# Inference in belief networks

## Chapter 14b

# Outline

◇ Exact inference by enumeration

◇ Approximate inference by stochastic simulation

◇ Likelihood weighting

◇ Approximate inference using MCMC (SKIP)

# Inference tasks

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

Conjunctive queries: $\mathbf{P}(X_i, X_j|\mathbf{E}=\mathbf{e}) = \mathbf{P}(X_i|\mathbf{E}=\mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E}=\mathbf{e})$

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

Value of information: which evidence to seek next?

Explanation: why do I need a new starter motor?

Sensitivity analysis: which probability values are most critical?

# Inference by enumeration

Simple query on the burglary network:

$\mathbf{P}(B|J = true, M = true)$

$= \mathbf{P}(B, J = true, M = true)/P(J = true, M = true)$

$= \alpha\mathbf{P}(B, J = true, M = true)$

$= \alpha \Sigma_e \Sigma_a \mathbf{P}(B, e, a, J = true, M = true)$

# Inference by enumeration

Simple query on the burglary network:

$\mathbf{P}(B|J = true, M = true)$

$= \mathbf{P}(B, J = true, M = true)/P(J = true, M = true)$

$= \alpha\mathbf{P}(B, J = true, M = true)$

$= \alpha\Sigma_e\Sigma_a\mathbf{P}(B, e, a, J = true, M = true)$

Let's use the non-vector notation for $B = true$:

$= \alpha\Sigma_e\Sigma_a P(B = true, e, a, J = true, M = true)$

Rewrite full joint entries using product of CPT entries:

$= \alpha\Sigma_e\Sigma_a P(B = true)P(e)P(a|B = true, e)P(J = true|a)P(M = true|a)$

$= \alpha P(B = true)\Sigma_e P(e)\Sigma_a P(a|B = true, e)P(J = true|a)P(M = true|a)$

for $E = \{e, \neg e\}$

for $A = \{a, \neg a\}$

# Inference by enumeration

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

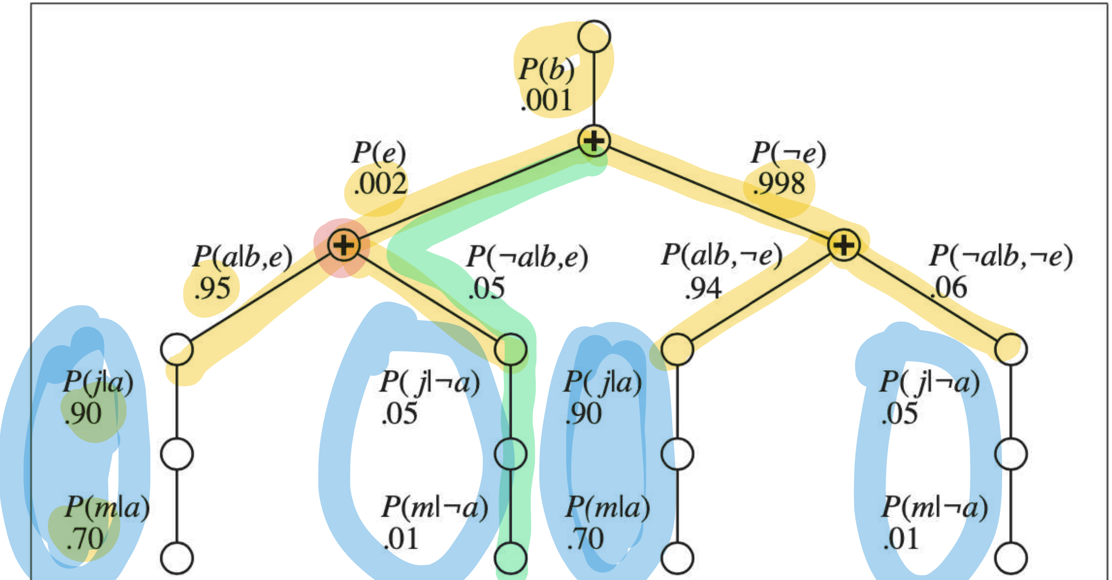



**Figure 14.8** The structure of the expression shown in Equation (14.4). The evaluation proceeds top down, multiplying values along each path and summing at the "+" nodes. Notice the repetition of the paths for $j$ and $m$.

# 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} ∪ E ∪ Y

   Q(X) ← 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) ← 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 ← FIRST(vars)
   if Y has value y in e
        then return P(y | Pa(Y)) × ENUMERATE-ALL(REST(vars), e)      // y is an evid. var. of query
        else return Σ_y P(y | Pa(Y)) × ENUMERATE-ALL(REST(vars), e_y)  // y is a hidden var e.g. e
             where e_y is e extended with Y = y
```

# Inference by variable elimination

**Enumeration is inefficient**: repeated computation
e.g., computes $P(J = true|a)P(M = true|a)$ for each value of $e$

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

We will skip the mechanism for speeding this and concentrate on the use approximate inference (stochastic methods).

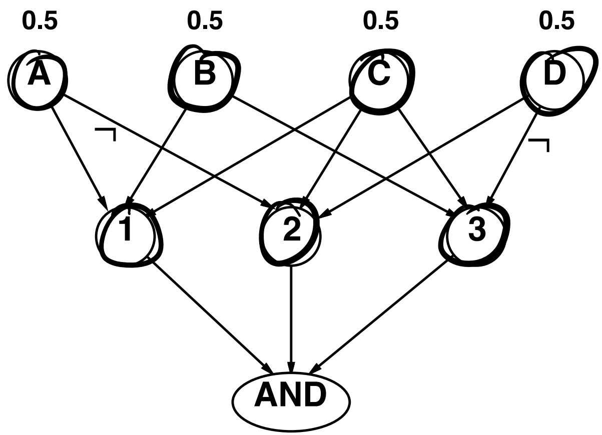
# Complexity of Exact Inference

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 $\Rightarrow$ NP-hard

$\sigma =$

1. (A v B v C) $\wedge$
2. (C v D v ¬A) $\wedge$
3. (B v C v ¬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 $P$

Methods:

– Sampling when there is no evidence variables
– Rejection sampling: reject samples disagreeing with evidence
– Likelihood weighting: use evidence to weight samples
– MCMC: sample from a stochastic process whose stationary (Skipped) distribution is the true posterior

# Sampling with No Evidence Variables

**function** PRIORSAMPLE($bn$) **returns** an event sampled from $\mathbf{P}(X_1, \ldots, X_n)$ specified by $bn$

  $\mathbf{x} \leftarrow$ an event with $n$ elements

  **for** $i = 1$ **to** $n$ **do**

    $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid Parents(X_i))$

  **return** $\mathbf{x}$

P(Sprinkler = true | Cloudy = true) = 0.10

true false

$\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$

sample $\rightarrow$ *true*

true false

$\mathbf{P}(Sprinkler|Cloudy) = \langle 0.1, 0.9 \rangle$ = true

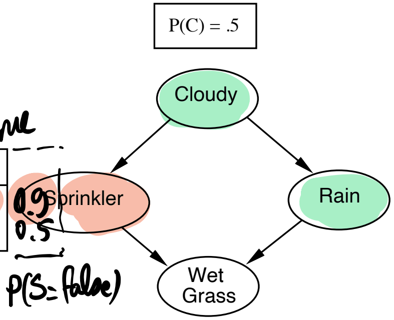
sample $\rightarrow$ *false*

$\mathbf{P}(Rain|Cloudy) = \langle 0.8, 0.2 \rangle$ = true

sample $\rightarrow$ *true*

$\mathbf{P}(WetGrass|\neg Sprinkler, Rain) = \langle 0.9, 0.1 \rangle$

sample $\rightarrow$ *true*




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

true

0.9

0.5

P(S = false)

P(Sprinkler = true | Cloudy = false) = 0.5

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

| S | R | P(W) |
|---|---|---|
| T | T | .99 |
| T | F | .90 |
| F | T | .90 |
| F | F | .00 |

| simulated sample : | Cloudy | Sprinkler | Rain | Wetgrass | |
|---|---|---|---|---|---|
| | [ true | false | true | true ] | × 1000 |

Following the topology, I start from Cloudy to sampling.

# Sampling from an empty network

**The probability of a particular event can be estimated as the fraction of all events generated by the sampling process that match the particular event.**

E.g. Let's say we generate 1000 samples from the sprinkler network and
511 of them have Rain=True, then
the estimate of P(Rain=True) = 0.511.

# Sampling from an empty network contd.

**Proof given for completeness - slide not covered, just know the result -last line.**

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

Let $N_{PS}(\mathbf{Y} = \mathbf{y})$ be the number of samples generated for which $\mathbf{Y} = \mathbf{y}$, for any set of variables $\mathbf{Y}$.

Then $\hat{P}(\mathbf{Y} = \mathbf{y}) = N_{PS}(\mathbf{Y} = \mathbf{y})/N$ and

$$\begin{aligned}\lim_{N\to\infty} \hat{P}(\mathbf{Y} = \mathbf{y}) &= \Sigma_{\mathbf{h}} S_{PS}(\mathbf{Y} = \mathbf{y}, \mathbf{H} = \mathbf{h}) \\ &= \Sigma_{\mathbf{h}} P(\mathbf{Y} = \mathbf{y}, \mathbf{H} = \mathbf{h}) \\ &= P(\mathbf{Y} = \mathbf{y})\end{aligned}$$

That is, estimates derived from PriorSample are consistent

# Rejection sampling with evidence vars

Prior sampling can give us the prior probabilities of the random variables in the BBN, but **more often we will be interested in estimating the probability given some evidence variables.**

→ Sample as before, but estimate $\hat{\mathbf{P}}(X|\mathbf{e})$ from samples agreeing with $\mathbf{e}$

e.g. $P(Rain \mid WetGrass = true)$

1000 samples

| Rain | WetGrass |
|---|---|
| true | true |
| | ~~false~~ |
| | ~~false~~ |
| false | true |
| | ~~false~~ |

In 1/2 samples w/
WGrass = true
Rain = true

# Rejection sampling with evidence vars

**function** REJECTIONSAMPLING($X$,$\mathbf{e}$,$bn$,$N$) **returns** an approximation to $P(X|\mathbf{e})$
  $\mathbf{N}[X] \leftarrow$ a vector of counts over $X$, initially zero
  **for** $j = 1$ to $N$ **do**
    $\mathbf{x} \leftarrow$ PRIORSAMPLE($bn$)   // x is full sample
    **if** $\mathbf{x}$ is consistent with $\mathbf{e}$ **then**
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where $x$ is the value of $X$ in $\mathbf{x}$
  **return** NORMALIZE($\mathbf{N}[X]$)

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

$$\hat{\mathbf{P}}(Rain|Sprinkler = true) = \text{NORMALIZE}(\langle 8, 19 \rangle) = \langle \underset{\frac{8}{27}}{\underset{\|}{0.296}} + \underset{\frac{19}{27}}{\underset{\|}{0.704}} = 1 \rangle$$

# Analysis of rejection sampling

**Proof given for completeness - slide not covered, just know the result - last 2 lines.**

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

Hence: Rejection sampling returns consistent posterior estimates.

**Problem: hopelessly expensive if $P(\mathrm{e})$ is small.**

# Likelihood weighting

Likelihood weighting avoids the inefficiency of rejection sampling by generating only those events that are consistent with the evidence.

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

**function** WeightedSample($bn$,$\mathbf{e}$) **returns** an event and a weight
  $\mathbf{x} \leftarrow$ an event with $n$ elements; $w \leftarrow 1$
  **for** $i = 1$ **to** $n$ **do**
    **if** $X_i$ has a value $x_i$ in $\mathbf{e}$ // if $X_i$ is an evidence var
      **then** $w \leftarrow w \times P(X_i = x_i \mid Parents(X_i))$ // update $w$
      **else** $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid Parents(X_i))$
  **return** $\mathbf{x}$, $w$

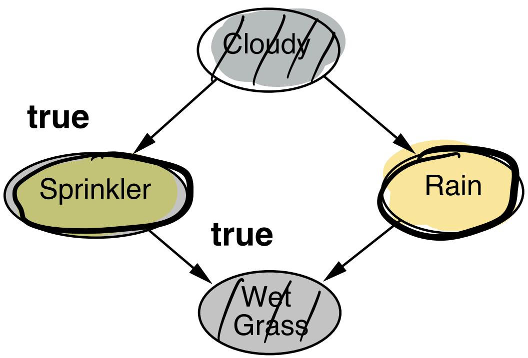
**function** LikelihoodWeighting($X$,$\mathbf{e}$,$bn$,$N$) **returns** an approximation to $P(X|\mathbf{e})$
  $\mathbf{W}[X] \leftarrow$ a vector of weighted counts over $X$, initially zero
  **for** $j = 1$ to $N$ **do**
    $\mathbf{x}, w \leftarrow$ WeightedSample($bn$)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where $x$ is the value of $X$ in $\mathbf{x}$
  **return** Normalize($\mathbf{W}[X]$)

# Likelihood weighting example

Evidences

Estimate $\mathbf{P}(Rain|Cloudy = true, WetGrass = true)$

P(C) = .5




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

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

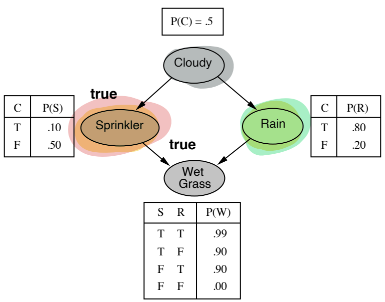
| S | R | P(W) |
|---|---|---|
| T | T | .99 |
| T | F | .90 |
| F | T | .90 |
| F | F | .00 |

# Likelihood weighting example contd.

Estimate $\mathbf{P}(Rain|Cloudy = true, WetGrass = true)$

Cloudy
⟨0.9 0.1⟩
true

1. $w \leftarrow 1.0$
2. Cloudy is an evidence variable with value $true$, so set
   $w \leftarrow w \times P(Cloudy = true)$ ; $w$ becomes 0.5
3. $Sprinkler$ is not an evidence variable, so
   sample from $\mathbf{P}(Sprinkler|Cloudy = true)$; suppose this returns $false$
4. Sample $\mathbf{P}(Rain|Cloudy = true) = \langle 0.8, 0.2\rangle$; suppose this returns $true$
5. $WetGrass$ is an evidence variable with value $true$, so
   $w \leftarrow w \times P(WetGrass = true|Sprinkler = false, Rain = true)$;
   $w$ becomes 0.45




$sample_1$: C=t, Spr=false, R=true, WG=t
w = 0.45

$Sample_2$

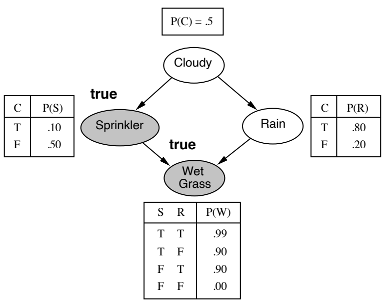
# Likelihood weighting analysis

WEIGHTED-SAMPLE returns the event [true,false,true,true] with weight 0.45; and this is counted under $Rain = true$

# Likelihood weighting example contd.

**Different example in agrrement with the figure:** Estimate $\mathbf{P}(Rain|Sprinkler=$

1. $w \leftarrow 1.0$
2. Cloudy is not an evidence variable, Sample $\mathbf{P}(Cloudy) = \langle 0.5, 0.5\rangle$; say $true$
3. $Sprinkler$ has value $true$, so
   $w \leftarrow w \times P(Sprinkler=true|Cloudy=true) = 0.1$
4. Sample $\mathbf{P}(Rain|Cloudy=true) = \langle 0.8, 0.2\rangle$; say $true$
5. $WetGrass$ has value $true$, so
   $w \leftarrow w \times P(WetGrass=true|Sprinkler=true, Rain=true) = 0.099$



# Likelihood weighting analysis

Likelihood weighting returns consistent estimates, but performance degrades with many evidence variables

Note that when sampling Sprinkler and Rain, the analysis ignores the evidence in the child variable WetGrass with value True.

It may thus generate many samples with Sprinkler=False and Rain=False despite the evidence (WetGrass=True). But those samples will have zero weight.

# Approximate inference using MCMC

The state-of-art in stochastic algorithms is **Markov Chain Monte Carlo (MCMC) approximation**, which s widely used in AI, Bayesian inference, and statistical modeling.

But it will be covered very lightly.

# 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

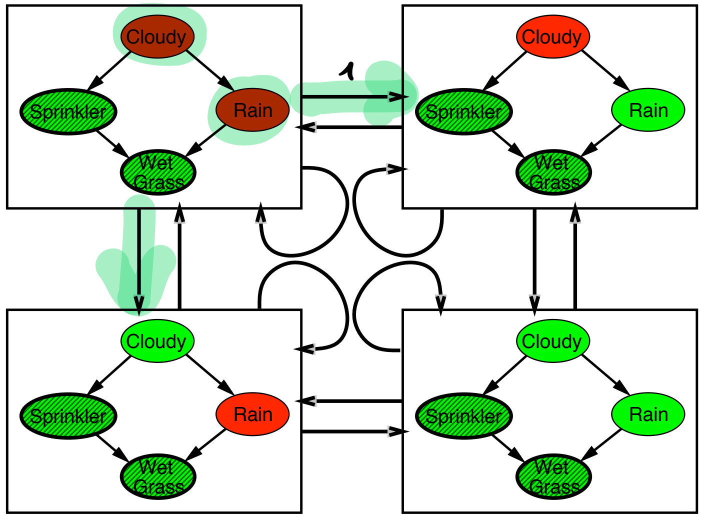
```
function MCMC-Ask(X,e,bn,N) returns an approximation to P(X|e)
   local variables: N[X], a vector of counts over X, initially zero
                    Y, the nonevidence variables in bn
                    x, the current state of the network, initially copied from e

   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 Y_i in Y do
             sample the value of Y_i in x from P(Y_i|MB(Y_i)) given the values of
MB(Y_i) in x
   return Normalize(N[X])
```

Fraction of time spent in each state is proportional to its posterior probability

## The Markov chain

initial state $\{c=f, r=f, s=t, wg=t\}$

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




next state: $\{ \quad \}$

Wander about for a while, average what you see



## After obtaining the MCMC samples

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

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 $Rain = false$

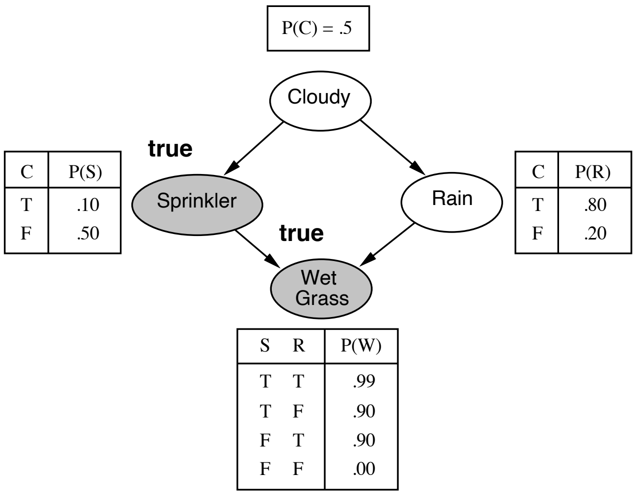
$\hat{\mathbf{P}}(Rain|Sprinkler = true, WetGrass = true)$
$= \text{Normalize}(\langle 31, 69 \rangle) = \langle 0.31, 0.69 \rangle$

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

But:
1. Difficult to tell when samples have converged. Theorem only applies in limit, and it could take time to "settle in".
2. Can also be inefficient if each state depends on many other variables.

# MCMC Example

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




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

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

| S | R | P(W) |
|---|---|---|
| T | T | .99 |
| T | F | .90 |
| F | T | .90 |
| F | F | .00 |

Reminder: Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents

# MCMC example contd.

Random initial state: $Cloudy = true$ and $Rain = false$

1. $\mathbf{P}(Cloudy|MB(Cloudy)) = \mathbf{P}(Cloudy|Sprinkler, \neg Rain)$
   sample $\rightarrow$ $false$

2. $\mathbf{P}(Rain|MB(Rain)) = \mathbf{P}(Rain|\neg Cloudy, Sprinkler, WetGrass)$
   sample $\rightarrow$ $true$

Visit 100 states
    31 have $Rain = true$, 69 have $Rain = false$

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