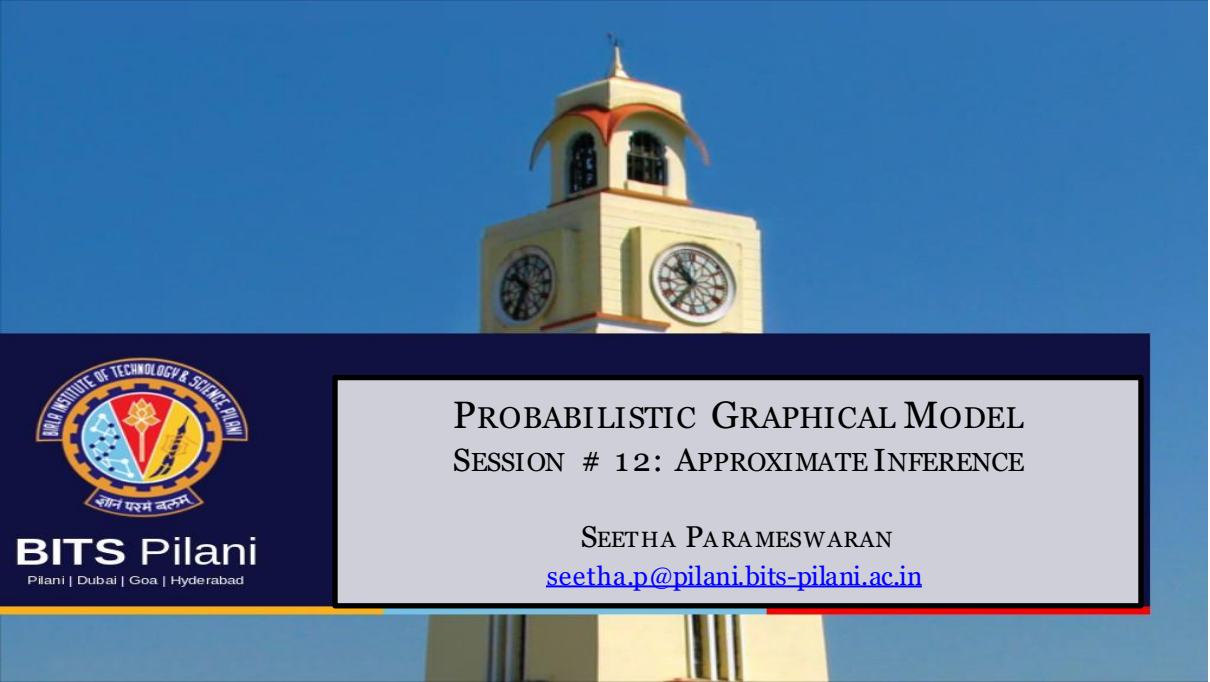


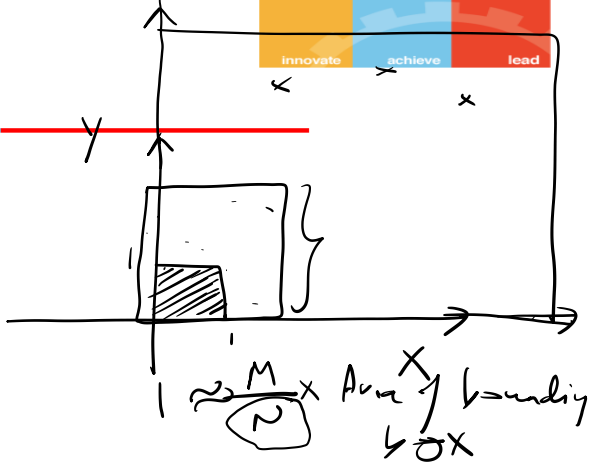
BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Probabilistic Graphical Model

Session # 12: Approximate Inference

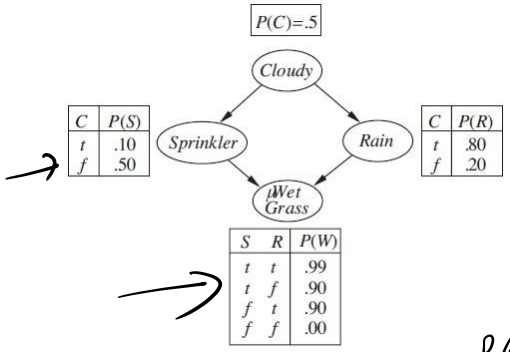
Seetha Parameswaran
seetha.p@pilani.bits-pilani.ac.in

Table of Contents

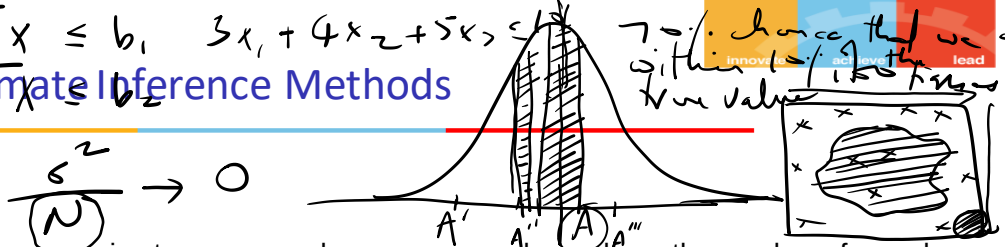


Approximate Inference Methods

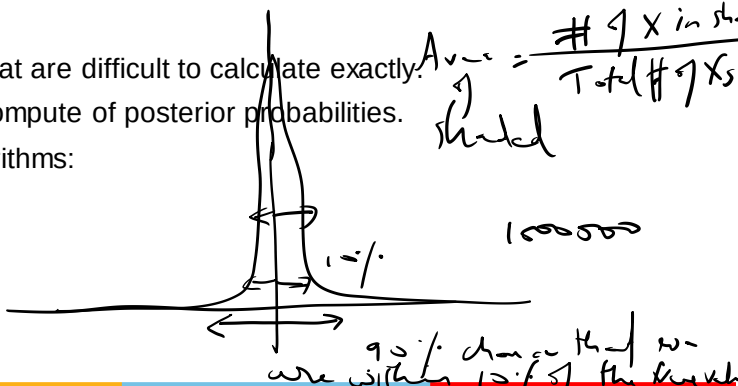
- Given the intractability of exact inference in large, multiply connected networks, it is essential to consider approximate inference methods.



Approximate Inference Methods



- Provide approximate answers whose accuracy depends on the number of samples generated.
- Used to estimate quantities that are difficult to calculate exactly.
- Sampling can be applied to compute of posterior probabilities.
- Two families of sampling algorithms:
  1. Direct sampling
  2. Markov chain sampling



Direct Sampling in Bayesian Networks

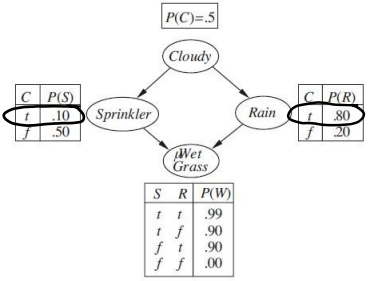
- The primitive element in any sampling algorithm is the generation of samples from a known **probability distribution**.
- Given a source of random numbers uniformly distributed in the range [0, 1], it is a simple matter to sample any distribution on a single variable, whether discrete or continuous.
- PRIOR-SAMPLE generates samples from the prior joint distribution specified by the network.

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

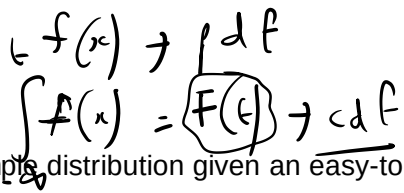
  $\mathbf{x} \leftarrow$ an event with n elements
  foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return $\mathbf{x}$

Direct Sampling - Example

- Assume ordering [*Cloudy*, *Sprinkler*, *Rain*, *WetGrass*]
- Sample from $P(Cloudy) = \langle 0.5, 0.5 \rangle$.
- Sampled value is *true*.
- Sample from $P(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$. Sampled value is *false*.
- Sample from $P(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$.
- Sampled value is *true*.
- Sample from $P(WetGrass \mid Sprinkler = false, Rain = true) = \langle 0.9, 0.1 \rangle$. Sampled value is *true*.
- PRIOR-SAMPLE returns the event [*true*, *false*, *true*, *true*].



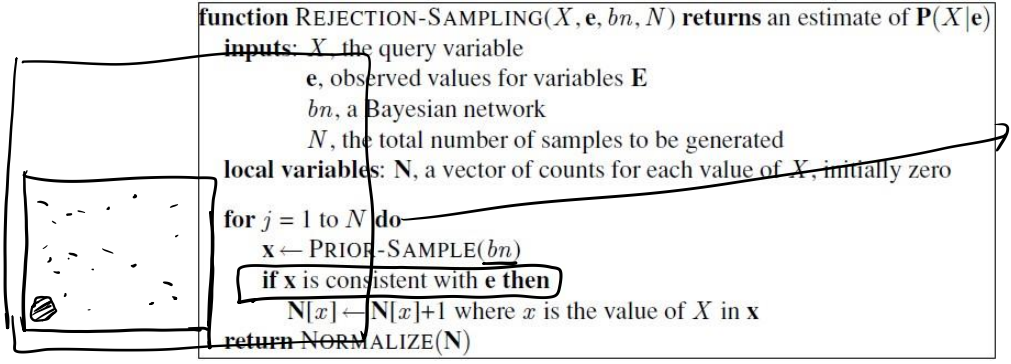
Rejection Sampling in Bayesian Networks



- Produce samples from a hard-to-sample distribution given an easy-to-sample distribution.
- Used to compute conditional probabilities $P(X \mid e)$.
- Rejection sampling produces a consistent estimate of the true probability.

Rejection Sampling in Bayesian Networks

- First, it generates samples from the prior distribution specified by the network. Then, it rejects all those that do not match the evidence. Finally, the estimate $\hat{P}(X \mid e)$ is obtained by counting how often $X = x$ occurs in the remaining samples.

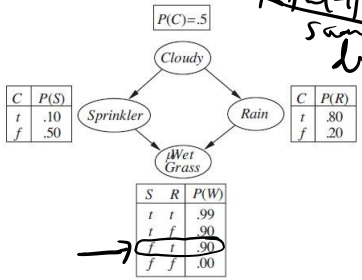


function REJECTION-SAMPLING(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network
    N, the total number of samples to be generated
  local variables: **N**, a vector of counts for each value of X, initially zero

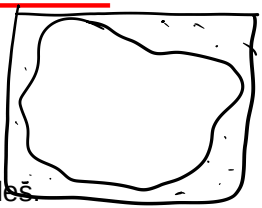
  for $j = 1$ to N **do**
    $\mathbf{x} \leftarrow$ PRIOR-SAMPLE(bn)
    if **x** is consistent with **e** **then**
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where x is the value of X in **x**
  return NORMALIZE(**N**)

Rejection Sampling - Example

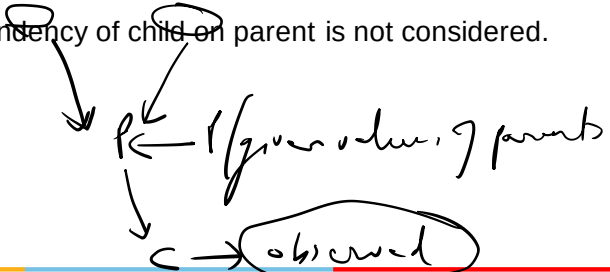
- Assume that we wish to estimate $P(Rain \mid Sprinkler = true)$, using 100 samples.
- Of the 100 that we generate, suppose that 73 have $Sprinkler = false$ and are rejected, while 27 have $Sprinkler = true$. ;
- Of the 27 accepted samples, 8 have $Rain = true$ and 19 have $Rain = false$. Hence, $P(Rain \mid Sprinkler = true) \approx Normalize(< 8, 19 >) = < 0.296, 0.704 >$.



Rejection Sampling - Issues



- Rejection sampling is expensive, as it rejects many samples.
- Unusable for complex problems.
- When the child is observed, the dependency of child on parent is not considered.



Likelihood Weighting in Bayesian Networks

- Likelihood weighting avoids the inefficiency of rejection sampling by generating only events that are consistent with the evidence *e*.
- It is a particular instance of the general statistical technique of **importance sampling**, tailored for inference in Bayesian networks.
- LIKELIHOOD WEIGHTING fixes the values for the evidence variables *E* and samples only the nonevidence variables. This guarantees that each event generated is consistent with the evidence.
- Each event is weighted by the likelihood that the event accords to the evidence, as measured by the product of the conditional probabilities for each evidence variable, given its parents. Events in which the actual evidence appears unlikely should be given less weight.

Likelihood Weighting in Bayesian Networks

function LIKELIHOOD-WEIGHTING(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$
    N, the total number of samples to be generated
  local variables: **W**, a vector of weighted counts for each value of X, initially zero

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

---

function WEIGHTED-SAMPLE(bn, **e**) **returns** an event and a weight
  $w \leftarrow 1$; **x** $\leftarrow$ an event with n elements initialized from **e**
  foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    if X_i is an evidence variable with value x_i in **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 **x**, w

justification for this

Likelihood Weighting - Example

- Assume Query is $P(Rain \mid Cloudy = true, WetGrass = true)$ and the ordering is [*Cloudy*, *Sprinkler*, *Rain*, *WetGrass*].
- First, the weight w is set to 1.0.
- Then an event is generated:
  1. *Cloudy* is an evidence variable with value *true*. Therefore, $w \leftarrow w \times P(Cloudy = true) = 0.5$
  2. *Sprinkler* is not an evidence variable, so sample from $P(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$; suppose this returns *false*.
  3. Sample from $P(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*.
  4. *WetGrass* is an evidence variable with value *true*. Therefore, $w \leftarrow w \times P(WetGrass = true \mid Sprinkler = false, Rain = true) = 0.45$.
- WEIGHTED-SAMPLE returns the event [*true*, *false*, *true*, *true*] with weight 0.45, and this is tallied under *Rain* = *true*.

$P(\text{WetGrass} : \text{True} \mid \text{Sprinkler} = \text{False}, \text{Rain} = \text{True})$

$P(\text{Cloudy} = \text{true})$

$= 0.5 \times 0.9$

$$\frac{1}{M}\sum f(x[i])\,\frac{p(x[i])}{\Delta x_i M} \qquad \int f(x)\,p(x)\,dx$$

Some theory about Importance Sampling

We would like to estimate the expectation of some function f with respect to a distribution $p(x)$

$$E_p(f) = \frac{1}{M}\sum_{m=1}^{M} f(x[m])$$

What if p is not known or is very difficult to sample from?

We use another distribution ϕ and adjust for it

Some theory about Importance Sampling

$$E_{p(x)}\left[f(x)\right] = E_{q(x)}\left[f(x)\frac{p(x)}{q(x)}\right] = \int f(x)\frac{p(x)}{q(x)}q(x)\,dx$$

We use the standard estimator for expectations relative to q. Let $D = \{x[1], x[2], \ldots x[M]\}$ be a set of samples drawn from q

$$\hat{E}_D(f) = \frac{1}{M}\sum_{m=1}^{M} f(x[m])\frac{p(x[m])}{q(x[m])}$$

Some theory about Importance Sampling

This is the unnormalized Importance Sampling estimator

$$w(x) = \frac{p(x)}{q(x)}$$

Proposition 12.1: For data set D sampled from q we have $E_D\left(\hat{E}_D(f)\right) = E_{q(x)}\left[f(x)w(x)\right] = E_{p(x)}\left(f(x)\right)$

Proof: $E_q\left[\frac{1}{M}\sum_{m=1}^{M} f(x[m])\frac{p(x[m])}{q(x[m])}\right] = \frac{1}{M}\sum_{m=1}^{M} E\left(f(x[m])\frac{p(x[m])}{q(x[m])}\right)$

Some theory on importance sampling

We need to take the expectation over the joint distribution on M-samples. The probability of selecting a particular sample $x[1], x[2] \ldots x[M]$ is $\prod_{i=1}^{M} q(x[i])$ and therefore

$$E_D\left(\hat{I}(f)\right) = \frac{1}{M} \sum_{x[1]} \sum_{x[2]} \cdots \sum_{x[M]} \left(\frac{f(x[1])P(x[1])}{q(x[1])} + \cdots \frac{f(x[M])P(x[M])}{q(x[M])} \right) \prod_{i=1}^{M} q(x[i])$$

$\hat{E}_D(f)$ $P(D)$

We can rewrite this expression as in the next slide

Some theory on importance sampling

$$= \frac{1}{M} \sum_{x[1]} \sum_{x[2]} \cdots \sum_{x[M]} f(x[1]) p(x[1]) \prod_{i=1, i \neq 1}^{M} Q(x[i])$$

$$+ \frac{1}{M} \sum_{x[1]} \sum_{x[2]} \cdots \sum_{x[M]} f(x[2]) p(x[2]) \prod_{i=1, i \neq 2}^{M} Q(x[i])$$

$$+ \cdots \frac{1}{M} \sum_{x[1]} \cdots \sum_{x[M]} f(x[M]) p(x[M]) \prod_{i=1, i \neq M}^{M} Q(x[i])$$

There are M terms in the above sum which are all very similar → they each evaluate to the same quantity.

Some theory on importance sampling

Each term can be simplified to a term like the following

$$\left[\sum_{x[1]} \frac{1}{M} f(x[1]) p(x[1])\right] \underbrace{\left[\sum_{x[2]} \sum_{x[3]} \cdots \sum_{x[M]} \prod_{i=2}^{M} q(x[i])\right]}_{\text{this evaluates to 1}}$$

We are left with $\sum_{x[1]} \frac{1}{M} f(x[1]) p(x[1]) = \frac{1}{M} E_{p(x)}(f(x))$

Thus we have $E_{p(x)}(f(x))$ overall

Note: this derivation is strictly valid for discrete RVs only

$Var(X+Y) = Var(X) + Var(Y)$ $\quad Var(kX) = k^2 Var(X)$

Some theory about Importance Sampling

Letting $\varepsilon_D = \hat{E}_D[F] - E_p[f(x)]$ we see that the distribution of ε_D is $N\left(0, \frac{\sigma_q^2}{M}\right)$ where

$$\sigma_q^2 = E_{q(x)}\left[(f(x)w(x))^2\right] - E^2_{q(x)}\left[f(x)w(x)\right] = Var(f(x)w(x))$$

Why is this true?

$\sigma_q^2 = Var\left(\hat{E}_D(F)\right) = \frac{1}{M^2} M \cdot Var\left[F(x)w(x)\right]$ since we

$\hat{E}_D(F)$ is the average of M independent samples $f(x_i)w(x_i)$

$$\sum_x P_B(x, e) = P_B(e) \neq 1$$

Normalized Importance Sampling

We do not have access to a distribution $P(x)$, only $\tilde{P}(x)$ which is not normalized

$$Z P(x) = \tilde{P}(x) \text{ where } Z \text{ is a normalizing constant}$$

In a Bayesian network $P(x)$ could be the posterior distribution $P_B(x/e)$ and $\tilde{P}(x)$ could be $P_B(x, e)$

Let us define $w(x) = \frac{\tilde{P}(x)}{q(x)}$

$$P(e) P_B(x/e) = P_B(x, e)$$

Normalized Importance Sampling

$$E_q\left(w(x)\right) = \int \frac{\tilde{p}(x)}{q(x)} q(x)\,dx \quad \text{or} \quad \sum \frac{\tilde{p}(x)}{q(x)} q(x)$$

$$= \int \tilde{p}(x)\,dx \quad \text{or} \quad \sum \tilde{p}(x) \rightarrow \text{both are equal to } Z$$

$$E_{p(x)}\left[f(x)\right] = \sum_x f(x)\,p(x) = \sum_x f(x)\,\frac{p(x)}{q(x)}\,q(x)$$

$$= \frac{1}{Z} \sum_x \frac{f(x)\,\tilde{p}(x)}{q(x)}\,q(x) = \frac{1}{Z} E_{q(x)}\left[f(x)w(x)\right]$$

Normalized Importance Sampling

$$E_{p(x)}[f(x)] = \sum_{m=1}^{M} \frac{f(x[m])\,w(x[m])}{\sum_{m=1}^{M} w(x[m])} \rightarrow ?$$ where

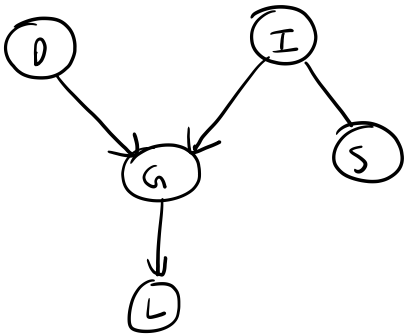
$$w(x) = \frac{\tilde{p}(x)}{q(x)} =$$

$x[1], x[2], \cdots x[m]$ are all samples drawn from $q(x)$

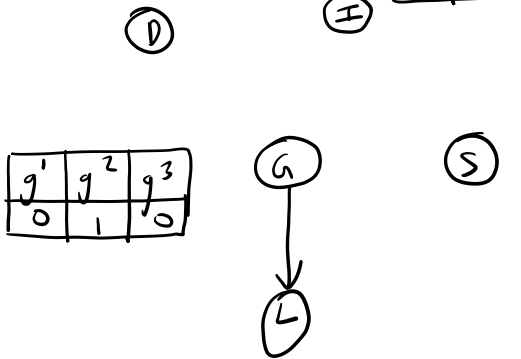
The bias of the normalized importance sampling estimator is not zero

G, I

Link to Likelihood Weighting



Original network B



Mutilated Network $B_{I=3}$

Link to Likelihood Weighting

Let ξ be a sample generated by the Likelihood Weighting algorithm and let w be its weight. Then the distribution over ξ is as defined by the network $B_{Z=z}$ and $w(\xi) = \frac{P_B(\xi)}{P_{B_{Z=z}}(\xi)} = \frac{\tilde{P}}{Q}$

Proof: Note that here $P_B \rightarrow \tilde{P}$ in the previous slides and $P_{B_{Z=z}}(\xi) \Rightarrow Q$ $\quad \frac{\tilde{P}}{Q}$

Link to Likelihood Weighting

Proof Continued

The only difference between sampling from $P_{B_{Z=z}}(\xi)$ and $P_B(\xi)$ concerns what happens when we draw a sample $z_i \in$ Evidence Variables / Parents(z_i)

In $P_{B_{Z=z}}(\xi)$, we draw this sample z_i with probability $=1$

In $P_B(\xi)$ we draw this sample with probability $P_B(z_i / \text{parent}(z_i))$

Link to Likelihood weighting

The weight in the Algorithm gets multiplied by

$$P_B(z_i / \text{parents}(z_i))$$

When z_i is not in evidence, $P_B(z_i / \text{parents}(z_i)) = P_{B_{z=z}}(z_i / \text{pare}$

So we have $\frac{1}{1}$.

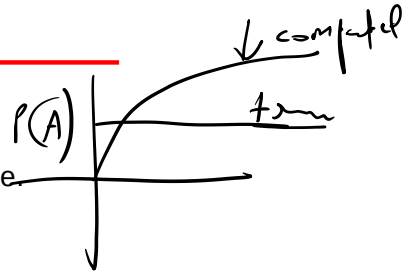
Multiplying the weights together, we get $w(\xi) = \frac{P_B(\xi)}{P_{B_{z=z}}(\xi)}$

Sampling-based Approximate Methods

- The methods using instantiations are generally known as **particle-based methods**.
- Each instantiation is known as a **particle**.
- Either create particles using a deterministic process, or sample particles from some distribution.
- Complete assignments to all of the network variables is commonly known as **full particles**.
- Disadvantage of full particle is that each particle covers only a very small part of the space.
- A collapsed particle specifies an assignment w only to some subset of the variables W, associating with it the conditional distribution $P(X \mid w)$.
- Assignments only to a subset $P(X \mid w)$ of variables of the network representing the conditional probability are commonly known as **collapsed particles**.

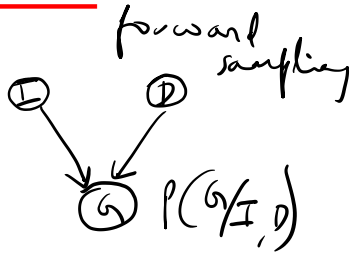
Table of Contents

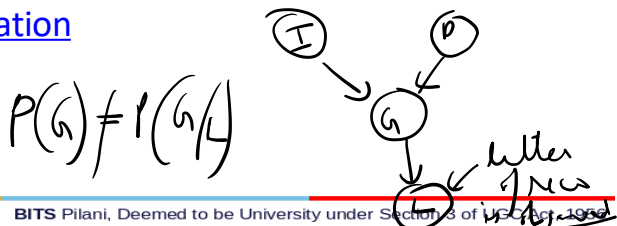
Propagation-Based Approximation



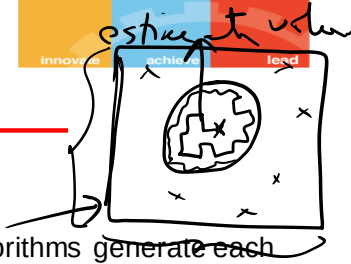
- Use the same message propagation as in exact inference.
- Use Cluster graph instead of clique tree.
- Use **Loopy belief propagation**.
- Message propagation process may not converge in two passes, since information from one pass will circulate and affect the next round.
- In some cases, the propagation of beliefs may not converge at all.
- Use Bethe cluster graph to eliminate the loops.

Table of Contents





Markov Chain Monte Carlo Methods



- Generate a **sequence** of samples.
- Instead of generating each sample from scratch, MCMC algorithms generate each sample by making a random change to the preceding sample.
- Think of an MCMC algorithm as being in a particular **current state** specifying a value for every variable and generating a **next state** by making random changes to the current state.
- Markov chain methods apply equally well to directed and to undirected models.
- Works for both Bayesian and Markov networks.
- A particular form of MCMC is called **Gibbs sampling**, which is especially well suited for Bayesian networks.

Gibbs Sampling Algorithm

- Starts out by generating a sample of the unobserved variables from some initial distribution.
- Starting from initial sample, iterate over each of the unobserved variables, sampling a new value for each variable given our current sample for all other variables.
- This allows information to **flow** across the network as each variable is sampled.
- The Gibbs sampling algorithm for Bayesian networks starts with an arbitrary state (with the evidence variables fixed at their observed values) and generates a next state by randomly sampling a value for one of the nonevidence variables X_i.
- The sampling for X_i is done conditioned on the current values of the variables in the Markov blanket of X_i.
- The algorithm therefore wanders randomly around the state space flipping one variable at a time, but keeping the evidence variables fixed.

Gibbs Sampling Algorithm

function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$
  local variables: **N**, a vector of counts for each value of X, initially zero
    Z, the nonevidence variables in bn
    x, the current state of the network, initially copied from **e**

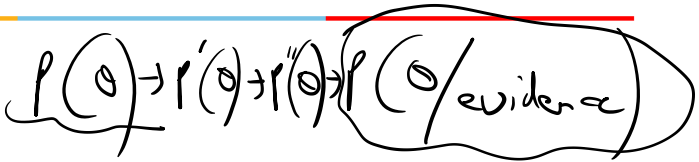
  initialize **x** with random values for the variables in **Z**
  for $j = 1$ to N **do**
    for each Z_i in **Z** **do**
      set the value of Z_i in **x** by sampling from $\mathbf{P}(Z_i|mb(Z_i))$
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in **x**
  return NORMALIZE(**N**)

not from Daphne-Koller

Gibbs Sampling - Example

- Assume Query is $P(Rain \mid Cloudy = true, WetGrass = true)$ and the ordering is [*Cloudy*, *Sprinkler*, *Rain*, *WetGrass*].
- The evidence variables *Sprinkler* and *WetGrass* are fixed to their observed values,
- The nonevidence variables *Cloudy* and *Rain* are initialized randomly; say to *true* and *false* respectively. The initial state is [*true*, *true*, *false*, *true*].
- The nonevidence variables are sampled repeatedly in an arbitrary order.
  1. *Cloudy* is sampled from $P(Cloudy \mid Sprinkler = true, Rain = false)$. Suppose the result is *Cloudy* = *false*. Then the new current state is [*false*, *true*, *false*, *true*].
  2. *Rain* is sampled from $P(Rain \mid Cloudy = false, Sprinkler = true, WetGrass = true)$. Suppose this yields *Rain* = *true*. The new current state is [*false*, *true*, *true*, *true*]
- 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 $Normalize(< 20, 60 >) = < 0.25, 0.75 >$.

Markov Chain Monte Carlo (MCMC)



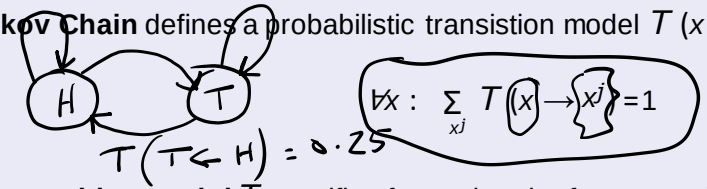
- MCMC framework generates samples from the posterior distribution, where we cannot efficiently sample from the posterior directly.
- Construct an iterative process that gradually samples from distributions that are closer and closer to the posterior.
- The number of iterations is to be detemined.

Markov Chain

- A Markov chain is defined in terms of a graph of states over which the sampling algorithm takes a random walk.

Definition

Markov Chain defines a probabilistic transistion model $T(x \rightarrow x^j)$ over a state x



The **transition model** T specifies for each pair of state x, x^j the probability $T(x \rightarrow x^j)$ of going from x to x^j. This transition probability applies whenever the chain is in state x.

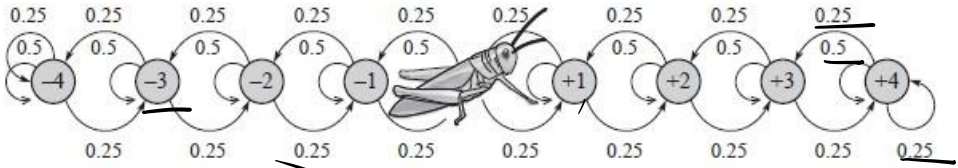
- MCMC is defined on **homogeneous** systems, where the system dynamics do not change over time.

Markov Chain

- Visualize the state space as a graph, with probability-weighted directed edges corresponding to transitions between different states.
- Generate a random sequence of states $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$.
- The state of the process at step t is random variable $X^{(t)}$.
- Initial state $X^{(0)}$ is distributed according to some initial state distribution $P^{(0)}(X^{(0)})$.
- Subsequent states are defined by distributions $P^{(1)}(X^{(1)}), P^{(2)}(X^{(2)}), \ldots$.

$$P^{(t+1)}(X^{t+1} = x^j) = \sum_{x \in Val(X)} P^{(t)}(X^t = x)\, T(x \rightarrow x^j)$$

Grasshopper Example



Time	-2	-1	0	1	2
$P^{(0)}$	0	0	1	0	0
$P^{(1)}$	0	0.25	0.5	0.25	0
$P^{(2)}$	0.25^2	$2 * 0.5 * 0.25$	$0.5^2 + 2 * 0.25^2$	$2 * 0.5 * 0.25$	0.25^2
	=0.0625	= 0.25	=0.375	=0.25	=0.0625

increasing time

Grasshopper Example

$$p^{(2)}(-1) = p'(-1)T(-1 \to -1) + p'(1)T(1 \to -1) + p'(0)T(0 \to -1)$$

(all these probabilities in p' are 0)

Now $T(1 \to -1) = 0$

$$\therefore p^2(-1) = (0.5)(0.25) + (0.5)(0.25)$$

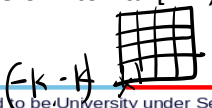
$$= 0.25$$

Markov Chain Monte Carlo (MCMC)

- Markov chain Monte carlo (MCMC) sampling is a process that mirrors the dynamics of the Markov chain.
- The sample $X^{(t)}$ is drawn from the distribution $P^{(t)}$.
- Find out whether $P^{(t)}$ converges, and if so, to what limit.

$$P^{(t+1)}(X^{t+1} = x^j) = \sum_{x \in Val(x)} P^{(t)}(X^t = x)\, T(x \rightarrow x^j)$$

- Apply on unifrom distribution.
- Expected time required for a chain to reach boundaries of interval $[-K, K]$ is K^2 steps.



Markov Chain Monte Carlo (MCMC)

Algorithm 12.5 Generating a Markov chain trajectory

Procedure MCMC-Sample (
  $P^{(0)}(\boldsymbol{X})$, // Initial state distribution
  $\mathcal{T}$, // Markov chain transition model
  T // Number of time steps
)

1 Sample $\boldsymbol{x}^{(0)}$ from $P^{(0)}(\boldsymbol{X})$
2 **for** $t = 1, \ldots, T$
3 Sample $\boldsymbol{x}^{(t)}$ from $\mathcal{T}(\boldsymbol{x}^{(t-1)} \rightarrow \boldsymbol{X})$
4 **return** $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T)}$

Stationary Distributions

- As MCMC process converges, $P^{(t+1)}$ will be close to $P^{(t)}$.

$$P^{(t)}(x^j) \approx P^{(t+1)}(X^{t+1} = x^j) = \sum_{x \in Val(X)} P^{(t)}(X^t = x) T(x \rightarrow x^j)$$

- The resulting distribution is $\Pi(x)$.
- At equilibrium, for any state x^j, $P^{(t+1)}$ will be almost equal to $P^{(t)}$.
- Example : Grasshopper Markov Chain

Stationary Distributions

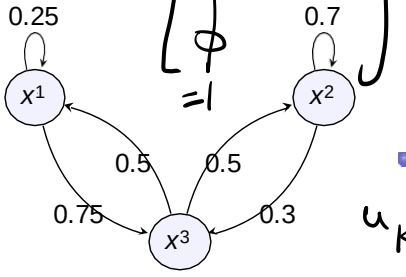
Definition (Stationary Distribution)

A distribution $\pi(x)$ is a stationary distribution for a Markov chain T if it satisfies

$$\pi(X = x^j) = \sum_{x \in Val(x)} \pi(X = x) T(x \to x^j)$$

A stationary distribution is also called an invariant distribution.

Stationary Distribution - Example



- Less uniform distribution.

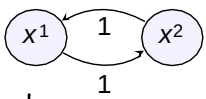
$$\pi(x^1) = 0.25\pi(x^1) + 0.5\pi(x^3)$$
$$\pi(x^2) = 0.7\pi(x^2) + 0.5\pi(x^3)$$
$$\pi(x^3) = 0.75\pi(x^1) + 0.3\pi(x^2)$$
$$\pi(x^1) + \pi(x^2) + \pi(x^3) = 1$$

- Unique solution

$$\pi(x^1) = 0.2$$
$$\pi(x^2) = 0.5$$
$$\pi(x^3) = 0.3$$

Periodic Markov Chains

- Markov Chain with two states x^1 and x^2



$$T(x^1 \rightarrow x^2) = 1$$

$$T(x^2 \rightarrow x^1) = 1$$

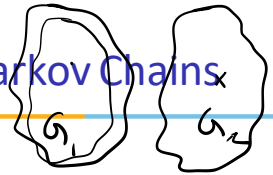
Let $P^{(0)}(x^1) = 1$

$$P^{(t)}(x^1) = 1 \text{ if } t \text{ is even}$$

$$P^{(t)}(x^2) = 1 \text{ if } t \text{ is odd}$$

- No convergence. So not a stationary distribution.
- Markov chains that exhibit a fixed cyclic behavior, are called **periodic Markov chains**.

Regular Markov Chains

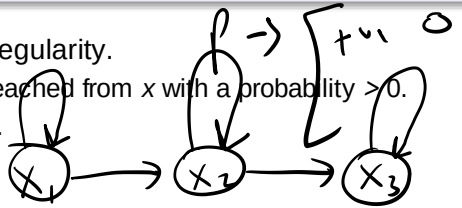


- A Markov chain is said to be regular if there exists some number k such that, for every x, $x^j \in Val(x)$, the probability of getting from x to x^j in exactly k steps is > 0.
- Example: Grasshopper Markov chain $\rightarrow$ $k = 9$.

Theorem

If a finite state Markov chain T is regular, then it has a unique stationary distribution.

- Two conditions that together guarantee regularity.
  - Every state are connected. x^j can be reached from x with a probability > 0.
  - For each state, there is a self-transition.



Using Markov Chains

- Goal: Compute $P(x \in S)$
  - P is too hard to sample from directly.
- Construct a regular Markov Chain T whose unique stationary distriution is P.
- Sample $x^{(0)}$ from some arbitrary distribution Q.
- For $t = 0, 1, 2, \ldots$
  - Generate $x^{(t+1)}$ from $T(x^{(t)} \rightarrow x^j)$.

Using Samples

- Once the MCMC finds equilibrium, all samples $x^{(t)}$ are from stationary distribution $\pi(x)$.
- Collect and use samples $x^{(t)}$.
- Ideally collect every 100^{th} sample, as nearby samples are correlated.

MCMC Algorithm I

- For $c = 1, 2, \ldots, C$ We are talking about C different Markov chains
  - Sample $x^{(0)}$ from arbitrary distribution Q (or Gibbs distribution can be used).
- Repeat until equilibrium
  - For $c = 1, 2, \ldots, C$
    - Generate $x^{(c,t+1)}$ from from $T(x^{(c,t)} \rightarrow x^{j})$.
  - Check for equilibrium
  - $t := t + 1$

MCMC Algorithm II

- Repeat until sufficient samples
  - Dataset D :=, empty
  - For $c = 1, 2, \ldots, C$ We have C different Markov chains
    - Generate $x^{(c,t+1)}$ from from T $(x^{(c,t)} \rightarrow x^{j})$.
    - $D := D \cup x^{(c,t+1)}$
  - $t := t + 1$
- Compute Expectation
  - Let $D := \{x[1], x[2], \ldots x[m]\}$
  - Estimate expectation

$$\hat{E}_{\mathbf{P}}[f] = \frac{1}{M} \sum_{i=1}^{M} f(x[m])$$