Lecture 10: Bayesian Networks and Inference

CS 580 (001) - Spring 2018

Amarda Shehu

Department of Computer Science
George Mason University, Fairfax, VA, USA

May 02, 2018

1. Outline of Today's Class – Bayesian Networks and Inference

2. Bayesian Networks
   - Syntax
   - Semantics
   - Parameterized Distributions

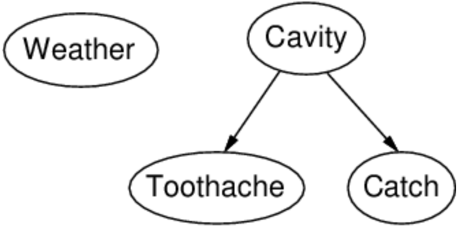
3. Inference on Bayesian Networks
   - Exact Inference by Enumeration
   - Exact Inference by Variable Elimination
   - Approximate Inference by Stochastic Simulation
   - Approximate Inference by Markov Chain Monte Carlo (MCMC)
   - Digging Deeper...

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 $\approx$ "directly influences")
a conditional distribution for each node given its parents:
$\mathbf{P}(X_i | Parents(X_i))$

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

Topology of network encodes conditional independence assertions:



Weather is independent of the other variables

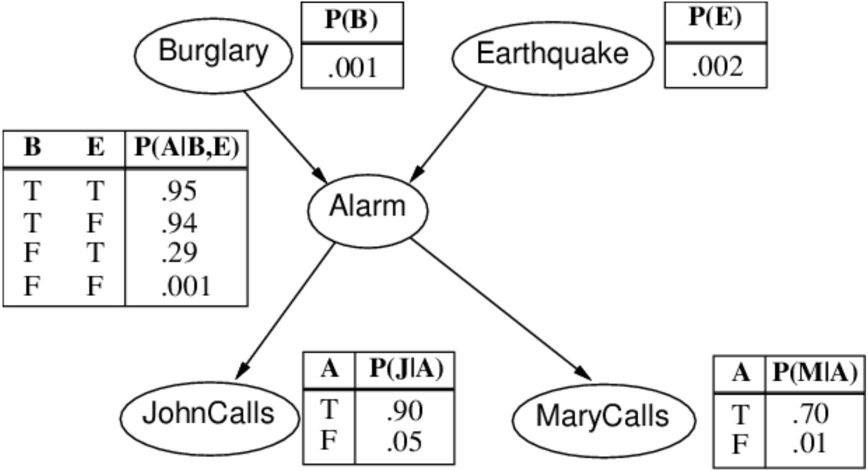
Toothache and *Catch* are conditionally independent given *Cavity*

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: *Burglar*, *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



P(B)
.001

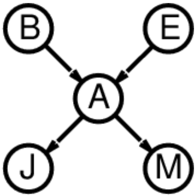
P(E)
.002

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

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 $1 - p$)

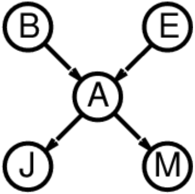
If each variable has no more than k parents,
the complete network requires $O(n \cdot 2^k)$ numbers

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. $2^5 - 1 = 31$)

Global Semantics

Global semantics defines the full joint



distribution

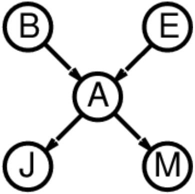
as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$

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

$$=$$

"Global" semantics defines the full joint



distribution

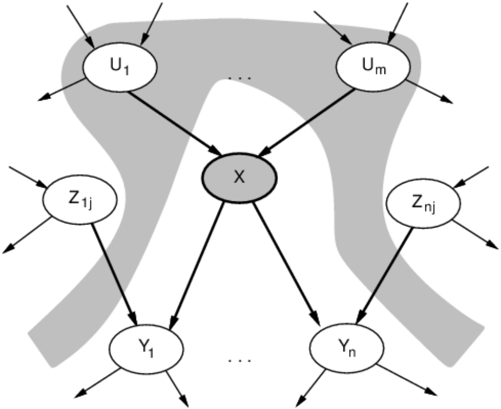
as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$

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

$$\begin{aligned} &= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \\ &\approx 0.00063 \end{aligned}$$

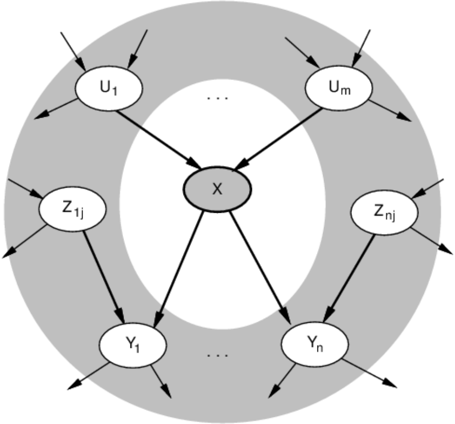
Local semantics: each node is conditionally independent of its nondescendants given its parents



Theorem: Local semantics $\Leftrightarrow$ global semantics

Markov Blanket

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



Constructing Bayesian Networks

Need a method such that a series of locally testable assertions of conditional independence guarantees the required global semantics

1. Choose an ordering of variables $X_1, \ldots, X_n$
2. For $i = 1$ to n

add X_i to the network

select parents from $X_1, \ldots, X_{i-1}$ such that

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

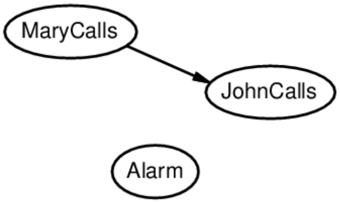
This choice of parents guarantees the global semantics:

$$\begin{aligned} \mathbf{P}(X_1, \ldots, X_n) &= \prod_{i=1}^{n} \mathbf{P}(X_i | X_1,\ \ldots,\ X_{i-1}) \quad \text{(chain rule)} \\ &= \prod_{i=1}^{n} \mathbf{P}(X_i | Parents(X_i)) \quad \text{(by construction)} \end{aligned}$$

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

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

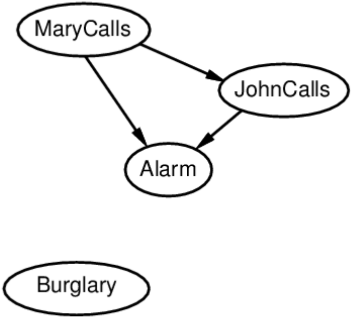
Suppose we choose the ordering M, J, A, B, E



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

Example

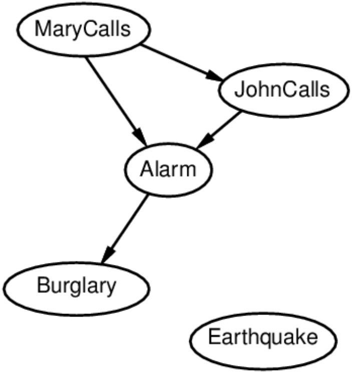
Suppose we choose the ordering M, J, A, B, E



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

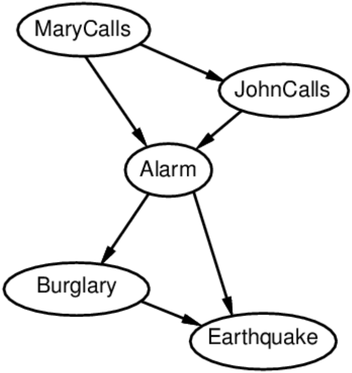
Example

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

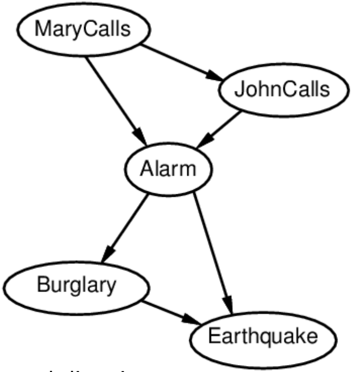


$P(J|M) = P(J)$? No
$P(A|J, M) = P(A|J)$? $P(A|J, M) = P(A)$? No
$P(B|A, J, M) = P(B|A)$? Yes
$P(B|A, J, M) = P(B)$? No
$P(E|B, A, J, M) = P(E|A)$?
$P(E|B, A, J, M) = P(E|A, B)$?

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



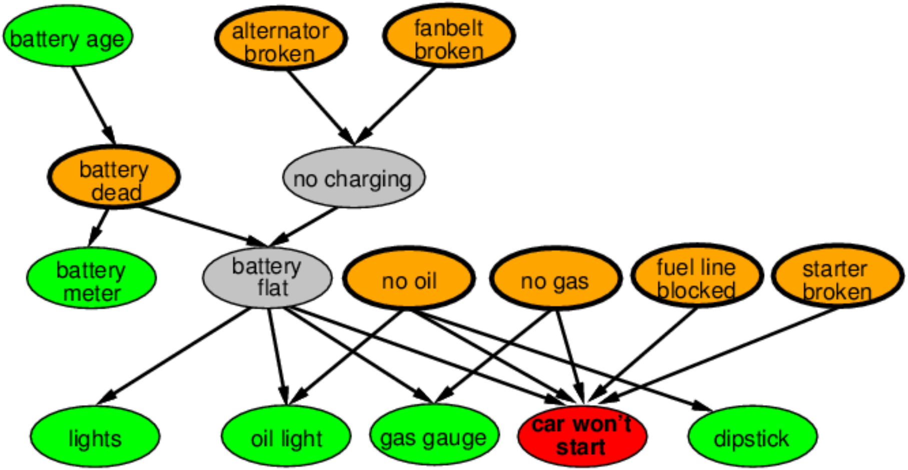
$P(J|M) = P(J)$? No
$P(A|J,M) = P(A|J)$? $P(A|J,M) = P(A)$? No
$P(B|A,J,M) = P(B|A)$? Yes
$P(B|A,J,M) = P(B)$? No
$P(E|B,A,J,M) = P(E|A)$? No
$P(E|B,A,J,M) = P(E|A,B)$? Yes



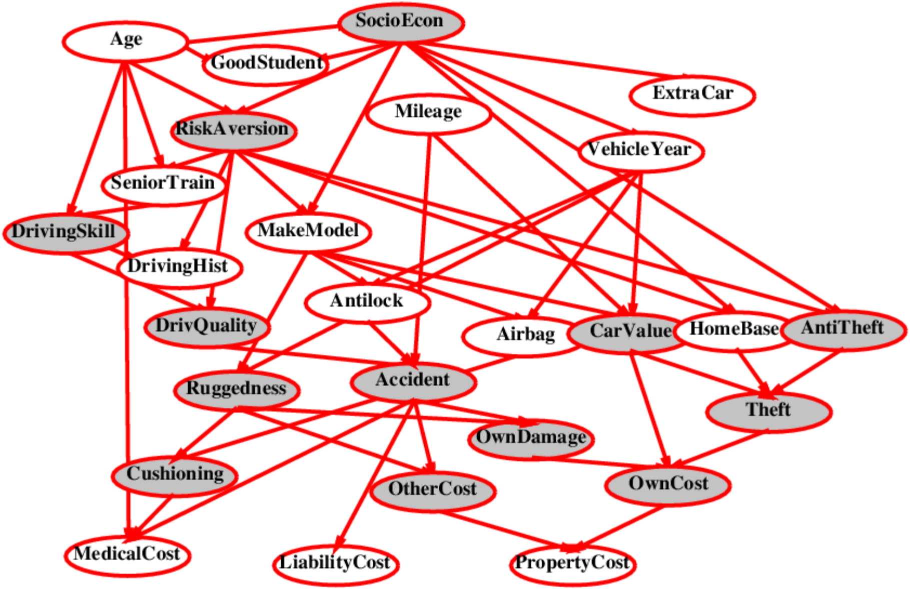
Deciding conditional independence is hard in noncausal directions
(Causal models and conditional independence seem hardwired for humans!)
Assessing conditional probabilities is hard in noncausal directions
Network is less compact: $1+2+4+2+4=13$ numbers needed

Example: Car Diagnosis

Initial evidence: car won't start
Testable variables (green), "broken, so fix it" variables (orange)
Hidden variables (gray) ensure sparse structure, reduce parameters



Example: Car Insurance



Compact Conditional Distributions

CPT grows exponentially with number of parents
CPT becomes infinite with continuous-valued parent or child

Solution: canonical distributions that are defined compactly

Deterministic nodes are the simplest case:

$X = f(Parents(X))$ for some function f

E.g., Boolean functions

$NorthAmerican \;\; \Leftrightarrow \;\; Canadian \vee US \vee Mexican$

E.g., numerical relationships among continuous variables

$$\frac{\partial Level}{\partial t} = \text{ inflow + precipitation - outflow - evaporation}$$

Compact Conditional Distributions

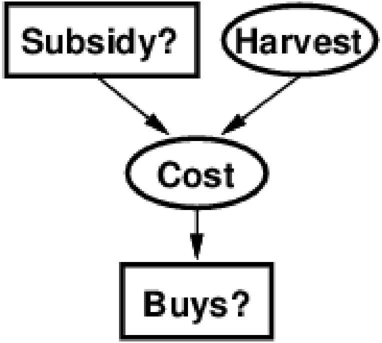
Noisy-OR distributions model multiple noninteracting causes
1) Parents $U_1 \ldots U_k$ include all causes (can add leak node)
2) Independent failure probability q_i for each cause alone

$$\Longrightarrow P(X|U_1 \ldots U_j, \neg U_{j+1} \ldots \neg U_k) = 1 - \prod_{i=1}^{j} q_i$$

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	F	F	$\mathbf{0.0}$	1.0
F	F	T	0.9	$\mathbf{0.1}$
F	T	F	0.8	$\mathbf{0.2}$
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	$\mathbf{0.6}$
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

Number of parameters **linear** in number of parents

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?*)

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

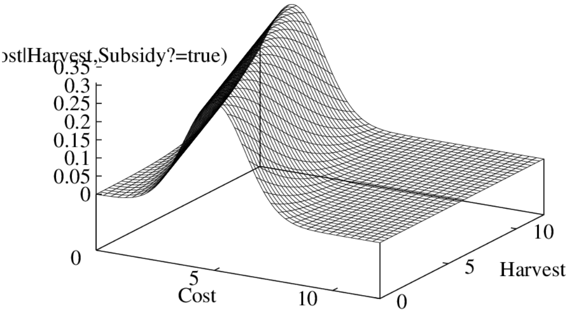
Most common is the linear Gaussian model, e.g.,:

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

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

Linear variation is unreasonable over the full range

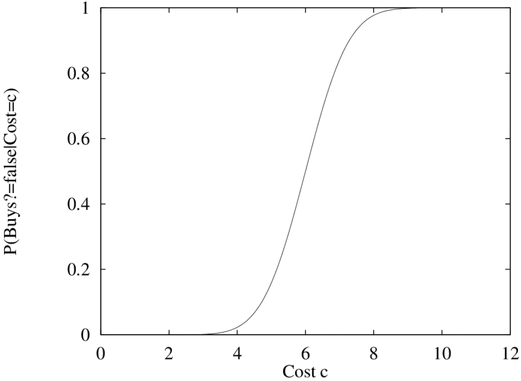
but works OK if the **likely** range of *Harvest* is narrow



All-continuous network with LG distributions
$\Longrightarrow$ 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

Probability of *Buys?* given *Cost* should be a "soft" threshold:



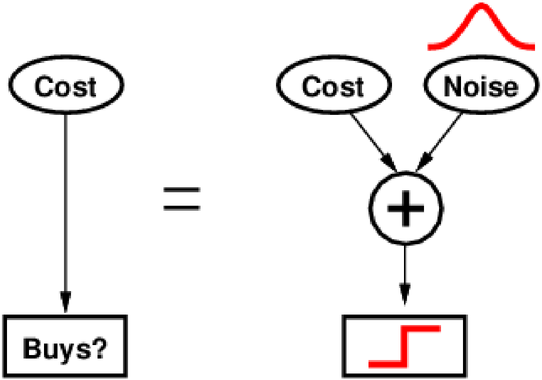
Probit distribution uses integral of Gaussian:

$\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$

$P(Buys? = true \mid Cost = c) = \Phi((-c + \mu)/\sigma)$

Why the probit?

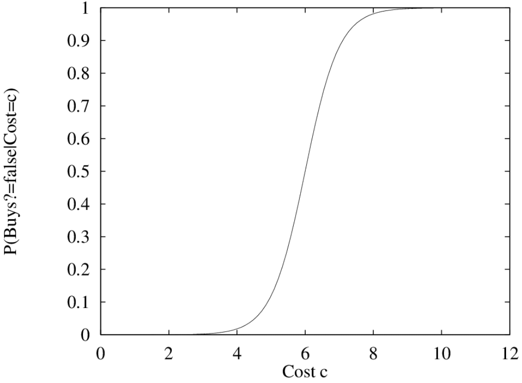
1. It's sort of the right shape
2. Can view as hard threshold whose location is subject to noise



Sigmoid (or logit) distribution also used in neural networks:

$$P(Buys? = true \mid Cost = c) = \frac{1}{1 + exp(-2\frac{-c+\mu}{\sigma})}$$

Sigmoid has similar shape to probit but much longer tails:



Summary on Bayesian Networks

Bayes nets provide a natural representation for (causally induced) conditional independence

Topology + CPTs = compact representation of joint distribution

Generally easy for (non)experts to construct

Canonical distributions (e.g., noisy-OR) = compact representation of CPTs

Continuous variables $\Longrightarrow$ parameterized distributions (e.g., linear Gaussian)

Next: Inference on Bayesian Networks

Summary on Bayesian Networks

Bayes nets provide a natural representation for (causally induced) conditional independence

Topology + CPTs = compact representation of joint distribution

Generally easy for (non)experts to construct

Canonical distributions (e.g., noisy-OR) = compact representation of CPTs

Continuous variables $\implies$ parameterized distributions (e.g., linear Gaussian)

Next: Inference on Bayesian Networks

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 $P(outcome|action, evidence)$

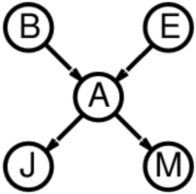
Value of information: which evidence to seek next?

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

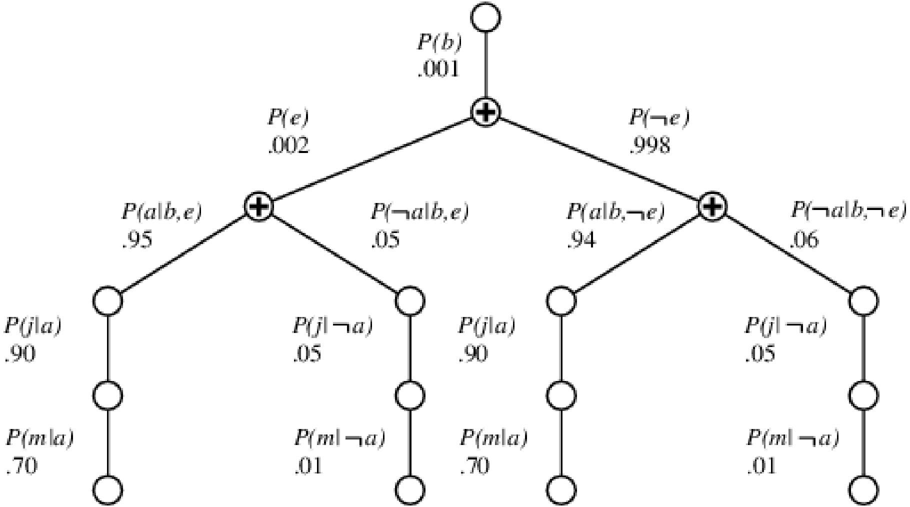
function ENUMERATION-ASK(X, $\mathbf{e}$, bn) **returns** a distribution over X
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

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

---

function ENUM-ALL($vars$, $\mathbf{e}$) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in $\mathbf{e}$
    then return $P(y \mid Pa(Y)) \times$ ENUM-ALL(REST($vars$), $\mathbf{e}$)
    else return $\sum_y P(y \mid Pa(Y)) \times$ ENUM-ALL(REST($vars$), $\mathbf{e}_y$)
      where $\mathbf{e}_y$ is $\mathbf{e}$ extended with $Y = y$

Evaluation Tree



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

Inference by Variable Elimination

Variable elimination refers to a heuristic to reduce complexity of exact inference

Use of memoization to avoid redundant calculations (stored in factors)

$$\mathbf{P}(B|j,m)$$
$$= \alpha \underbrace{\mathbf{P}(B)}_{B} \sum_e \underbrace{P(e)}_{E} \sum_a \underbrace{\mathbf{P}(a|B,e)}_{A} \underbrace{P(j|a)}_{J} \underbrace{P(m|a)}_{M}$$
$$= \alpha f_1(B) \sum_e f_2(E) \sum_a f_3(A,B,E) f_4(A) f_5(A)$$ pointwise product and sum out A
$$= \alpha f_1(B) \sum_e f_2(E) f_6(B,E)$$ sum out E
$$= \alpha f_1(B) f_7(B)$$

Basic operations: pointwise product and summation of factors

Direction: Carry out summations right-to-left

Example of factors:
$f_4(A)$ is $\langle P(j|a), P(j|\neg a)\rangle = \langle 0.90, 0.05\rangle$ $f_5(A)$ is $\langle P(m|a), P(m|\neg a)\rangle = \langle 0.70, 0.01\rangle$
$f_3(A,b,E)$ is a matrix of two rows, $\langle P(a|b,e), P(\neg a|b,e)\rangle$ and $\langle P(a|b,\neg e), P(\neg a|b,\neg e)\rangle$
$f_3(A,B,E)$ is a $2x2x2$ matrix (considering also b and $\neg b$).

Pointwise product $f_4(A) \times f_5(A) = \langle P(j|a) \cdot P(m|a), P(j|\neg a) \cdot P(m|\neg a) \rangle$

Corresponding entries in vectors are multiplied, yielding another same-size vector

equivalent to going bottom-up in tree, keeping track of both children in a vector, and multiplying child with parent to "roll up" to higher level.

Generally:
Pointwise product of factors f_1 and f_2:
$f_1(x_1, \ldots, x_j, y_1, \ldots, y_k) \times f_2(y_1, \ldots, y_k, z_1, \ldots, z_l)$
$= f(x_1, \ldots, x_j, y_1, \ldots, y_k, z_1, \ldots, z_l)$ vars are unions

Example: $f_1(a, b) \times f_2(b, c) = f(a, b, c)$

Rewrite $f_4(A)$ as $f(j, A)$ and $f_5(A)$ as $f(m, A)$
Rule suggests $f(j, A) \times f_2(m, A) = f(j, m, A)$

Correct: $P(j|A) \times P(m|A) = P(j, m|A)$ (because J and M are conditionally indepedent given their parent set A)

Consider $f_3(A, b, E)$ which is a 2x2 matrix:
$\langle P(a|b, e), P(\neg a|b, e)\rangle$
$\langle P(a|b, \neg e), P(\neg a|b, \neg e)\rangle$ (each row corresponds to branching point in search tree)

"Summing out" A means pointwise product on each branch and sum up at parent

Example: What is $= \sum_a f_3(A, b, E) f_4(A) f_5(A)$?

Let $f_4(A) \times f_5(A)$ be $f(j, m, A) =< P(j, m|a), P(j, m|\neg a >$ (from previous slide)
Take pointwise product of first row of $f_3(A, b, E)$ with $f(j, m, A)$
Take pointwise product of second row of $f_3(A, b, E)$ with $f(j, m, A)$
Sum the two rows to get a new factor $f_6(b, E)$

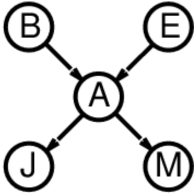
Generally, summing out a variable from a product of factors:
move any constant factors outside the summation
add up submatrices in pointwise product of remaining factors

$\sum_x f_1 \times \cdots \times f_k = f_1 \times \cdots \times f_i \sum_x f_{i+1} \times \cdots \times f_k = f_1 \times \cdots \times f_i \times f_{\bar{X}}$
assuming $f_1, \ldots, f_i$ do not depend on X
summation needed to account for all values of hidden variables (A, E)

```
function ELIMINATION-ASK(X, e, bn) returns a distribution over X
   inputs: X, the query variable
           e, evidence specified as an event
           bn, a belief network specifying joint distribution P(X_1, ..., X_n)

   factors ← [ ]; vars ← REVERSE(VARS[bn])
   for each var in vars do
        factors ← [MAKE-FACTOR(var, e)|factors]
        if var is a hidden variable then factors ← SUM-OUT(var, factors)
   return NORMALIZE(POINTWISE-PRODUCT(factors))
```

Every choice of ordering for variables yields a sound algorithm
Different orderings give different intermediate factors
Certain variable orderings can introduce irrelevant calculations
Intractable to find optimal ordering, but heuristics exist



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

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

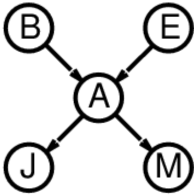
Sum over m is identically 1; M is **irrelevant** to the query

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

Hence the name, **variable elimination** algorithm

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

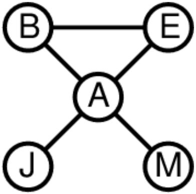
Thm 1: Y is irrelevant unless $Y \in Ancestors(\{X\} \cup \mathbf{E})$

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

Hence the name, **variable elimination** algorithm

Defn: moral graph of Bayes net: marry all parents and drop arrows

Defn: **A** is m-separated from **B** by **C** iff separated by **C** in the moral graph



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

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

Complexity of Exact Inference

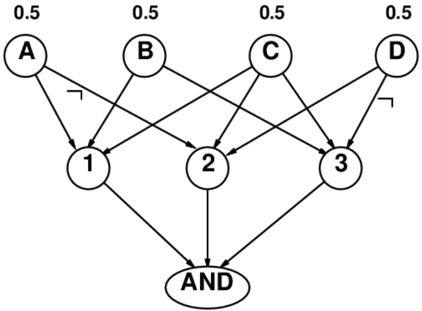
Singly connected networks (or polytrees):
– any two nodes are connected by at most one (undirected) path
– worst-case time and space cost of a query is $O(n)$
– worst-case time and space cost of n queries is $O(n^2)$

Multiply connected networks:
– worst-case time and space cost are exponential, $O(n \cdot d^n)$ (n queries, d values per r.v.)
– NP-hard and #P-complete
– can reduce 3SAT to exact inference $\Longrightarrow$ NP-hard
– equivalent to **counting** 3SAT models $\Longrightarrow$ #P-complete

1. A v B v C

2. C v D v ¬A

3. B v C v ¬D



Taming Exact Inference

How to reduce time? Identify structure in BN similar to CSP setting: group variables together to "reduce" network to a polytree

How? Cluster variables together (joint tree algorithms)

Parents of a node can be grouped into a meta-parent node (meganode)

As in CSP, meganodes may share variables, so special inference algorithm is needed

Algorithm takes care of constraint propagation so that meganodes agree on posterior probability of shared variables

No free lunch, so what gives?

The exponential time cost is hidden in the combined CPTs, which can become exponentially large

Taming Exact Inference

How to reduce time? Identify structure in BN similar to CSP setting: group variables together to "reduce" network to a polytree

How? Cluster variables together (joint tree algorithms)

Parents of a node can be grouped into a meta-parent node (meganode)

As in CSP, meganodes may share variables, so special inference algorithm is needed

Algorithm takes care of constraint propagation so that meganodes agree on posterior probability of shared variables

No free lunch, so what gives?

The exponential time cost is hidden in the combined CPTs, which can become exponentially large

Or...

Give up on exact inference

Giving up on Exact Inference: Go for Approximate Instead

Or...

Give up on exact inference

Go for approximate inference algorithms...

Giving up on Exact Inference: Go for Approximate Instead

Or...

Give up on exact inference

Go for approximate inference algorithms...

that use sampling (Monte Carlo-based) to estimate posterior probabilities

Or...

Give up on exact inference

Go for approximate inference algorithms...

that use sampling (Monte Carlo-based) to estimate posterior probabilities

Inference by Stochastic Simulation (Sampling-based)

Basic idea:

1) Draw N samples from a sampling distribution S
   Can you draw N samples for the r.v. Coin
   from the probability distribution $\mathrm{P(Coin)} = [0.5, 0.5]$?

2) Compute an approximate posterior probability $\hat{P}$

3) Show this converges to the true probability P

Outline:

– Direct Sampling: 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

Direct Sampling: Sampling from an Empty Network

Empty refers to the absence of any evidence: used to estimate joint probabilities

Main idea:

Sample each r.v. in turn, in topological order, from parents to children

Once parent is sampled, its value is fixed and used to sample child

Events generated via this direct sampling, observing joint probability distribution

To get (prior) probability of an event, have to sample many times, so frequency of "observing" it among samples approaches its probability

Example next

Direct Sampling: Sampling from an Empty Network

Empty refers to the absence of any evidence: used to estimate joint probabilities

Main idea:

Sample each r.v. in turn, in topological order, from parents to children

Once parent is sampled, its value is fixed and used to sample child

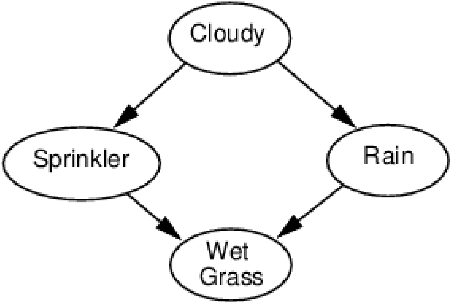
Events generated via this direct sampling, observing joint probability distribution

To get (prior) probability of an event, have to sample many times, so frequency of "observing" it among samples approaches its probability

Example next

Direct Sampling Example

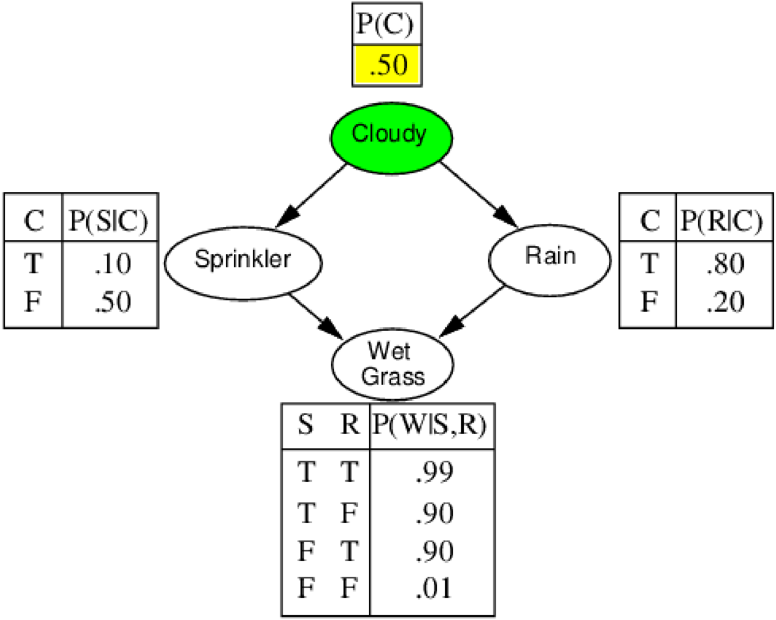
P(C)
.50



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

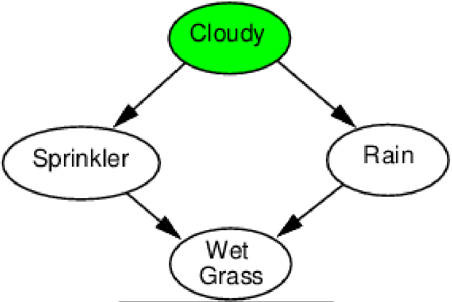
C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01


P(C)
.50
Cloudy
C P(S|C)
T .10
F .50
Sprinkler
Rain
C P(R|C)
T .80
F .20
Wet Grass
S R P(W|S,R)
T T .99
T F .90
F T .90
F F .01

Direct Sampling Example

P(C)
.50

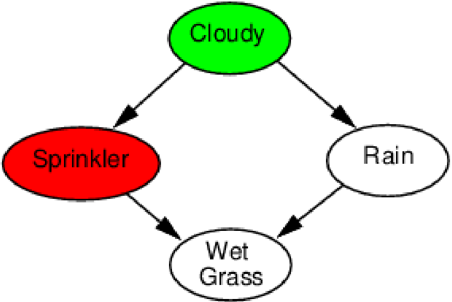


C	P(S|C)
T	.10
F	.50

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

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

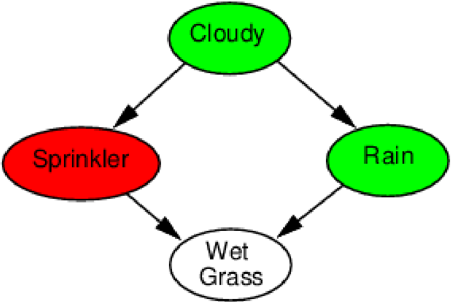
P(C)
.50



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

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

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01



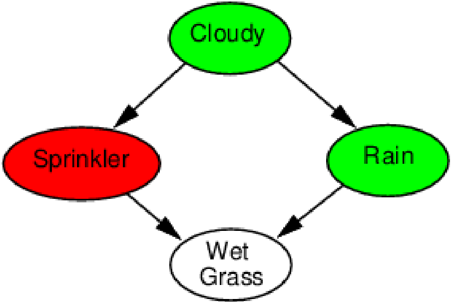
P(C)
.50

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

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

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

P(C)
.50

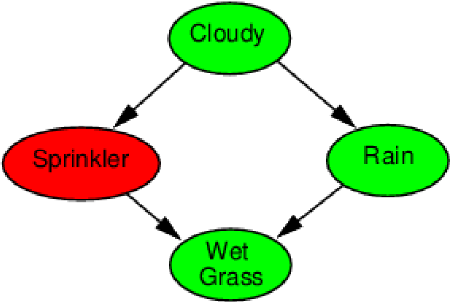


C	P(S|C)
T	.10
F	.50

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

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

P(C)
.50



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

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

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

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 $\mathbf{x}$

Direct Sampling Continued

Probability that PriorSample generates a particular event $x_1 \ldots x_n$:
$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i)) = P(x_1 \ldots x_n)$
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
(becomes exact in large-sample limit)

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

Problem: N needs to be sufficiently large to sample "rare events"

Rejection Sampling (for Conditional Probabilities $P(X|e)$)

Main idea:

Given distribution too hard to sample directly from it: use an easy-to-sample distribution for direct sampling, and then reject samples based on hard-to-sample distribution

(1) Direct sampling to sample (X, E) events from prior distribution in BN
(2) Determine whether (X, E) is consistent with given evidence e
(3) Get $\hat{\mathbf{P}}(X|E = e)$ by counting how often $(E = e)$ and $(X, E = e)$ occur
as per Bayes' rule: $\hat{\mathbf{P}}(X|E = e) = \frac{N(X,E=e)}{N(E=e)}$

Example: estimate $\mathbf{P}(Rain|Sprinkler = true)$ using 100 samples

Generate 100 samples for *Cloudy*, *Sprinkler*, *Rain*, *WetGrass* via direct sampling
27 samples have $Sprinkler = true$ event of interest
Of these, 8 have $Rain = true$ and 19 have $Rain = false$.

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

Similar to a basic real-world empirical estimation procedure

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

Analysis of Rejection Sampling

$\hat{\mathbf{P}}(X|\mathbf{e}) = \alpha \mathbf{N}_{PS}(X, \mathbf{e})$ (algorithm defn.)

$= \mathbf{N}_{PS}(X, \mathbf{e}) / N_{PS}(\mathbf{e})$ (normalized by $N_{PS}(\mathbf{e})$)

$\approx \mathbf{P}(X, \mathbf{e}) / P(\mathbf{e})$ (property of PriorSample)

$= \mathbf{P}(X|\mathbf{e})$ (defn. of conditional probability)

Hence rejection sampling returns consistent posterior estimates
Standard deviation of error in each probability proportional to $1/\sqrt{n}$ (number of r.v.s)

Problem:

If e is very rare event, most samples rejected; hopelessly expensive if $P(\mathbf{e})$ is small

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

Rejection sampling is unusable for complex problems $\rightarrow$ **Likelihood Weighting** instead

A form of **importance sampling** (for BNs)

Main idea:

Generate only events that are consistent with given values e of evidence variables E

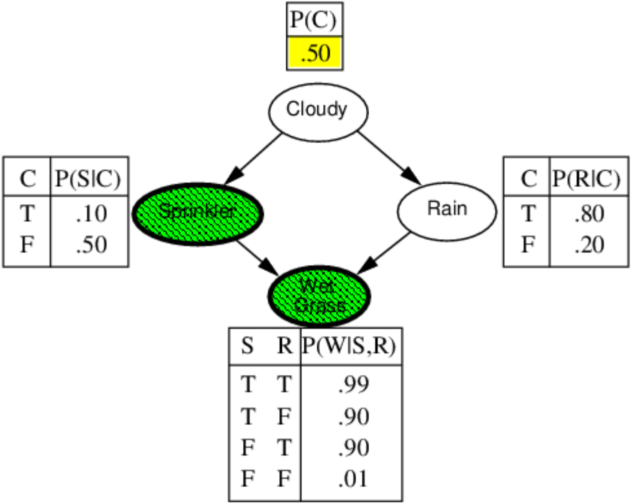
Fix evidence variables to given values, sample only nonevidence variables

Weight each sample by the likelihood it accords the evidence (how likely e is)

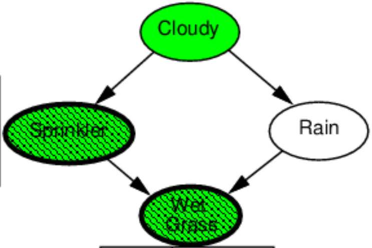
Example: Query $P(Rain|Cloudy = true, WetGrass = true)$

Consider r.v.s in some topological ordering
Set $w = 1.0$ (weight will be a running product)

If r.v. X_i is in given evidence variables (Cloudy or WetGrass in this example),
$w = w \times P(X_i|Parents(X_i))$
Else, sample X_i from $P(X_i|$ evidence)
When all r.v.s considered, normalized weights to turn to probabilities



Cloudy considered first, sample, $w = 1.0$ because nonevidence



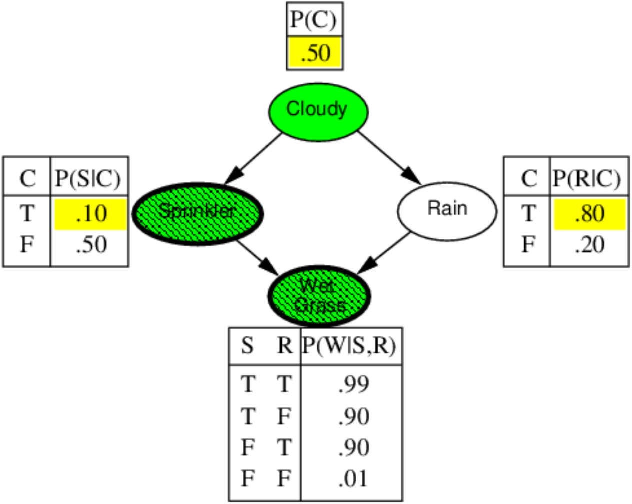
C	P(S|C)
T	.10
F	.50

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

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Cloudy considered first, sample, $w = 1.0$ because nonevidence
Say, Cloudy=T sampled

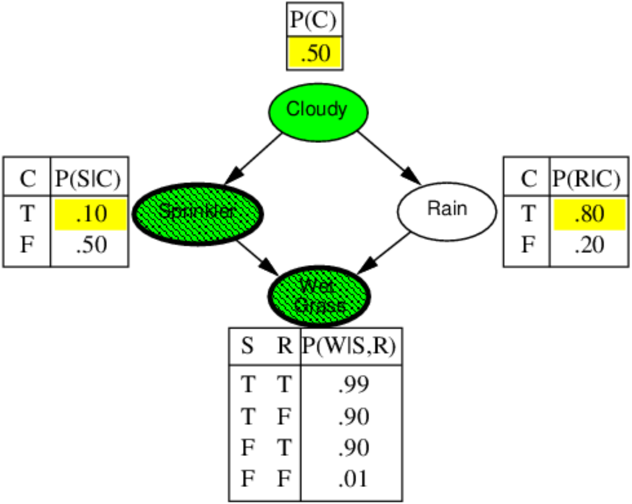
Likelihood Weighting Example: $P(Rain|Sprinkler = t, WetGrass = t)$



Sprinkler considered next, evidence variable, so update w

$w = w \times P(Sprinkler = t|Parents(Sprinkler))$

$w = 1.0$



P(C)
.50

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

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

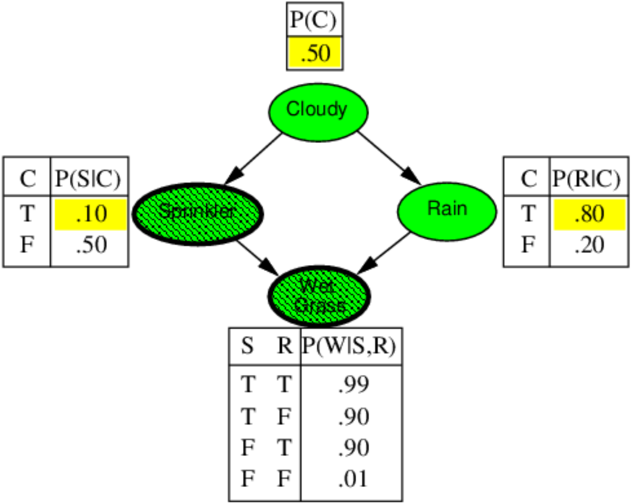
S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sprinkler considered next, evidence variable, so update w

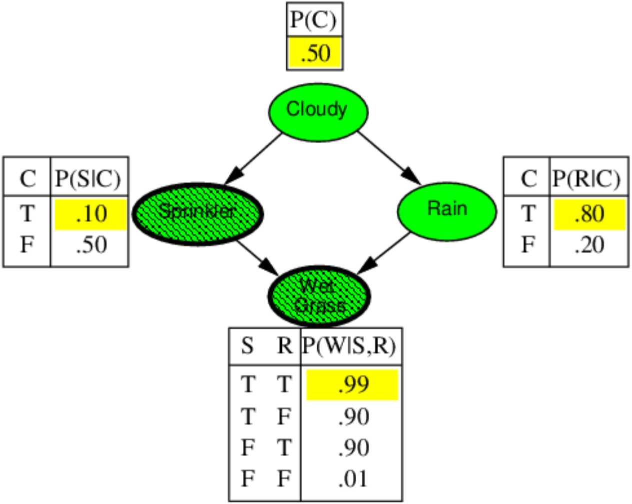
$w = w \times P(Sprinkler = t|Parents(Sprinkler)) = P(Sprinkler = t|Cloudy = t)$

$w = 1.0 \times 0.1$

Likelihood Weighting Example: $P(Rain|Sprinkler = t, WetGrass = t)$



Rain considered next, nonevidence, so sample from BN, w does not change
$w = 1.0 \times 0.1$



P(C)
.50

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

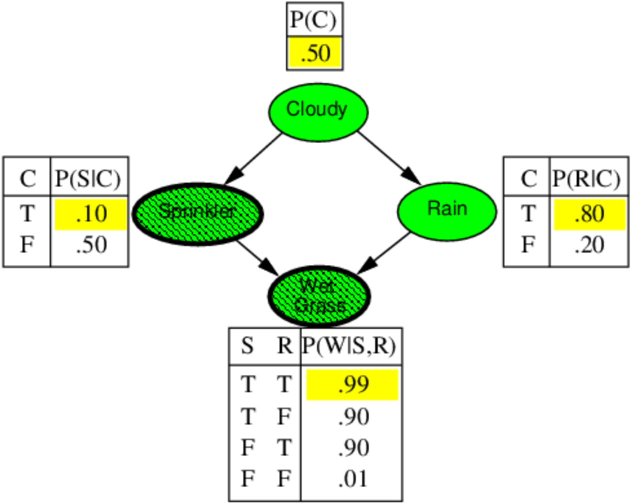
C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Sample Rain, note Cloudy $=$ t from before

Say, Rain $=$ t sampled

$w = 1.0 \times 0.1$



Last r.v. WetGrass, evidence variable, so update w

$w = w \times P(WetGrass = t | Parents(WetGrass)) = P(W = t | S = t, R = t)$

$w = 1.0 \times 0.1 \times 0.99 = 0.099$ (this is not probability but weight of *this* sample)

Likelihood Weighting Algorithm

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($\mathbf{W}[X]$)

---

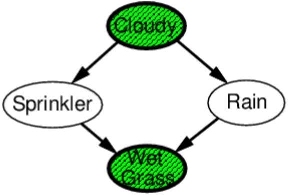
function WEIGHTED-SAMPLE(bn, $\mathbf{e}$) **returns** an event and a weight

  $\mathbf{x} \leftarrow$ 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

Likelihood Weighting 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



$\Longrightarrow$ 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:
$S_{WS}(\mathbf{z}, \mathbf{e}) w(\mathbf{z}, \mathbf{e})$
$= \prod_{i=1}^{l} P(z_i | parents(Z_i)) \prod_{i=1}^{m} P(e_i | parents(E_i))$
$= P(\mathbf{z}, \mathbf{e})$ (by standard global semantics of network)

Likelihood weighting returns consistent estimates

Order actually matters

Degradation in performance as number of evidence variables increases

A few samples have nearly all the total weight

Most samples will have very low weights, and weight estimate will be dominated by tiny fraction of samples that contribute little likelihood to evidence

Exacerbated when evidence variables occur late in the ordering

Nonevidence variables will have no evidence in their parents to guide generation of samples

Samples in simulations will bear little resemblance to reality suggested by evidence

Change framework: do not directly sample (from scratch), but modify preceding sample

Approximate Inference using MCMC

Main idea:

Markov Chain Monte Carlo (MCMC) algorithm(s) generate each sample by making a random change to a preceding sample

Concept of *current state*: specifies value for every r.v.

"State" of network = current assignment to all variables

Random change to *current state* yields *next state*

A form of MCMC: Gibbs Sampling

Initial state has evidence variables assigned as provided

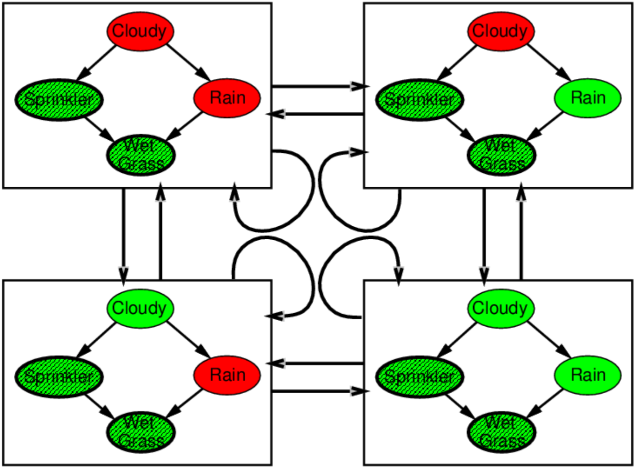
Next state generated by randomly sampling values for nonevidence variables

Each nonevidence variable Z sampled in turn, given its Markov blanket mb

```
function GIBBS-ASK(X, e, bn, N) returns an estimate of P(X|e)
    local variables: N[X], a vector of counts over X, initially zero
                     Z, nonevidence variables in bn
                     x, current state of network, initially copied from e

    initialize x with random values for the variables in Z
    for j = 1 to N do
        for each Z_i in Z do
            sample the value of Z_i in x from P(Z_i|mb(Z_i))
                given the values of MB(Z_i) in x
            N[x] ← N[x] + 1 where x is the value of X in x
    return NORMALIZE(N[X])
```

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



Wander about for a while, average what you see

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.

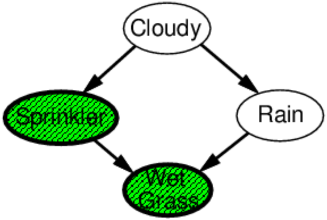
E.g., visit 100 states
31 have $Rain = true$, 69 have $Rain = false$
$\hat{\mathbf{P}}(Rain|Sprinkler = true, WetGrass = true)$
$= \textsc{Normalize}(\langle 31, 69\rangle) = \langle 0.31, 0.69\rangle$

Theorem: chain approaches stationary distribution

long-run fraction of time spent in each state is exactly
proportional to its posterior probability

Markov Blanket Sampling

Markov blanket of *Cloudy* is *Sprinkler* and *Rain*



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

Probability given the Markov blanket is calculated as follows:
$P(x_i'|mb(X_i)) = P(x_i'|parents(X_i)) \prod_{Z_j \in Children(X_i)} P(z_j|parents(Z_j))$
Easily implemented in message-passing parallel systems, brains

Main computational problems:
1) Difficult to tell if convergence has been achieved
2) Can be wasteful if Markov blanket is large:
$P(X_i|mb(X_i))$ won't change much (law of large numbers)

Summary on Inference on Bayesian Networks

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

For Those that Want to Dig Deeper...

- ♢ MCMC Analysis
- ♢ Stationarity
- ♢ Detailed Balance
- ♢ General Gibbs Sampling

MCMC Analysis: Outline

Transition probability $q(\mathbf{x} \to \mathbf{x}')$
Occupancy probability $\pi_t(\mathbf{x})$ at time t

Equilibrium condition on π_t defines stationary distribution $\pi(\mathbf{x})$
Note: stationary distribution depends on choice of $q(\mathbf{x} \to \mathbf{x}')$

Pairwise detailed balance on states guarantees equilibrium

Gibbs sampling transition probability:
sample each variable given current values of all others
$\Longrightarrow$ detailed balance with the true posterior

For Bayesian networks, Gibbs sampling reduces to
sampling conditioned on each variable's Markov blanket

Stationary Distribution

$\pi_t(\mathbf{x})$ = probability in state $\mathbf{x}$ at time t

$\pi_{t+1}(\mathbf{x}')$ = probability in state $\mathbf{x}'$ at time $t+1$

π_{t+1} in terms of π_t and $q(\mathbf{x} \to \mathbf{x}')$

$$\pi_{t+1}(\mathbf{x}') = \sum_{\mathbf{x}} \pi_t(\mathbf{x}) q(\mathbf{x} \to \mathbf{x}')$$

Stationary distribution: $\pi_t = \pi_{t+1} = \pi$

$$\pi(\mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \to \mathbf{x}') \qquad \text{for all } \mathbf{x}'$$

If π exists, it is unique (specific to $q(\mathbf{x} \to \mathbf{x}')$)

In equilibrium, expected "outflow" = expected "inflow"

Detailed Balance

"Outflow" = "inflow" for each pair of states:

$$\pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') = \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x}) \qquad \text{for all } \mathbf{x},\ \mathbf{x}'$$

Detailed balance $\Longrightarrow$ stationarity:

$$\begin{aligned}\sum_{\mathbf{x}} \pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') &= \sum_{\mathbf{x}} \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x}) \\ &= \pi(\mathbf{x}')\sum_{\mathbf{x}} q(\mathbf{x}' \to \mathbf{x}) \\ &= \pi(\mathbf{x}')\end{aligned}$$

MCMC algorithms typically constructed by designing a transition probability q that is in detailed balance with desired π

Gibbs Sampling

Sample each variable in turn, given **all other variables**

Sampling X_i, let $\bar{\mathbf{X}}_i$ be all other nonevidence variables
Current values are x_i and $\bar{\mathbf{x}}_i$; $\mathbf{e}$ is fixed

Transition probability is given by

$$q(\mathbf{x} \to \mathbf{x}') = q(x_i, \bar{\mathbf{x}}_i \to x_i', \bar{\mathbf{x}}_i) = P(x_i' | \bar{\mathbf{x}}_i, \mathbf{e})$$

This gives detailed balance with true posterior $P(\mathbf{x}|\mathbf{e})$:

$$\begin{aligned}
\pi(\mathbf{x})q(\mathbf{x} \to \mathbf{x}') &= P(\mathbf{x}|\mathbf{e})P(x_i'|\bar{\mathbf{x}}_i, \mathbf{e}) = P(x_i, \bar{\mathbf{x}}_i|\mathbf{e})P(x_i'|\bar{\mathbf{x}}_i, \mathbf{e}) \\
&= P(x_i|\bar{\mathbf{x}}_i, \mathbf{e})P(\bar{\mathbf{x}}_i|\mathbf{e})P(x_i'|\bar{\mathbf{x}}_i, \mathbf{e}) \quad \text{(chain rule)} \\
&= P(x_i|\bar{\mathbf{x}}_i, \mathbf{e})P(x_i', \bar{\mathbf{x}}_i|\mathbf{e}) \qquad \text{(chain rule backwards)} \\
&= q(\mathbf{x}' \to \mathbf{x})\pi(\mathbf{x}') = \pi(\mathbf{x}')q(\mathbf{x}' \to \mathbf{x})
\end{aligned}$$

Performance of Approximation Algorithms

Absolute approximation: $|P(X|\mathbf{e}) - \hat{P}(X|\mathbf{e})| \leq \epsilon$

Relative approximation: $\frac{|P(X|\mathbf{e}) - \hat{P}(X|\mathbf{e})|}{P(X|\mathbf{e})} \leq \epsilon$

Relative $\implies$ absolute since $0 \leq P \leq 1$ (may be $O(2^{-n})$)

Randomized algorithms may fail with probability at most δ

Polytime approximation: $\text{poly}(n, \epsilon^{-1}, \log \delta^{-1})$

Theorem (Dagum and Luby, 1993): both absolute and relative approximation for either deterministic or randomized algorithms are NP-hard for any $\epsilon, \delta < 0.5$

(Absolute approximation polytime with no evidence—Chernoff bounds)