

Introduction to Artificial Intelligence

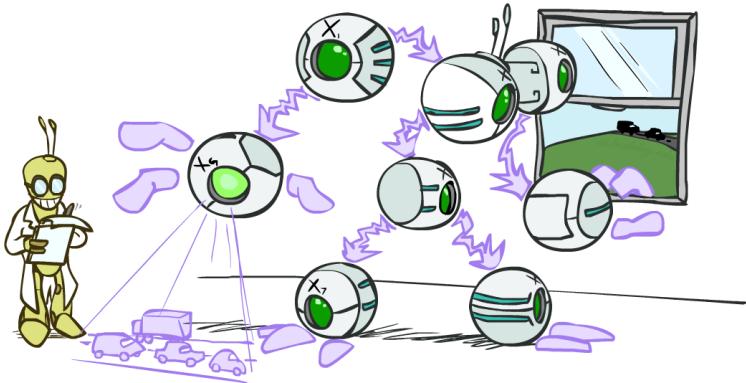
Lecture 6: Inference in Bayesian networks

Prof. Gilles Louppe
g.louppe@uliege.be



Today

- Exact inference
 - Inference by enumeration
 - Inference by variable elimination
- Continuous variables
- Approximate inference
 - Ancestral sampling
 - Rejection sampling
 - Likelihood weighting
 - Gibbs sampling



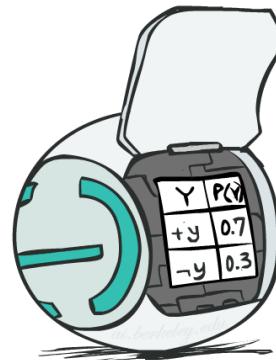
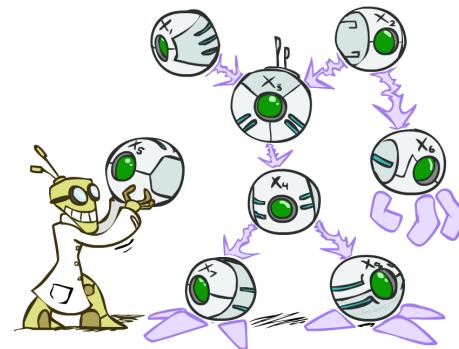
Bayesian networks

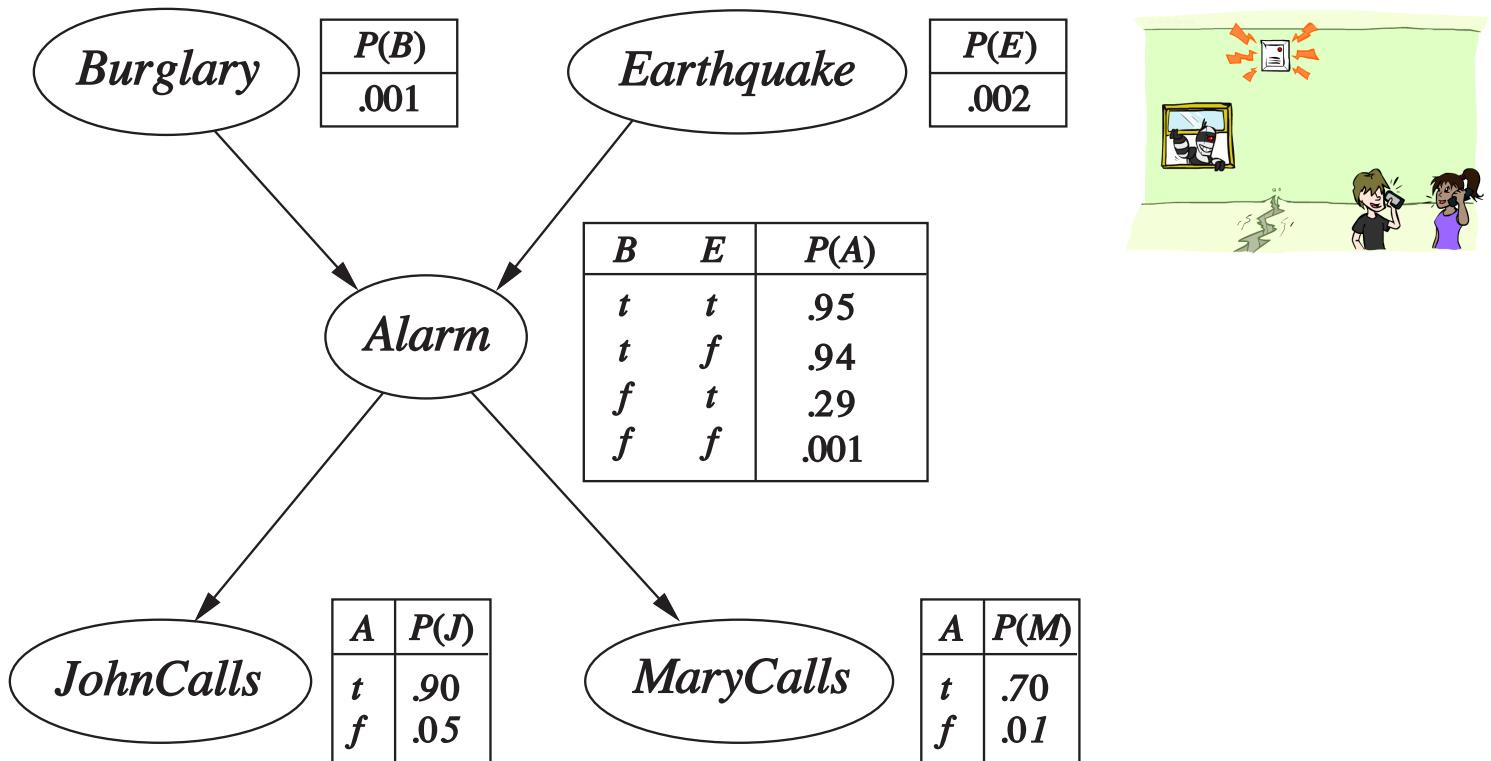
A Bayesian network is a **directed acyclic graph** in which:

- Each node corresponds to a **random variable** X_i .
- Each node X_i is annotated with a **conditional probability distribution** $P(X_i|\text{parents}(X_i))$ that quantifies the effect of the parents on the node.

A Bayesian network implicitly **encodes** the full joint distribution as the product of the local distributions:

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





$$\begin{aligned}
 P(b, \neg e, a, \neg j, m) &= P(b)P(\neg e)P(a|b, \neg e)P(\neg j|a)P(m, a) \\
 &= 0.001 \times 0.998 \times 0.94 \times 0.1 \times 0.7
 \end{aligned}$$

Exact inference

Inference

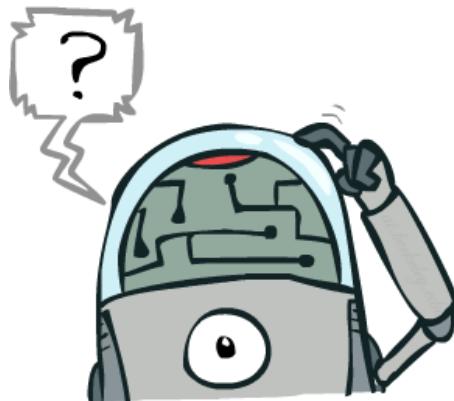
Inference is concerned with the problem **computing a marginal and/or conditional probability** from a joint probability distribution:

Simple queries: $P(X_i|e)$

Conjunctive queries: $P(X_i, X_j|e) = P(X_i|e)P(X_j|X_i, e)$

Most likely explanation: $\arg \max_q P(q|e)$

Optimal decisions: $\arg \max_a \mathbb{E}_{p(s'|s,a)} [V(s')]$



Inference by enumeration

Start from the joint distribution $P(Q, E_1, \dots, E_k, H_1, \dots, H_r)$.

1. Select the entries consistent with the evidence $E_1, \dots, E_k = e_1, \dots, e_k$.
2. Marginalize out the hidden variables to obtain the joint of the query and the evidence variables:

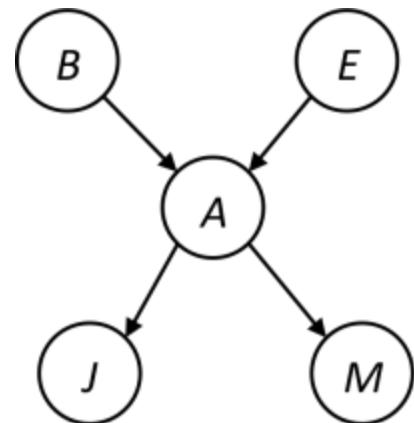
$$P(Q, e_1, \dots, e_k) = \sum_{h_1, \dots, h_r} P(Q, h_1, \dots, h_r, e_1, \dots, e_k).$$

3. Normalize:

$$\begin{aligned} Z &= \sum_q P(q, e_1, \dots, e_k) \\ P(Q|e_1, \dots, e_k) &= \frac{1}{Z} P(Q, e_1, \dots, e_k) \end{aligned}$$

Consider the alarm network and the query $P(B|j, m)$:

$$\begin{aligned} P(B|j, m) &= \frac{1}{Z} \sum_e \sum_a P(B, j, m, e, a) \\ &\propto \sum_e \sum_a P(B, j, m, e, a) \end{aligned}$$



Using the Bayesian network, the full joint entries can be rewritten as the product of CPT entries:

$$\begin{aligned} P(B|j, m) &\propto \sum_e \sum_a P(B)P(e)P(a|B, e)P(j|a)P(m|a) \\ &\propto P(B) \sum_e P(e) \sum_a P(a|B, e)P(j|a)P(m|a) \end{aligned}$$

Same complexity as DFS: $O(n)$ in space, $O(d^n)$ in 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 Bayes net with variables $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$ /* $\mathbf{Y} = \text{hidden variables}$ */

$\mathbf{Q}(X) \leftarrow$ a distribution over X , initially empty

for each value x_i of X **do**

$\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL($bn.\text{VARS}, \mathbf{e}_{x_i}$)

where \mathbf{e}_{x_i} is \mathbf{e} extended with $X = x_i$

return NORMALIZE($\mathbf{Q}(X)$)

function ENUMERATE-ALL($vars, \mathbf{e}$) **returns** a real number

if EMPTY?($vars$) **then return** 1.0

$Y \leftarrow \text{FIRST}(vars)$

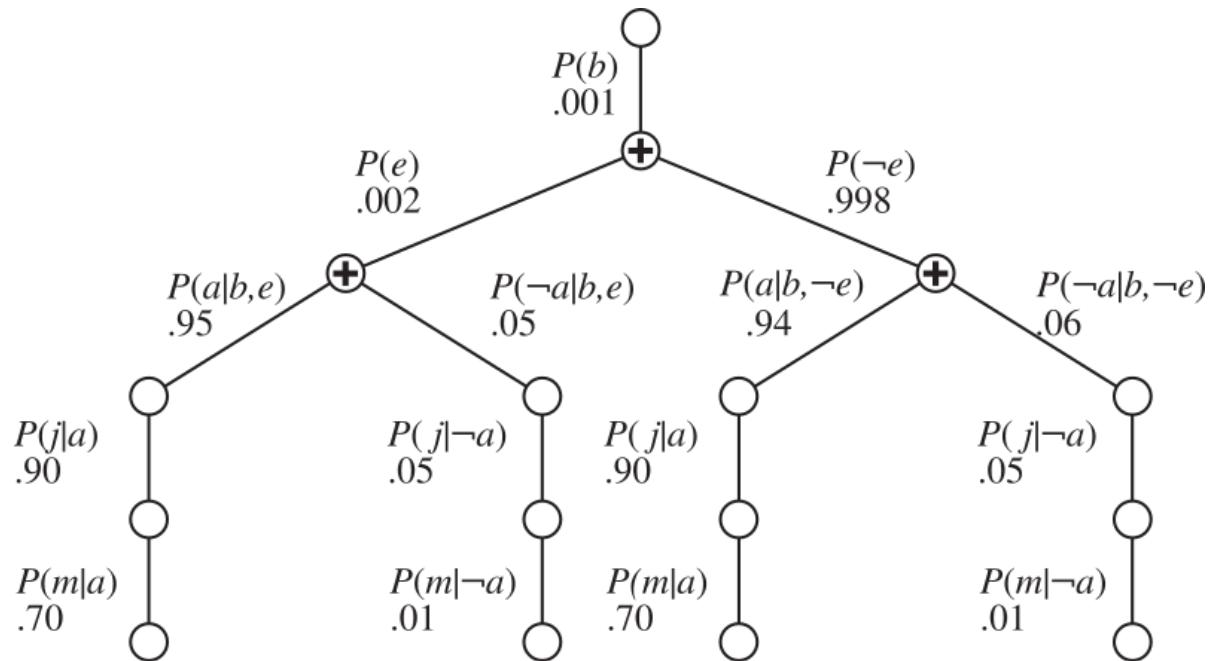
if Y has value y in \mathbf{e}

then return $P(y | parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), \mathbf{e})

else return $\sum_y P(y | parents(Y)) \times$ ENUMERATE-ALL(REST($vars$), \mathbf{e}_y)

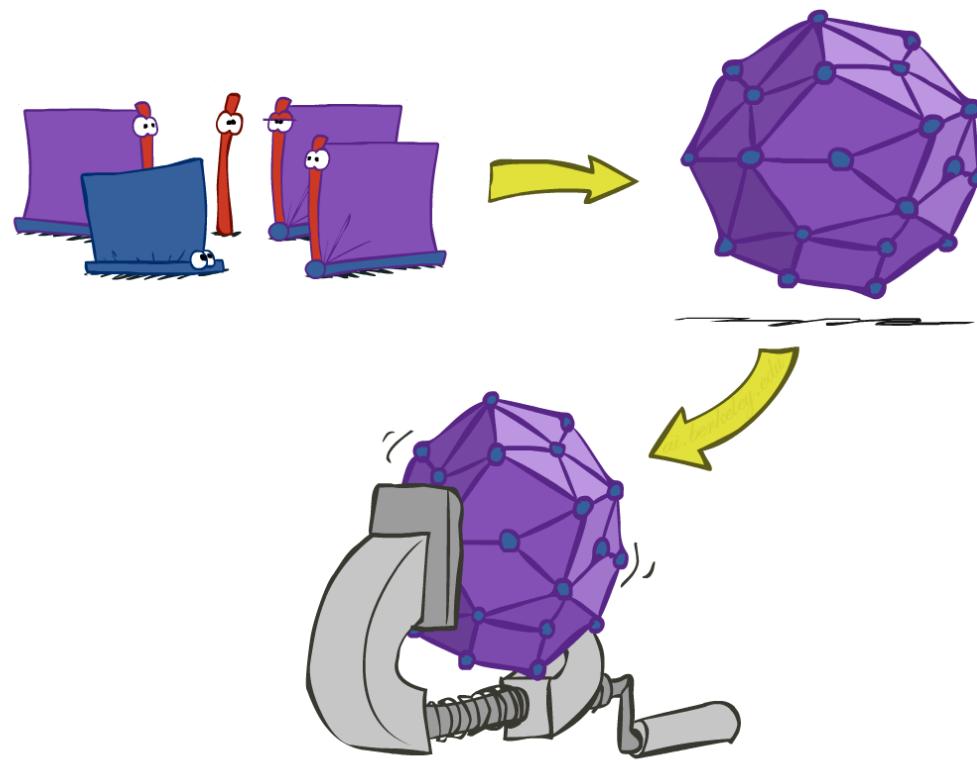
where \mathbf{e}_y is \mathbf{e} extended with $Y = y$

Evaluation tree for $P(b|j, m)$



Enumeration is **inefficient**: there are repeated computations!

- e.g., $P(j|a)P(m|a)$ is computed twice, once for e and once for $\neg e$.
- These can be avoided by **storing intermediate results**.

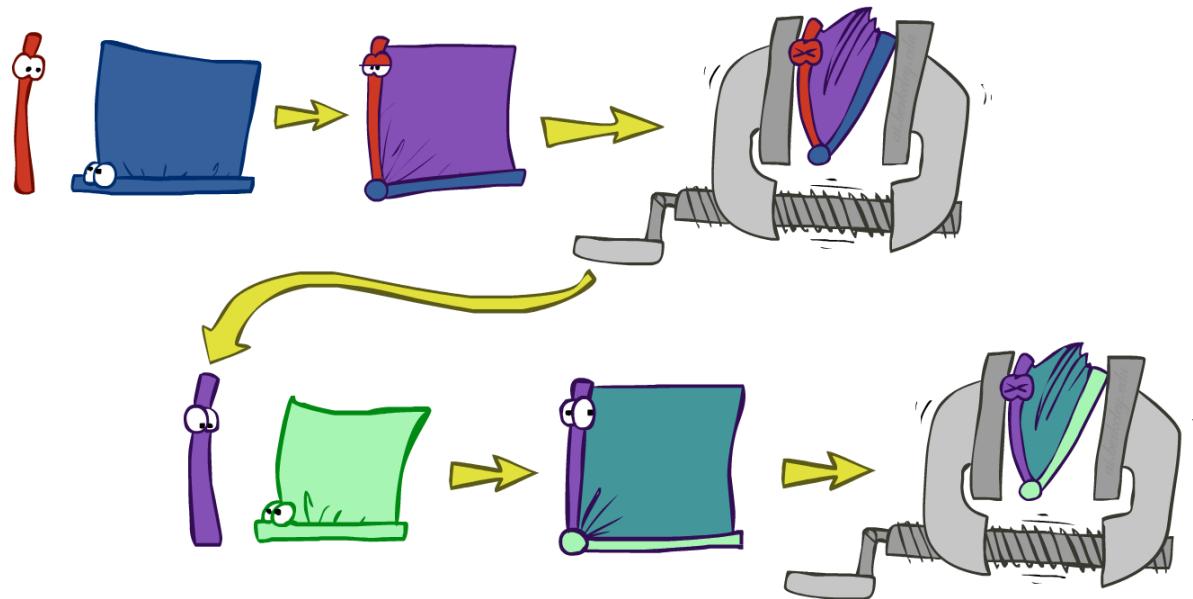


Inference by enumeration is slow because the whole joint distribution is joined up before summing out the hidden variables.

Inference by variable elimination

The **variable elimination** (VE) algorithm carries out summations right-to-left and stores intermediate results (called **factors**) to avoid recomputations. The algorithm interleaves:

- Joining sub-tables
- Eliminating hidden variables



Example

$$\begin{aligned} P(B|j, m) &\propto P(B, j, m) \\ &= P(B) \sum_e P(e) \sum_a P(a|B, e) P(j|a) P(m|a) \\ &= \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(e) \times \sum_a \mathbf{f}_3(a, B, e) \times \mathbf{f}_4(a) \times \mathbf{f}_5(a) \\ &= \mathbf{f}_1(B) \times \sum_e \mathbf{f}_2(e) \times \mathbf{f}_6(B, e) \quad (\text{sum out } A) \\ &= \mathbf{f}_1(B) \times \mathbf{f}_7(B) \quad (\text{sum out } E) \end{aligned}$$

Factors

- Each factor \mathbf{f}_i is a tensor indexed by the values of its argument variables. E.g.:

$$\mathbf{f}_4 = \mathbf{f}_4(A) = \begin{pmatrix} P(j|a) \\ P(j|\neg a) \end{pmatrix} = \begin{pmatrix} 0.90 \\ 0.05 \end{pmatrix}$$

$$\mathbf{f}_4(a) = 0.90$$

$$\mathbf{f}_4(\neg a) = 0.5$$

- Factors are initialized with the CPTs annotating the nodes of the Bayesian network, conditioned on the evidence.

Join

The pointwise product \times , or **join**, of two factors \mathbf{f}_1 and \mathbf{f}_2 yields a new factor \mathbf{f} .

- Exactly like a **database join**!
- The variables of \mathbf{f} are the **union** of the variables in \mathbf{f}_1 and \mathbf{f}_2 .
- The elements of \mathbf{f} are given by the product of the corresponding elements in \mathbf{f}_1 and \mathbf{f}_2 .

A	B	$\mathbf{f}_1(A, B)$	B	C	$\mathbf{f}_2(B, C)$	A	B	C	$\mathbf{f}_3(A, B, C)$
T	T	.3	T	T	.2	T	T	T	$.3 \times .2 = .06$
T	F	.7	T	F	.8	T	T	F	$.3 \times .8 = .24$
F	T	.9	F	T	.6	T	F	T	$.7 \times .6 = .42$
F	F	.1	F	F	.4	T	F	F	$.7 \times .4 = .28$
						F	T	T	$.9 \times .2 = .18$
						F	T	F	$.9 \times .8 = .72$
						F	F	T	$.1 \times .6 = .06$
						F	F	F	$.1 \times .4 = .04$

Figure 14.10 Illustrating pointwise multiplication: $\mathbf{f}_1(A, B) \times \mathbf{f}_2(B, C) = \mathbf{f}_3(A, B, C)$.

Elimination

Summing out, or eliminating, a variable from a factor is done by adding up the submatrices formed by fixing the variable to each of its values in turn.

For example, to sum out A from $\mathbf{f}_3(A, B, C)$, we write:

$$\begin{aligned}\mathbf{f}(B, C) &= \sum_a \mathbf{f}_3(a, B, C) = \mathbf{f}_3(a, B, C) + \mathbf{f}_3(\neg a, B, C) \\ &= \begin{pmatrix} 0.06 & 0.24 \\ 0.42 & 0.28 \end{pmatrix} + \begin{pmatrix} 0.18 & 0.72 \\ 0.06 & 0.04 \end{pmatrix} = \begin{pmatrix} 0.24 & 0.96 \\ 0.48 & 0.32 \end{pmatrix}\end{aligned}$$

General Variable Elimination algorithm

- Query: $P(Q|e_1, \dots, e_k)$.
- Start with the initial factors.
 - The local CPTs, instantiated by the evidence.
- While there are still hidden variables:
 - Pick a hidden variable H
 - Join all factors mentioning H
 - Eliminate H
- Join all remaining factors
- Normalize

Example

(blackboard example)

Relevance

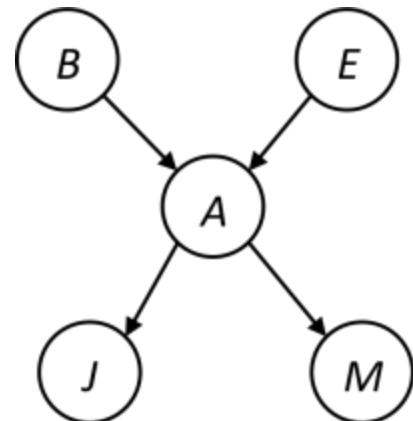
Consider the query $P(J|b)$:

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

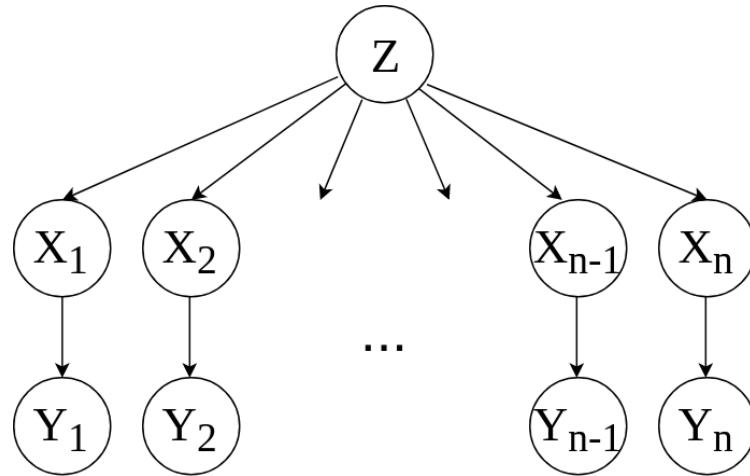
- $\sum_m P(m|a) = 1$, therefore M is **irrelevant** for the query.
- In other words, $P(J|b)$ remains unchanged if we remove M from the network.

Theorem

H is irrelevant for $P(Q|e)$ unless
 $H \in \text{ancestors}(\{Q\} \cup E)$.



Complexity



Consider the query $P(X_n | y_1, \dots, y_n)$. Work through the two elimination orderings:

- Z, X_1, \dots, X_{n-1}
- X_1, \dots, X_{n-1}, Z

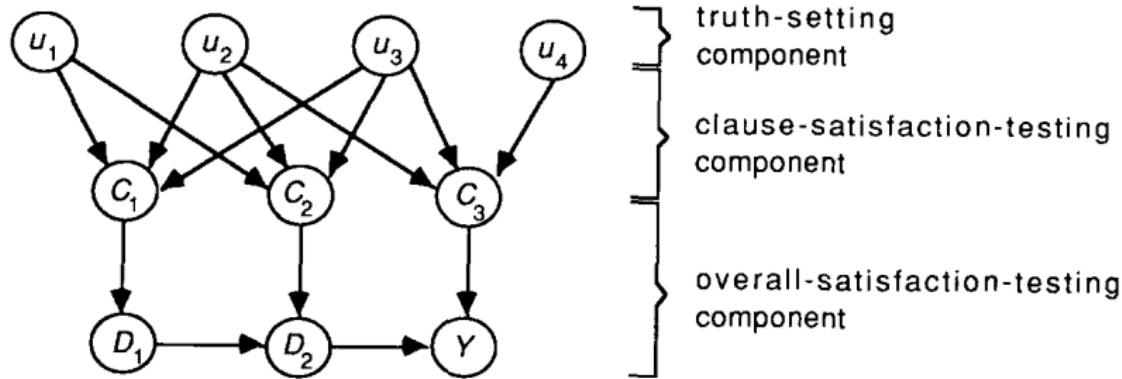
What is the size of the maximum factor generated for each of the orderings?

- Answer: 2^{n+1} vs. 2^2 (assuming boolean values)

The computational and space complexity of variable elimination is determined by the largest factor.

- The elimination ordering can greatly affect the size of the largest factor.
- Does there always exist an ordering that only results in small factors? No!
- Singly connected networks (polytrees):
 - Any two nodes are connected by at most one (undirected path).
 - For these networks, time and space complexity of variable elimination are $O(nd^k)$.

Worst-case complexity?



3SAT is a special case of inference:

- CSP: $(u_1 \vee u_2 \vee u_3) \wedge (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_2 \vee \neg u_3 \vee u_4)$
- $P(U_i = 0) = P(U_i = 1) = 0.5$
- $C_1 = U_1 \vee U_2 \vee U_3; C_2 = \neg U_1 \vee \neg U_2 \vee U_3; C_3 = U_2 \vee \neg U_3 \vee U_4$
- $D_1 = C_1; D_2 = D_1 \wedge C_2$
- $Y = D_2 \wedge C_3$

If we can answer whether $P(Y = 1) > 0$, then we answer whether 3SAT has a solution.

- By reduction, inference in Bayesian networks is therefore **NP-hard**.
- There is no known efficient probabilistic inference algorithm in general.

Continuous variables

Random variables

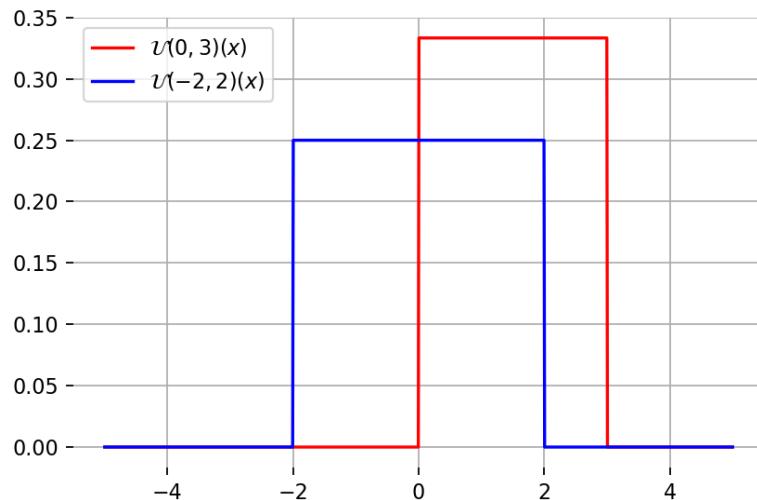
Let $X : \Omega \rightarrow D_X$ be a random variable.

- When D_X is finite or countably infinite, X is called a discrete random variable.
- Its probability distribution is described by a probability mass function that assigns a probability to each value $x \in D_X$.
- When D_X is uncountably infinite (e.g., $D_X = \mathbb{R}$), X is called a continuous random variable.
- If X is absolutely continuous, its probability distribution is described by a density function p that assigns a probability to any interval $[a, b] \subseteq D_X$ such that

$$P(a < X \leq b) = \int_a^b p(x)dx,$$

where p is non-negative piecewise continuous and such that $\int_{D_X} p(x)dx = 1$.

Uniform

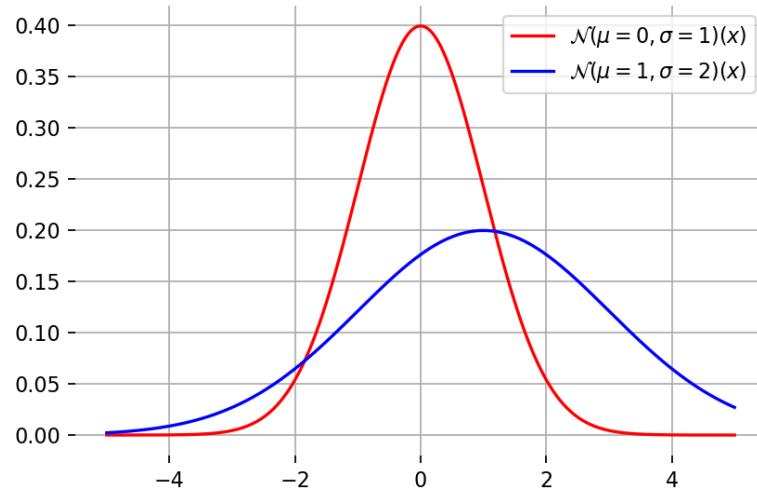


The uniform distribution $\mathcal{U}(a, b)$ is described by the density function

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

where $a \in \mathbb{R}$ and $b \in \mathbb{R}$ are the bounds of its support.

Normal

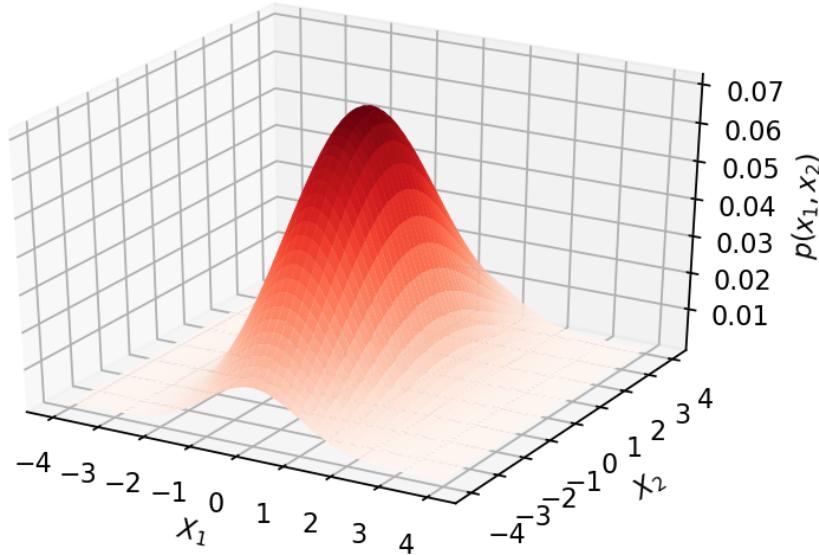


The normal (or Gaussian) distribution $\mathcal{N}(\mu, \sigma)$ is described by the density function

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$ are its mean and standard deviation parameters.

Multivariate normal



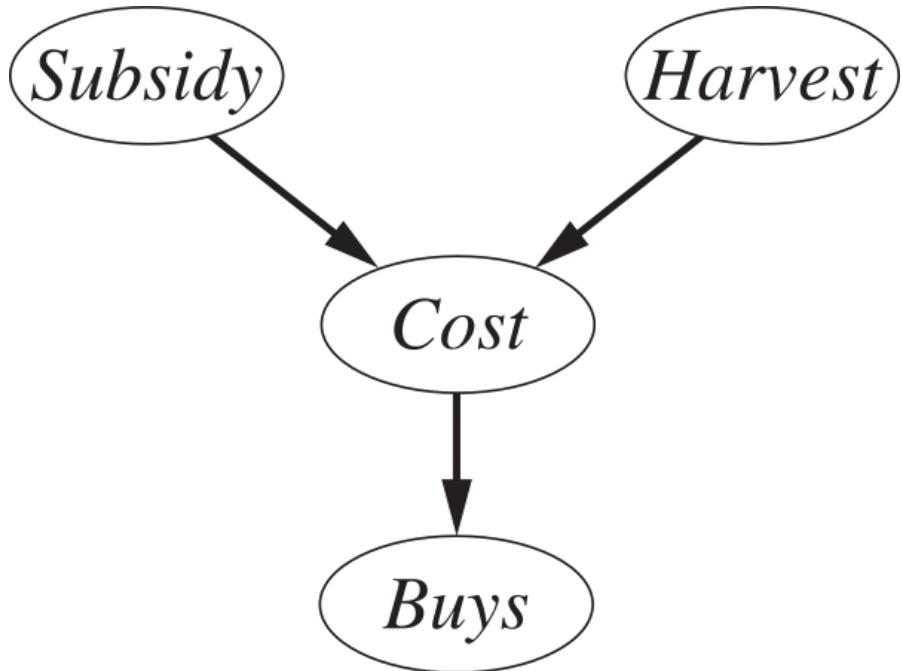
The multivariate normal distribution generalizes to N random variables. Its (joint) density function is defined as

$$p(\mathbf{x} = x_1, \dots, x_n) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

where $\boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ is positive semi-definite.

- The (multivariate) Normal density is the only density for real random variables that is **closed under marginalization and multiplication**.
- Also, a linear (or affine) function of a Normal random variable is Normal; and, a sum of Normal variables is Normal.
- For these reasons, the algorithms we will discuss will be tractable only for finite random variables or Normal random variables.

Hybrid Bayesian networks



What if we have both **discrete** (e.g., **Subsidy** and **Buys**) and **continuous** variables (e.g., **Harvest** and **Cost**) in a same Bayesian network?

Options

- Discretization: transform continuous variables into discrete variables.
 - Issues: possibly large errors due to precision loss, large CPTs.
- Define the conditional distribution with a **finitely parameterized** canonical distribution.
 - e.g., assume it is a Gaussian distribution.
- Use a non-parametric representation.

Continuous child variables

- We need to specify a conditional density function for each continuous child variable given continuous parents, for each possible assignment to discrete parents.
 - e.g., we need to specify both $p(c|h, s)$ and $p(c|h, \neg s)$
- Common choice: the **linear Gaussian model** (LG):
 - $p(c|h, s) = \mathcal{N}(a_t h + b_t, \sigma_t^2)(c)$
 - $p(c|h, \neg s) = \mathcal{N}(a_f h + b_f, \sigma_f^2)(c)$

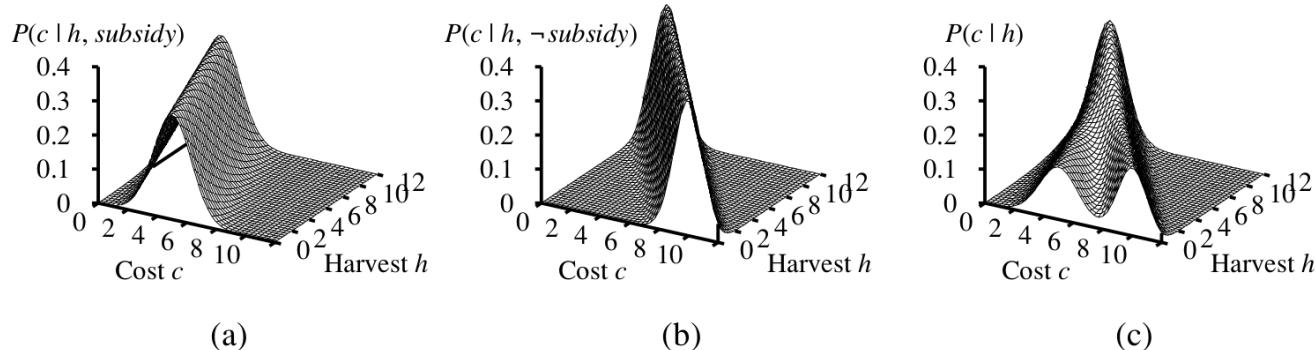


Figure 14.6 The graphs in (a) and (b) show the probability distribution over *Cost* as a function of *Harvest* size, with *Subsidy* true and false, respectively. Graph (c) shows the distribution $P(\text{Cost} | \text{Harvest})$, obtained by summing over the two subsidy cases.

Conditional Gaussian network

- The joint distribution of an all-continuous network with linear Gaussian distributions is a multivariate Gaussian.
- The joint distribution of a network with discrete or linear Gaussian continuous variables is a **conditional Gaussian network**.
 - i.e., a multivariate Gaussian over all continuous variables for each combination of the discrete variable values.

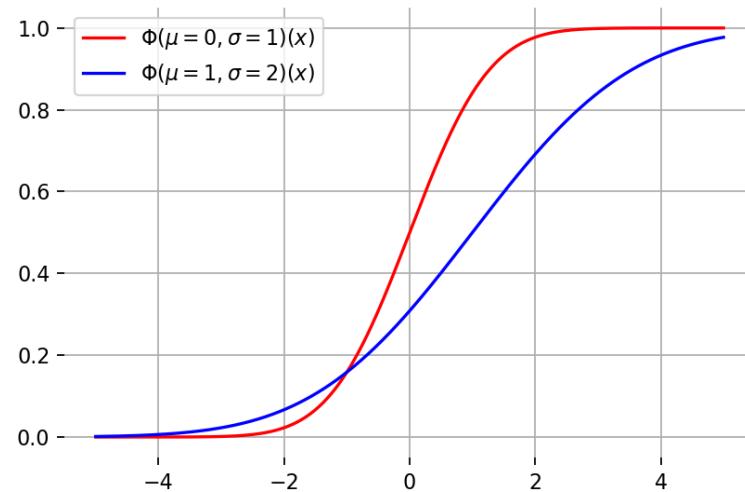
Discrete child variables, with continuous parents

- We need to specify a conditional distribution for each discrete child variable, given continuous parents.
- It is often reasonable to assume that the probability values of the discrete outcomes are almost piece-wise constant but **vary smoothly in intermediate regions**.

For example, if B is binary, $P(b|c)$ could be a "soft" threshold, such as the **probit distribution** for which

$$P(b|c) = \Phi((c - \mu)/\sigma),$$

where Φ is the cumulative distribution function of the (standard) normal distribution.



Variable elimination

Variable elimination in hybrid Bayesian networks can be conducted similarly as in the discrete case, by replacing **summations with integrations**.

- Exact inference remains possible **under some assumptions**.
 - e.g., for linear Gaussian models, queries can all be derived analytically.
- However, this often **does not scale** to arbitrary continuous distributions.
 - e.g., numerical approximations of integrals amount to discretize continuous variables.

Approximate inference

a.k.a. Monte Carlo methods

Exact inference is **intractable** for most probabilistic models of practical interest.
(e.g., involving many variables, continuous and discrete, undirected cycles, etc).

Solution

Abandon exact inference and develop **approximate** but **faster** inference algorithms:

- **Sampling methods**: produce answers by repeatedly generating random numbers from a distribution of interest.
- **Variational methods**: formulate inference as an optimization problem.
- **Belief propagation methods**: formulate inference as a message-passing algorithm.

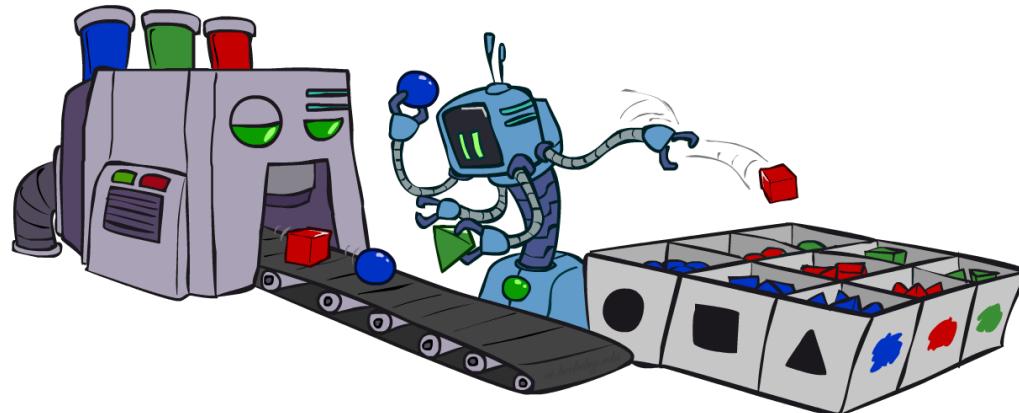
Sampling methods

Basic idea:

- Draw N samples from a sampling distribution S .
- Compute an approximate posterior probability \hat{P} .
- Show this approximate converges to the true probability distribution P .

Why sampling?

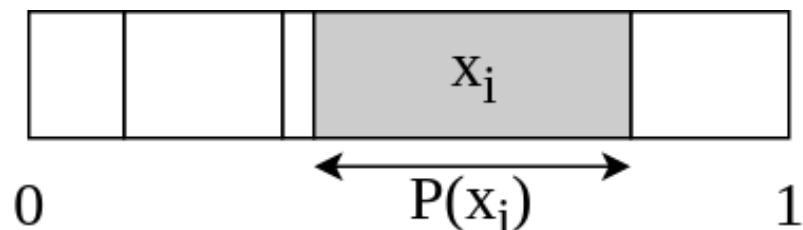
Generating samples is often much faster than computing the right answer (e.g., with variable elimination).



Sampling

How to sample from the distribution of a discrete variable X ?

- Assume k discrete outcomes x_1, \dots, x_k with probability $P(x_i)$.
- Assume sampling from $\mathcal{U}(0, 1)$ is possible.
 - e.g., as enabled by a standard `rand()` function.
- Divide the $[0, 1]$ interval into k regions, with region i having size $P(x_i)$.
- Sample $u \sim \mathcal{U}(0, 1)$ and return the value associated to the region in which u falls.



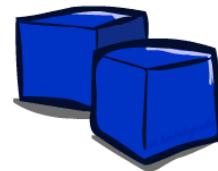
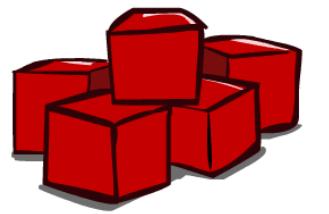
$P(C)$

C	P
red	0.6
green	0.1
blue	0.3

$0 \leq u < 0.6 \rightarrow C = \text{red}$

$0.6 \leq u < 0.7 \rightarrow C = \text{green}$

$0.7 \leq u < 1 \rightarrow C = \text{blue}$



The same algorithm extends to continuous variables, assuming access to the **inverse cumulative distribution function** F^{-1} .

- for $u \in [0, 1]$, $F^{-1}(u) = b$ such that $F(b) = u$, where F is the cumulative distribution function

$$F(b) = P(X < b) = \int_0^b p(x)dx.$$

- F^{-1} is known analytically for most canonical distributions.

[Q] How to extend to arbitrary multivariate distributions?

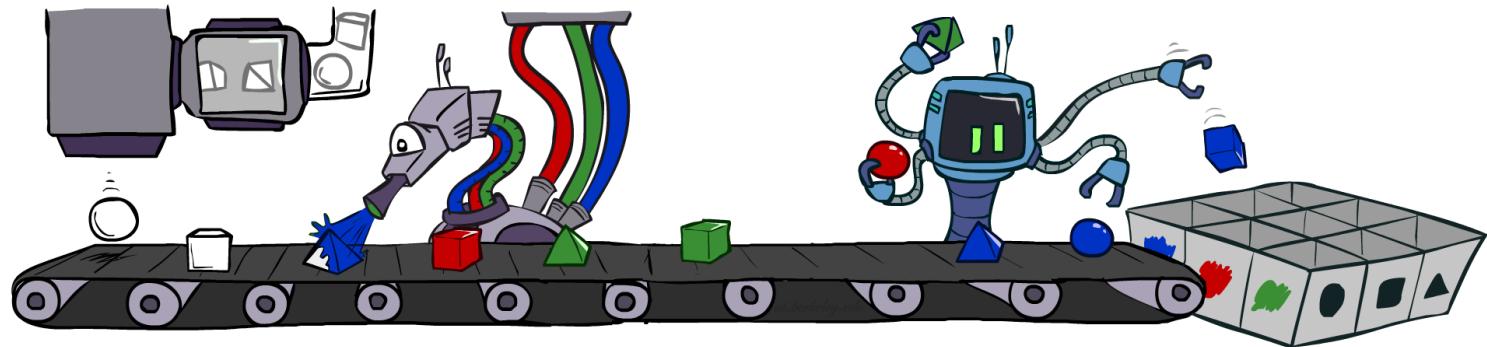
Prior sampling

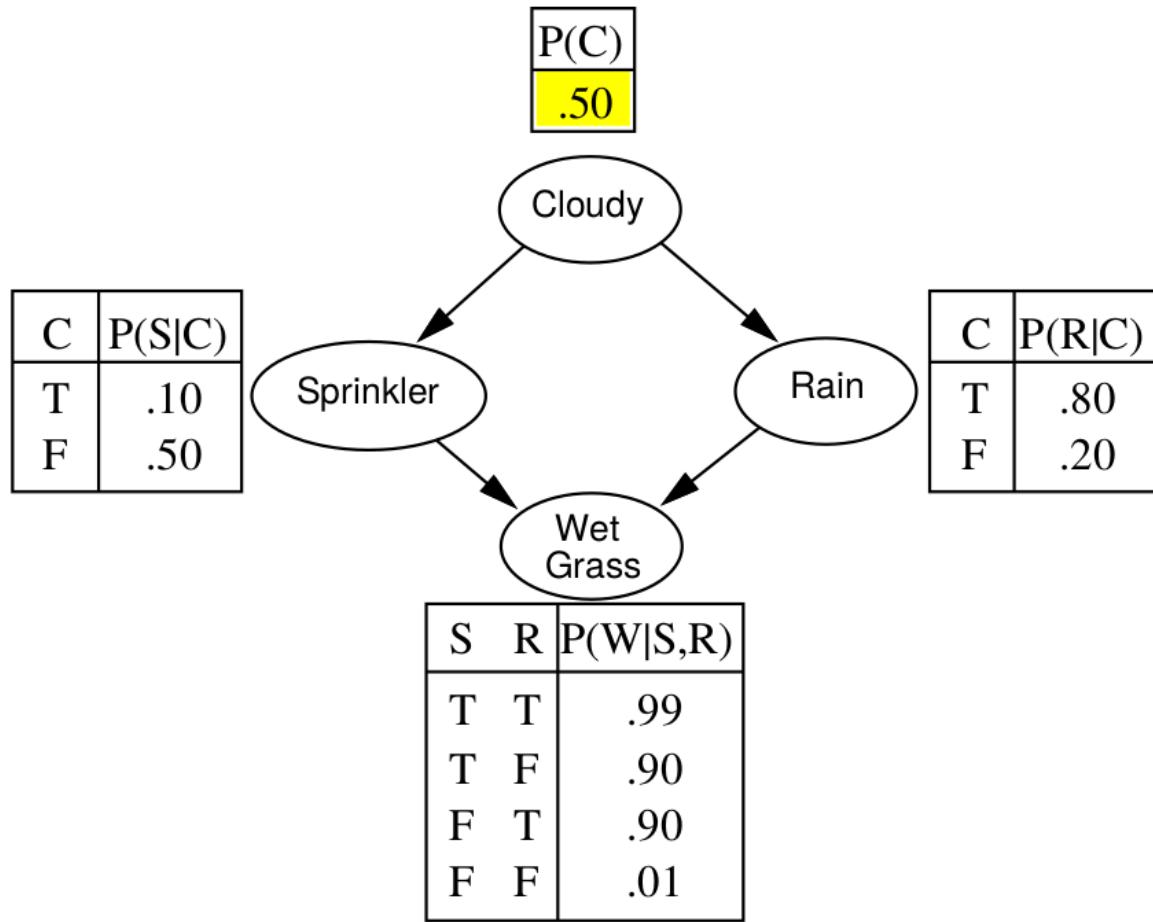
Sampling from a Bayesian network, [without observed evidence](#):

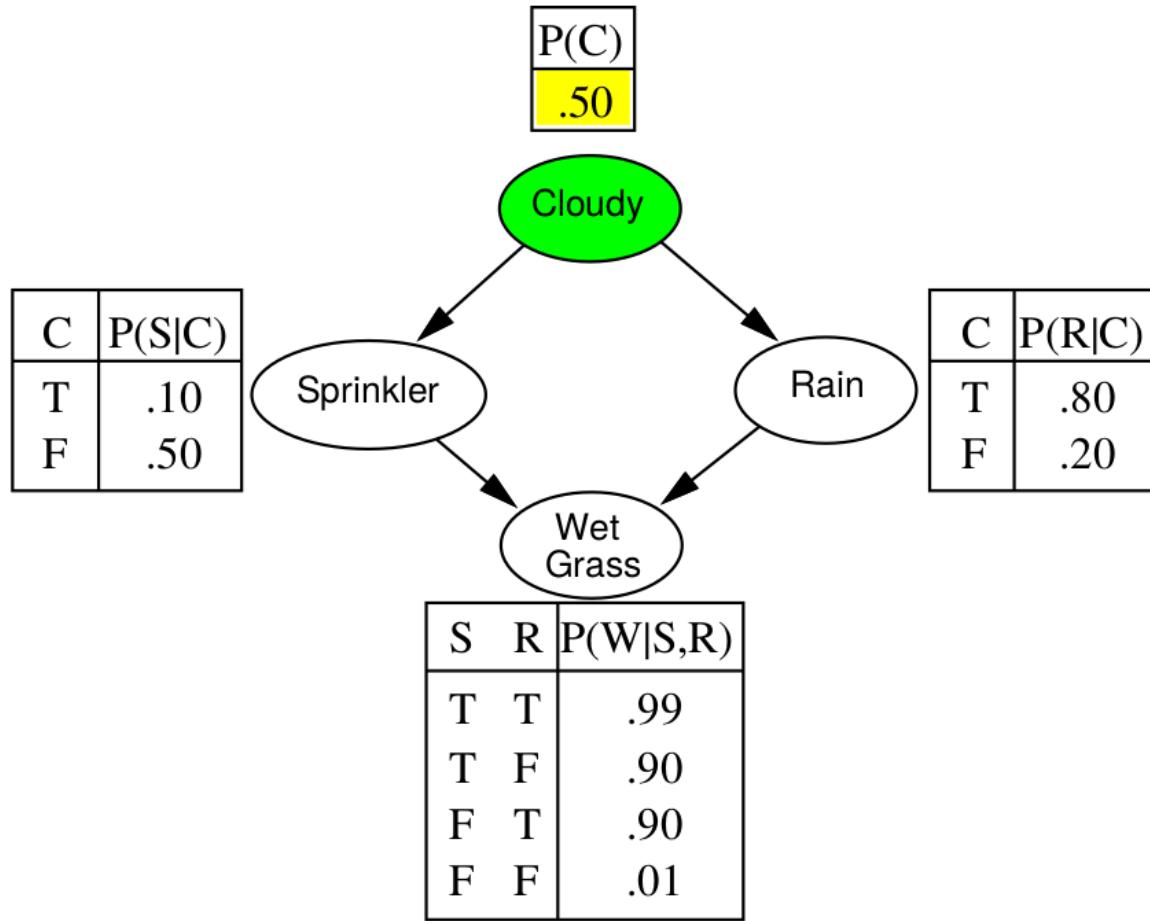
- Sample each variable in turn, [in topological order](#).
- The probability distribution from which the value is sampled is conditioned on the values already assigned to the variable's parents.

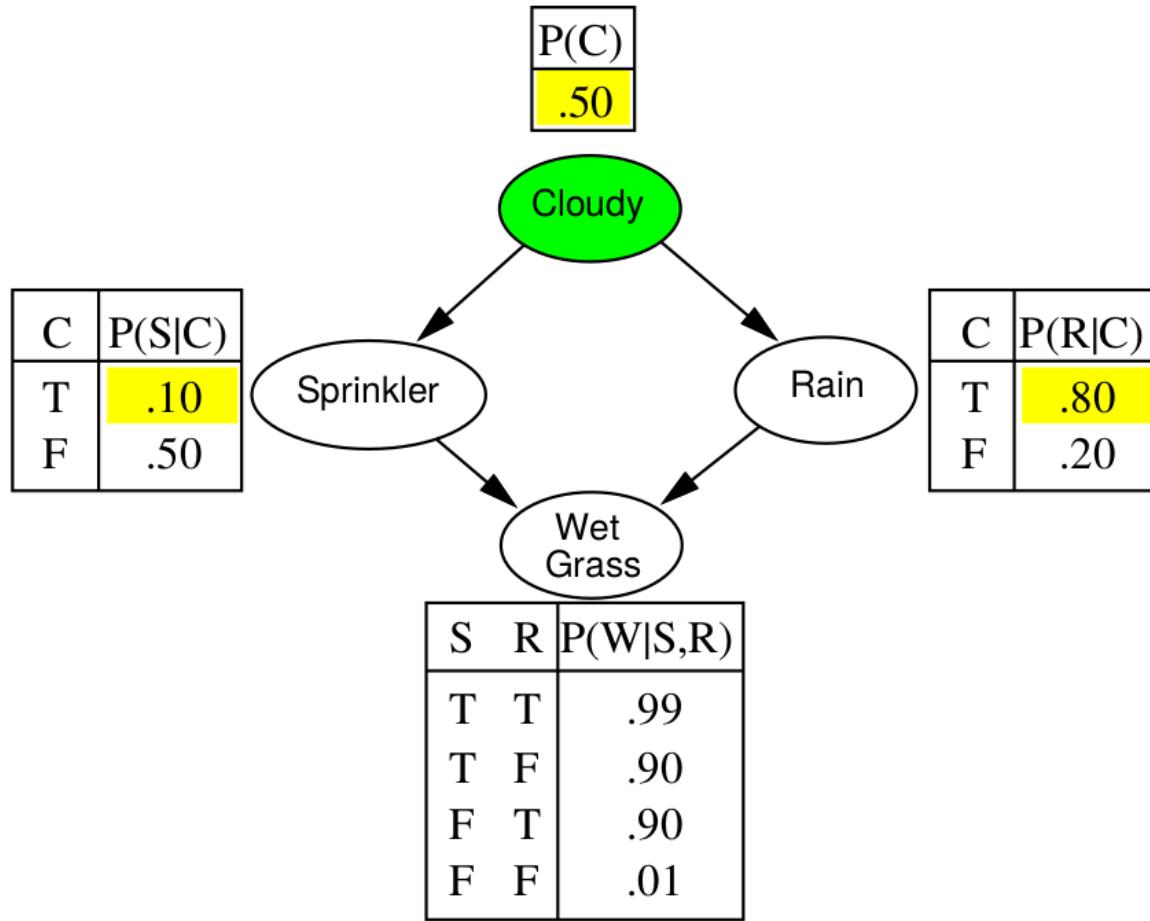
function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
inputs: bn , a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \dots, X_n)$

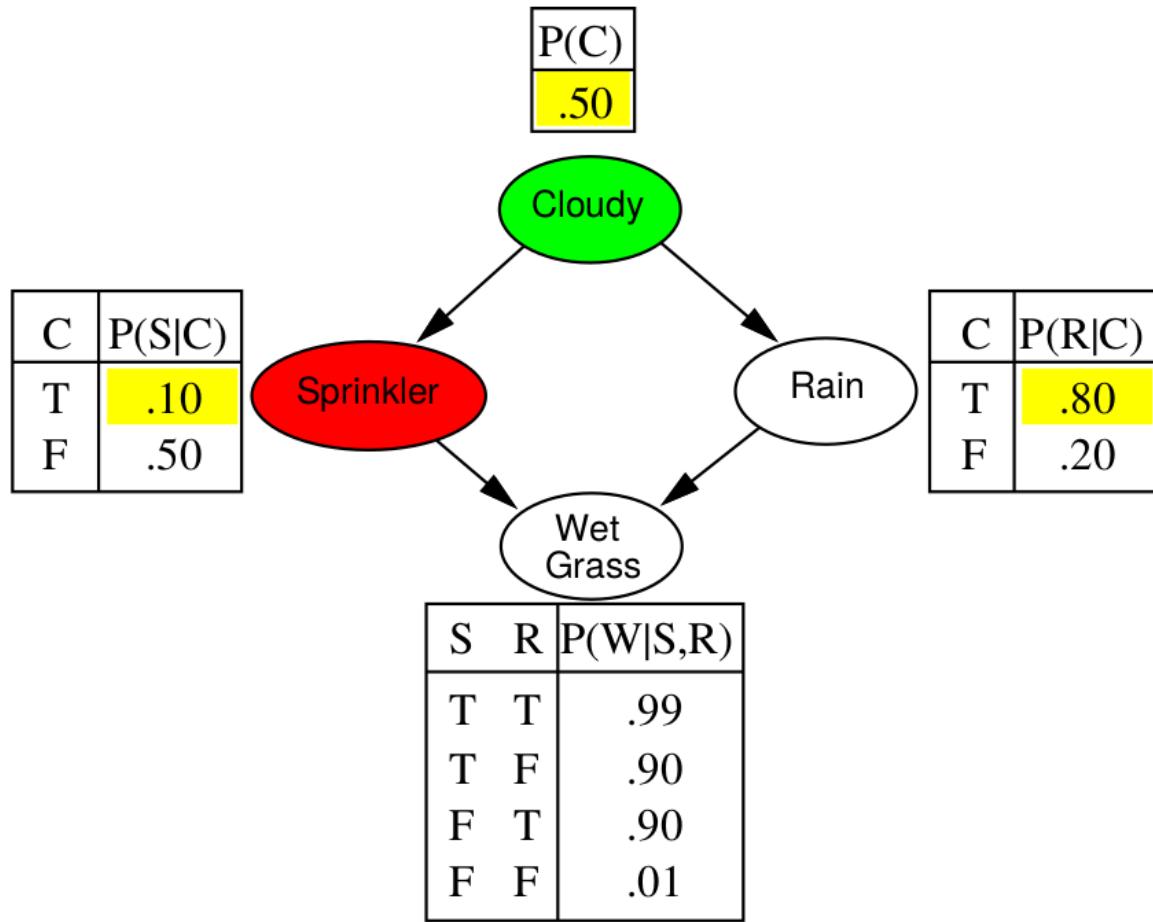
```
x  $\leftarrow$  an event with  $n$  elements
foreach variable  $X_i$  in  $X_1, \dots, X_n$  do
     $x[i] \leftarrow$  a random sample from  $\mathbf{P}(X_i \mid \text{parents}(X_i))$ 
return x
```

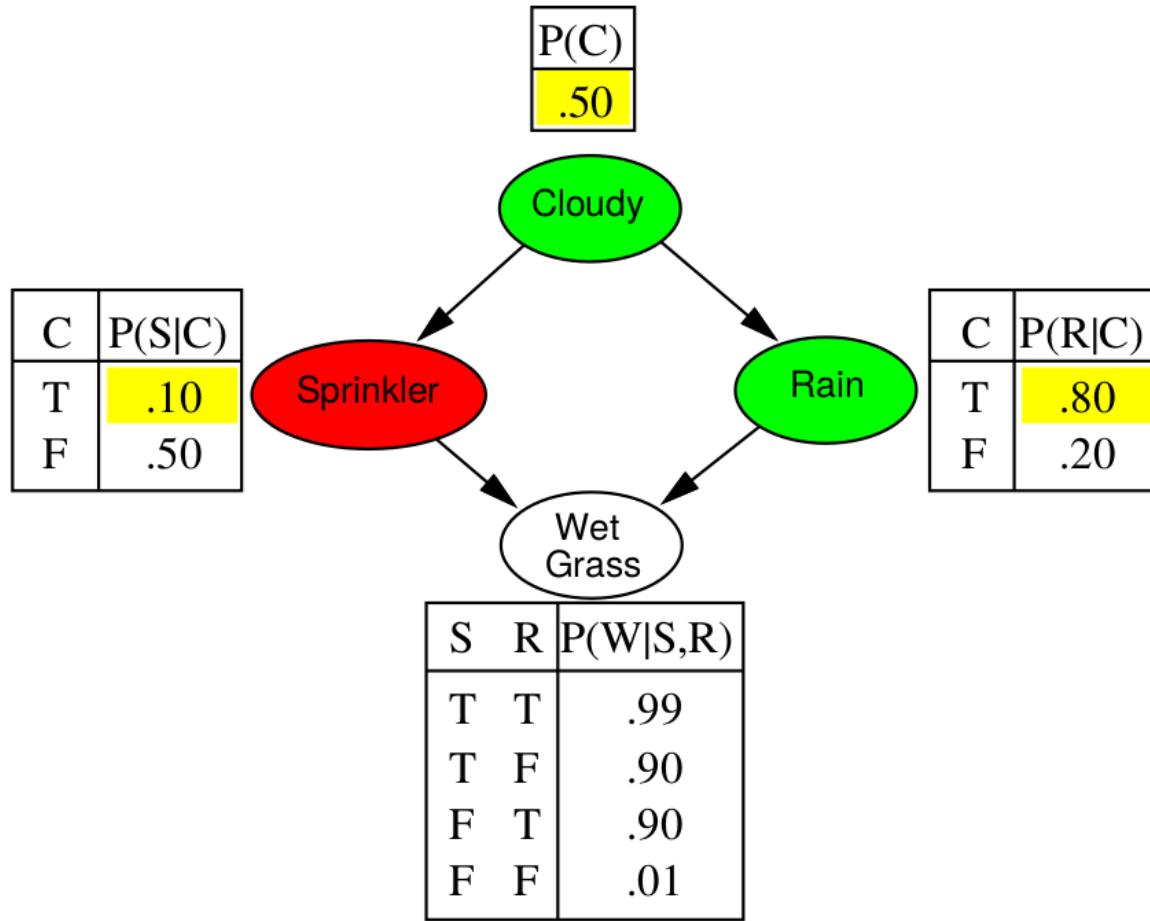


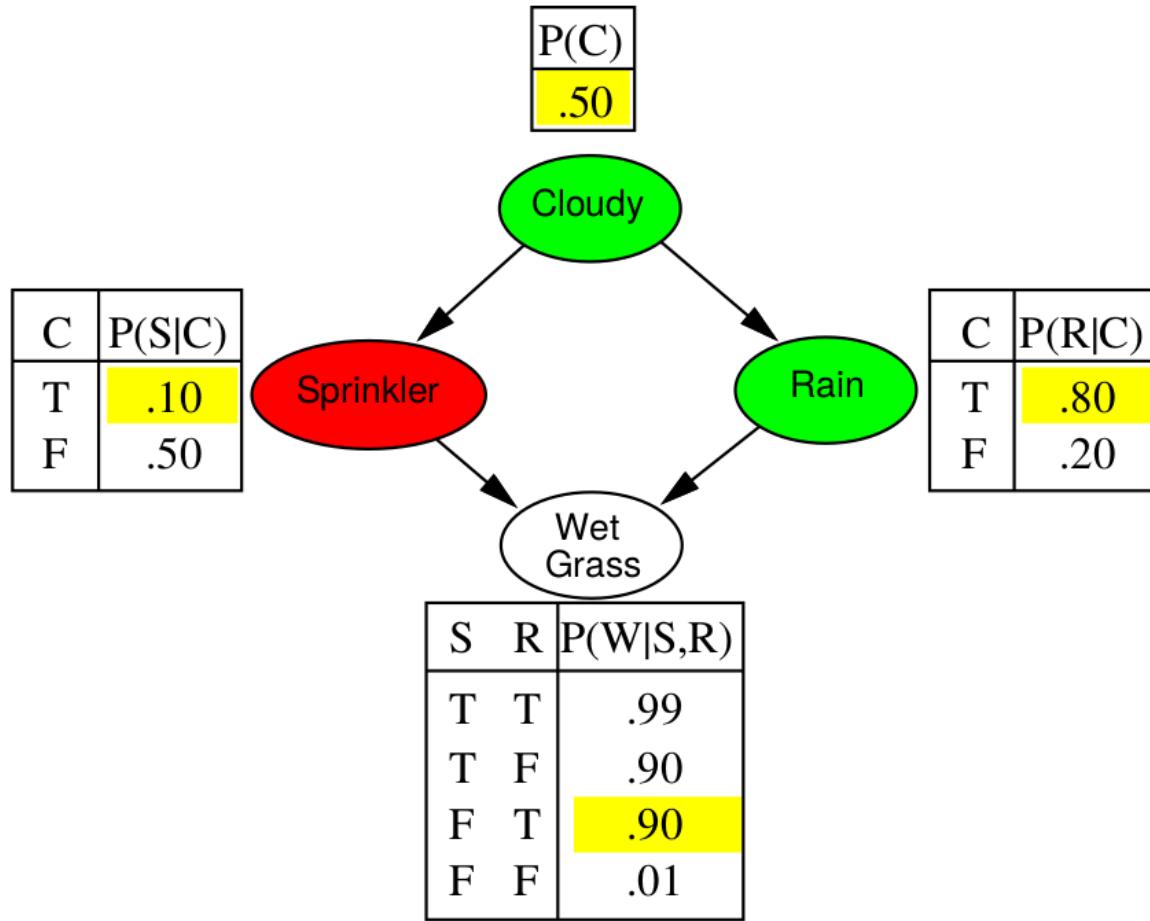


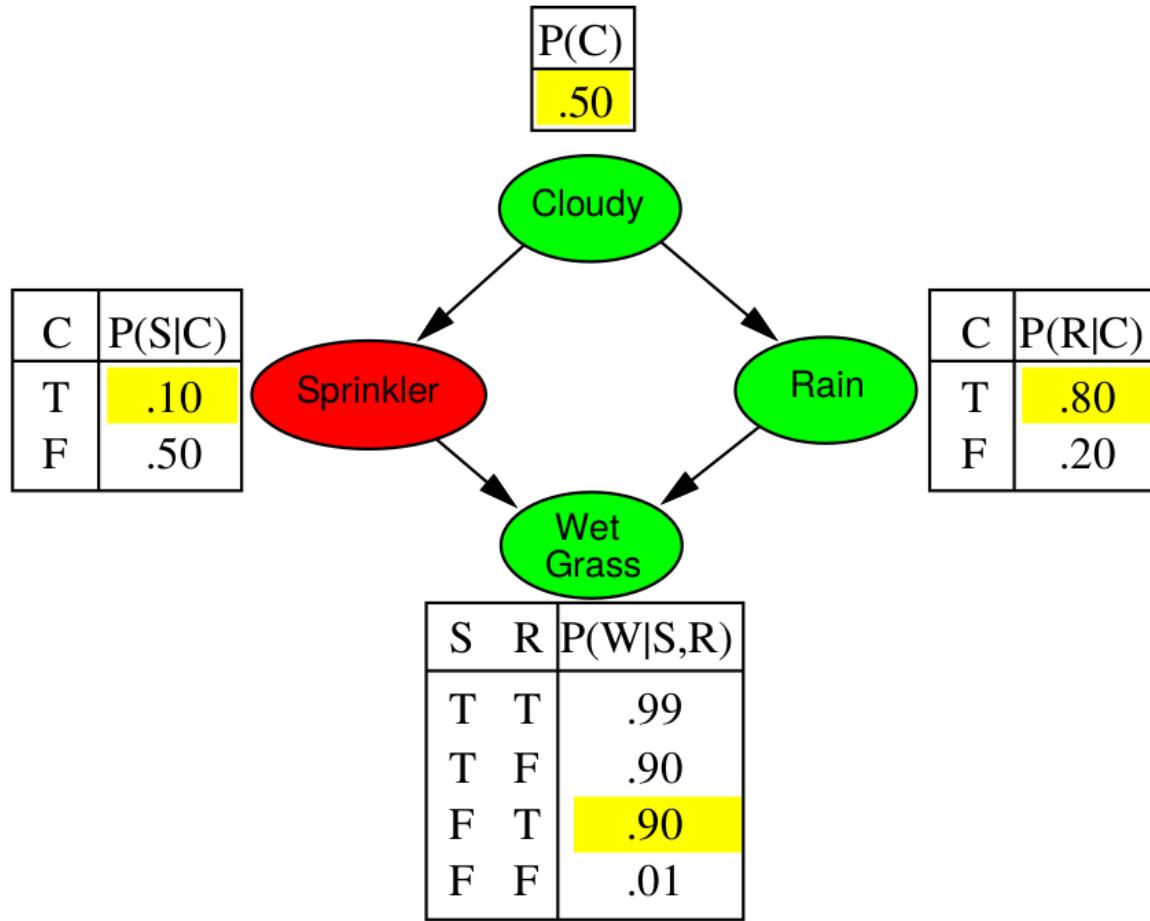












Example

We will collect a bunch of samples from the Bayesian network:

$c, \neg s, r, w$

c, s, r, w

$\neg c, s, r, \neg w$

$c, \neg s, r, w$

$\neg c, \neg s, \neg r, w$

If we want to know $P(W)$:

- We have counts $\langle w : 4, \neg w : 1 \rangle$
- Normalize to obtain $\hat{P}(W) = \langle w : 0.8, \neg w : 0.2 \rangle$
- This will get closer to the true distribution $P(W)$ as we collect more samples.

Analysis

The probability that prior sampling generates a particular event is

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

i.e., the Bayesian network's joint probability.

Let $N_{PS}(x_1, \dots, x_n)$ denote the number of samples of an event. We define the probability **estimate**

$$\hat{P}(x_1, \dots, x_n) = N_{PS}(x_1, \dots, x_n)/N.$$

Then,

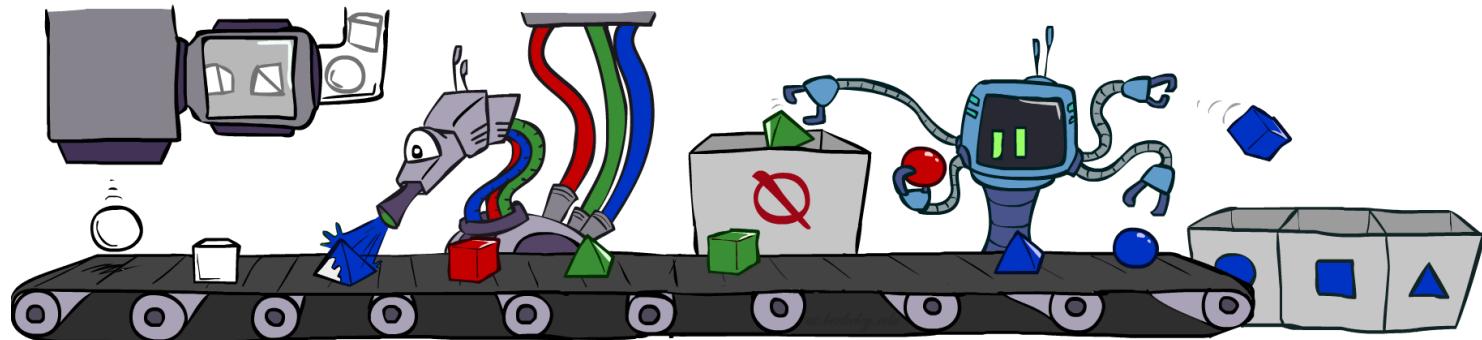
$$\begin{aligned}\lim_{N \rightarrow \infty} \hat{P}(x_1, \dots, x_n) &= \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n)/N \\ &= S_{PS}(x_1, \dots, x_n) \\ &= P(x_1, \dots, x_n)\end{aligned}$$

Therefore, prior sampling is consistent:

$$P(x_1, \dots, x_n) \approx N_{PS}(x_1, \dots, x_n)/N$$

Rejection sampling

Using prior sampling, an estimate $\hat{P}(x|e)$ can be formed from the proportion of samples x agreeing with the evidence e among all samples agreeing with the evidence.



function REJECTION-SAMPLING(X, \mathbf{e}, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$

inputs: X , the query variable

\mathbf{e} , observed values for variables \mathbf{E}

bn , a Bayesian network

N , the total number of samples to be generated

local variables: \mathbf{N} , a vector of counts for each value of X , initially zero

for $j = 1$ to N **do**

$\mathbf{x} \leftarrow \text{PRIOR-SAMPLE}(bn)$

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})

Analysis

Let consider the posterior probability estimate $\hat{P}(x|e)$ formed by rejection sampling:

$$\begin{aligned}\hat{P}(x|e) &= N_{PS}(x, e)/N_{PS}(e) \\ &= \frac{N_{PS}(x, e)}{N} / \frac{N_{PS}(e)}{N} \\ &\approx P(x, e)/P(e) \\ &= P(x|e)\end{aligned}$$

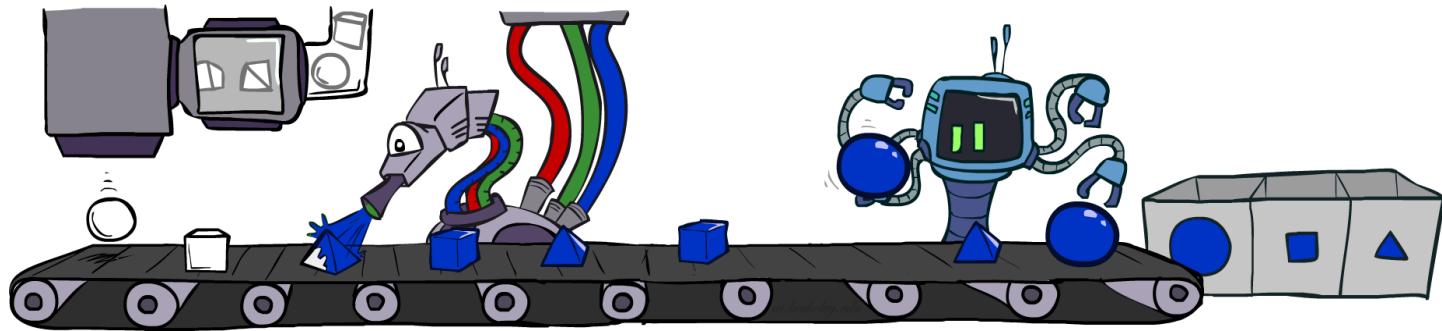
Therefore, rejection sampling returns **consistent** posterior estimates.

- The standard deviation of the error in each probability is $O(1/\sqrt{n})$.
- **Problem:** many samples are rejected!
 - Hopelessly expensive if the evidence is unlikely, i.e. if $P(e)$ is small.
 - Evidence is not exploited when sampling.

Likelihood weighting

Idea: **clamp** the evidence variables, sample the rest.

- Problem: the resulting sampling distribution is not consistent.
- Solution: **weight** by probability of evidence given parents.



function LIKELIHOOD-WEIGHTING(X, \mathbf{e}, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$

inputs: X , the query variable
 \mathbf{e} , observed values for variables \mathbf{E}
 bn , a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \dots, X_n)$
 N , the total number of samples to be generated

local variables: \mathbf{W} , a vector of weighted counts for each value of X , initially zero

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

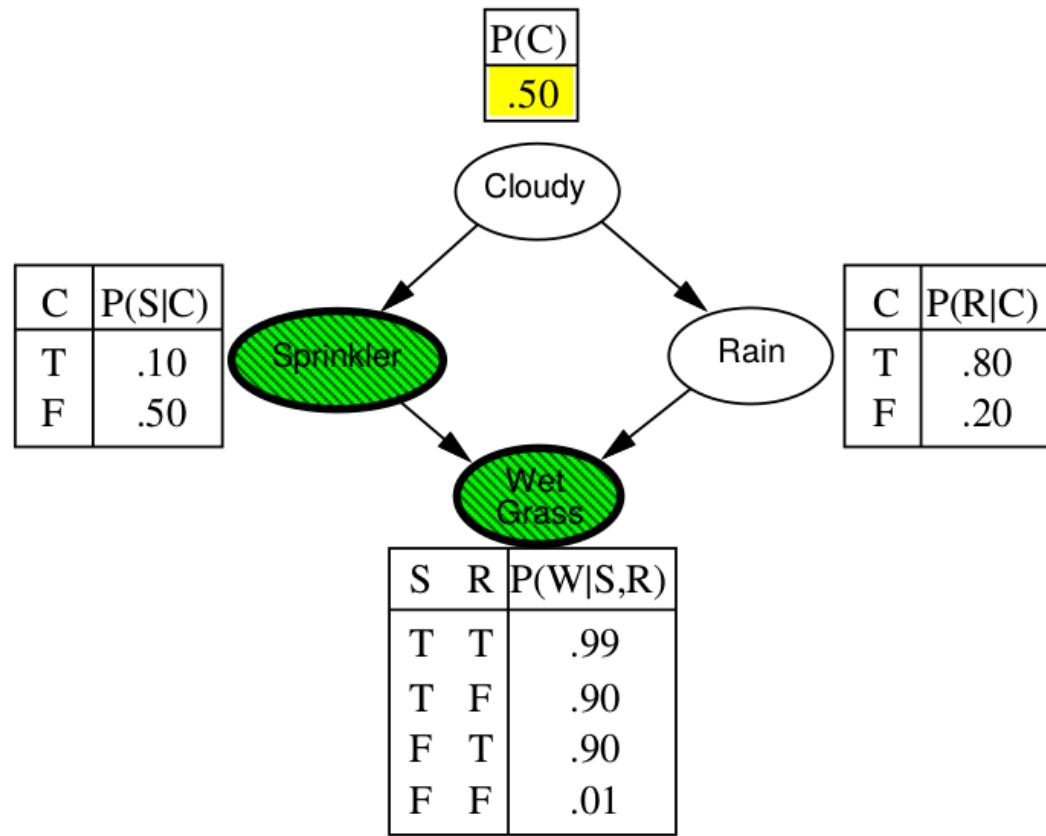
return NORMALIZE(\mathbf{W})

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

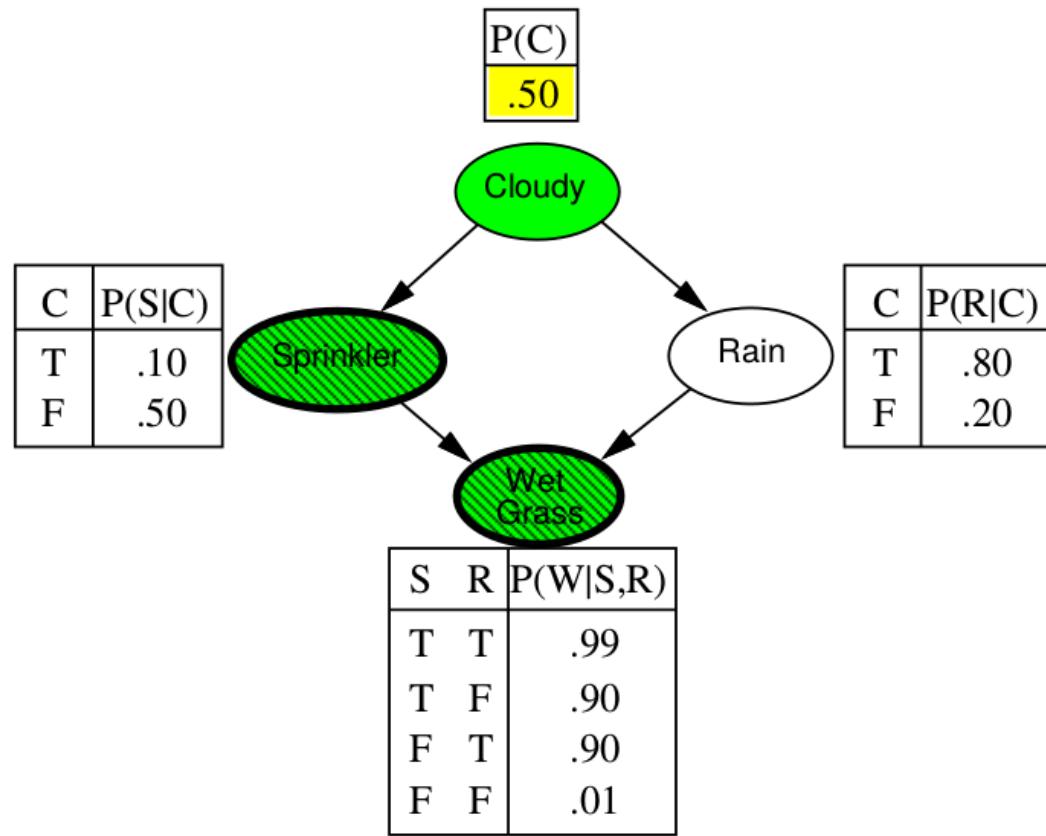
$w \leftarrow 1; \mathbf{x} \leftarrow$ an event with n elements initialized from \mathbf{e}

foreach variable X_i **in** X_1, \dots, X_n **do**
if X_i is an evidence variable with value x_i in \mathbf{e}
then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
else $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$

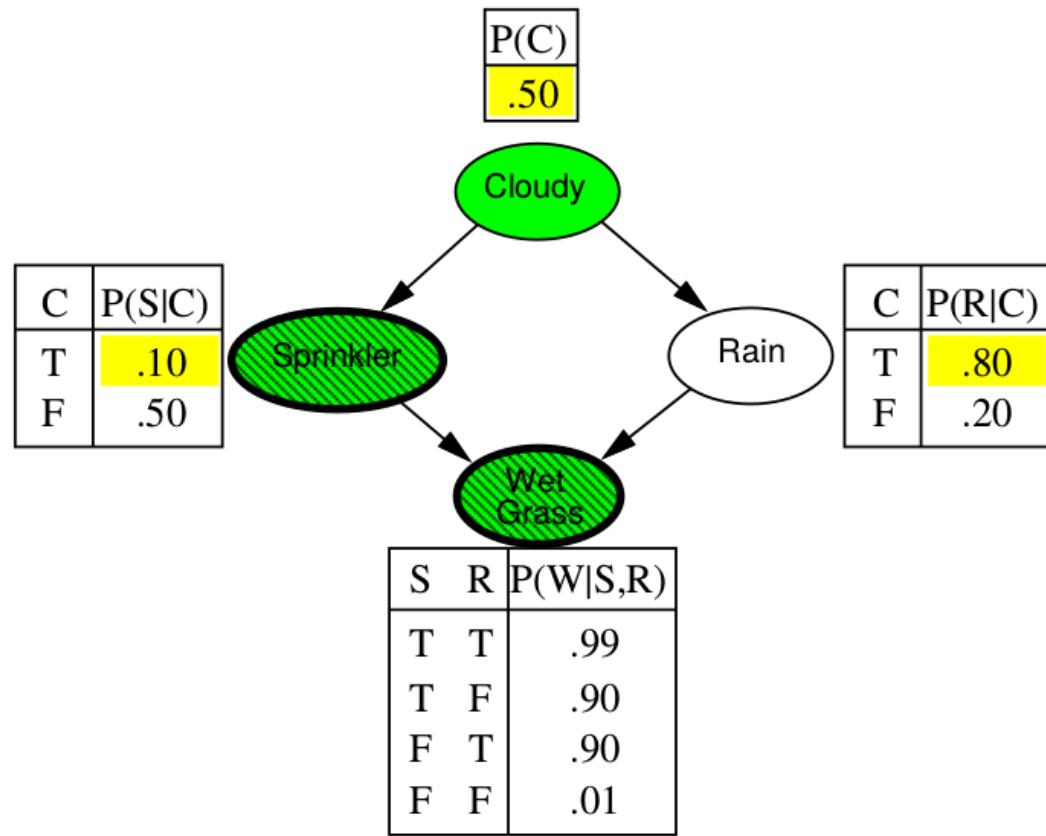
return \mathbf{x}, w



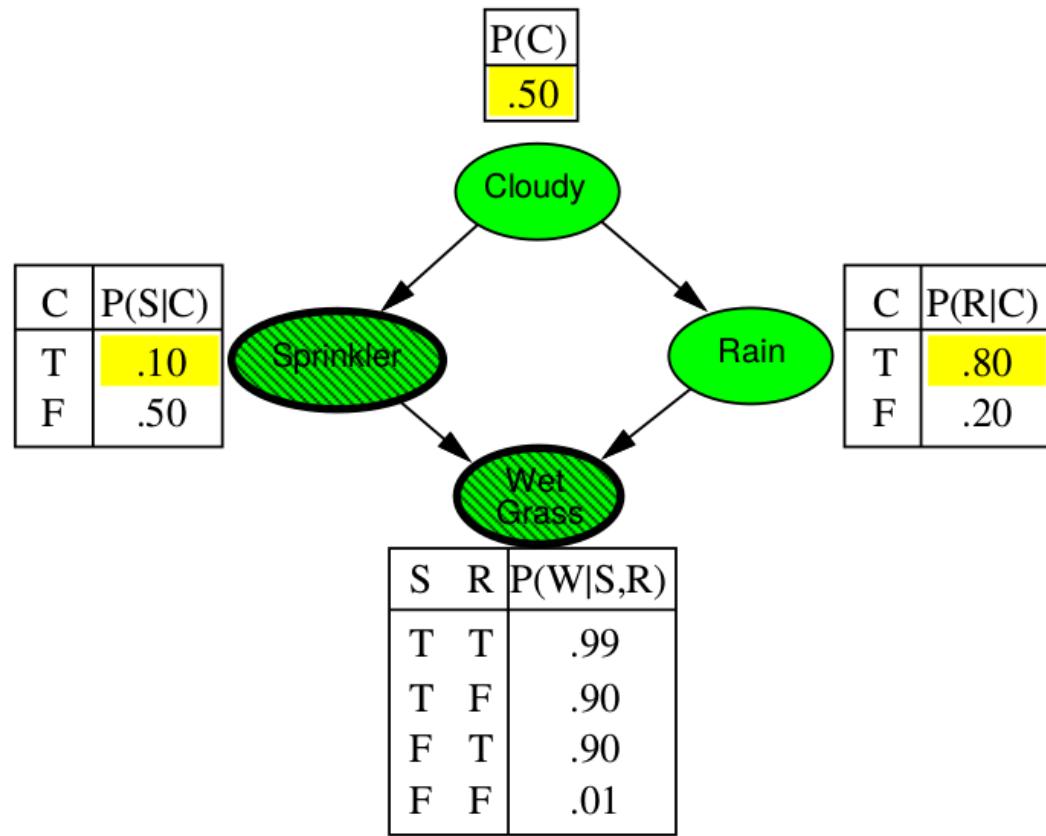
$$w = 1.0$$



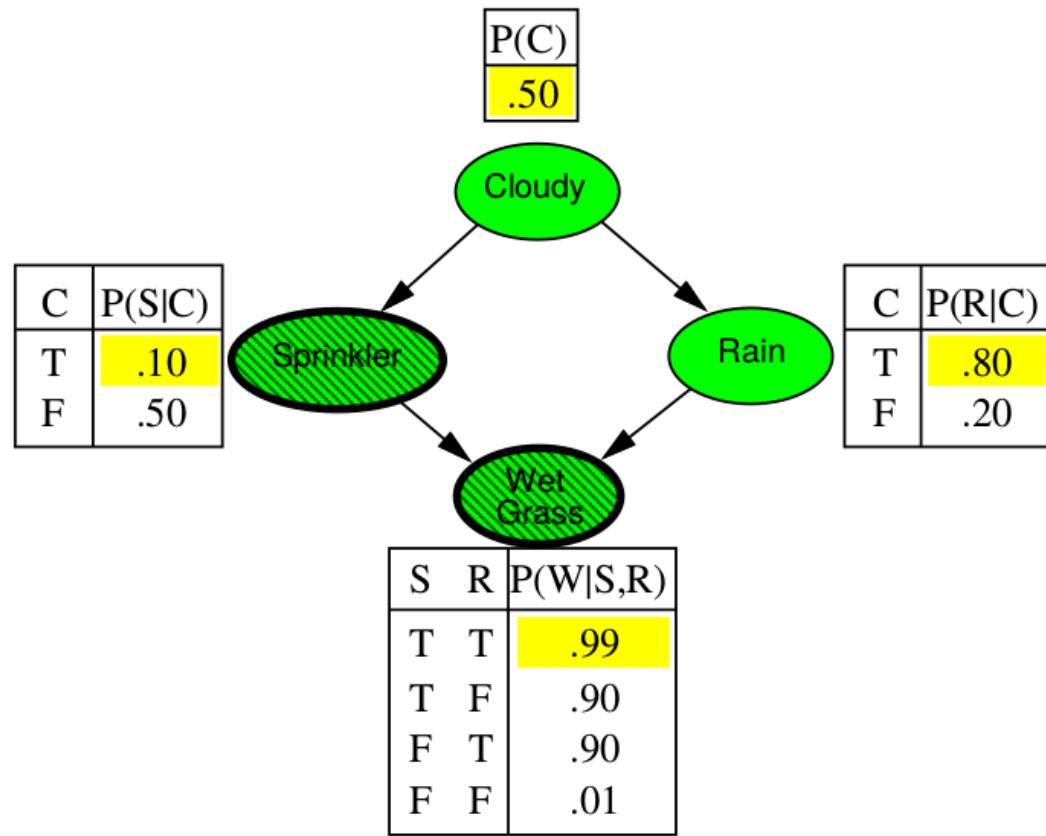
$$w = 1.0$$



$$w = 1.0$$



$$w = 1.0 \times 0.1$$



$$w = 1.0 \times 0.1 \times 0.99 = 0.099$$

Analysis

The sampling probability for an event with likelihood weighting is

$$S_{WS}(x, e) = \prod_{i=1}^l P(x_i | \text{parents}(X_i)),$$

where the product is over the non-evidence variables. The weight for a given sample x, e is

$$w(x, e) = \prod_{i=1}^m P(e_i | \text{parents}(E_i)),$$

where the product is over the evidence variables.

The weighted sampling probability is

$$\begin{aligned} S_{WS}(x, e)w(x, e) &= \prod_{i=1}^l P(x_i | \text{parents}(X_i)) \prod_{i=1}^m P(e_i | \text{parents}(E_i)) \\ &= P(x, e) \end{aligned}$$

The estimated joint probability is computed as follows:

$$\begin{aligned}\hat{P}(x, e) &= N_{WS}(x, e)w(x, e)/N \\ &\approx S_{WS}(x, e)w(x, e) \\ &= P(x, e)\end{aligned}$$

From this, the estimated posterior probability is given by:

$$\begin{aligned}\hat{P}(x|e) &= \hat{P}(x, e)/\hat{P}(e) \\ &\approx P(x, e)/P(e) = P(x|e)\end{aligned}$$

Hence likelihood weighting returns **consistent** estimates.

- Performance **still degrades** with many evidence variables.
- A few samples have nearly all the total weight.

Comments

- Likelihood weighting is **good**:
 - The evidence is taken into account to generate a sample.
 - More of the samples will reflect the state of the world suggested by the evidence.
- Likelihood weighting **does not solve all problems**:
 - The evidence influences the choice of downstream variables, but not upstream ones.
- We would like to consider evidence when we sample **every variable**.

Gibbs sampling

Procedure

- Keep track of a full instance x_1, \dots, x_n .
- Start with an arbitrary instance consistent with the evidence.
- Sample one variable at a time, conditioned on all the rest, but keep the evidence fixed.
- Keep repeating this for a long time.

The sampling process settles into a **dynamic equilibrium** in which the long-run fraction of time spent in each state is exactly proportional to its posterior probability.

```

function GIBBS-ASK( $X, \mathbf{e}, bn, N$ ) returns an estimate of  $\mathbf{P}(X|\mathbf{e})$ 
  local variables:  $\mathbf{N}$ , a vector of counts for each value of  $X$ , 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  $\mathbf{Z}$ 
  for  $j = 1$  to  $N$  do
    for each  $Z_i$  in  $\mathbf{Z}$  do
      set the value of  $Z_i$  in  $\mathbf{x}$  by sampling from  $\mathbf{P}(Z_i|mb(Z_i))$ 
       $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$  where  $x$  is the value of  $X$  in  $\mathbf{x}$ 
  return NORMALIZE( $\mathbf{N}$ )

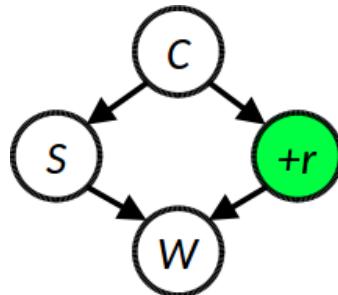
```

Rationale

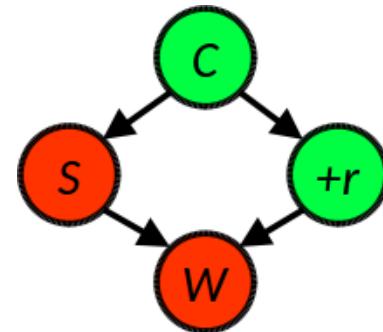
- Both upstream and downstream variables condition on evidence.
- In contrast, likelihood weighting only conditions on upstream evidence, and hence the resulting weights might be very small.

Example

- 1) Fix the evidence.

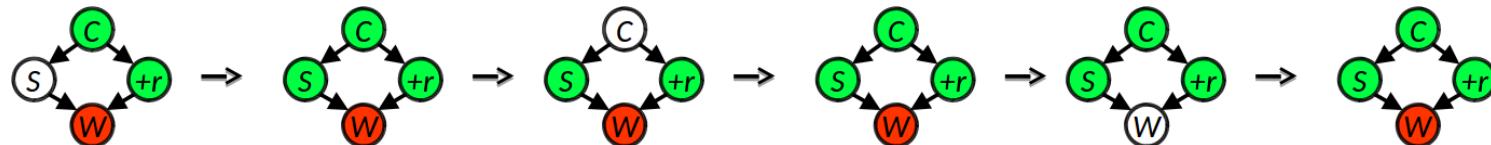


- 2) Randomly initialize the other variables



- 3) Repeat

- Choose a non-evidence variable X .
- Resample X from $P(X|\text{all other variables})$.



Demo

(See code)

Further reading

- Gibbs sampling produces samples from the query distribution $P(X|e)$ in the limit of re-sampling infinitely often.
- Gibbs sampling is a special case of a more general set of methods called **Markov chain Monte Carlo** (MCMC) methods.
- Metropolis-Hastings is one of the most famous MCMC methods.
 - Gibbs sampling is a special case of Metropolis-Hastings.

Summary

- Exact inference by variable elimination .
 - NP-hard on general graphs, but polynomial on polytrees.
 - space = time, very sensitive to topology.
- Approximate inference gives reasonable estimates of the true posterior probabilities in a network and can cope with much larger networks than can exact algorithms.
 - LW does poorly when there is lots of evidence.
 - LW and GS generally insensitive to topology.
 - Convergence can be slow with probabilities close to 1 or 0.
 - Can handle arbitrary combinations of discrete and continuous variables.
- Want to know more about sampling methods? Follow [MATH2022](#).

References

- Cooper, Gregory F. "The computational complexity of probabilistic inference using Bayesian belief networks." *Artificial intelligence* 42.2-3 (1990): 393-405.