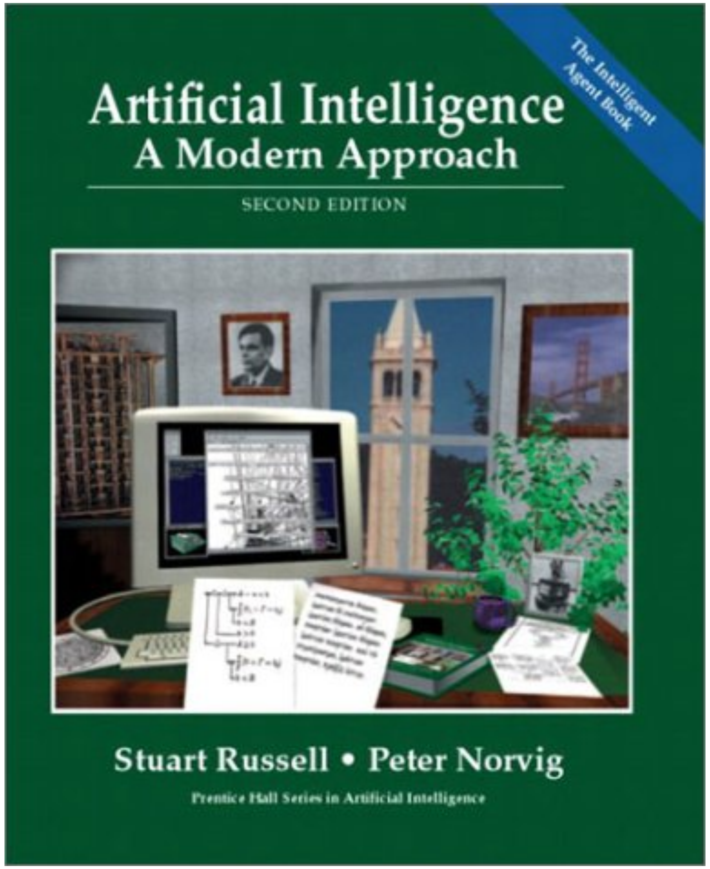
Intelligent Autonomous Agents
Agents and Rational Behavior
Lecture 4: Bayesian Networks

Ralf Möller, Rainer Marrone
Hamburg University of Technology

Literature



- Chapter 14 (Section 1 and 2)

Outline

- Agents
- Uncertainty
- Probability
- Syntax and Semantics
- Inference
- Independence and Bayes' Rule
- Bayesian Networks

Issues

- If a state is described by n propositions, then a belief space contains 2^n states for boolean domains (possibly, some have probability 0)
- → Modeling difficulty: many numbers must be entered in the first place
- → Computational issue: memory size and time

	toothache		¬toothache	
	pcatch	¬pcatch	pcatch	¬pcatch
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

- Toothache and Pcatch are independent given cavity (or ¬cavity), but this relation is hidden in the numbers ! [we will verify this]
- Bayesian networks explicitly represent independence among propositions to reduce the number of probabilities defining a belief state

	toothache		¬toothache	
	pcatch	¬pcatch	pcatch	¬pcatch
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

$$P(toothache, pcatch, cavity) = \frac{P(toothache, cavity) * P(pcatch, cavity)}{P(cavity)}$$

$$0{,}108 = \frac{((0{,}108 + 0{,}012) * (0{,}108 + 0{,}072))}{(0{,}108 + 0{,}012 + 0{,}072 + 0{,}008)}$$

$$0{,}108 = \frac{0{,}12 * 0{,}18}{0{,}2} = \frac{0{,}0216}{0{,}2} = 0{,}108$$

Bayesian networks

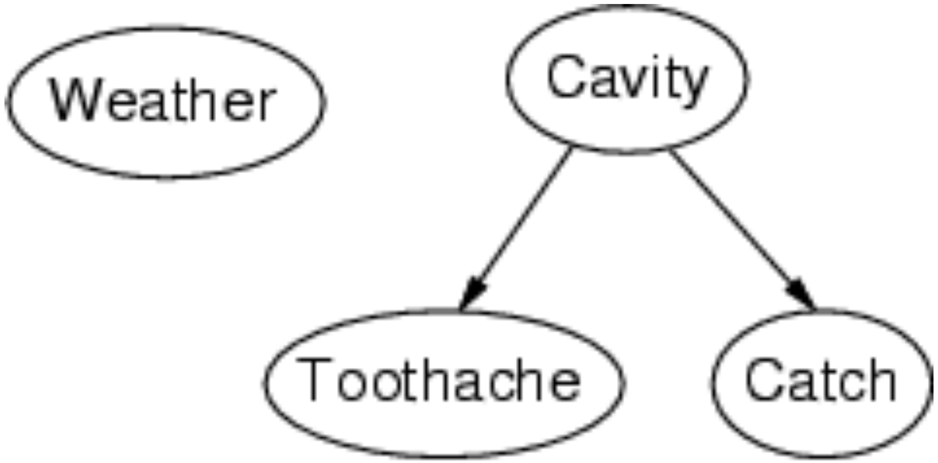
- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions

- Syntax:
  - a set of nodes, one per variable
  - a directed, acyclic graph (link ≈ "directly influences")
  - a conditional distribution for each node given its parents:
    $\mathbf{P}(X_i \mid Parents(X_i))$

- In the simplest case, conditional distribution represented as a conditional probability table (CPT) giving the distribution over X_i for each combination of parent values

Example

- Topology of network encodes conditional independence assertions:



- *Weather* is independent of the other variables
- *Toothache* and *Catch* are conditionally independent given *Cavity*

Remember: Conditional Independence

Random variable X is independent of random variable Y given random variable Z if, for all $x_i \in dom(X)$, $y_j \in dom(Y)$, $y_k \in dom(Y)$ and $z_m \in dom(Z)$,

$$\begin{aligned} & P(X = x_i | Y = y_j \wedge Z = z_m) \\ = \quad & P(X = x_i | Y = y_k \wedge Z = z_m) \\ = \quad & P(X = x_i | Z = z_m). \end{aligned}$$

That is, knowledge of Y's value doesn't affect your belief in the value of X, given a value of Z.

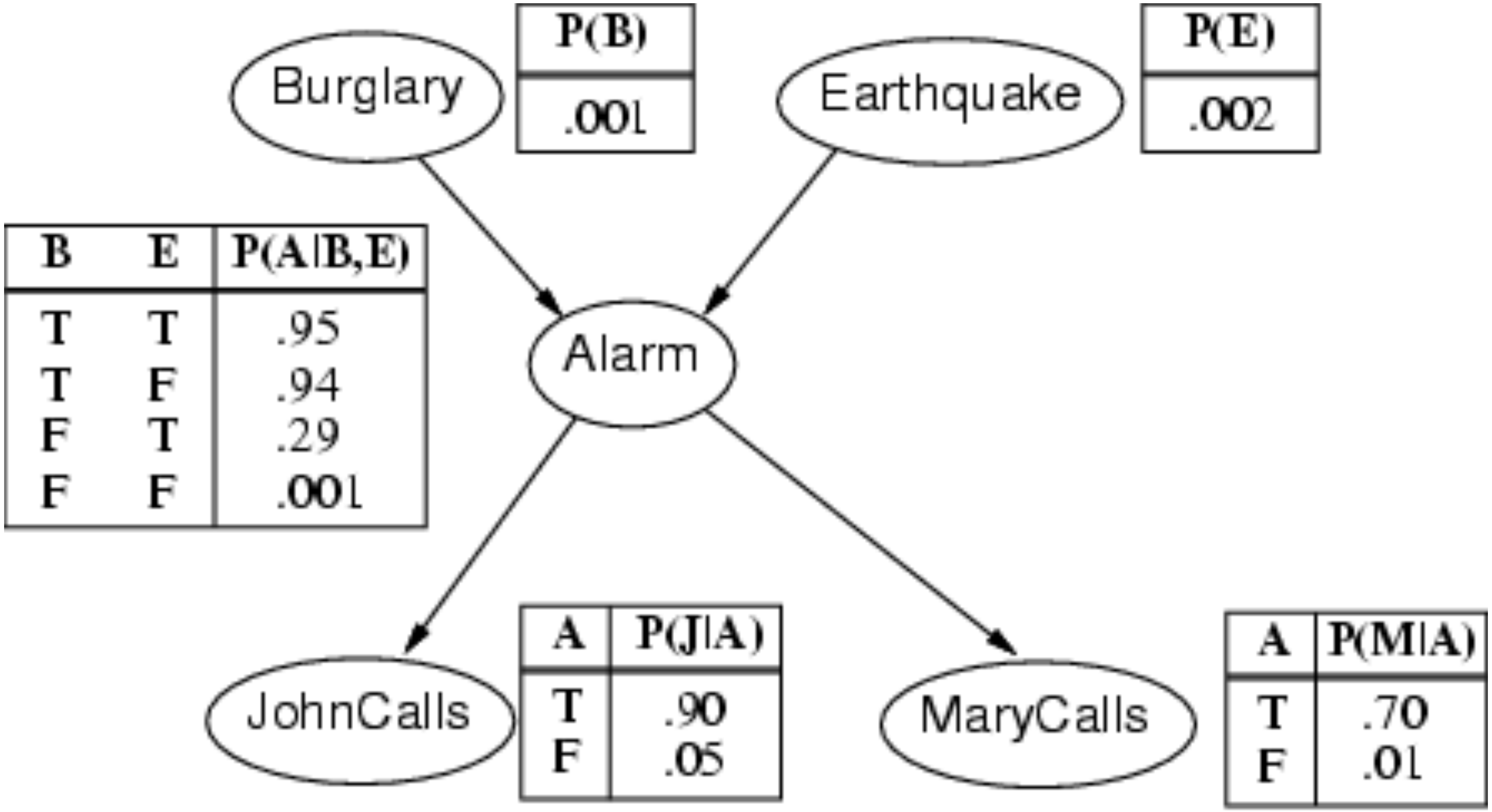
Example

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: *Burglary*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*

- Network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call

Example contd.



P(B)
.001

P(E)
.002

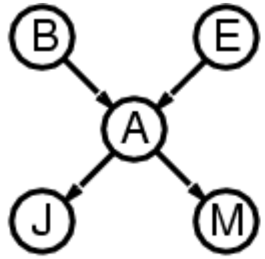
B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

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

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

Compactness

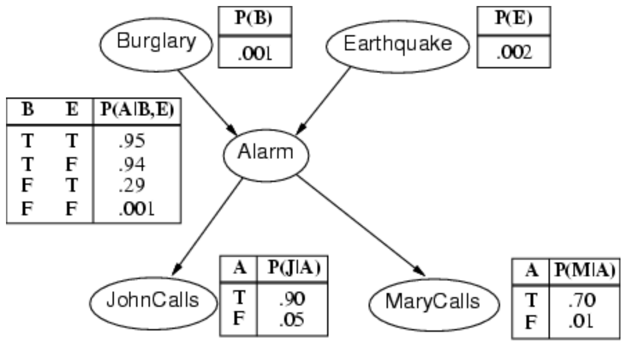
- A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values



- Each row requires one number p for $X_i = true$ (the number for $X_i = false$ is just $1-p$)
- If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers
- i.e., grows linearly with n, vs. $O(2^n)$ for the full joint distribution
- For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5-1 = 31$)

Semantics

The full joint distribution is defined as the product of the local conditional distributions:



$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i \mid Parents(X_i))$$

e.g., $\mathbf{P}(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$= \mathbf{P}(j \mid a)\, \mathbf{P}(m \mid a)\, \mathbf{P}(a \mid \neg b, \neg e)\, \mathbf{P}(\neg b)\, \mathbf{P}(\neg e)$

$= 0.90 \text{x} 0.7 \text{x} 0.001 \text{x} 0.999 \text{x} 0.998$

≈ 0.00063

Constructing Bayesian networks

- 1. Choose an ordering of variables $X_1, \dots, X_n$
- 2. For $i = 1$ to n
  - add X_i to the network
  - select parents from $X_1, \dots, X_{i-1}$ such that

$$\mathbf{P}(X_i \mid Parents(X_i)) = \mathbf{P}(X_i \mid X_1, \dots X_{i-1})$$

This choice of parents guarantees:

$$\mathbf{P}(X_1, \dots, X_n) = \pi_{i=1}^{n} \mathbf{P}(X_i \mid X_1, \dots, X_{i-1})$$

(chain rule)

$$= \pi_{i=1} \mathbf{P}(X_i \mid Parents(X_i))$$

(by construction)

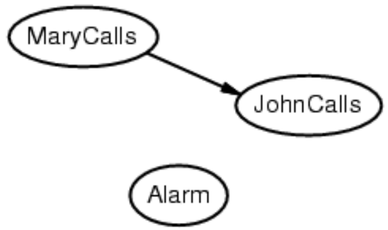
Example

- Suppose we choose the ordering *M, J, A, B, E*

***P**(J | M) = **P**(J)?*

Example

- Suppose we choose the ordering *M, J, A, B, E*

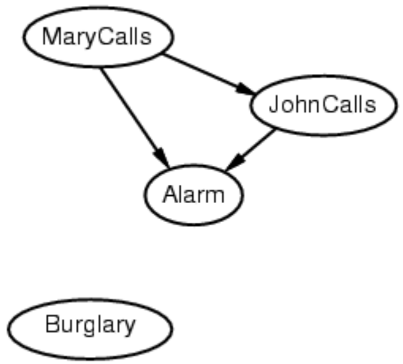


$\mathbf{P}(J \mid M) = \mathbf{P}(J)$? **No**

$\mathbf{P}(A \mid J, M) = \mathbf{P}(A \mid J)$? $\mathbf{P}(A \mid J, M) = \mathbf{P}(A)$?

Example

- Suppose we choose the ordering *M, J, A, B, E*



$\mathbf{P}(J \mid M) = \mathbf{P}(J)$? **No**

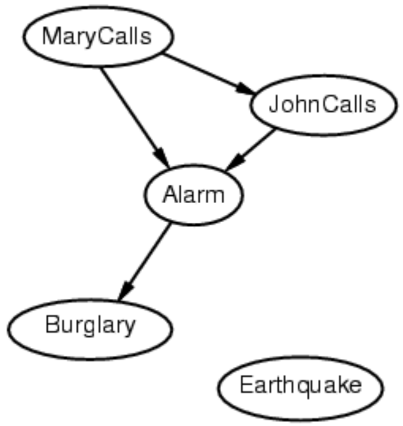
$\mathbf{P}(A \mid J, M) = \mathbf{P}(A \mid J)$? $\mathbf{P}(A \mid J, M) = \mathbf{P}(A)$? **No**

$\mathbf{P}(B \mid A, J, M) = \mathbf{P}(B \mid A)$?

$\mathbf{P}(B \mid A, J, M) = \mathbf{P}(B)$?

Example

- Suppose we choose the ordering M, J, A, B, E



P*(J | M) =* ***P****(J)?* **No**

P*(A | J, M) =* ***P****(A | J)?* ***P****(A | J, M) =* ***P****(A)?* **No**

P*(B | A, J, M) =* ***P****(B | A)?* **Yes**

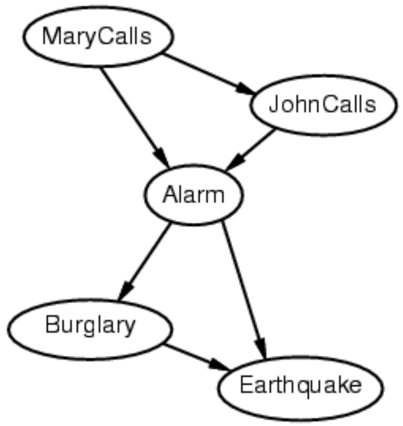
P*(B | A, J, M) =* ***P****(B)?* **No**

P*(E | B, A ,J, M) =* ***P****(E | A)?*

P*(E | B, A, J, M) =* ***P****(E | A, B)?*

Example

- Suppose we choose the ordering M, J, A, B, E



P*(J | M) =* ***P****(J)?* **No**

P*(A | J, M) =* ***P****(A | J)?* ***P****(A | J, M) =* ***P****(A)?* **No**

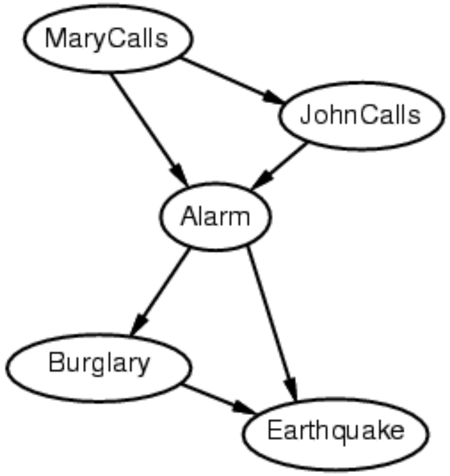
P*(B | A, J, M) =* ***P****(B | A)?* **Yes**

P*(B | A, J, M) =* ***P****(B)?* **No**

P*(E | B, A ,J, M) =* ***P****(E | A)?* **No**

P*(E | B, A, J, M) =* ***P****(E | A, B)?* **Yes**

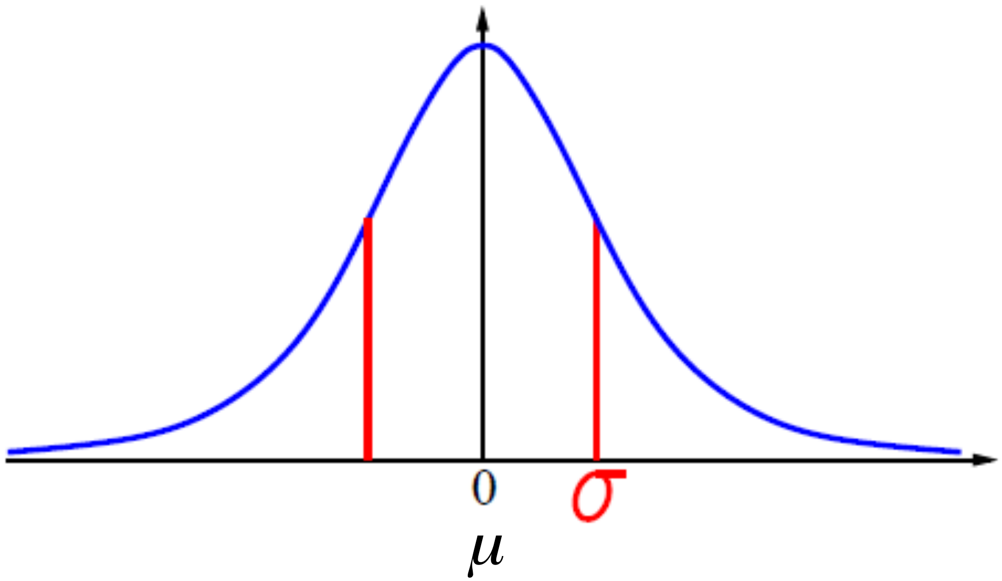
Example contd.



- Deciding conditional independence is hard in noncausal directions
- (Causal models and conditional independence seem hardwired for humans!)
- Network is less compact: 1 + 2 + 4 + 2 + 4 = 13 numbers needed instead of 10.

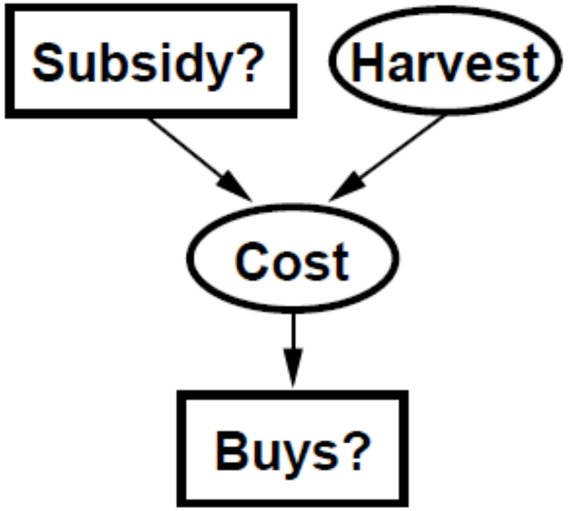
Gaussian density

$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$



Hybrid (discrete+contionous) networks

Discrete (*Subsidy?* and *Buys?*); continuous (*Harvest* and *Cost*)



Option 1: discretization—possibly large errors, large CPTs
Option 2: finitely parameterized canonical families

1) Continuous variable, discrete+continuous parents (e.g., *Cost*)
2) Discrete variable, continuous parents (e.g., *Buys?*)

Continuous child variables

Need one conditional density function for child variable given continuous parents, for each possible assignment to discrete parents

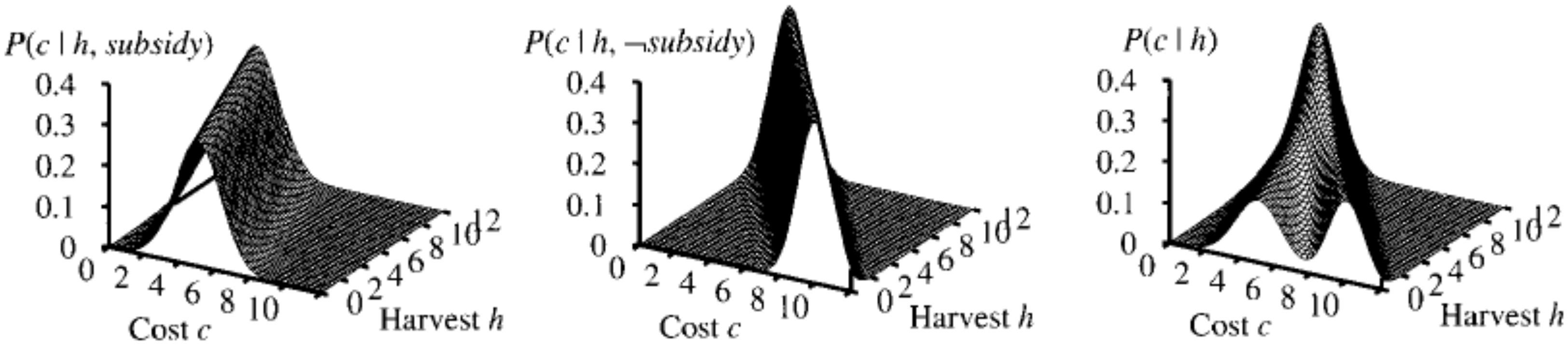
Most common is the linear Gaussian model, e.g.,:

$$
\begin{aligned}
&P(Cost = c | Harvest = h, Subsidy? = true) \\
&\quad = N(a_t h + b_t, \sigma_t)(c) \\
&\quad = \frac{1}{\sigma_t \sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right)
\end{aligned}
$$

Mean $Cost$ varies linearly with $Harvest$, variance is fixed

Linear variation is unreasonable over the full range
  but works OK if the **likely** range of $Harvest$ is narrow

Continuous child variables



All-continuous network with LG distributions

$\Rightarrow$ full joint distribution is a multivariate Gaussian

Discrete+continuous LG network is a conditional Gaussian network i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

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?

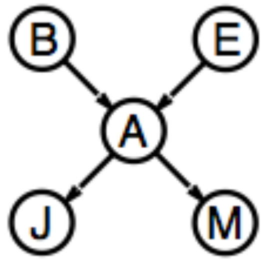
Sensitivity analysis: which probability values are most critical?

Explanation: why do I need a new starter motor?

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:



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

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

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 $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

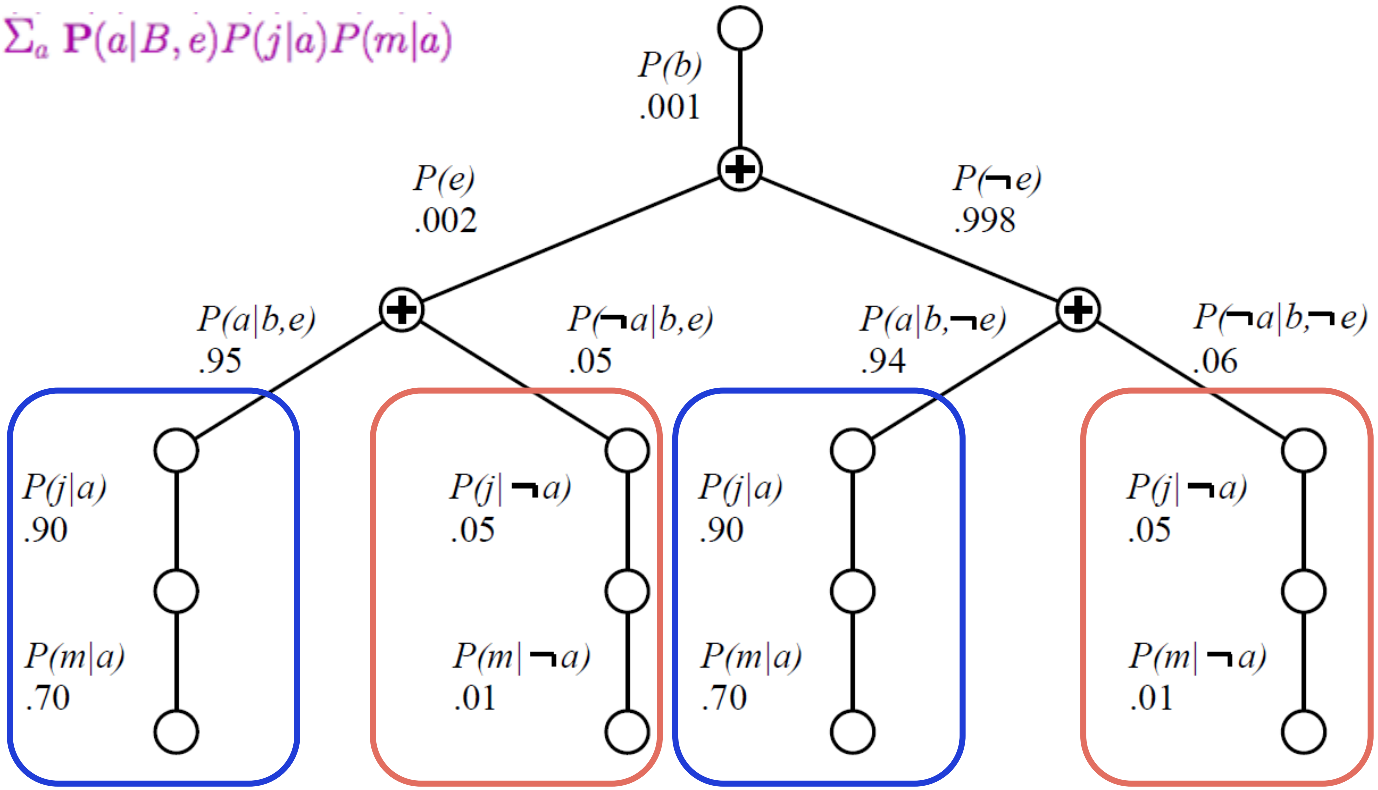
  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    extend **e** with value x_i for X
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(VARS[bn], **e**)
  return NORMALIZE($\mathbf{Q}(X)$)

---

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 $P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), **e**)
    else return $\Sigma_y\ P(y \mid Pa(Y)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is **e** extended with $Y = y$

Evaluation Tree

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

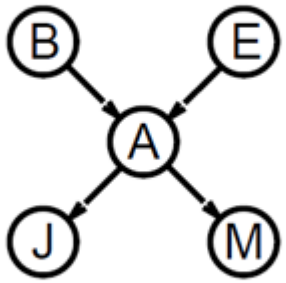


Enumeration is inefficient: repeated computation

e.g., computes $P(j|a)P(m|a)$ for each value of e

Basic Objects

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



- Track objects called factors
- Initial factors are local CPTs

$$\underbrace{P(B)}_{f_B(B)} \quad \underbrace{P(J|A)}_{f_J(A,J)} \quad \underbrace{P(A|B,E)}_{f_A(A,B,E)}$$

- During elimination create new factors

Basic Operations: Pointwise Product

- Pointwise Product of factors $\mathbf{f}_1$ and $\mathbf{f}_2$
  - for example: $\mathbf{f}_1(A,B) * \mathbf{f}_2(B,C) = \mathbf{f}(A,B,C)$
  - in general:

    $\mathbf{f}_1(X_1,\ldots,X_j,Y_1,\ldots,Y_k) * \mathbf{f}_2(Y_1,\ldots,Y_k,Z_1,\ldots,Z_l) =$
    $\mathbf{f}_1(X_1,\ldots,X_j,Y_1,\ldots,Y_k,Z_1,\ldots,Z_l)$
  - has 2^{j+k+l} entries (if all variables are binary)

Join by pointwise product

A	$f_{JM}(A)$
T	.9 * .7
F	.05 * .01

=

A	$f_J(A)$
T	.9
F	.05

A	$f_M(A)$
T	.7
F	.01

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

A	B	E	$f_{AJM}(A,B,E)$
T	T	T	.95 * .63
T	T	F	.94 * .63
T	F	T	.29 * .63
T	F	F	.001 * .63
F	T	T	.05 * .0005
F	T	F	.06 * .0005
F	F	T	.71 * .0005
F	F	F	.999 * .0005

=

A	B	E	$f_{AJM}(A,B,E)$
T	T	T	.95
T	T	F	.94
T	F	T	.29
T	F	F	.001
F	T	T	.05
F	T	F	.06
F	F	T	.71
F	F	F	.999

A	$f_{JM}(A)$
T	.63
F	.0005

Basic Operations: Summing out

- 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

$$\Sigma_x f_1 * \ldots * f_k = f_1 * \ldots * f_i * \Sigma_x f_{i+1} * \ldots * f_k$$
$$= f_1 * \ldots * f_i * f_{\bar{X}}$$

assuming $\mathbf{f}_1, \ldots, \mathbf{f}_i$ do not depend on X

Summing out

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

A	B	E	$f_{AJM}(A,B,E)$
T	T	T	.95 * .63
T	T	F	.94 * .63
T	F	T	.29 * .63
T	F	F	.001 * .63
F	T	T	.05 * .0005
F	T	F	.06 * .0005
F	F	T	.71 * .0005
F	F	F	.999 * .0005

Summing out *a*

$\Rightarrow$

B	E	$f_{\bar{A}JM}(B,E)$
T	T	.95 * .63 + .05 * .0005 = .5985
T	F	.94 * .63 + .06 * .0005 = .5922
F	T	.29 * .63 + .71 * .0005 = .1830
F	F	.001 * .63 + .999 * .0005 = .001129

What we have done

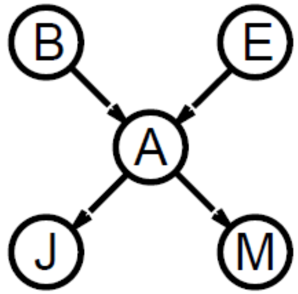
$$
\begin{aligned}
&\mathbf{P}(B|j,m)\\
&\quad = \alpha\, \underbrace{\mathbf{P}(B)}_{B} \textstyle\sum_e \underbrace{P(e)}_{E} \textstyle\sum_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M}\\
&\quad = \alpha \mathbf{P}(B) \textstyle\sum_e P(e) \textstyle\sum_a \mathbf{P}(a|B,e) P(j|a) f_M(a)\\
&\quad = \alpha \mathbf{P}(B) \textstyle\sum_e P(e) \textstyle\sum_a \mathbf{P}(a|B,e) f_J(a) f_M(a)\\
&\quad = \alpha \mathbf{P}(B) \textstyle\sum_e P(e) \textstyle\sum_a f_A(a,b,e) f_J(a) f_M(a)\\
&\quad = \alpha \mathbf{P}(B) \textstyle\sum_e P(e) f_{\bar{A}JM}(b,e) \text{ (sum out } A\text{)}\\
&\quad = \alpha \mathbf{P}(B) f_{\bar{E}\bar{A}JM}(b) \text{ (sum out } E\text{)}\\
&\quad = \alpha\, f_B(b) \times f_{\bar{E}\bar{A}JM}(b)
\end{aligned}
$$

Variable ordering

- Different selection of variables lead to different factors of different size.
- Every choice yields a valid execution
  - Different intermediate factors
- Time and space requirements depend on the largest factor constructed
- Heuristic may help to decide on a good ordering
- What else can we do?????

Irrelevant variables

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



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

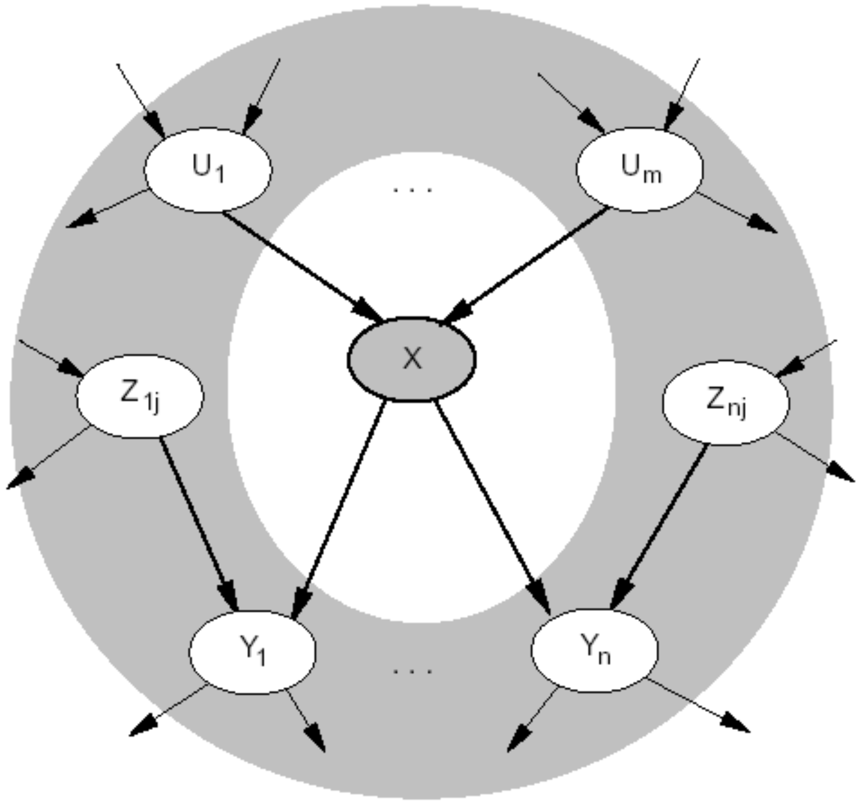
Sum over m is identically 1; M is **irrelevant** to the query

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

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

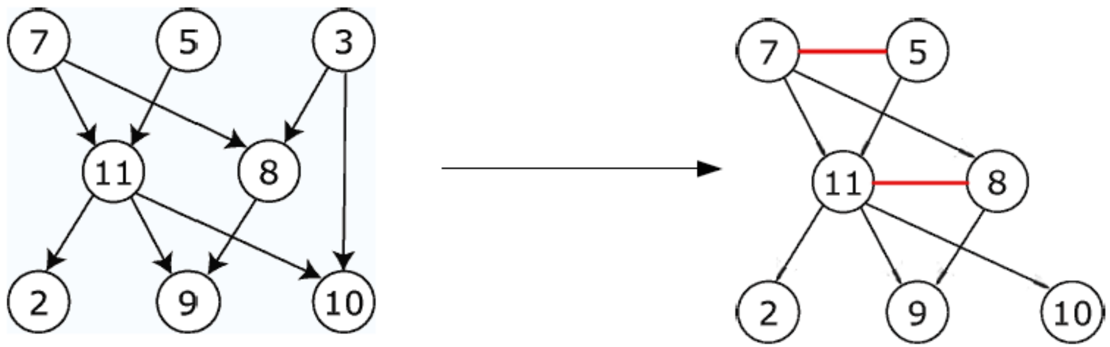
Markov Blanket

- Markov blanket: Parents + children + children's parents
- Node is conditionally independent of all other nodes in network, given its Markov Blanket



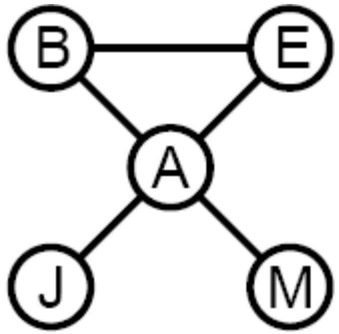
Moral Graph

- The moral graph is an undirected graph that is obtained as follows:
  - connect all parents of all nodes
  - make all directed links undirected
- Note:
  - the moral graph connects each node to all nodes of its Markov blanket
    - it is already connected to parents and children
    - now it is also connected to the parents of its children



Irrelevant variables continued:

- m-separation:
  - A is m-separated from B by C iff it is separated by C in the moral graph
- Example:
  - J is m-separated from E by A



Theorem 2: *Y* is irrelevant if it is m-separated from *X* by *E*

- Example:

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

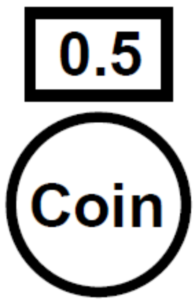
Approximate Inference In Bayesian Network

- Monte Carlo algorithm
  - Widely used to estimate quantities that are difficult to calculate exactly
  - Randomized sampling algorithm
  - Accuracy depends on the number of samples
  - Two families
    - Direct sampling
    - Markov chain sampling

Inference by stochastic simulation

Basic idea:

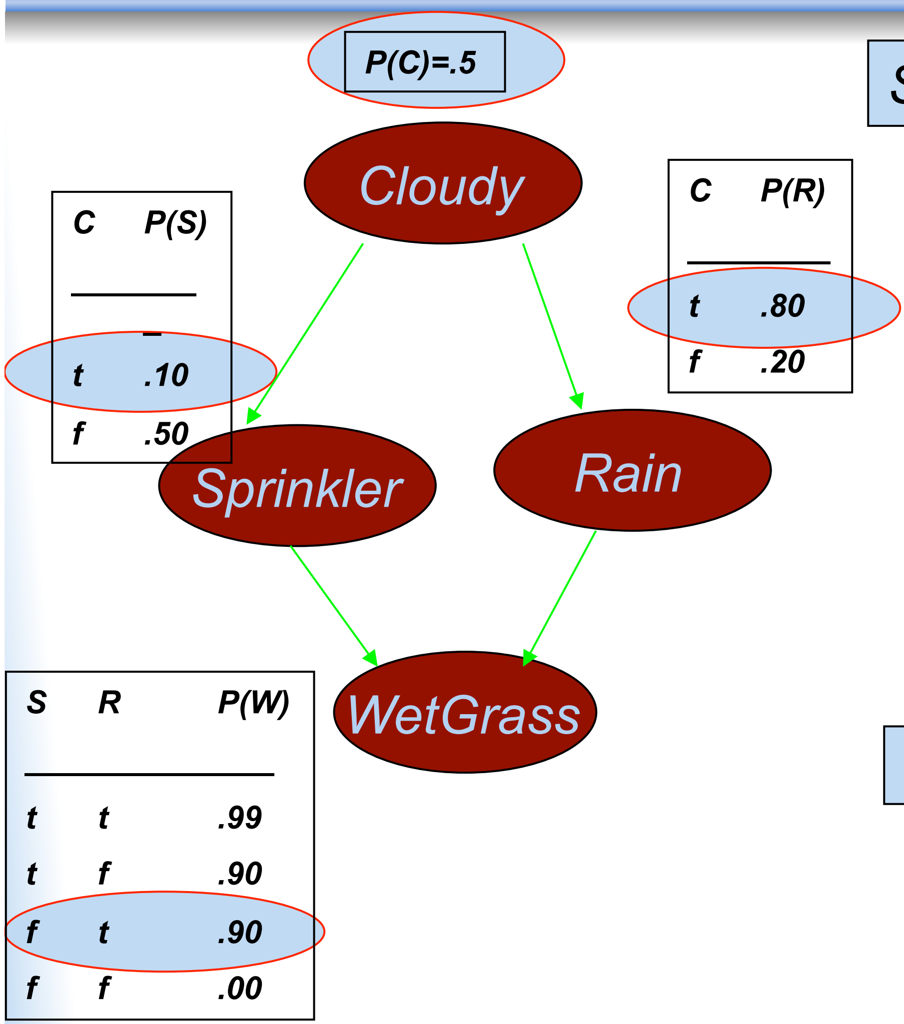
1) Draw N samples from a sampling distribution S
2) Compute an approximate posterior probability $\hat{P}$
3) Show this converges to the true probability P



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 in simple case



Sampling

[Cloudy, Sprinkler, Rain, WetGrass]

[true, , ,]

[true, false, ,]

[true, false, true,]

[true, false, true, true]

Estimating

N = 1000

N(Rain=true) = N([_ , _ , true, _]) = 511

P(Rain=true) = 0.511

Sampling from empty network

- Generating samples from a network that has no evidence associated with it (*empty* network)
- Basic idea
  - sample a value for each variable in topological order
  - using the specified conditional probabilities

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 $\mathbf{x}$

Properties

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

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

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

That is, estimates derived from PRIORSAMPLE are consistent

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

What if evidence is given?

- Sampling as defined above would generate cases that cannot be used

Rejection Sampling

- Used to compute conditional probabilities
- Procedure
  - Generating sample from prior distribution specified by the Bayesian Network
  - Rejecting all that do not match the evidence
  - Estimating probability

Rejection Sampling

$\hat{\mathbf{P}}(X|\mathbf{e})$ estimated from samples agreeing with $\mathbf{e}$

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

  for $j = 1$ to N **do**
    $\mathbf{x} \leftarrow$ PRIOR-SAMPLE(bn)
    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($\mathbf{N}[X]$)

Rejection Sampling Example

- Let us assume we want to estimate P(Rain|Sprinkler = true) with 100 samples

- 100 samples
  - 73 samples => Sprinkler = false
  - 27 samples => Sprinkler = true
    - 8 samples => Rain = true
    - 19 samples => Rain = false

- P(Rain|Sprinkler = true) = NORMALIZE((8,19)) = (0.296,0.704)

- Problem
  - It rejects too many samples

Analysis of rejection sampling

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

Hence rejection sampling returns consistent posterior estimates

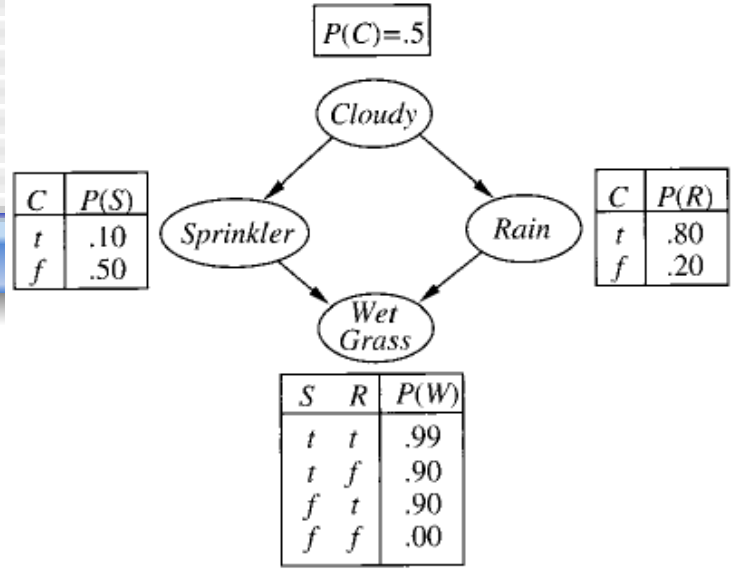
Problem: hopelessly expensive if $P(\mathbf{e})$ is small

$P(\mathbf{e})$ drops off exponentially with number of evidence variables!

Likelihood Weighting

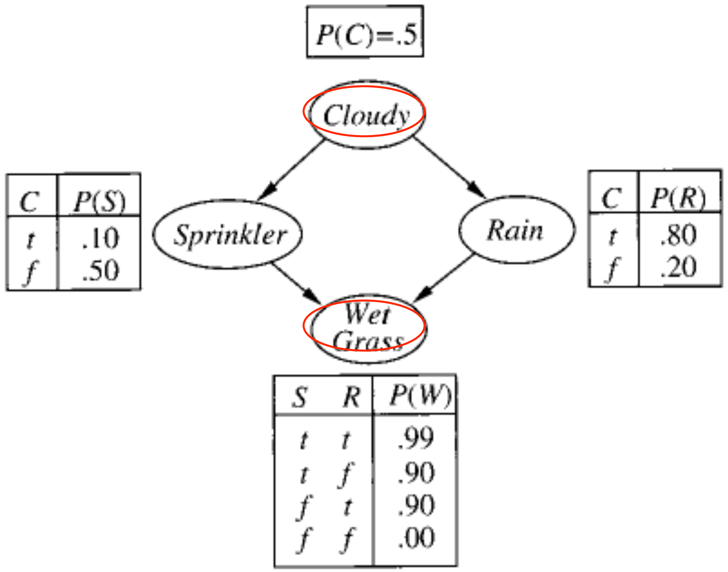
- Goal
  - Avoiding inefficiency of rejection sampling
- Idea
  - Generating only events consistent with evidence
  - Each event is weighted by likelihood that the event accords to the evidence

Likelyhood Weighting Example



- **P**(*Rain*|*Sprinkler=true, WetGrass = true)?*
- Sampling
  - The weight is set to 1.0
  - Sample from **P**(*Cloudy*) = (0.5,0.5) => true
  - *Sprinkler* is an evidence variable with value *true*
    w ← *w* * *P*(*Sprinkler=true* | *Cloudy = true*) = 0.1
  - Sample from P(*Rain*|*Cloudy=true*)=(0.8,0.2) => *true*
  - *WetGrass* is an evidence variable with value *true*
    w ← *w* * *P*(*WetGrass=true* |*Sprinkler=true, Rain = true*) = 0.099
  - [true, true, true, true] with weight 0.099
- Estimating
  - Accumulating weights to either Rain=true or Rain=false
  - Normalize

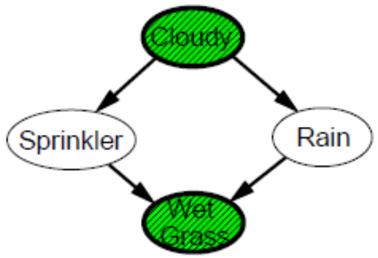
Likelyhood Weighting Example



- **P**(*Rain*|*Cloudy=true*, *WetGrass = true*)?
- Sampling
  - Cloudy is an evidence
    w ← *w* * *P*(*Cloudy = true*) = 0.5
  - *Sprinkler* no evidence
    Sample from P(Sprinkler| Cloudy=true)=(0.1, 0.9) false
  - Sample from P(*Rain|Cloudy=true*)=(0.8,0.2) => *true*
  - *WetGrass* is an evidence variable with value *true*
    w ←*w* * *P*(*WetGrass=true* |*Sprinkler=false, Rain = true*) = 0.45
  - [true, false, true, true] with weight 0.45

Likelihood analysis

Sampling probability for WEIGHTEDSAMPLE is
$S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i | parents(Z_i))$
Note: pays attention to evidence in **ancestors** only
$\Rightarrow$ somewhere "in between" prior and posterior distribution



Weight for a given sample $\mathbf{z}, \mathbf{e}$ is
$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i | parents(E_i))$

Weighted sampling probability is

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

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

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 ← WEIGHTED-SAMPLE(bn)
       W[x] ← W[x] + w where x is the value of X in x
   return NORMALIZE(W[X])
```

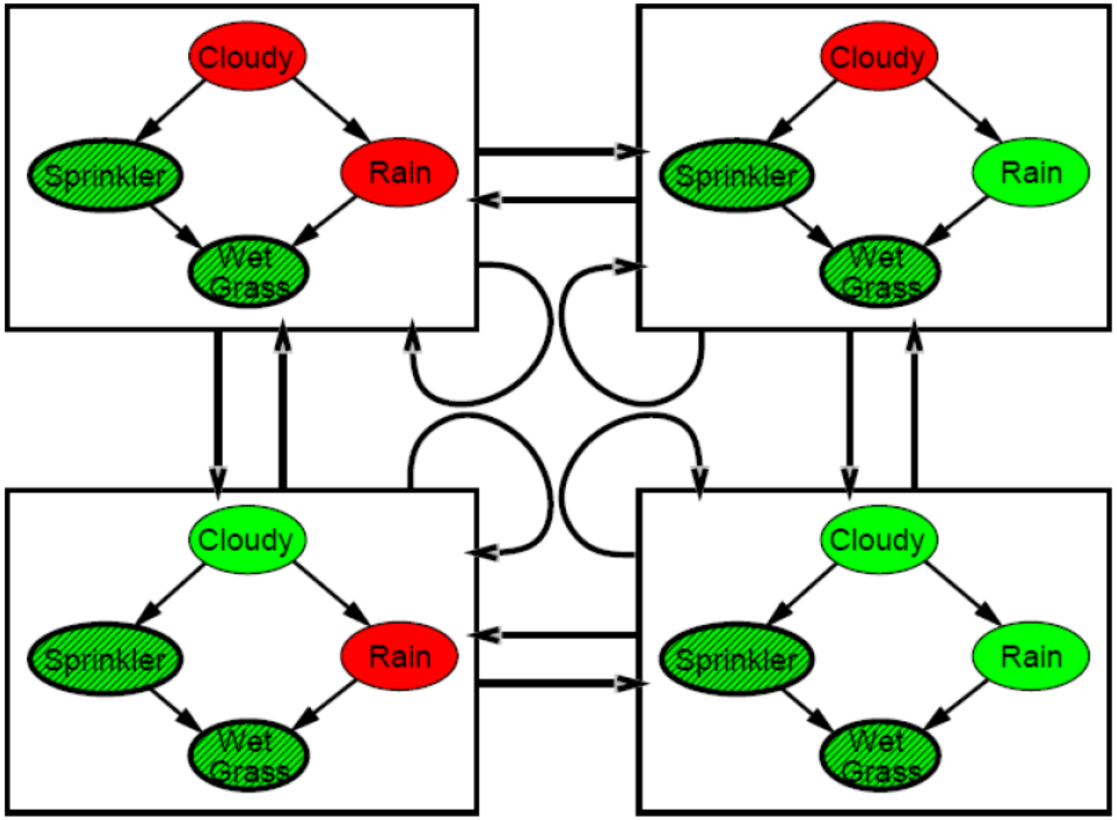
```
function WEIGHTED-SAMPLE(bn, e) returns an event and a weight

   x ← an event with n elements; w ← 1
   for i = 1 to n do
       if X_i has a value x_i in e
           then w ← w × P(X_i = x_i | parents(X_i))
           else x_i ← a random sample from P(X_i | parents(X_i))
   return x, w
```

Markov Chain Monte Carlo

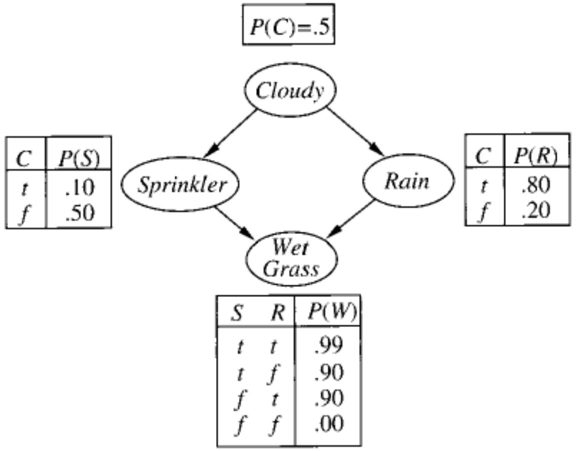
- Let's think of the network as being in a particular current state specifying a value for every variable
- MCMC generates each event by making a random change to the preceding event
- The next state is generated by randomly sampling a value for one of the nonevidence variables X_i, conditioned on the current values of the variables in the MarkovBlanket of X_i
- Likelihood Weighting only takes into account the evidences of the parents.

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



Wander about for a while, average what you see

Markov Chain Monte Carlo *Example*



- Query P(Rain|Sprinkler = true, WetGrass = true)
- Initial state is [true, true, false, true] [Cloudy,Sprinkler,Rain,WetGrass]

- The following steps are executed repeatedly:
  - Cloudy is sampled, given the current values of its MarkovBlanket variables

    So, we sample from *P(Cloudy|Sprinkler= true, Rain=false)*

    Suppose the result is Cloudy = false.
  - Now current state is [false, true, false, true] and counts are updated
  - Rain is sampled, given the current values of its MarkovBlanket variables

    So, we sample from *P(Rain|Cloudy=false,Sprinkler=true, WetGrass=true)*

    Suppose the result is Rain = true.
  - Then current state is [false, true, true, true]
- After all the iterations, let's say the process visited 20 states where rain is true and 60 states where rain is false then the answer of the query is NORMALIZE((20,60))=(0.25,0.75)

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 Z
  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($\mathbf{N}[X]$)

Can also choose a variable to sample at random each time

Summary

- Bayesian networks provide a natural representation for (causally induced) conditional independence
- Topology + CPTs = compact representation of joint distribution
- Generally easy for domain experts to construct
- Exact inference by variable elimination
  - polytime on polytrees, NP-hard on general graphs
  - space can be exponential as well
- Approximate inference based on sampling and counting help to overcome complexity of exact inference