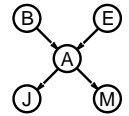


Inference by enumeration

Slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Simple query on the burglary network:

$$\begin{aligned} & \mathbf{P}(B|j, m) \\ &= \mathbf{P}(B, j, m) / P(j, m) \\ &= \alpha \mathbf{P}(B, j, m) \\ &= \alpha \sum_e \sum_a \mathbf{P}(B, e, a, j, m) \end{aligned}$$



Rewrite full joint entries using product of CPT entries:

$$\begin{aligned} & \mathbf{P}(B|j, m) \\ &= \alpha \sum_e \sum_a \mathbf{P}(B)P(e)\mathbf{P}(a|B, e)P(j|a)P(m|a) \\ &= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B, e)P(j|a)P(m|a) \end{aligned}$$

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

Outline

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

Enumeration algorithm

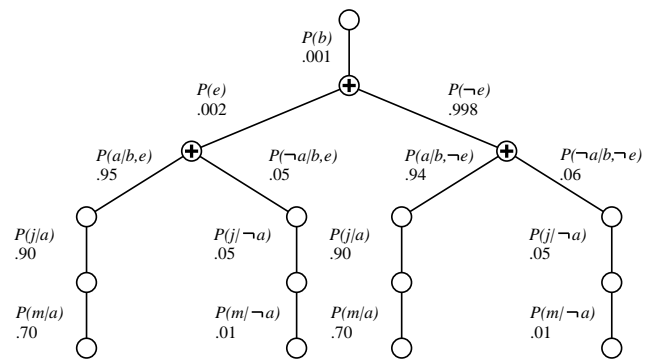
```
function ENUMERATION-ASK( $X, e, bn$ ) returns a distribution over  $X$ 
inputs:  $X$ , the query variable
        $e$ , observed values for variables  $E$ 
        $bn$ , a Bayesian network with variables  $\{X\} \cup E \cup Y$ 
 $Q(X) \leftarrow$  a distribution over  $X$ , initially empty
for each value  $x_i$  of  $X$  do
  extend  $e$  with value  $x_i$  for  $X$ 
   $Q(x_i) \leftarrow$  ENUMERATE-ALL(VARS[ $bn$ ],  $e$ )
return NORMALIZE( $Q(X)$ )

function ENUMERATE-ALL( $vars, e$ ) returns a real number
if EMPTY?( $vars$ ) then return 1.0
 $Y \leftarrow$  FIRST( $vars$ )
if  $Y$  has value  $y$  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  $e_y$  is  $e$  extended with  $Y = y$ 
```

Inference tasks

- Simple queries: compute posterior marginal $\mathbf{P}(X_i|\mathbf{E} = e)$
e.g., $P(\text{NoGas}|\text{Gauge} = \text{empty}, \text{Lights} = \text{on}, \text{Starts} = \text{false})$
- Conjunctive queries: $\mathbf{P}(X_i, X_j|\mathbf{E} = e) = \mathbf{P}(X_i|\mathbf{E} = e)\mathbf{P}(X_j|X_i, \mathbf{E} = e)$
- 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?

Evaluation 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

$$\begin{aligned}
 P(B|j, m) &= \alpha \underbrace{P(B)}_B \underbrace{\sum_e P(e)}_E \underbrace{\sum_a P(a|B, e)}_A \underbrace{P(j|a)}_J \underbrace{P(m|a)}_M \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) P(j|a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a P(a|B, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) \sum_a f_A(a, b, e) f_J(a) f_M(a) \\
 &= \alpha P(B) \sum_e P(e) f_{\bar{A}JM}(b, e) \quad (\text{sum out } A) \\
 &= \alpha P(B) f_{\bar{E}\bar{A}JM}(b) \quad (\text{sum out } E) \\
 &= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
 \end{aligned}$$

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

$$\sum_x f_1 \times \dots \times f_k = f_1 \times \dots \times f_i \sum_x f_{i+1} \times \dots \times f_k = f_1 \times \dots \times f_i \times f_{\bar{x}}$$

assuming f_1, \dots, f_i do not depend on X

Pointwise product of factors f_1 and f_2 :

$$\begin{aligned}
 f_1(x_1, \dots, x_j, y_1, \dots, y_k) \times f_2(y_1, \dots, y_k, z_1, \dots, z_l) \\
 = f(x_1, \dots, x_j, y_1, \dots, y_k, z_1, \dots, z_l)
 \end{aligned}$$

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

Variable elimination algorithm

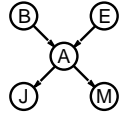
function ELIMINATION-ASK(X, e, bn) **returns** a distribution over X
inputs: X , the query variable
 e , evidence specified as an event
 bn , a belief network specifying joint distribution $P(X_1, \dots, X_n)$
 $factors \leftarrow []$; $vars \leftarrow REVERSE(VARS[bn])$
for each var **in** $vars$ **do**
 $factors \leftarrow [MAKE-FACTOR(var, e)|factors]$
 if var is a hidden variable **then** $factors \leftarrow SUM-OUT(var, factors)$
return NORMALIZE(POINTWISE-PRODUCT($factors$))

Irrelevant variables

Consider the query $P(JohnCalls|Burglary = true)$

$$P(J|b) = \alpha P(b) \sum_e P(e) \sum_a P(a|b, e) P(J|a) \sum_m P(m|a)$$

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



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

Here, $X = JohnCalls$, $E = \{Burglary\}$, and
 $Ancestors(\{X\} \cup E) = \{Alarm, Earthquake\}$
 so $MaryCalls$ is irrelevant

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

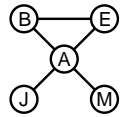
Irrelevant variables contd.

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 $P(JohnCalls|Alarm = true)$, both $Burglary$ and $Earthquake$ are irrelevant



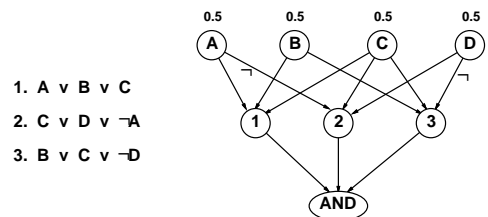
Complexity of exact inference

Singly connected networks (or polytrees):

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

Multiply connected networks:

- can reduce 3SAT to exact inference \Rightarrow NP-hard
- equivalent to **counting** 3SAT models \Rightarrow #P-complete



1. $A \vee B \vee C$
2. $C \vee D \vee \neg A$
3. $B \vee C \vee \neg D$

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

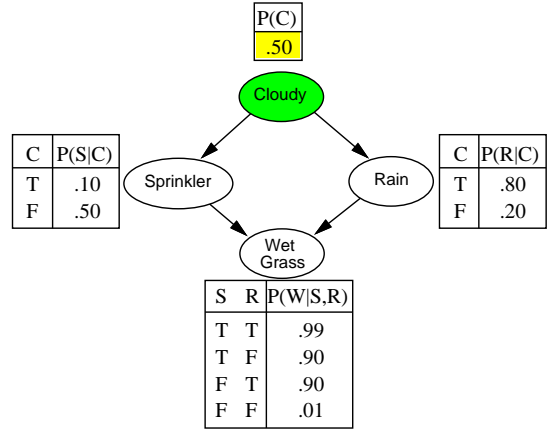
0.5



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

Example

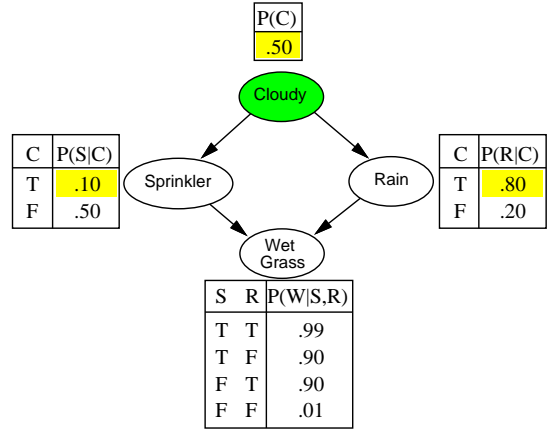


Sampling from an empty network

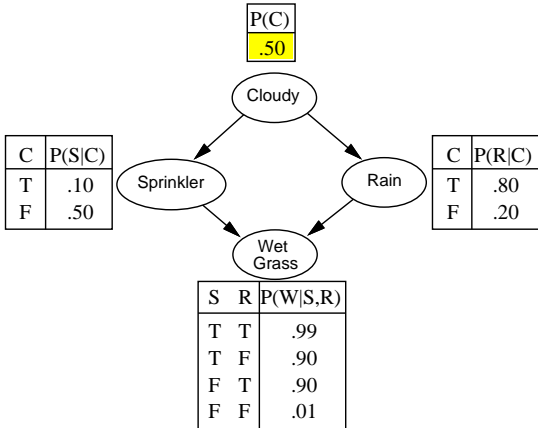
```

function PRIOR-SAMPLE(bn) returns an event sampled from bn
inputs: bn, a belief network specifying joint distribution  $P(X_1, \dots, X_n)$ 
x ← an event with n elements
for i = 1 to n do
   $x_i$  ← a random sample from  $P(X_i \mid \text{parents}(X_i))$ 
  given the values of Parents( $X_i$ ) in x
return x
    
```

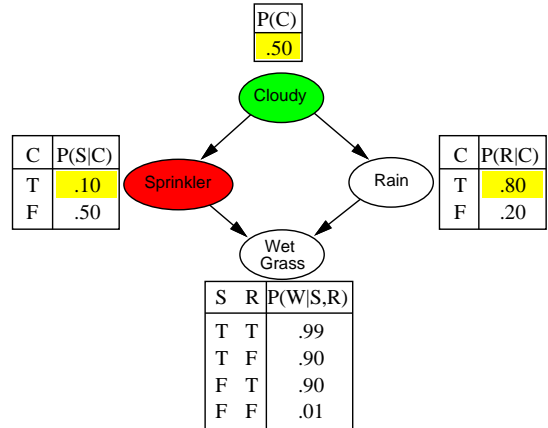
Example



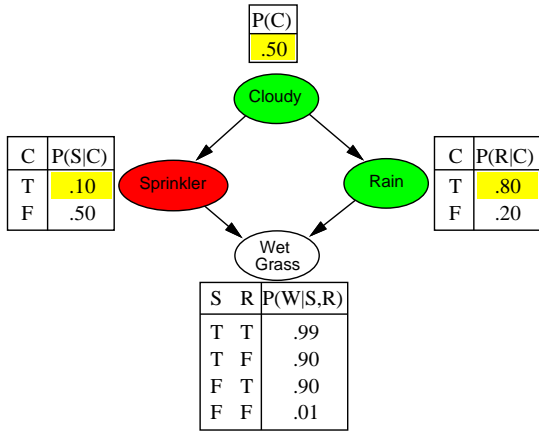
Example



Example



Example



Sampling from an empty network contd.

Probability that PRIORSAMPLE generates a particular event
 $S_{PS}(x_1 \dots x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i)) = P(x_1 \dots x_n)$
 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 $N_{PS}(x_1 \dots x_n)$ be the number of samples generated for event x_1, \dots, x_n

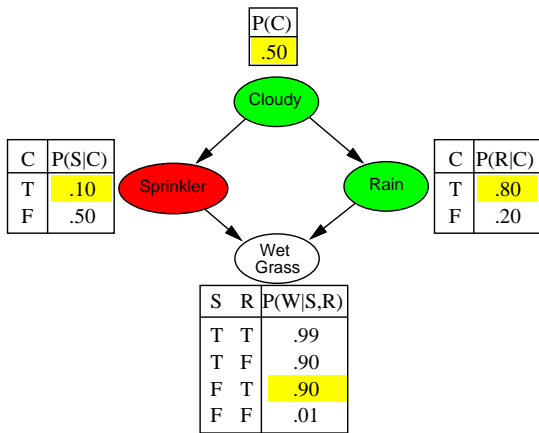
Then we have

$$\begin{aligned} \lim_{N \rightarrow \infty} \hat{P}(x_1, \dots, x_n) &= \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n) / N \\ &= S_{PS}(x_1, \dots, x_n) \\ &= P(x_1 \dots x_n) \end{aligned}$$

That is, estimates derived from PRIORSAMPLE are consistent

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

Example



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 ← 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 $P(\text{Rain} | \text{Sprinkler} = \text{true})$ using 100 samples

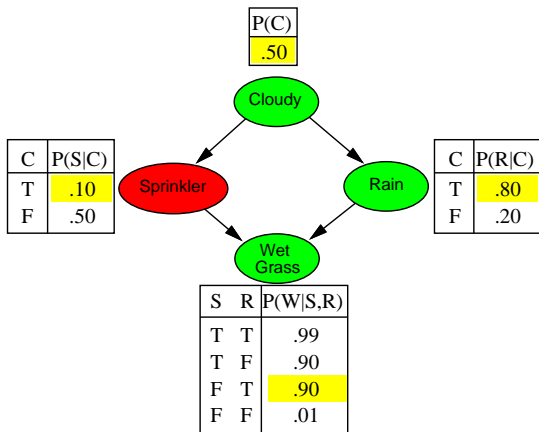
27 samples have $\text{Sprinkler} = \text{true}$

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

Example



Analysis of rejection sampling

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

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

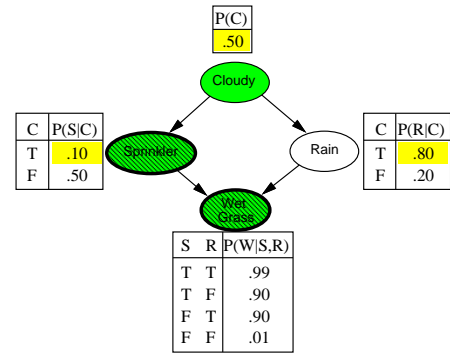
Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

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**
 $x, w \leftarrow$ WEIGHTED-SAMPLE(bn)
 $W[x] \leftarrow W[x] + w$ where x is the value of X in x
return NORMALIZE($W[X]$)

function WEIGHTED-SAMPLE(bn, e) **returns** an event and a weight
 $x \leftarrow$ an event with n elements; $w \leftarrow 1$
for $i = 1$ to n **do**
 if X_i has a value x_i in e
 then $w \leftarrow w \times P(X_i = x_i \mid \text{parents}(X_i))$
 else $x_i \leftarrow$ a random sample from $P(X_i \mid \text{parents}(X_i))$
return x, w

Chapter 14.4-5 25

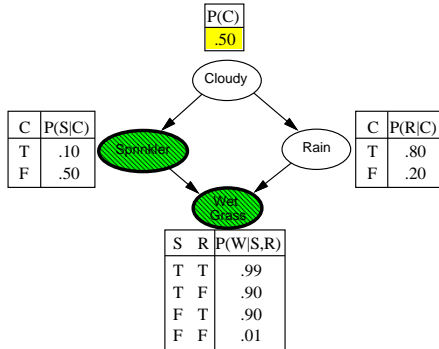
Likelihood weighting example



$w = 1.0$

Chapter 14.4-5 28

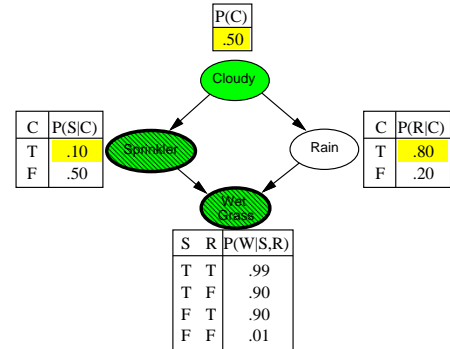
Likelihood weighting example



$w = 1.0$

Chapter 14.4-5 26

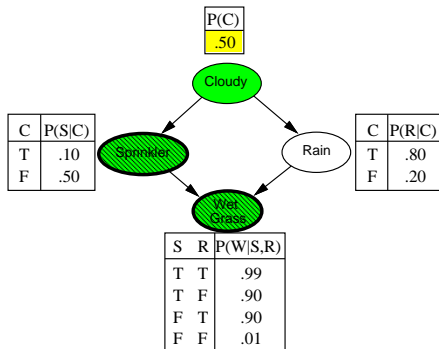
Likelihood weighting example



$w = 1.0 \times 0.1$

Chapter 14.4-5 29

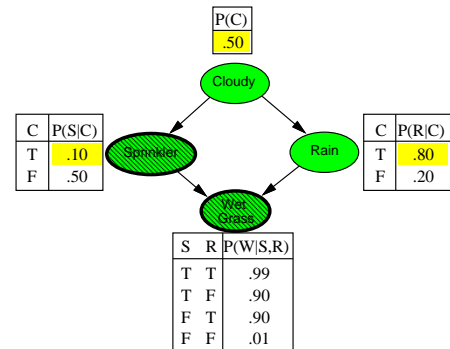
Likelihood weighting example



$w = 1.0$

Chapter 14.4-5 27

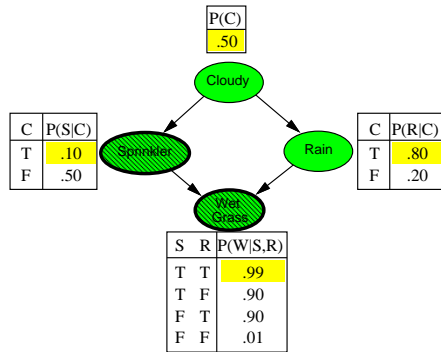
Likelihood weighting example



$w = 1.0 \times 0.1$

Chapter 14.4-5 30

Likelihood weighting example



$$w = 1.0 \times 0.1$$

Approximate inference using MCMC

"State" of network = current assignment to all variables.

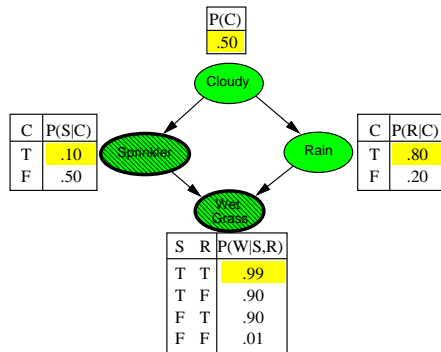
Generate next state by sampling one variable given Markov blanket
Sample each variable in turn, keeping evidence fixed

```

function MCMC-Ask(X, e, 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 Zi in Z do
      sample the value of Zi in x from P(Zi|mb(Zi))
      given the values of MB(Zi) in x
      N[x] ← N[x] + 1 where x is the value of X in x
  return NORMALIZE(N[X])
    
```

Can also choose a variable to sample at random each time

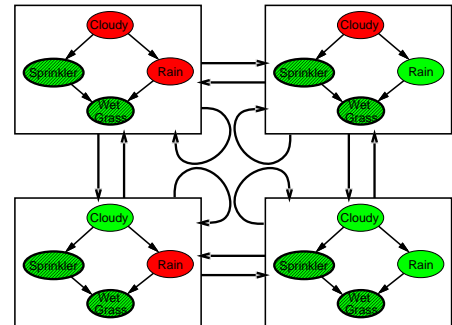
Likelihood weighting example



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

The Markov chain

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



Wander about for a while, average what you see

Likelihood weighting analysis

Sampling probability for WEIGHTEDSAMPLE is

$$S_{WS}(z, e) = \prod_{i=1}^m P(z_i | \text{parents}(Z_i))$$

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

⇒ somewhere "in between" prior and posterior distribution

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

$$w(z, e) = \prod_{i=1}^m P(e_i | \text{parents}(E_i))$$

Weighted sampling probability is

$$\begin{aligned}
 S_{WS}(z, e)w(z, e) &= \prod_{i=1}^m P(z_i | \text{parents}(Z_i)) \prod_{i=1}^m P(e_i | \text{parents}(E_i)) \\
 &= P(z, e) \text{ (by standard global semantics of network)}
 \end{aligned}$$

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



MCMC example contd.

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

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*

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

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:

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

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