

430.457

Introduction to Intelligent Systems

Prof. Songhwai Oh
ECE, SNU

APPROXIMATE INFERENCE IN BAYESIAN NETWORKS

Monte Carlo

- Sampling algorithm
- Law of large number
- Useful when exact inference is intractable
- Applications:
 - Optimization (simulated annealing)
 - Numerical integration (posterior probability)
 - Sample generation

[Wikipedia] The modern version of the Monte Carlo method was invented in the late 1940s by Stanislaw Ulam, while he was working on nuclear weapon projects at the Los Alamos National Laboratory. It was named, by Nicholas Metropolis, after the **Monte Carlo Casino** in Monaco, where Ulam's uncle often gambled.



Direct Sampling (Ancestral Sampling)

An example of an event from a Bayesian network

1. Sample from $\mathbf{P}(\textit{Cloudy}) = \langle 0.5, 0.5 \rangle$, value is *true*.
2. Sample from $\mathbf{P}(\textit{Sprinkler} \mid \textit{Cloudy} = \textit{true}) = \langle 0.1, 0.9 \rangle$, value is *false*.
3. Sample from $\mathbf{P}(\textit{Rain} \mid \textit{Cloudy} = \textit{true}) = \langle 0.8, 0.2 \rangle$, value is *true*.
4. Sample from $\mathbf{P}(\textit{WetGrass} \mid \textit{Sprinkler} = \textit{false}, \textit{Rain} = \textit{true}) = \langle 0.9, 0.1 \rangle$, value is *true*.

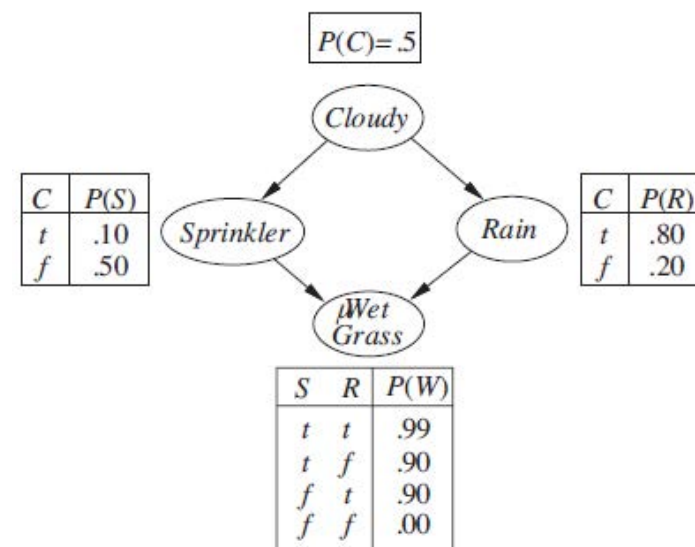
- Let $S_{PS}(x_1, \dots, x_n)$ be the probability that a specific event is generated. Then, we have

$$S_{PS}(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i \mid \textit{Parents}(X_i)) = P(x_1, \dots, x_n).$$

- Let $N_{PS}(x_1, \dots, x_n)$ be the number of times the event $x_1, \dots, x_n$ occurs. Then, by the law of large numbers,

$$\lim_{N \rightarrow \infty} \frac{1}{N} N_{PS}(x_1, \dots, x_n) = S_{PS}(x_1, \dots, x_n) = P(x_1, \dots, x_n).$$

It is a **consistent** estimate.



```
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
```

Rejection Sampling

```
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  
  foreach 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  $\mathbf{x}$ 
```

```
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  
     $\mathbf{x} \leftarrow$  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( $\mathbf{N}$ )
```

-
- Let $\hat{\mathbf{P}}$ be the estimated distribution from rejection sampling. Then we have

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

- Since $\lim_{N \rightarrow \infty} \frac{1}{N} N_{PS}(x_1, \dots, x_n) = P(x_1, \dots, x_n)$, $\mathbf{N}_{PS}(X, \mathbf{e}) \rightarrow \mathbf{P}(X, \mathbf{e})$ and $N_{PS}(\mathbf{e}) \rightarrow P(\mathbf{e})$ as $N \rightarrow \infty$.
- Hence,

$$\hat{\mathbf{P}}(X|\mathbf{e}) \approx \frac{\mathbf{P}(X, \mathbf{e})}{P(\mathbf{e})} = \mathbf{P}(X|\mathbf{e}).$$

- However, the algorithm gives poor performance if a large number of samples are rejected. For instance, if the number of evidence variables grows, the fraction of samples consistent with the evidence drops exponentially. Thus, rejection sampling is not suitable for complex problems.

Likelihood Weighting (Importance Sampling)

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**

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

$\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ 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 $w \leftarrow w \times P(X_i = x_i \mid \text{parents}(X_i))$

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

return $\mathbf{x}, w$

Example

Suppose that we want to compute $P(Rain|Cloudy = true, WetGrass = true)$. Set the weight $w = 1$. Then generate an event as follows while updating w .

1. *Cloudy* is an evidence variable with value *true*. Therefore, we set

$$w \leftarrow w \times P(Cloudy = true) = 0.5 .$$

2. *Sprinkler* is not an evidence variable, so sample from $\mathbf{P}(Sprinkler | Cloudy = true) = \langle 0.1, 0.9 \rangle$; suppose this returns *false*.
3. Similarly, sample from $\mathbf{P}(Rain | Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*.
4. *WetGrass* is an evidence variable with value *true*. Therefore, we set

$$w \leftarrow w \times P(WetGrass = true | Sprinkler = false, Rain = true) = 0.45 .$$

For this run, the algorithm returns an event $[true, false, true, true]$ with weight 0.45.

- Let $\mathbf{Z}$ be the nonevidence variables ($\mathbf{Z} = \mathbf{X} \setminus \mathbf{E}$).
- Each event is generated with probability $S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^l P(z_i | \text{parents}(Z_i))$.
- Weight is $w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^m P(e_i | \text{parents}(E_i))$.
- Hence,

$$S_{WS}(\mathbf{z}, \mathbf{e})w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^l P(z_i | \text{parents}(Z_i)) \prod_{i=1}^m P(e_i | \text{parents}(E_i)) = P(\mathbf{z}, \mathbf{e}) \leftarrow$$

- The estimated posterior distribution is:

$$\begin{aligned} \hat{P}(x | \mathbf{e}) &= \alpha \sum_{\mathbf{y}} N_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) && \text{from LIKELIHOOD-WEIGHTING} \\ &\approx \alpha' \sum_{\mathbf{y}} S_{WS}(x, \mathbf{y}, \mathbf{e}) w(x, \mathbf{y}, \mathbf{e}) && \text{for large } N \\ &= \alpha' \sum_{\mathbf{y}} P(x, \mathbf{y}, \mathbf{e}) && \text{by Equation } \leftarrow \\ &= \alpha' P(x, \mathbf{e}) = P(x | \mathbf{e}) . \end{aligned}$$

Markov Chain Monte Carlo (MCMC)

- Generate samples by executing a Markov chain
- Algorithms:
 - Metropolis
 - Metropolis-Hastings
 - Gibbs Sampling
- General structure of MCMC
 - Iterate
 - Propose a new state
 - Go to a new state with some probability, otherwise, stay at the current state, i.e., it simulates a Markov chain.
 - Output desired sample means

Gibbs Sampling

function GIBBS-ASK(X, e, bn, N) **returns** an estimate of $P(X|e)$

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**

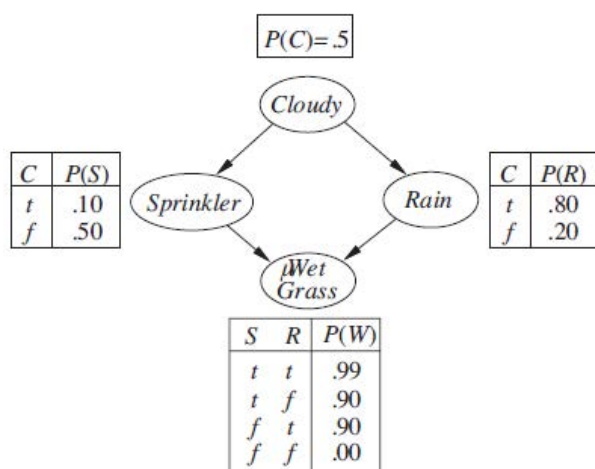
for each Z_i in Z **do**

 set the value of Z_i in x by sampling from $P(Z_i|mb(Z_i))$

$N[x] \leftarrow N[x] + 1$ where x is the value of X in x

return NORMALIZE(N)

Markov blanket



Query: $P(Rain|Sprinkler = true, WetGrass = true)$.

Initial state: $[true, true, false, true]$.

- Sample *Cloudy* from $P(Cloudy|Sprinkler = true, Rain = false)$. *Cloudy* = *false*. New state: $[false, true, false, true]$.
- Sample *Rain* from $P(Rain|Cloudy = false, Sprinkler = true, WetGrass = true)$. *Rain* = *true*. New state: $[false, true, true, true]$.

If we have visited 20 states with *Rain* = *true* and 60 states with *Rain* = *false*.
Then we can estimate that

$$P(Rain|Sprinkler = true, WetGrass = true) = \langle 0.25, 0.75 \rangle.$$

Some Facts about Markov Chains

- Let $q(\mathbf{x} \rightarrow \mathbf{x}')$ be the transition probability from $\mathbf{x}$ to $\mathbf{x}'$.
- Let $\pi_t(\mathbf{x})$ be the probability that the Markov chain is in state $\mathbf{x}$ at time t .

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

- The Markov chain reaches its **stationary distribution** if $\pi_t = \pi_{t+1}$. π is a stationary distribution of the Markov chain if

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

- **Fact:** If the chain is **ergodic**, i.e., every state is reachable from every other state and there are no periodic cycles, then there exists a unique stationary distribution.
- **Detailed balance:** $\pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') = \pi(\mathbf{x}') q(\mathbf{x}' \rightarrow \mathbf{x})$ for all $\mathbf{x}$ and $\mathbf{x}'$
- Detailed balance implies stationarity:

$$\sum_{\mathbf{x}} \pi(\mathbf{x}) q(\mathbf{x} \rightarrow \mathbf{x}') = \sum_{\mathbf{x}} \pi(\mathbf{x}') q(\mathbf{x}' \rightarrow \mathbf{x}) = \pi(\mathbf{x}') \sum_{\mathbf{x}} q(\mathbf{x}' \rightarrow \mathbf{x}) = \pi(\mathbf{x}')$$

Why Gibbs Sampling Works

Gibbs sampling satisfies the detailed balance condition with stationary distribution $P(\mathbf{x}|\mathbf{e})$.

- Gibbs sampling: $q_i(\mathbf{x} \rightarrow \mathbf{x}') = q_i((x_i, \bar{\mathbf{x}}_i) \rightarrow (x'_i, \bar{\mathbf{x}}_i)) = P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e})$.

$$\begin{aligned}\pi(\mathbf{x})q_i(\mathbf{x} \rightarrow \mathbf{x}') &= P(\mathbf{x} | \mathbf{e})P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e}) = P(x_i, \bar{\mathbf{x}}_i | \mathbf{e})P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e}) \\ &= P(x_i | \bar{\mathbf{x}}_i, \mathbf{e})P(\bar{\mathbf{x}}_i | \mathbf{e})P(x'_i | \bar{\mathbf{x}}_i, \mathbf{e}) && \text{(using the chain rule on the first term)} \\ &= P(x_i | \bar{\mathbf{x}}_i, \mathbf{e})P(x'_i, \bar{\mathbf{x}}_i | \mathbf{e}) && \text{(using the chain rule backward)} \\ &= \pi(\mathbf{x}')q_i(\mathbf{x}' \rightarrow \mathbf{x}).\end{aligned}$$

- Hence, the detailed balance condition is satisfied and the stationary distribution is the desired posterior $P(\mathbf{x}|\mathbf{e})$.
- Now if we make sure that the chain is ergodic, then $P(\mathbf{x}|\mathbf{e})$ is the unique stationary distribution and the Markov chain converges to this distribution from any initial state.