# Artificial Intelligence (CS303)

Lecture 9: Inference in Bayesian Networks

# Hints for this lecture

- Calculate/estimate posterior probability distribution.
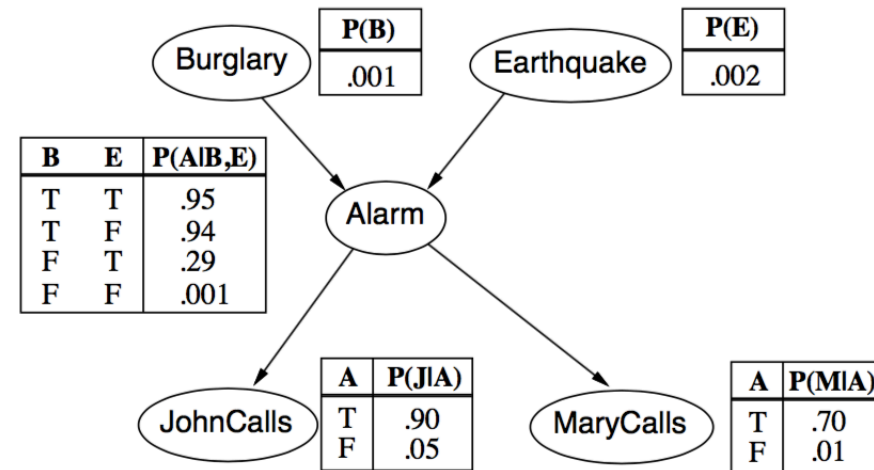
# Outline of this lecture

- Problem Statement

- Exact Inference

- Approximate Inference

# I. Problem Statement

# What are we looking for?

- Given a Bayesian Network, and an (or some) observed events, which specifies the value for evidence variables, we want to know the probability distribution of one (or several) query variables X, P (X | events).

- E. g. :  $\mathbf{P}(Burglary \mid JohnCalls = true, MaryCalls = true)$



| | P(B) |
|---|---|
| Burglary | .001 |

| | P(E) |
|---|---|
| Earthquake | .002 |

| B | E | P(A|B,E) |
|---|---|---|
| T | T | .95 |
| T | F | .94 |
| F | T | .29 |
| F | F | .001 |

Alarm

| A | P(J|A) |
|---|---|
| T | .90 |
| F | .05 |

JohnCalls

| A | P(M|A) |
|---|---|
| T | .70 |
| F | .01 |

MaryCalls

# More Examples

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?

Sensitivity analysis: which probability values are most critical?

Explanation: why do I need a new starter motor?
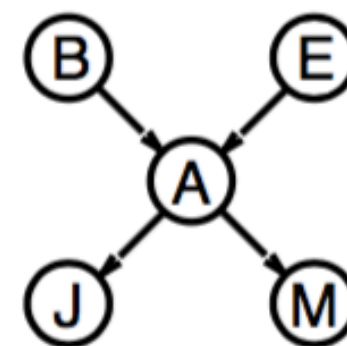
# II. Exact Inference

# Enumeration

Simple query on the burglary network:

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



Rewrite full joint entries using product of CPT entries:

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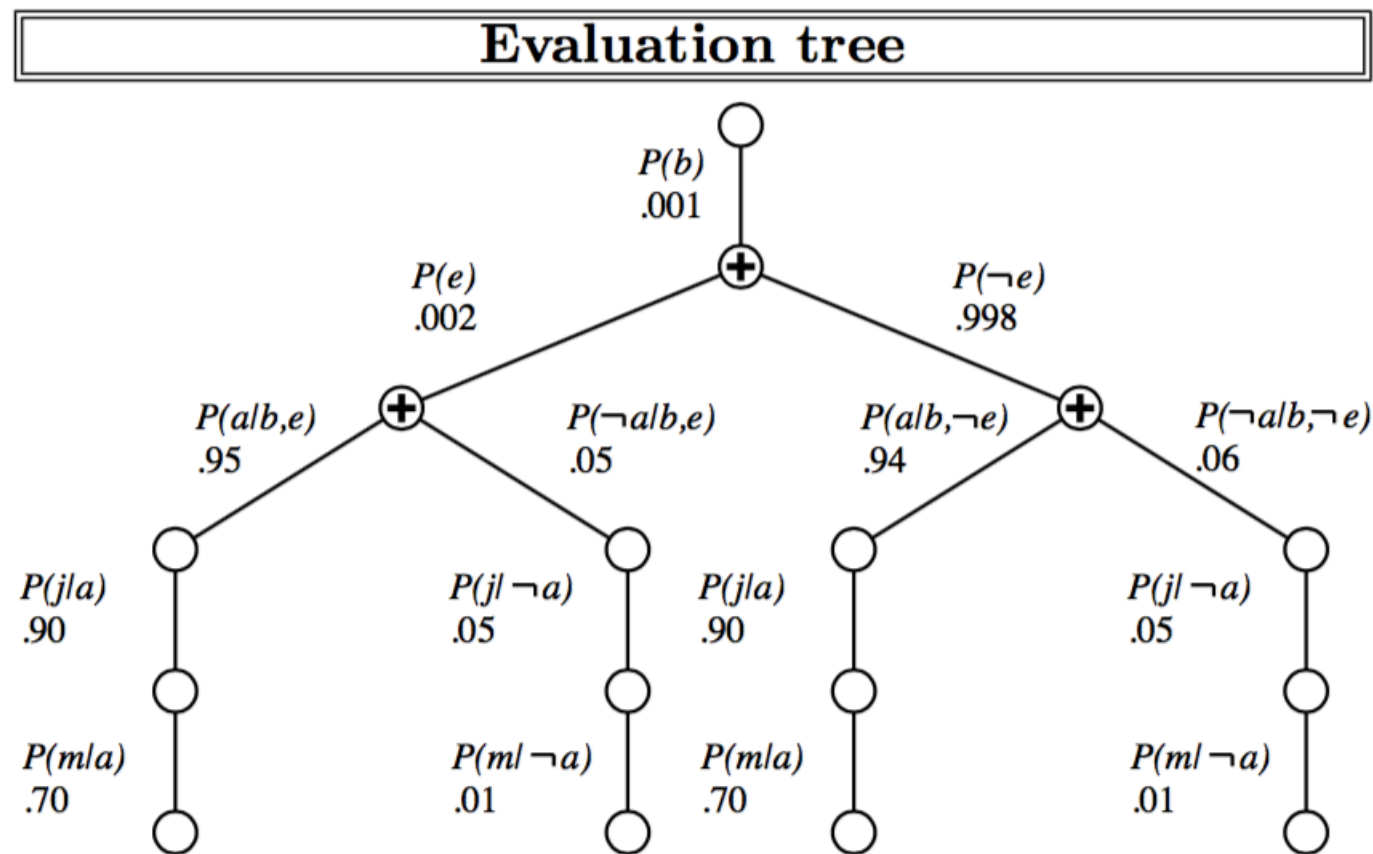$= \alpha \sum_e \sum_a \mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a)$
$= \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e)P(j|a)P(m|a)$

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

d=2 for Boolean variables

# Enumeration



**Evaluation tree**

$P(b)$
.001

$P(e)$
.002

$P(\neg e)$
.998

$P(a|b,e)$
.95

$P(\neg a|b,e)$
.05

$P(a|b,\neg e)$
.94

$P(\neg a|b,\neg e)$
.06

$P(j|a)$
.90

$P(j|\neg a)$
.05

$P(j|a)$
.90

$P(j|\neg a)$
.05

$P(m|a)$
.70

$P(m|\neg a)$
.01

$P(m|a)$
.70

$P(m|\neg a)$
.01

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

# Enumeration by Variable Elimination

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

$$\mathbf{P}(B|j,m)$$

$$= \alpha \underbrace{\mathbf{P}(B)}_{B} \Sigma_e \underbrace{P(e)}_{E} \Sigma_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M}$$

$$= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) P(j|a) f_M(a)$$

$$= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a \mathbf{P}(a|B,e) f_J(a) f_M(a)$$

$$= \alpha \mathbf{P}(B) \Sigma_e P(e) \Sigma_a f_A(a,b,e) f_J(a) f_M(a)$$

$$= \alpha \mathbf{P}(B) \Sigma_e P(e) f_{\bar{A}JM}(b,e) \text{ (sum out } A)$$

$$= \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E)$$

$$= \alpha f_B(b) \times f_{\bar{E}\bar{A}JM}(b)$$

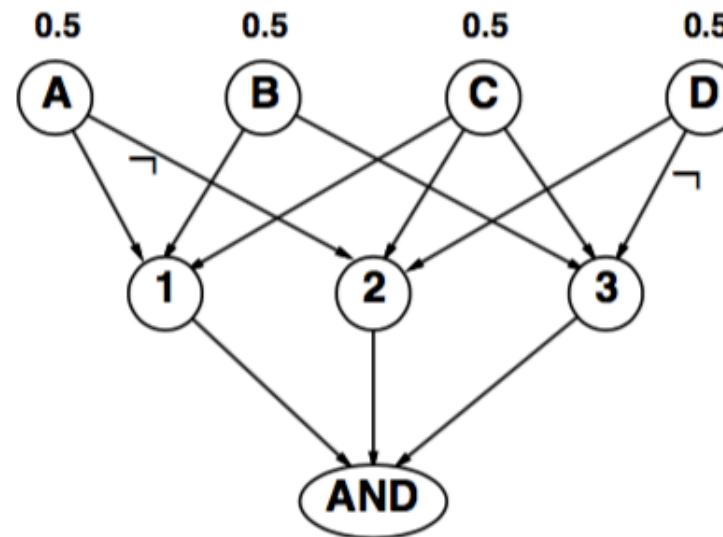# Complexity of Exact Inference

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
- – equivalent to **counting** 3SAT models  $\Rightarrow$  #P-complete



1. A ∨ B ∨ C

2. C ∨ D ∨ ¬A

3. B ∨ C ∨ ¬D

# III. Approximate Inference

# Basic Idea

- **Sampling/Monte Carlo/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 $P$

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

0.5

Coin

# Sampling from an empty network

**function** PRIOR-SAMPLE($bn$) **returns** an event sampled from $bn$
    **inputs:** $bn$, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

    $\mathbf{x} \leftarrow$ an event with $n$ elements
    **for** $i = 1$ **to** $n$ **do**
        $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
            given the values of $Parents(X_i)$ in $\mathbf{x}$
    **return** $\mathbf{x}$

# Sampling from an empty network

Probability that PRIORSAMPLE generates a particular event

$$S_{PS}(x_1 \ldots x_n) = \Pi^n_{i=1} P(x_i | parents(X_i)) = P(x_1 \ldots x_n)$$

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 $x_1, \ldots, x_n$

Then we have

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

That is, estimates derived from PRIORSAMPLE are consistent

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

# Rejection Sampling

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

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

Similar to a basic real-world empirical estimation procedure

$$\hat{\mathbf{P}}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e}) \qquad \text{(algorithm defn.)}$$
$$= \mathbf{N}_{PS}(X, \mathbf{e})/N_{PS}(\mathbf{e}) \qquad \text{(normalized by } N_{PS}(\mathbf{e}))$$
$$\approx \mathbf{P}(X, \mathbf{e})/P(\mathbf{e}) \qquad \text{(property of PRIORSAMPLE)}$$
$$= \mathbf{P}(X|\mathbf{e}) \qquad \text{(defn. of conditional probability)}$$

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!

15

# Likelihood Weighting

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

**function** LIKELIHOOD-WEIGHTING($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
    **local variables**: $\mathbf{W}$, a vector of weighted counts over $X$, initially zero

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

---

**function** WEIGHTED-SAMPLE($bn, \mathbf{e}$) **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 $\mathbf{e}$
            **then** $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
            **else** $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
    **return** $\mathbf{x}$, $w$

# Markov Chain Monte Carlo (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($X, \mathbf{e}, bn, N$) **returns** an estimate of $P(X|\mathbf{e})$
   **local variables:** $\mathbf{N}[X]$, a vector of counts over $X$, initially zero
                     $\mathbf{Z}$, the nonevidence variables 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 $\mathbf{Y}$
   **for** $j = 1$ to $N$ **do**
      **for each** $Z_i$ in $\mathbf{Z}$ **do**
         sample the value of $Z_i$ in $\mathbf{x}$ from $\mathbf{P}(Z_i|mb(Z_i))$
            given the values of $MB(Z_i)$ in $\mathbf{x}$
         $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where $x$ is the value of $X$ in $\mathbf{x}$
   **return** NORMALIZE($\mathbf{N}[X]$)
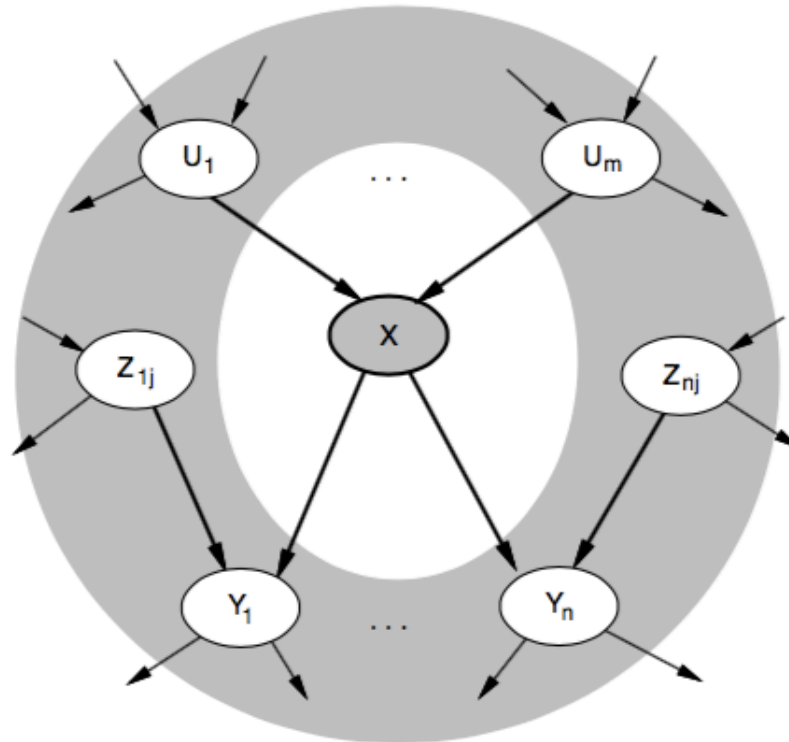
Markov Blanket

---

Can also choose a variable to sample at random each time

# Markov Chain Monte Carlo (MCMC)



**Markov blanket**

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

# Markov Chain Monte Carlo (MCMC)



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

Theorem: chain approaches stationary distribution: long-run fraction of time spent in each state is exactly proportional to its posterior probability
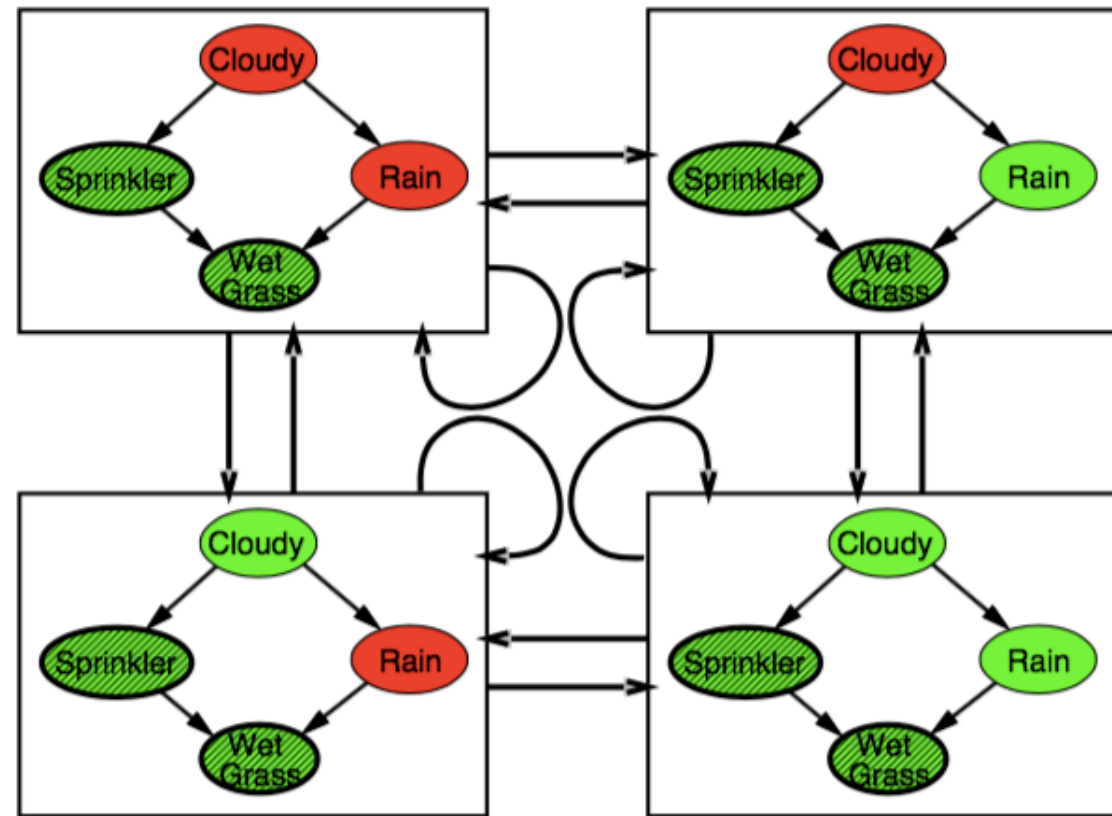
Wander about for a while, average what you see

# 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

# To be continued