Lecture 21: 2 April, 2024

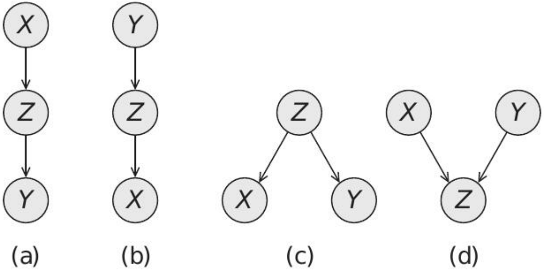
Madhavan Mukund

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

Data Mining and Machine Learning
January–April 2024

D-Separation

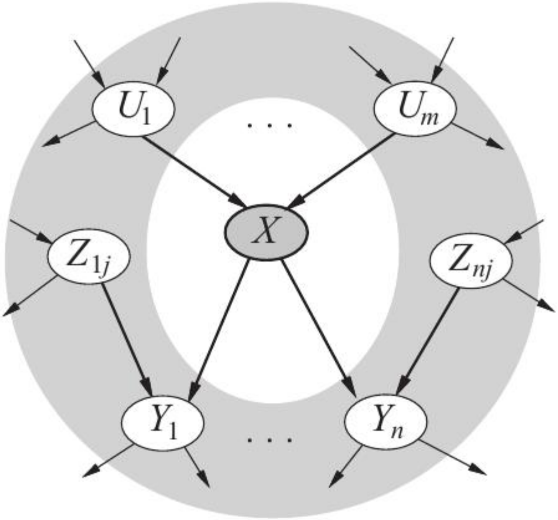
- Check if $X \perp Y \mid Z$
- Dependence should be blocked on every trail from X to Y
  - Each undirected path from X to Y is a sequence of basic trails
  - For (a), (b), (c), need Z present
  - For (d), need Z absent
  - In general, V-structure includes descendants of the bottom node



- 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 $y \in Y$

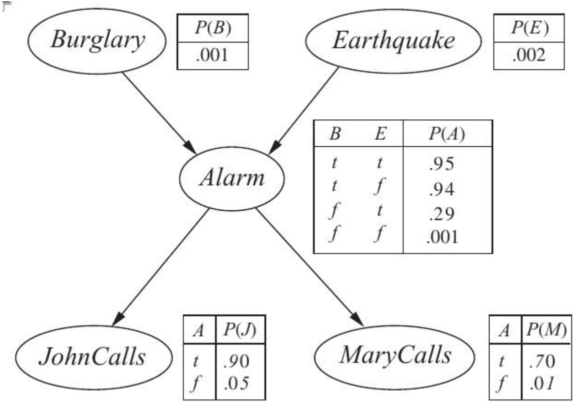
Markov blanket

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



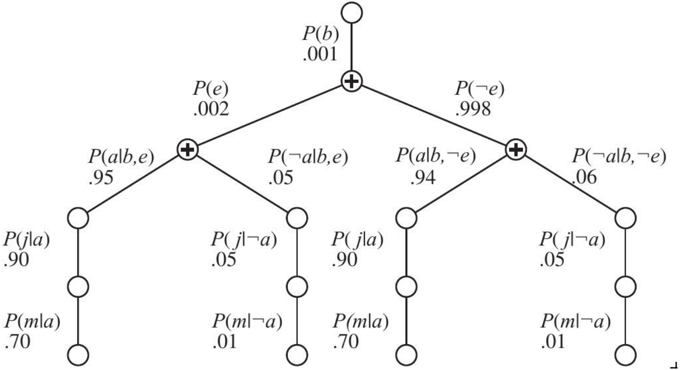
Computing with probabilistic graphical models

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



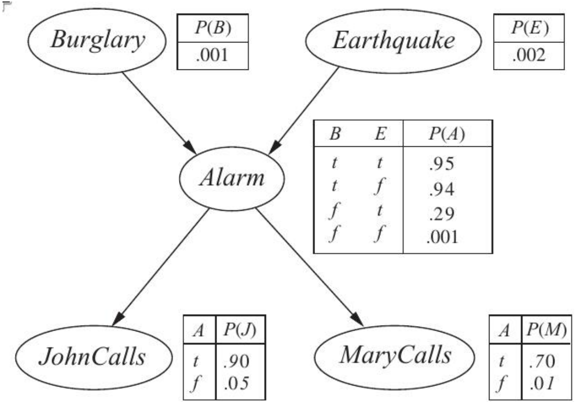
Computing with probabilistic graphical models

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



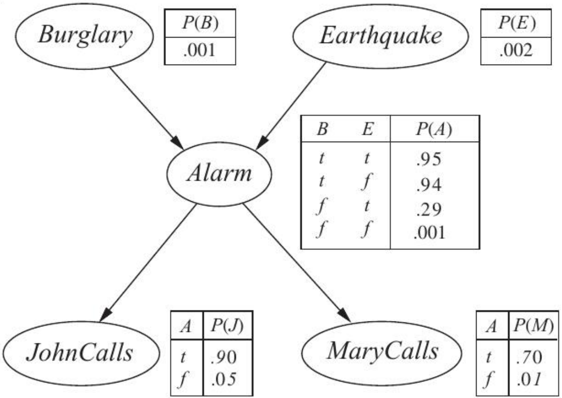
Approximate inference

- Generate random samples (b, e, a, m, j), count to estimate probabilities
- Random samples should respect conditional probabilities
- Fix parents of 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 $P(j \mid a)$, $P(m \mid a)$



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?



P(B)
.001

P(E)
.002

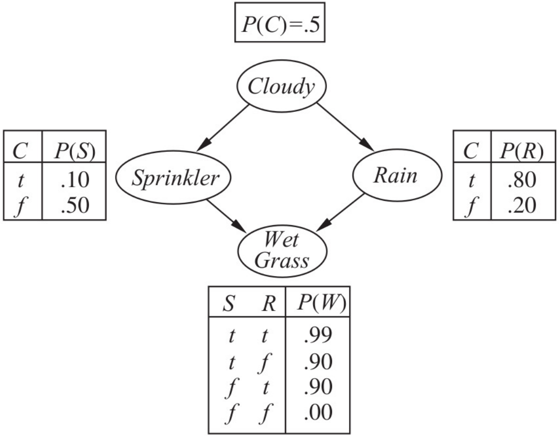
B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

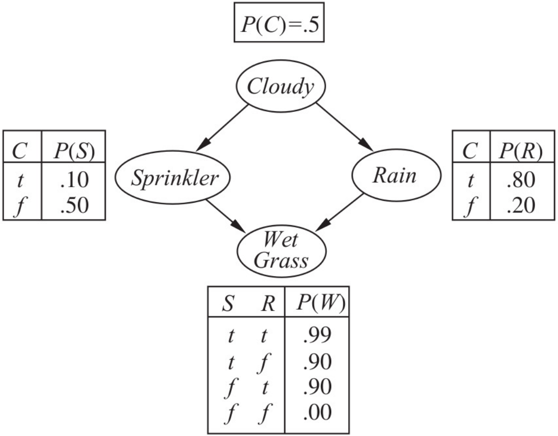
Rejection sampling

- $P(Rain \mid Cloudy, Wet\ Grass)$
- 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 — many samples are rejected



Likelihood weighted sampling

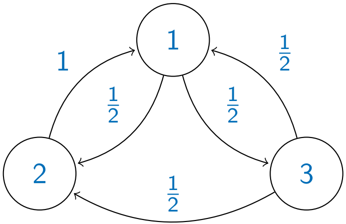
- $P(Rain \mid Cloudy, Wet\ Grass)$
- Fix evidence $Cloudy, Wet\ Grass$ true
- Then generate the other variables
- Suppose we generate $c, \neg s, r, w$
- Compute likelihood of evidence: $0.5 \times 0.9 = 0.45$
- 0.45 is likelihood weight of sample
- Samples $s_1, s_2, \ldots, s_N$ with weights $w_1,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



Approximate inference using Markov chains

Markov chains

- Finite set of states, with transition probabilities between states
- For us, a state will be an assignment of values to variables
- A three state Markov Chain



- Represent using a transition matrix — stochastic

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

- $P[j]$ is probability of being in state j
- Start in state 1, so initially $P = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

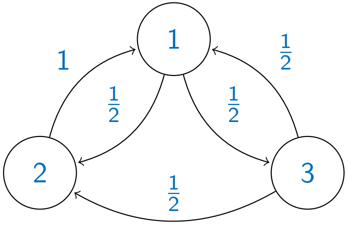
Markov chains ...

- After one step:

$$P^\top A = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

- After second step:

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix}$$

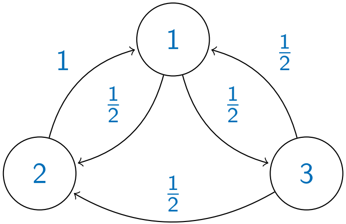


- After k steps, $P[j]$ is probability of being in state j
- Continuing our example,

$$\begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix} \rightarrow \begin{bmatrix} \frac{1}{4} & \frac{3}{8} & \frac{3}{8} \end{bmatrix} \rightarrow \begin{bmatrix} \frac{9}{16} & \frac{5}{16} & \frac{1}{8} \end{bmatrix}$$

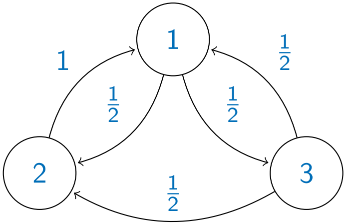
Ergodicity

- Is it the case that $P[j] > 0$ for all j continuously, after some point?
- Markov chain A is ergodic if there is some t_0 such that for every P, for all $t > t_0$, for every j, $(P^\top A^t)[j] > 0$.
  - No matter where we start, after $t > t_0$ steps, every state has a nonzero probability of being visited in step t
- Properties of ergodic Markov chains
  - There is a stationary distribution π^*, $(\pi^*)^\top A = \pi^*$
    - π^* is a left eigenvector of A
  - For *any* starting distribution P, $\lim_{t\to\infty} P^\top A^t = \pi^*$



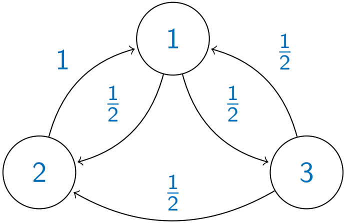
Ergodicity . . .

- How can ergodicity fail?
  - Starting from i, we reach a set of states from which there is no path back to i
  - We have a cycle $i \to j \to k \to i \to j \to k \cdots$, so we can only visit some states periodically
- Sufficient conditions for ergodicity
  - Irreducibility: When viewed as a directed graph, A is strongly connected
  - For all states i, j, there is a path from i to j and a path from j to i
  - Aperiodicity: For any pair of vertices i, j, the gcd of the lengths of all paths from i to j is 1
  - In particular, paths (loops) from i to i do not all have lengths that are multiples of some $k \geq 2$ — prevents bad cycles



Ergodicity . . .

- Can efficiently approximate $\lim_{t\to\infty} P^\top A^t$ by repeated squaring: $P^\top A^2$, $P^\top A^4$, $P^\top A^8$, . . . , $P^\top A^{2^k}$, . . .
  - Mixing time — how fast this converges to π^*
- Stationary distribution represents fraction of visits to each state in a long enough execution
- Can we create a Markov chain from a Bayesian network so that the stationary distribution is meaningful?



Approximate inference using Markov chains

- Bayesian network has variables $v_1, v_2, \ldots, v_n$
- Each assignment of values to the variables is a state
- Set up a Markov chain based on these states
- Stationary distribution should assign to state s the probability $P(s)$ in the Bayesian network
- How to reverse engineer the transition probabilities to achieve this?