}$
    for j = 1 to N **do**
        foreach Z_i **in Z do**
            set the value of Z_i in $\mathbf{x}$ by sampling from $\mathbf{P}(Z_i \mid mb(Z_i))$
            $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{x}$
    return NORMALIZE($\mathbf{N}[X]$)

Inference by Markov Chain Simulation

Example:

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



Wander about for a while, average what you see

Inference by Markov Chain Simulation

Example:

Estimate $\mathbf{P}(Rain \mid Sprinkler = true, WetGrass = true)$

- 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 $Rain = false$

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

Theorem: Chain approaches stationary distribution

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

Inference by Markov Chain Simulation

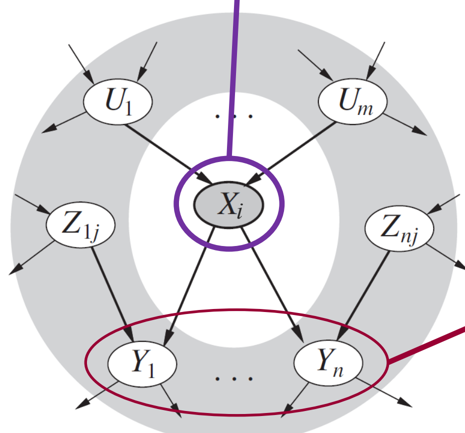
- Markov blanket sampling

$$\mathbf{P}(X_i \mid x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$
$$= \alpha\, \mathbf{P}(x_1, x_2, \ldots, x_{i-1}, X_i, x_{i+1}, \ldots, x_n)$$
$$= \alpha \left(\prod_{j=1}^{s} P(x_{c_j} \mid parents(X_{c_j})) \right) \mathbf{P}(X_i \mid parents(X_i)) \left(\prod_{k=1}^{t} \mathbf{P}(x_{d_k} \mid \boldsymbol{\pi}_{d_k}, X_i) \right)$$

the values of $Parents(X_{d_k}) - X_i$

$$X_i \notin Parents(X_{c_j})$$
$$X_{c_j} \neq X_i$$

$$X_i \in Parents(X_{d_k})$$



Therefore,

$$P(x_i \mid mb(X_i)) = \alpha\, P(x_i \mid parents(X_i)) \times \prod_{Y_j \in Children(X_i)} P(y_j \mid parents(Y_j))$$

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
  - Difficult to tell if convergence has been achieved with MCMC
  - 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