Lecture 23: 28 April, 2022

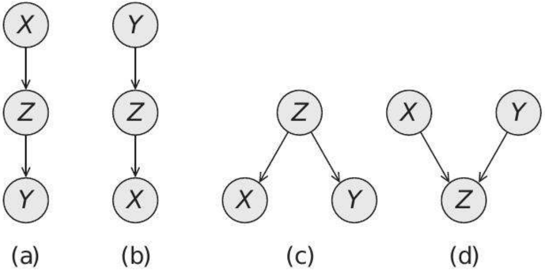
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
January–May 2022

D-Separation

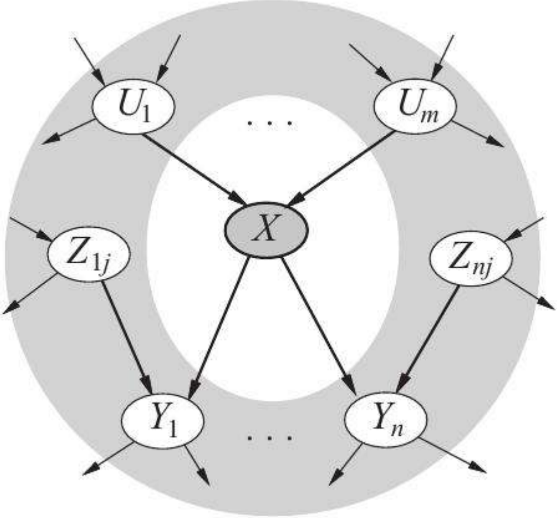
- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from X to Y
  - Each undirected path from X to Y is a sequence of basic trails
  - For (a), (b), (c), need Z present
  - For (d), need Z absent
  - In general, V-structure includes descendants of the bottom node



- x and y are D-separated given z if all trails are blocked
- Variation of breadth first search (BFS) to check if y is reachable from x through some trail
- Extends to sets — each $x \in X$ is D-separated from each $y \in Y$

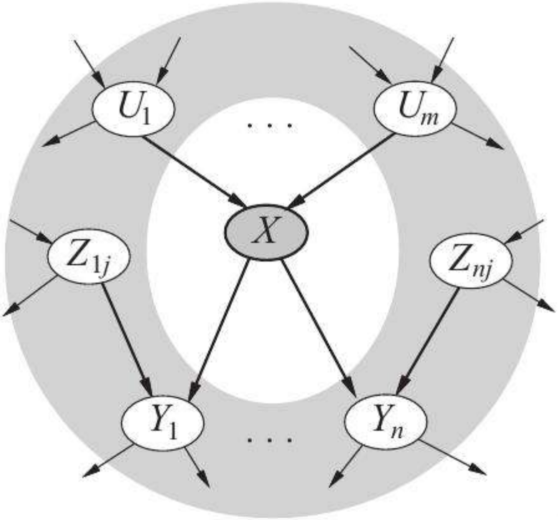
Markov blanket

- $MB(X)$ — Markov blanket of X



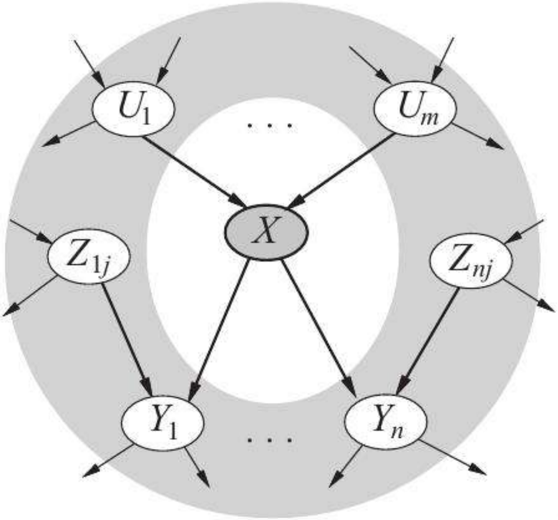
Markov blanket

- $MB(X)$ — Markov blanket of X
  - $Parents(X)$



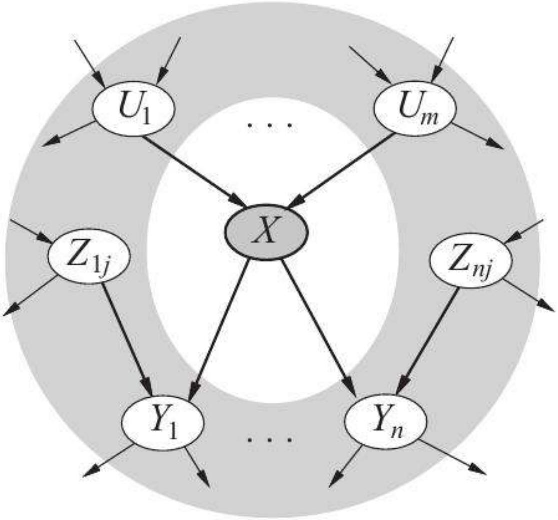
Markov blanket

- $MB(X)$ — Markov blanket of X
  - $Parents(X)$
  - $Children(X)$



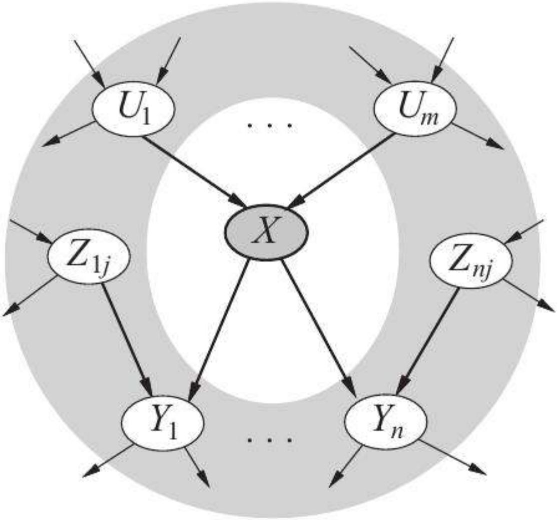
Markov blanket

- $MB(X)$ — Markov blanket of X
  - $Parents(X)$
  - $Children(X)$
  - $Parents\ of\ Children(X)$



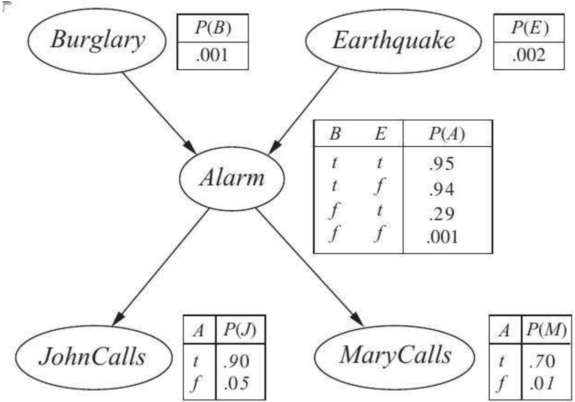
Markov blanket

- $MB(X)$ — Markov blanket of X
  - $Parents(X)$
  - $Children(X)$
  - $Parents\ of\ Children(X)$
- $X \perp \neg MB(X) \mid MB(X)$



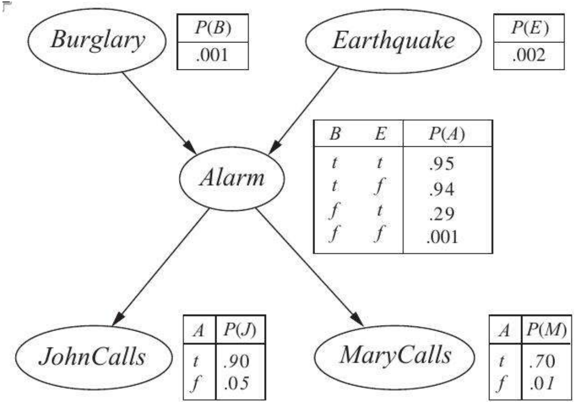
Computing with probabilistic graphical models

- John and Mary call Pearl. What is the probability that there has been a burglary?



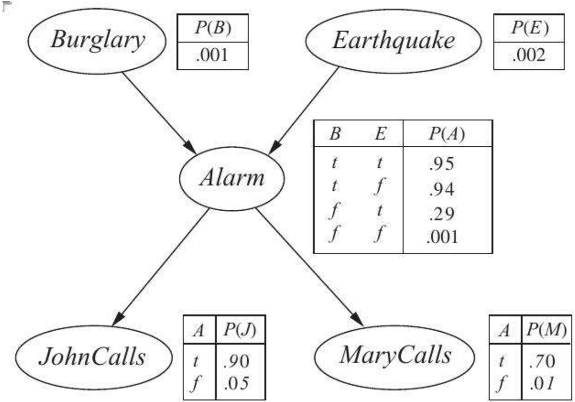
Computing with probabilistic graphical models

- John and Mary call Pearl. What is the probability that there has been a burglary?
- Want $P(b \mid m, j)$



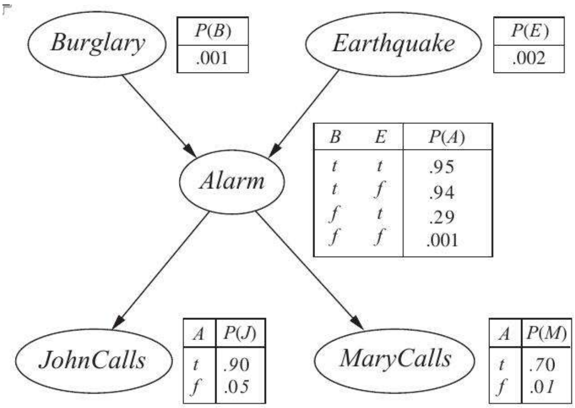
Computing with probabilistic graphical models

- John and Mary call Pearl. What is the probability that there has been a burglary?
- Want $P(b \mid m, j)$
- $\dfrac{P(b,m,j)}{P(m,j)}$



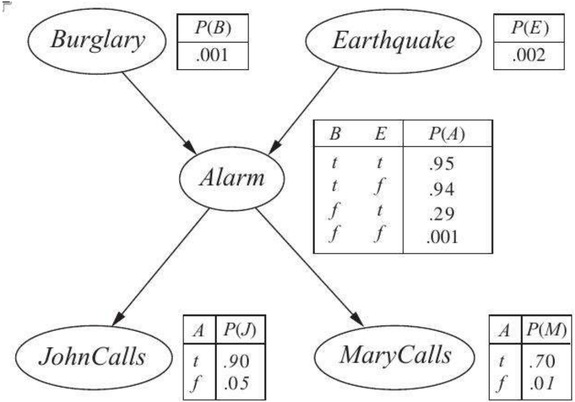
Computing with probabilistic graphical models

- John and Mary call Pearl. What is the probability that there has been a burglary?
- Want $P(b \mid m, j)$
- $\dfrac{P(b, m, j)}{P(m, j)}$
- Use chain rule to evaluate joint probabilities



Computing with probabilistic graphical models

- John and Mary call Pearl. What is the probability that there has been a burglary?
- Want $P(b \mid m, j)$
- $\dfrac{P(b, m, j)}{P(m, j)}$
- Use chain rule to evaluate joint probabilities
- Reorder variables appropriately, topological order of graph

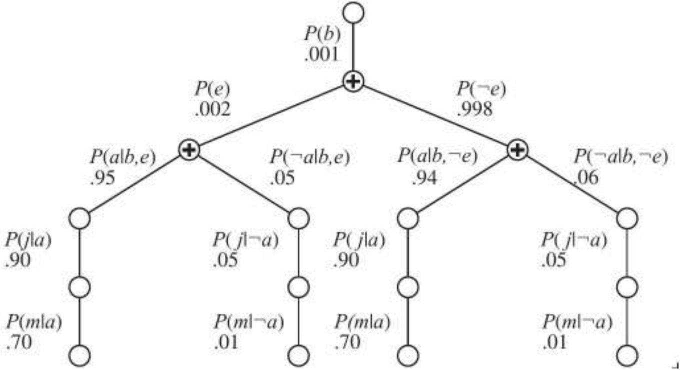


Computing with probabilistic graphical models

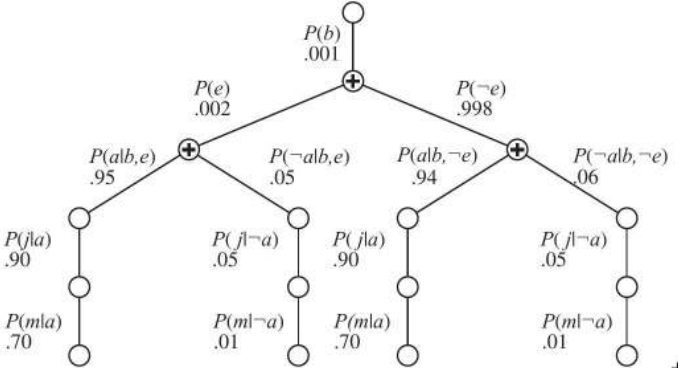
- $P(m, j, b) = P(b) \sum_{e=0}^{1} P(e) \sum_{a=0}^{1} P(a \mid b, e) P(m \mid a) P(j \mid a)$

Computing with probabilistic graphical models

- $P(m, j, b) = P(b) \sum_{e=0}^{1} P(e) \sum_{a=0}^{1} P(a \mid b, e) P(m \mid a) P(j \mid a)$
- Construct the computation tree

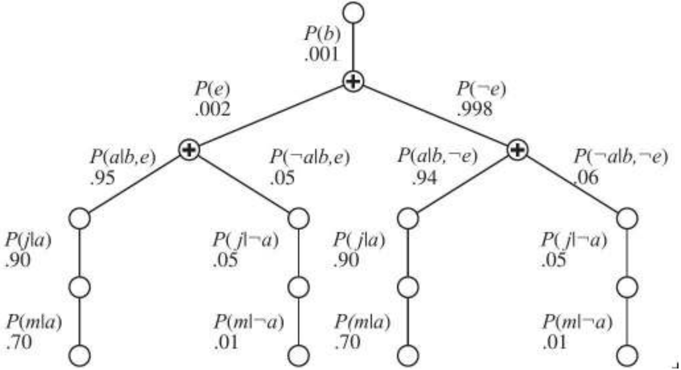


- $P(m, j, b) = P(b) \sum_{e=0}^{1} P(e) \sum_{a=0}^{1} P(a \mid b, e) P(m \mid a) P(j \mid a)$
- Construct the computation tree
- Use dynamic programming to avoid duplicated computations



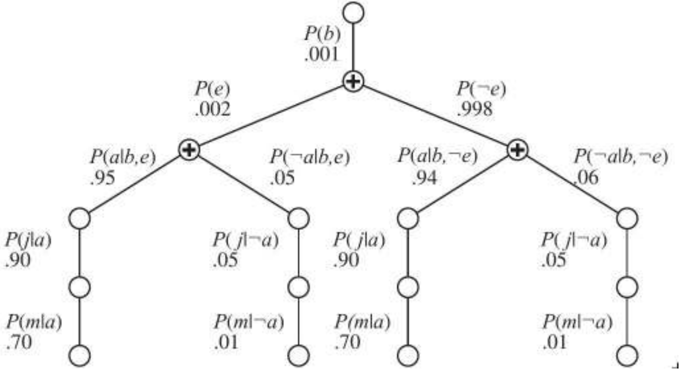
Computing with probabilistic graphical models

- $P(m, j, b) = P(b) \sum_{e=0}^{1} P(e) \sum_{a=0}^{1} P(a \mid b, e) P(m \mid a) P(j \mid a)$
- Construct the computation tree
- Use dynamic programming to avoid duplicated computations
- However, exact inference is NP-complete, in general



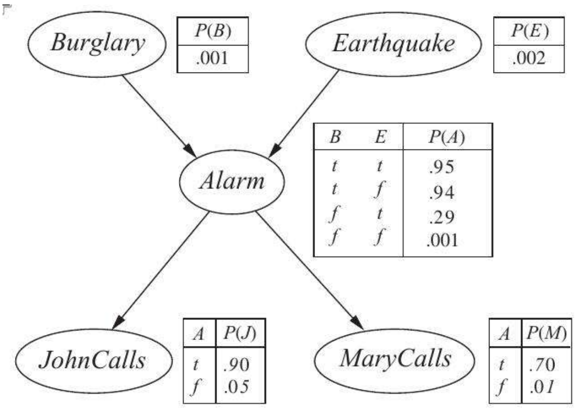
Computing with probabilistic graphical models

- $P(m, j, b) = P(b) \sum_{e=0}^{1} P(e) \sum_{a=0}^{1} P(a \mid b, e) P(m \mid a) P(j \mid a)$
- Construct the computation tree
- Use dynamic programming to avoid duplicated computations
- However, exact inference is NP-complete, in general
- Instead, approximate inference through sampling



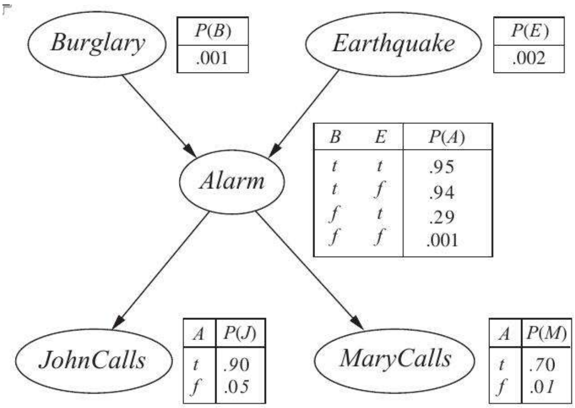
Approximate inference

- Generate random samples (b, e, a, m, j), count to estimate probabilities



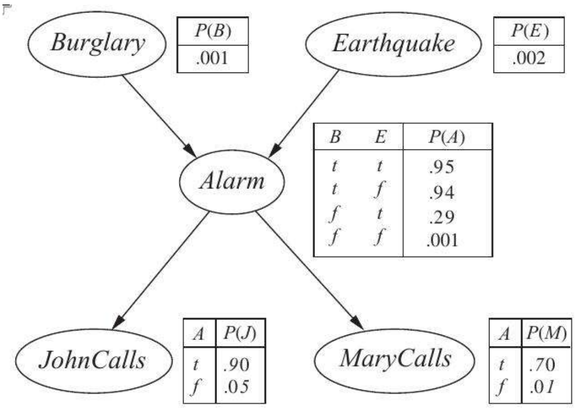
Approximate inference

- Generate random samples (b, e, a, m, j), count to estimate probabilities
- Random samples should respect conditional probabilities



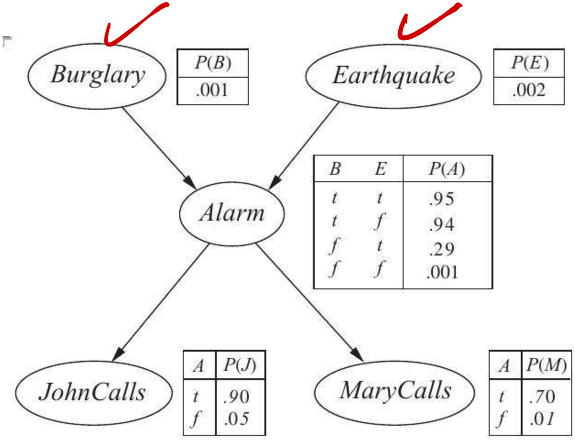
Approximate inference

- Generate random samples (b, e, a, m, j), count to estimate probabilities
- Random samples should respect conditional probabilities
- Fix $MB(x)$ before generating x



Approximate inference

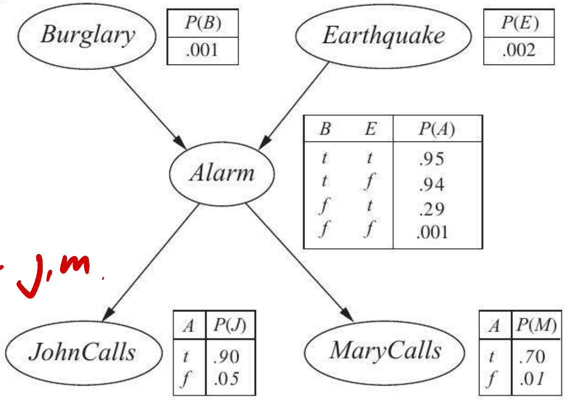
- Generate random samples (b, e, a, m, j), count to estimate probabilities
- Random samples should respect conditional probabilities
- ~~Fix $MB(x)$ before generating~~ x
- Generate in topological order
  - Generate b, e with probabilities $P(b)$ and $P(e)$
  - Generate a with probability $P(a \mid b, e)$
  - Generate j, m with probabilities $P(j \mid a)$, $P(m \mid a)$



- We are interested in $P(b \mid j, m)$

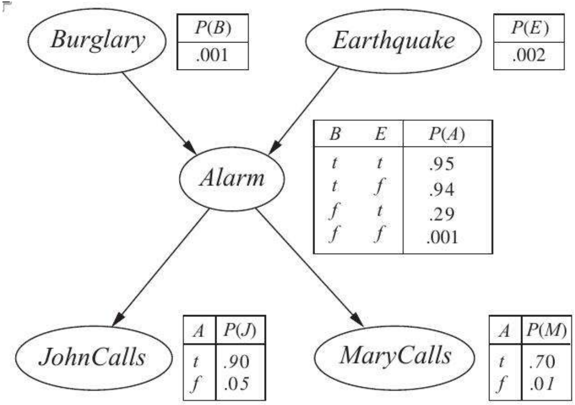
$$\frac{P(b,j,m)}{P(j,m)}$$

Among all sample with j, m
how many have b?



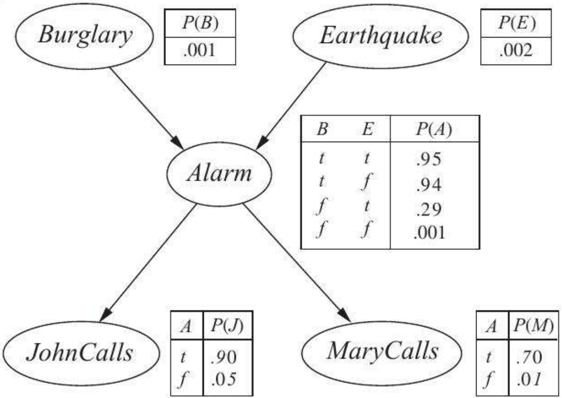
Approximate inference

- We are interested in $P(b \mid j, m)$
- Samples with $\neg j$ or $\neg m$ are useless



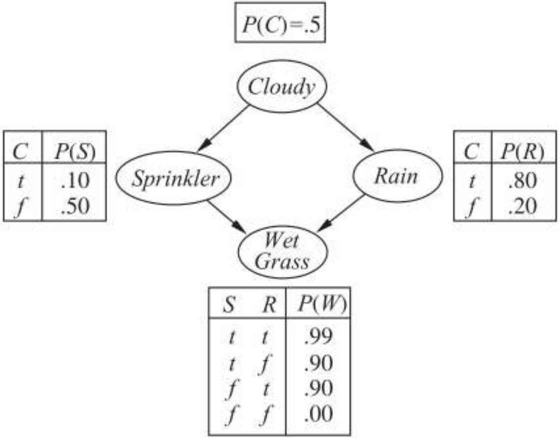
Approximate inference

- We are interested in $P(b \mid j, m)$
- Samples with $\neg j$ or $\neg m$ are useless
- Can we sample more efficiently?



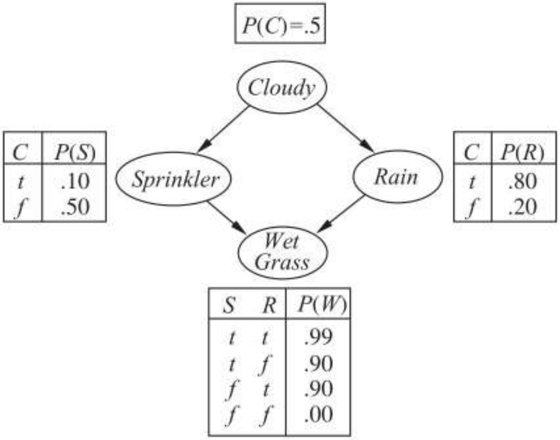
Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$



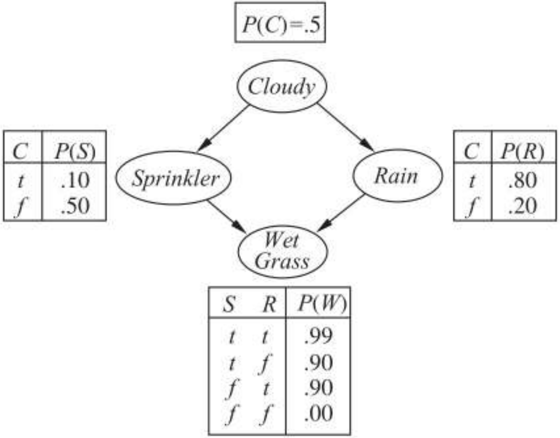
Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate *Cloudy*
  - Generate *Sprinkler*, *Rain*
  - Generate *Wet Grass*



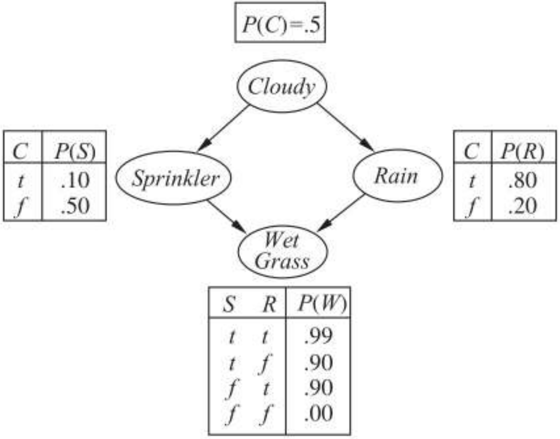
Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate $Cloudy$
  - Generate $Sprinkler$, $Rain$
  - Generate $Wet\ Grass$
- If we start with $\neg Cloudy$, sample is useless



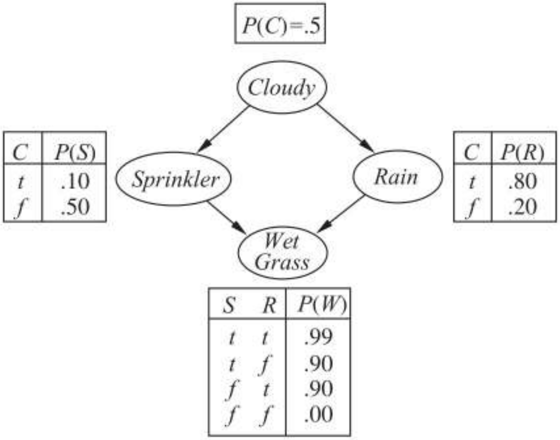
Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate *Cloudy*
  - Generate *Sprinkler*, *Rain*
  - Generate *Wet Grass*
- If we start with $\neg Cloudy$, sample is useless
- Immediately stop and reject this sample — rejection sampling



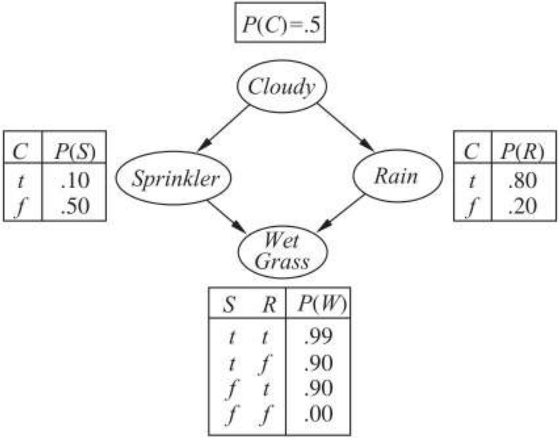
Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Topological order
  - Generate *Cloudy*
  - Generate *Sprinkler*, *Rain*
  - Generate *Wet Grass*
- If we start with $\neg Cloudy$, sample is useless
- Immediately stop and reject this sample — rejection sampling
- General problem with low probability situation — lots of samples



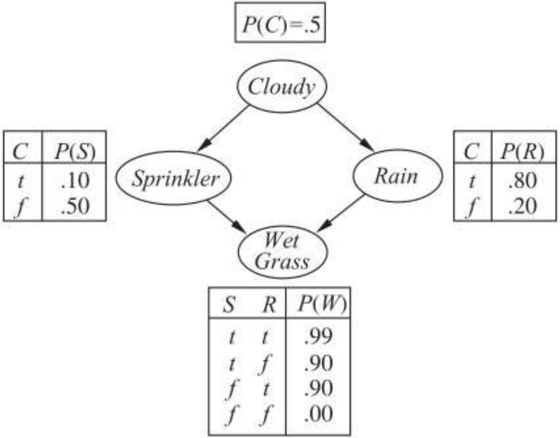
Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$



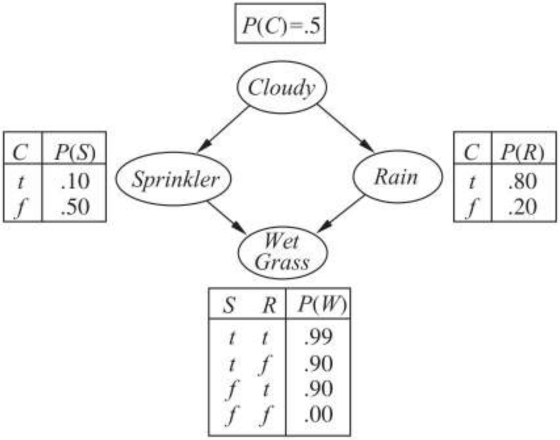
Likelihood weighted sampling

- $P(\mathit{Rain} \mid \mathit{Cloudy}, \mathit{Wet\ Grass})$
- Fix evidence $\mathit{Cloudy}, \mathit{Wet\ Grass}$ true



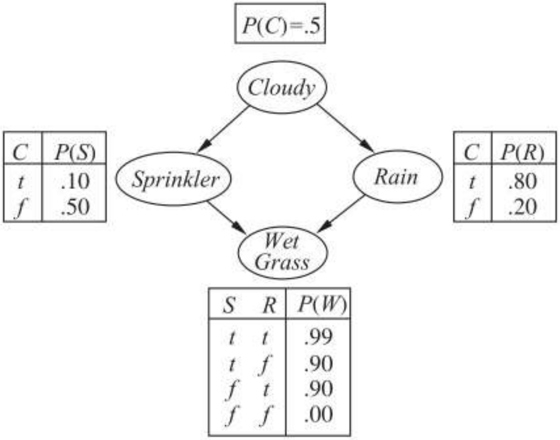
Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables



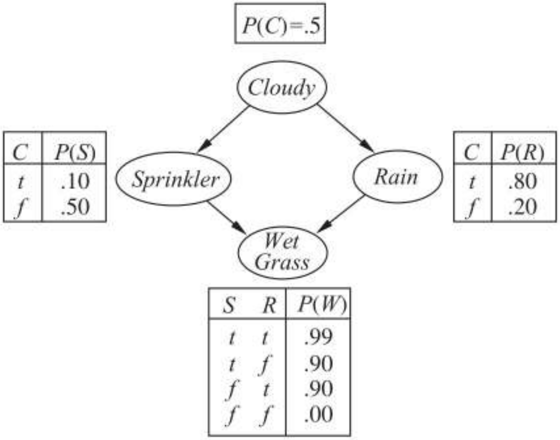
Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$



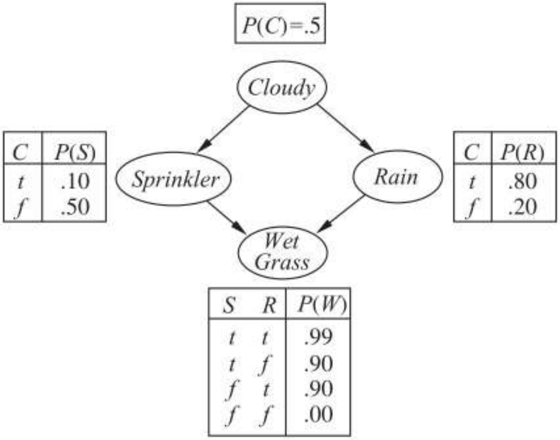
Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$



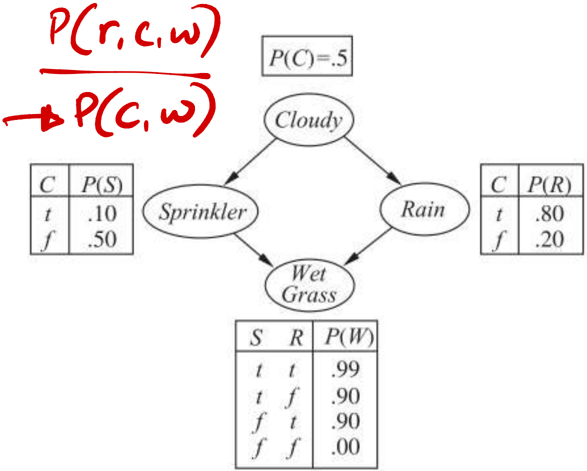
Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$
- 0.45 is likelihood weight of sample



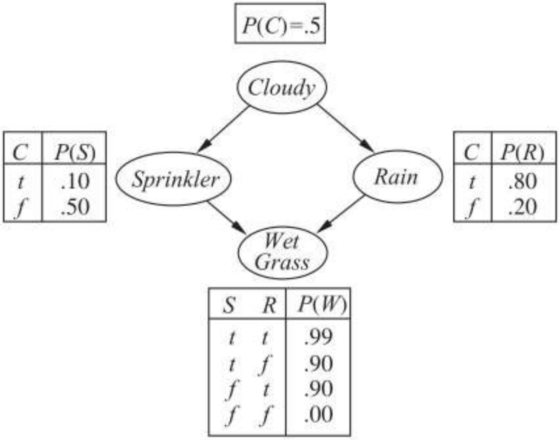
Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$
- 0.45 is likelihood weight of sample
- Samples $s_1, s_2, \ldots, s_N$ with weights $w_1, w_2, \ldots w_N$



Likelihood weighted sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$
- 0.45 is likelihood weight of sample
- Samples $s_1, s_2, \ldots, s_N$ with weights $w_1, w_2, \ldots w_N$
- $P(r \mid c, w) = \dfrac{\sum_{s_i\ has\ rain} w_i}{\sum_{1 \le j \le N} w_j}$

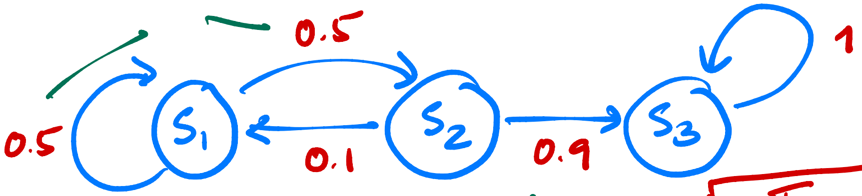


C	P(S)
t	.10
f	.50

C	P(R)
t	.80
f	.20

S	R	P(W)
t	t	.99
t	f	.90
f	t	.90
f	f	.00

Markov chains



Add up to 1

Each row sums to 1

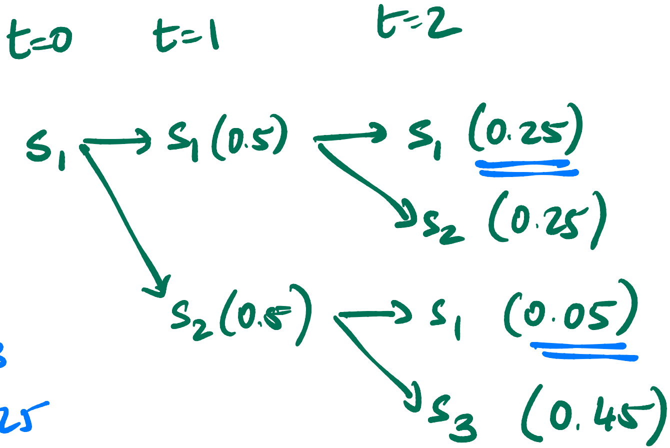
Stochastic Matrix

Transition Matrix

	s_1	s_2	s_3
s_1	0.5	0.5	0
s_2	0.1	0	0.9
s_3	0	0	1

Markov chains

	s_1	s_2	s_3
s_1	0.5	0.5	0
s_2	0.1	0	0.9
s_3	0	0	1



At t=2 $P(s_1) = 0.3$

$P(s_2) = 0.25$

$P(s_3) = 0.45$

State Distribution

$$\begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0.3 \\ 0.25 \\ 0.45 \end{bmatrix} \rightarrow \cdots$$

What happens in the limit?

In this example, intuitively $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Markov chains

	s_1	s_2	s_3
s_1	0.5	0.5	0
s_2	0.1	0	0.9
s_3	0	0	1

M

state transition matrix

π - state distribution

$\pi_0 = [1 \quad 0 \quad 0]$

$\pi_1 = ? \qquad \pi_0 \cdot M$

$$\pi_1[j] = \sum_{k=1}^{3} \pi_0 \cdot M_{kj}$$

$$\pi_{k+1} = \pi_k M$$

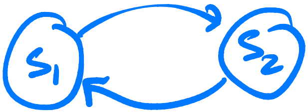
Stationary distribution

$$\pi^* = \pi^* M$$

M has a stationary distribution if it is ergodic

- irreducible / strongly connected
- aperiodic

Periodic



Aperiodic : $\exists k$ s.t. $\forall s_i, s_j$ there exist paths of length $k, k+1, k+2, \ldots$ from s_i to s_j

Claim If M is ergodic, M has a unique stationary distribution

$$xM = x$$

"Compute" π^* by running the Markov chain long enough from any initial state

Stationary distribution represents fraction of visits to each state in a long enough execution

How fast do we converge to π^*?

"Mixing" time

Markov chains

Bayesian network variables $V_1, V_2, \ldots, V_n$

Each assignment of values to V_i's is a sample

Samples $x_1, x_2, \ldots, x_N$ – all possible configurations of V_i's

$p_1 \quad p_2 \quad \ldots \quad p_N$

$$\sum p_i = 1$$

Markov chains

Design a Markov chain with N states

$$s_i \longleftrightarrow x_i$$

$$\Downarrow$$

$$M$$

$$\downarrow$$

$$\pi^* \quad \text{s.t.} \quad \pi^*_i = p_i$$