

# Artificial Intelligence

## Probabilistic Inference

Christopher Simpkins

Kennesaw State University



# Approximate Inference for Bayesian Networks

Exact inference in large Bayesian networks is intractable, so we need approximate inference methods. The methods we'll learn are randomized sampling algorithms, a.k.a., **Monte Carlo** algorithms, a.k.a., **stochastic simulation**. They work by

- ▶ generating random events based on the probabilities in the Bayes net and
- ▶ counting up the different answers found in those random events.

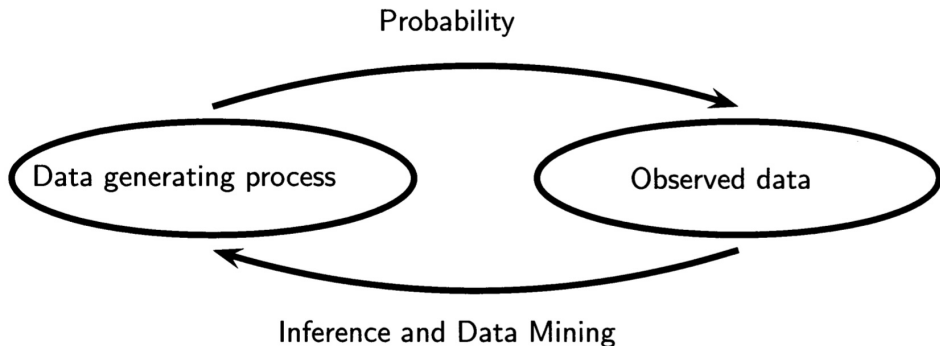
The more the samples, the closer to the true probability distribution.

We'll learn two families of Monte carlo algorithms:

- ▶ direct sampling, and
- ▶ Markov chain sampling.

# The Utility of Sampling

We observe data generated by some process.



1

Then we use those samples to make inferences about the data generating process.

- ▶ In supervised machine learning we assume a distribution for the data generating process and estimate its parameters.
- ▶ In probabilistic inference, we have a model of the full joint distribution (a Bayes net) and use it to generate samples to answer probabilistic queries.

---

<sup>1</sup>All of Statistics

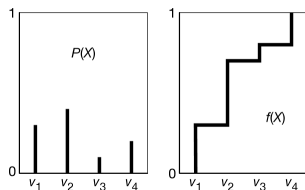
# Computational Sampling of a Univariate Distribution

The primitive element in any sampling algorithm is the generation of samples from a known probability distribution.

To generate samples from a single discrete or real-valued variable,  $X$ ,

- ▶ First totally order the values in the domain of  $X$ .
  - ▶ For discrete variables, if there is no natural order, just create an arbitrary ordering. Given this ordering, the cumulative probability distribution is a function of  $x$  defined by  $f(x) = P(X \leq x)$ .
- ▶ Select a uniform random number  $y$  in the domain  $[0, 1]$ .
- ▶ Let  $v$  be the value of  $X$  that maps to  $y$  in the cumulative probability distribution. That is,  $v$  is the element of  $domain(X)$  such that
  - ▶  $f(v) = y$  or, equivalently,  $v = f^{-1}(y)$
- ▶  $X = v$  is a random sample of  $X$ , chosen according to the distribution of  $X$ .

This procedure works for a single variable. If we have multiple variables and a Bayes net to represent the joint distribution over them, there are several algorithms



## Prior Sampling, a.k.a., Forward Sampling

To sample from a Bayes net with no associated evidence, sample each node in topological order.

- ▶ Sample unconditioned nodes according to their prior distributions.
  - ▶ Samples become fixed values for sampling CPTs of child nodes.
- ▶ Continue sampling child nodes, in order, until all variables are sampled.

**function** PRIOR-SAMPLE(*bn*) **returns** an event sampled from the prior specified by *bn*

**inputs:** *bn*, a Bayesian network specifying joint distribution  $\mathbf{P}(X_1, \dots, X_n)$

$\mathbf{x} \leftarrow$  an event with *n* elements

**for each** variable  $X_i$  **in**  $X_1, \dots, X_n$  **do**

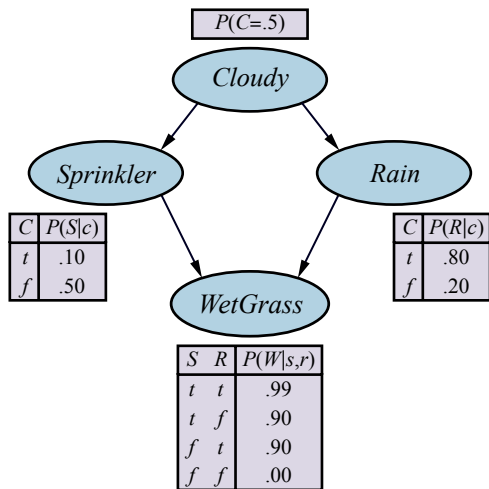
$\mathbf{x}[i] \leftarrow$  a random sample from  $\mathbf{P}(X_i \mid \text{parents}(X_i))$

**return**  $\mathbf{x}$

## Example: Generating a Sample from Sprinkler Bayes Net

1. Sample from  $P(Cloudy) = \langle 0.5, 0.5 \rangle$ 
  - ▶ Get value of true.
2. Sample from  $P(Sprinkler | c) = \langle 0.1, 0.9 \rangle$ 
  - ▶ Get value of false.
3. Sample from  $P(Rain | c) = \langle 0.8, 0.2 \rangle$ 
  - ▶ Get value of true.
4. Sample from  $P(WetGrass | \neg s, r) = \langle 0.9, 0.1 \rangle$ 
  - ▶ Get value of true.

Full sampled event:  $[true, false, true, true]$



## From Sampling to Probability Estimates

A Bayes net represents a full joint distribution, so a sample generated from a Bayes net is a sample from the prior joint distribution over all the variables. So, if  $S_{PS}$  is a sample generated by the PRIOR-SAMPLE algorithm, then:

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

To estimate these probabilities using samples, we simply generate  $N$  samples, count the number of events of interest among the  $N$  samples, and divide that number by  $N$ . This should converge to the true probability:

$$\lim_{N \rightarrow \infty} \frac{N_{PS}(x_1, \dots, x_n)}{N} = S_{PS}(x_1, \dots, x_n) = P(x_1, \dots, x_n)$$

From the CPTs in the Bayes net, the probability of the example event we sampled,  $[true, false, true, true]$  is:

$$S_{PS}(true, false, true, true) = 0.5 \cdot 0.9 \cdot 0.8 \cdot 0.9 = 0.324.$$

So if we generated  $N = 1000$  samples, we would expect to see  $[true, false, true, true]$  324 times.

## Consistent Probability Estimates

An estimate that becomes exact in the large-sample limit is called **consistent**. A consistent estimate of the probability of any partially specified event  $x_1, \dots, x_m$  for  $x \leq n$  is:

$$P(x_1, \dots, x_m) \approx \frac{N_{PS}(x_1, \dots, x_m)}{N} = \hat{P}(x_1, \dots, x_m) \quad (13.6)$$

That is,  $\hat{P}(x_1, \dots, x_m)$  is an approximation of  $P(x_1, \dots, x_m)$  that converges to the true probability as  $N$  approaches  $\infty$

- ▶ Forward sampling generates *prior* probabilities, that is, probabilities of events prior to observing evidence.
- ▶ What we usually want is *posterior* probabilities,  $P(X | e)$  – probability of  $X$  given that we have observed evidence  $e$ .

We now turn to algorithms for computing posteriors, which use prior sampling as a subroutine.

# Rejection Sampling

Rejection sampling is a general method for producing samples from a hard-to-sample distribution given an easy-to-sample distribution. In its simplest form, it can be used to compute conditional probabilities, that is, to determine  $P(X | e)$ .

Let

- ▶  $\hat{P}(X | e)$  be a vector of the probabilities for each value of  $x \in X$  and
- ▶  $N_{PS}(X, e)$  be a vector of sample counts for each  $x \in X$  where the sample is consistent with the evidence  $e$  (the variable  $\mathbf{C}$  in the algorithm).

Then:

$$\hat{P}(X | e) = \alpha N_{PS}(X, e) = \frac{N_{PS}(X, e)}{N_{PS}(e)}$$

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

**inputs:**  $X$ , the query variable

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

$bn$ , a Bayesian network

$N$ , the total number of samples to be generated

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

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

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

**if**  $\mathbf{x}$  is consistent with  $e$  **then**

$\mathbf{C}[j] \leftarrow \mathbf{C}[j] + 1$  where  $x_j$  is the value of  $X$  in  $\mathbf{x}$

**return** NORMALIZE( $\mathbf{C}$ )

## Rejection Sampling Example

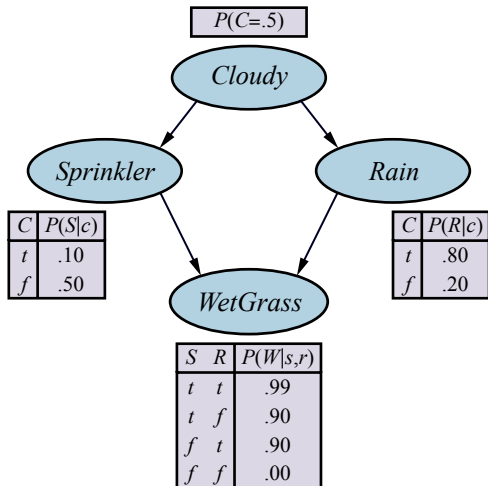
Say we use prior sampling to generate 100 samples and we want to estimate  $P(\text{Rain} \mid \text{Sprinkler} = \text{true})$ .

- ▶ 73 samples have  $\text{Sprinkler} = \text{false}$ , so are rejected.
- ▶ Of the 27 accepted samples:
  - ▶ 8 samples have  $\text{Rain} = \text{true}$
  - ▶ 19 samples have  $\text{Rain} = \text{false}$ .

Then:

$$\begin{aligned}\hat{P}(\text{Rain} \mid \text{Sprinkler} = \text{true}) &= \frac{N_{PS}(X, \mathbf{e})}{N_{PS}(\mathbf{e})} \\ &= \frac{\langle 8, 19 \rangle}{27} \\ &= \langle 0.296, 0.704 \rangle\end{aligned}$$

Calculating directly from the CPTs in the Bayes net would give us  $\langle 0.3, 0.7 \rangle$ .



## Importance Sampling

The general statistical technique of **importance sampling** aims to emulate the effect of sampling from a distribution  $P$  using samples from another distribution  $Q$  whose samples are easier to obtain.

We ensure that the answers are correct in the limit by applying a correction factor  $\frac{P(\mathbf{x})}{Q(\mathbf{x})}$ , also known as a **weight**, to each sample  $\mathbf{x}$  when counting up the samples.

Our query variable will always be one of the nonevidence variables,  $Z$ . The idea of importance sampling is to sample from  $P(\mathbf{z} | \mathbf{e})$ :

$$\hat{P}(\mathbf{z} | \mathbf{e}) = \frac{N_P(\mathbf{z})}{N} \approx P(\mathbf{z} | \mathbf{e})$$

but using an arbitrary distribution  $Q(\mathbf{z})$ :

$$\hat{P}(\mathbf{z} | \mathbf{e}) = \frac{N_Q(\mathbf{z})}{N} \frac{P(\mathbf{z} | \mathbf{e})}{Q(\mathbf{z})} \approx Q(\mathbf{z}) \frac{P(\mathbf{z} | \mathbf{e})}{Q(\mathbf{z})} = P(\mathbf{z} | \mathbf{e})$$

The question is: which  $Q$  to use? We want  $Q$  as close as possible to  $P(\mathbf{z} | \mathbf{e})$  with  $Q(\mathbf{z}) \neq 0$  whenever  $P(\mathbf{z} | \mathbf{e}) \neq 0$ . The most common approach is **likelihood weighting** ...

## Likelihood Weighting

Fix the values for the evidence variables  $E$  and sample all the nonevidence variables in topological order, each conditioned on its parents – guarantees each sample consistent with evidence.

Let the sampling distribution be  $Q_{WS}$  and nonevidence variables be  $Z = \{Z_1, \dots, Z_l\}$ . Then:

$$Q_{WS}(z) = \prod_{i=1}^l P(z_i \mid \text{parents}(Z_i))$$

Now we need to know the weight for each sample. By general scheme for importance sampling:

$$w(z) = \frac{P(z \mid e)}{Q_{WS}(z)} = \alpha \frac{P(z, e)}{Q_{WS}(z)}$$

where the normalizing factor  $\alpha = \frac{1}{P(e)}$  is the same for all samples. Since  $z$  and  $e$  cover all the variables in the Bayes net,  $P(z, e)$  is the product of the conditional probabilities for the nonevidence variables times the product of the conditional probabilities for the evidence variables:

$$\begin{aligned} w(z) &= \alpha \frac{P(z, e)}{Q_{WS}(z)} = \alpha \frac{\prod_{i=1}^l P(z_i \mid \text{parents}(Z_i)) \prod_{i=1}^m P(e_i \mid \text{parents}(E_i))}{\prod_{i=1}^l P(z_i \mid \text{parents}(Z_i))} \\ &= \alpha \prod_{i=1}^m P(e_i \mid \text{parents}(E_i)) \end{aligned} \tag{14.9}$$

The name of this method comes from the fact that probabilities of evidence are generally called likelihoods.

## Likelihood Weighting Algorithm

**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**

$\mathbf{x}, w \leftarrow \text{WEIGHTED-SAMPLE}(bn, \mathbf{e})$

$\mathbf{W}[j] \leftarrow \mathbf{W}[j] + w$  where  $x_j$  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, with values fixed from  $\mathbf{e}$

**for**  $i = 1$  **to**  $n$  **do**

**if**  $X_i$  is an evidence variable with value  $x_{ij}$  in  $\mathbf{e}$

**then**  $w \leftarrow w \times P(X_i = x_{ij} | \text{parents}(X_i))$

**else**  $\mathbf{x}[i] \leftarrow$  a random sample from  $\mathbf{P}(X_i | \text{parents}(X_i))$

**return**  $\mathbf{x}, w$

# Likelihood Weighting Example

Given the query

- ▶  $P(\text{Rain} \mid \text{Cloudy} = \text{true}, \text{WetGrass} = \text{true})$

and the ordering

- ▶  $\text{Cloudy}, \text{Sprinkler}, \text{Rain}, \text{WetGrass}$ ,

we initialize  $w = 1.0$  and proceed as follows:

1.  $\text{Cloudy}$  is an evidence variable with value true. Therefore, we set

$$w \leftarrow w \cdot P(c) = 0.5.$$

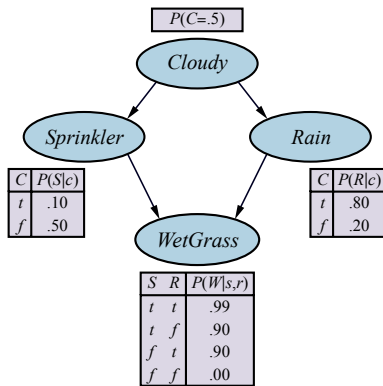
2.  $\text{Sprinkler}$  is not an evidence variable, so sample from  $P(\text{Sprinkler} \mid \text{Cloudy} = \text{true}) = \langle 0.1, 0.9 \rangle$ ; suppose this returns false.

3.  $\text{Rain}$  is not an evidence variable, so sample from  $P(\text{Rain} \mid \text{Cloudy} = \text{true}) = \langle 0.8, 0.2 \rangle$ ; suppose this returns true.

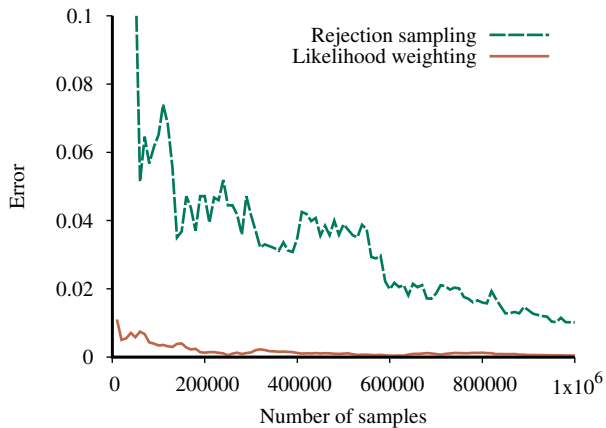
4.  $\text{WetGrass}$  is an evidence variable with value true. Therefore, we set

$$w \leftarrow w \times P(\text{WetGrass} = \text{true} \mid \neg s, r) = 0.5 \cdot 0.9 = 0.45.$$

Here WEIGHTED-SAMPLE returns the event  $[\text{true}, \text{false}, \text{true}, \text{true}]$  with weight 0.45, which we tally under  $\text{Rain} = \text{true}$ .



## Rejection vs. Importance Sampling



# Markov Chain Monte Carlo (MCMC) Algorithms

Instead of generating each sample from scratch, MCMC algorithms generate a sample by making a random change to the preceding sample.

Think of an MCMC algorithm as being in a particular current state that specifies a value for every variable and generating a next state by making random changes to the current state.

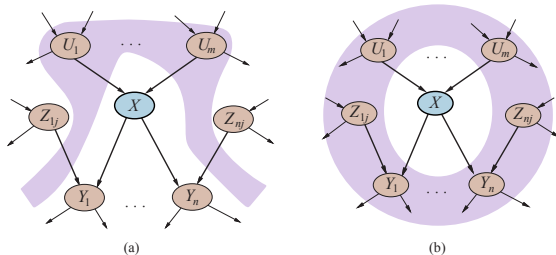
We'll learn two MCMC algorithms:

- ▶ Gibbs sampling, and
- ▶ Metropolis-Hastings

# Gibbs Sampling and Markov Blankets



# Conditional Independence Relations



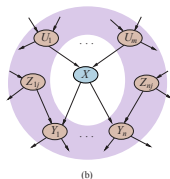
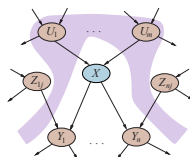
- ▶ (a) Each variable is conditionally independent of its non-descendants, given its parents.
- ▶ (b) A variable is conditionally independent of all other nodes in the network, given its parents, children, and children's parents – that is, given its **Markov blanket**.

For example, the variable *Burglary* is independent of *JohnCalls* and *MaryCalls*, given *Alarm* and *Earthquake*.

# Gibbs Sampling

- ▶ Fix the evidence variables.
- ▶ Start in an arbitrary state sampled from the nonevidence variables.
  - ▶ Each  $X_i$  is sampled from the values in its Markov blanket,  $mb(X_i)$

$$P(x_i | mb(X_i)) = \alpha P(x_i | \text{parents}(X_i)) \prod_{Y_j \in \text{Children}(X_i)} P(y_j | \text{parents}(Y_j^p)) \quad (13.10)$$



- ▶ Wander the state space randomly, keeping the evidence variables fixed and flipping one nonevidence variable by sampling from a distribution calculated Equation 13.10.

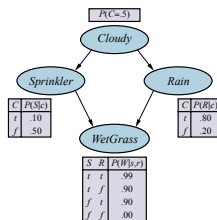
Gibbs sampling for  $X_i$  means sampling conditioned on the current values of the variables in its Markov blanket.

## Gibbs Sampling Example: Step 1

Consider the query  $P(\text{Rain} \mid \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$ .

- Evidence variables *Sprinkler* and *WetGrass* fixed to observed values (true), nonevidence variables *Cloudy* and *Rain* initialized randomly to, say, true and false respectively.

- Initial state is  $[\text{true}, \mathbf{\text{true}}, \text{false}, \mathbf{\text{true}}]$ . Fixed evidence variables marked in bold.



Now the nonevidence variables  $Z_i$  are sampled repeatedly in some random order according to a probability distribution  $\rho(i)$  for choosing variables. For example:

- Cloudy* is chosen and then sampled, given the current values of its Markov blanket: in this case, we sample from  $P(\text{Cloudy} \mid \text{Sprinkler} = \text{true}, \text{Rain} = \text{false})$  whose distribution is calculated using Equation 13.10:

$$P(x_i \mid mb(X_i)) = \alpha P(x_i \mid \text{parents}(X_i)) \prod_{Y_j \in \text{Children}(X_i)} P(y_j \mid \text{parents}(Y_j)) \quad (13.10)$$

$$P(c \mid s, \neg r) = \alpha P(c) P(s \mid c) P(\neg r \mid c) = \alpha 0.5 \cdot 0.1 \cdot 0.2$$

$$P(\neg c \mid s, \neg r) = \alpha P(\neg c) P(s \mid \neg c) P(\neg r \mid \neg c) = \alpha 0.5 \cdot 0.5 \cdot 0.8$$

yielding  $\alpha \langle 0.001, 0.020 \rangle \approx \langle 0.048, 0.952 \rangle$

- Suppose the result is *Cloudy* = false. Then the new current state is  $[\text{false}, \mathbf{\text{true}}, \text{false}, \mathbf{\text{true}}]$ .

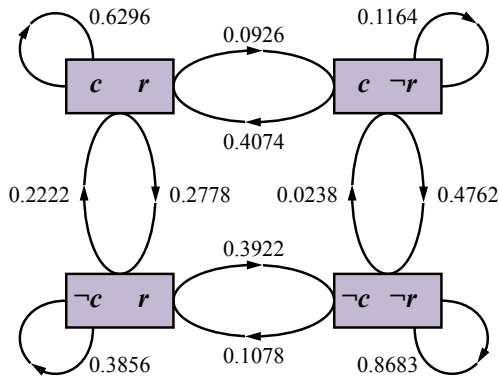
## Gibbs Sampling Example: Step 2

2. Sampling  $\rho(i)$  then chooses *Rain* as next variable. We sample from  $P(\text{Rain} \mid \text{Cloudy} = \text{false}, \text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$  using Equation 13.10 (just like Step 1, not shown here).

- Suppose this yields *Rain* = *true*. The new current state is [*false*, **true**, *true*, **true**].

Figure on the right shows the complete Markov chain for the case where variables are chosen uniformly, i.e.,  $\rho(\text{Cloudy}) = \rho(\text{Rain}) = 0.5$ .

- The algorithm wanders around in this graph, following links with stated probabilities.
- Each state visited during this process is a sample that contributes to the estimate for the query variable *Rain*.
  - If the process visits 20 states where *Rain* is true and 60 states where *Rain* is false, then the answer to the query is  $\text{NORMALIZE}(\langle 20, 60 \rangle) = \langle 0.25, 0.75 \rangle$ .



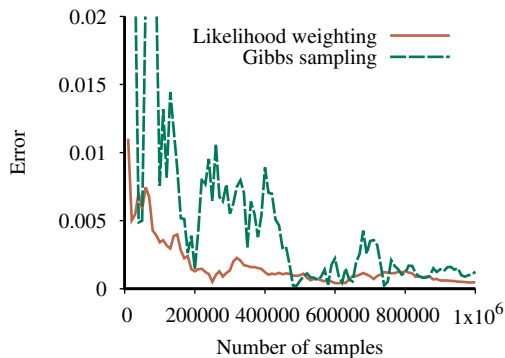
# Gibbs Sampling Algorithm

**function** GIBBS-ASK( $X, \mathbf{e}, bn, N$ ) **returns** an estimate of  $\mathbf{P}(X | \mathbf{e})$   
**local variables:**  $\mathbf{C}$ , 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, initialized from  $\mathbf{e}$

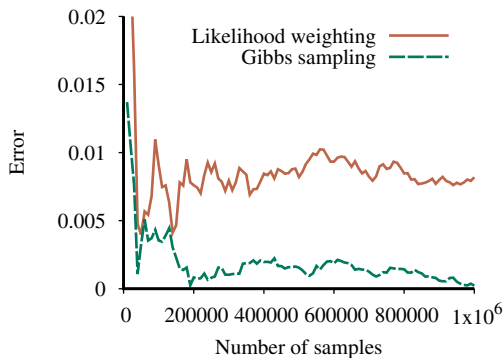
initialize  $\mathbf{x}$  with random values for the variables in  $\mathbf{Z}$   
**for**  $k = 1$  **to**  $N$  **do**  
    **choose** any variable  $Z_i$  from  $\mathbf{Z}$  according to any distribution  $\rho(i)$   
    set the value of  $Z_i$  in  $\mathbf{x}$  by sampling from  $\mathbf{P}(Z_i | mb(Z_i))$   
     $\mathbf{C}[j] \leftarrow \mathbf{C}[j] + 1$  where  $x_j$  is the value of  $X$  in  $\mathbf{x}$   
**return** NORMALIZE( $\mathbf{C}$ )

# Gibbs Sampling vs. Importance Sampling

Performance of Gibbs sampling compared to likelihood weighting on the car insurance network.



(a)

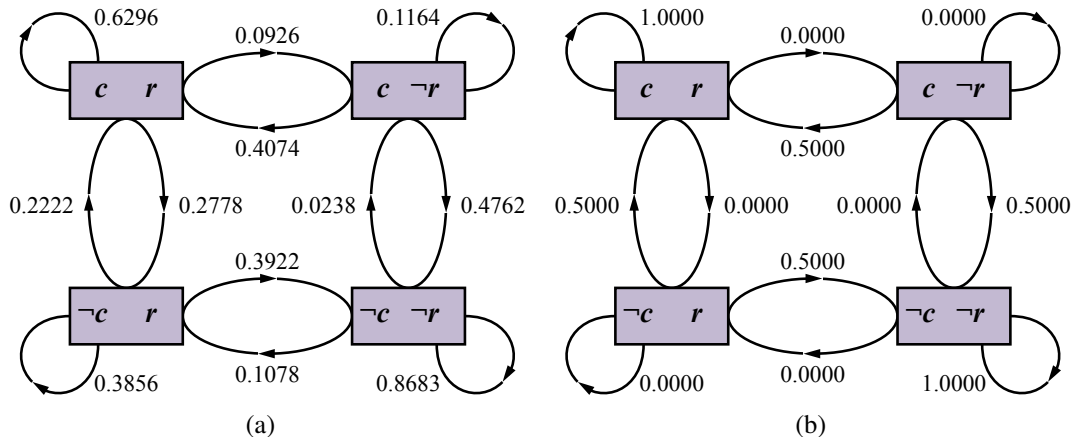


(b)

- ▶ (a) for the standard query on *PropertyCost*, and
- ▶ (b) for the case where the output variables are observed and *Age* is the query variable.

Gibbs sampling typically outperforms likelihood weighting when evidence is mostly downstream.

# Markov Chains



- ▶ (a) The states and transition probabilities of the Markov chain for the query  $P(Rain|Sprinkler = true, WetGrass = true)$ . Note the self-loops: the state stays the same when either variable is chosen and then resamples the same value it already has.
- ▶ (b) The transition probabilities when the CPT for  $Rain$  constrains it to have the same value as  $Cloudy$ .

# Metropolis



# Metropolis-Hastings Sampling

Like simulated annealing, MH has two stages in each iteration of the sampling process:

1. Sample a new state  $x'$  from a **proposal distribution**  $q(x'|x)$ , given the current state  $x$ .
2. Probabilistically accept or reject  $x'$  according to the **acceptance probability**

$$a(x'|x) = \min \left( 1, \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)} \right)$$

If the proposal is rejected, the state remains at  $x$ .

$q(x'|x)$  could be defined as follows:

- ▶ With probability 0.95, perform a Gibbs sampling step to generate  $x'$ .
- ▶ Otherwise, generate  $x'$  by running WEIGHTED-SAMPLE using likelihood weighting.

This has the effect of getting MH “unstuck.”