UIT2504 Artificial Intelligence

Bayesian Networks

C. Aravindan
<AravindanC@ssn.edu.in>

Professor of Information Technology
SSN College of Engineering

October 28, 2024

Introduction

- We have seen that using full joint distribution as our knowledge base is an intractable idea

Introduction

- We have seen that using full joint distribution as our knowledge base is an intractable idea
- We have also seen that independence, conditional independence, and Bayes rule can help in reducing the complexity

Introduction

- We have seen that using full joint distribution as our knowledge base is an intractable idea
- We have also seen that independence, conditional independence, and Bayes rule can help in reducing the complexity
- This is captured in a graphical data structure called as *Bayesian Networks*

Introduction

- We have seen that using full joint distribution as our knowledge base is an intractable idea
- We have also seen that independence, conditional independence, and Bayes rule can help in reducing the complexity
- This is captured in a graphical data structure called as *Bayesian Networks*
- The network captures the dependencies among the random variables and stores only the necessary conditional probability distributions called as *conditional proabability table (CPT)*

Introduction

- We have seen that using full joint distribution as our knowledge base is an intractable idea
- We have also seen that independence, conditional independence, and Bayes rule can help in reducing the complexity
- This is captured in a graphical data structure called as *Bayesian Networks*
- The network captures the dependencies among the random variables and stores only the necessary conditional probability distributions called as *conditional proability table (CPT)*
- Exact and approximate inferences can be carried out to answer a query

Simple Bayesian Network

- Basically, the nodes represent random variables and the edges represent the causal dependency of a variable on another

Simple Bayesian Network

- Basically, the nodes represent random variables and the edges represent the causal dependency of a variable on another
- For example, *Weather* is independent of any of the dentist variables
- Given *Cavity*, *Toothache* is conditionally independent of *Catch*

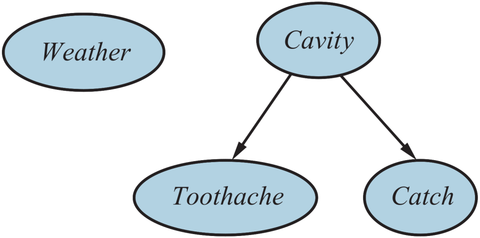


Figure: Simple Bayesian Network

Typical Bayesian Network

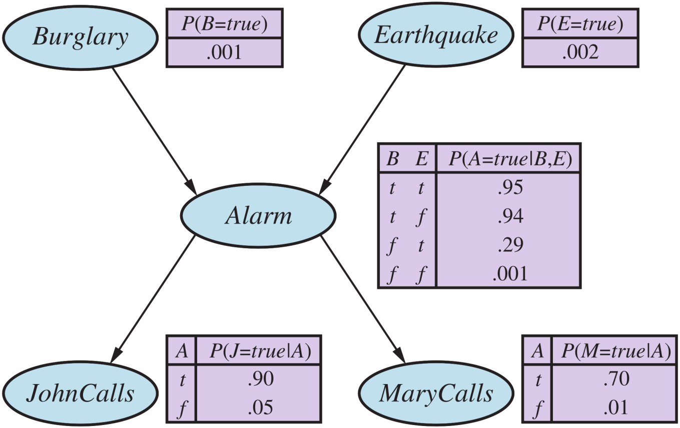


Figure: Typical Bayesian Network

Bayesian Network — Syntax

- Each node corresponds to a random variable (discrete or continuous)
- Each edge, say from X to Y, represents causal relationship (X causes Y). X is said to be a parent of Y.
- The graph contains no cycles and hence it is a *directed acyclic graph (DAG)*
- Each node X_i is associated with probability information $\theta(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node using a finite number of parameters (in discrete case, it is a CPT)
- Each row in a CPT contains the conditional probability of each node value for a *conditioning case*

Semantics of Bayesian Networks

- Actually a BN with n nodes (random variables) represents the full joint probability distributions of these n variables

Semantics of Bayesian Networks

- Actually a BN with n nodes (random variables) represents the full joint probability distributions of these n variables

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

Semantics of Bayesian Networks

- Actually a BN with n nodes (random variables) represents the full joint probability distributions of these n variables

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

- Here, $parents(X_i)$ stands for the values of $Parents(X_i)$ that appear in $(x_1, \ldots, x_n)$

Semantics of Bayesian Networks

- Actually a BN with n nodes (random variables) represents the full joint probability distributions of these n variables

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

- Here, $parents(X_i)$ stands for the values of $Parents(X_i)$ that appear in $(x_1, \ldots, x_n)$
- Consider: Alarm has sounded, but neither a burglary nor an earthquake has occurred, but both John and Mary call

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j|a)P(m|a)P(a|\neg b \wedge \neg e)P(\neg b)P(\neg e) \\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628 \end{aligned}$$

Semantics of Bayesian Networks

- Actually a BN with n nodes (random variables) represents the full joint probability distributions of these n variables

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

- Here, $parents(X_i)$ stands for the values of $Parents(X_i)$ that appear in $(x_1, \dots, x_n)$
- Consider: Alarm has sounded, but neither a burglary nor an earthquake has occurred, but both John and Mary call

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j|a)P(m|a)P(a|\neg b \wedge \neg e)P(\neg b)P(\neg e) \\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628 \end{aligned}$$

- It can be shown that $\theta(x_i|parents(X_i)) = P(x_i|parents(X_i))$

Semantics of Bayesian Networks

- Actually a BN with n nodes (random variables) represents the full joint probability distributions of these n variables

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

- Here, $parents(X_i)$ stands for the values of $Parents(X_i)$ that appear in $(x_1, \ldots, x_n)$
- Consider: Alarm has sounded, but neither a burglary nor an earthquake has occurred, but both John and Mary call

$$\begin{aligned} P(j, m, a, \neg b, \neg e) &= P(j|a)P(m|a)P(a|\neg b \wedge \neg e)P(\neg b)P(\neg e) \\ &= 0.90 \times 0.70 \times 0.001 \times 0.999 \times 0.998 = 0.000628 \end{aligned}$$

- It can be shown that $\theta(x_i|parents(X_i)) = P(x_i|parents(X_i))$
- So, when a BN is constructed, the CPT should be the actual conditional proababilities

Topological order

- A full joint probability can be expressed in terms of conditional probability using *chain rule*

$$
\begin{aligned}
P(x_1, \ldots, x_n) &= P(x_n | x_{n-1}, \ldots, x_1) P(x_{n-1} | x_{n-2}, \ldots, x_1) \ldots P(x_2 | x_1) P(x_1
\\
&= \prod_{i=1}^{n} P(x_i | x_{i-1}, \ldots, x_1)
\end{aligned}
$$

Topological order

- A full joint probability can be expressed in terms of conditional probability using *chain rule*

$$P(x_1, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}|x_{n-2}, \ldots, x_1) \ldots P(x_2|x_1)P(x_1$$
$$= \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1)$$

- In the case of Bayesian Network, this equation becomes

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

provided that $Parents(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$

Topological order

- A full joint probability can be expressed in terms of conditional probability using *chain rule*

$$P(x_1, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}|x_{n-2}, \ldots, x_1) \ldots P(x_2|x_1)P(x_1$$
$$= \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1)$$

- In the case of Bayesian Network, this equation becomes

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

provided that $Parents(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$

- Since Bayesian Network is a DAG, the nodes (random variables) can be listed in a *topological order*

Topological order

- A full joint probability can be expressed in terms of conditional probability using *chain rule*

$$P(x_1, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}|x_{n-2}, \ldots, x_1) \ldots P(x_2|x_1)P(x_1$$
$$= \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1)$$

- In the case of Bayesian Network, this equation becomes

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

provided that $Parents(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$
- Since Bayesian Network is a DAG, the nodes (random variables) can be listed in a *topological order*
- Example, (B, E, A, J, M); (E, B, A, M, J); etc.

Topological order

- A full joint probability can be expressed in terms of conditional probability using *chain rule*

$$P(x_1, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1)P(x_{n-1}|x_{n-2}, \ldots, x_1) \ldots P(x_2|x_1)P(x_1$$
$$= \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1)$$

- In the case of Bayesian Network, this equation becomes

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

provided that $Parents(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$

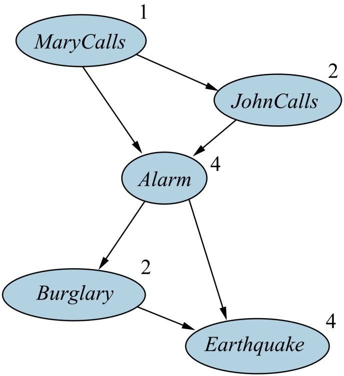
- Since Bayesian Network is a DAG, the nodes (random variables) can be listed in a *topological order*
- Example, (B, E, A, J, M); (E, B, A, M, J); etc.
- *Bayesian Network is a correct representation of a domain only if each node is conditionally independent of its other predecessors in the ordering, given its parents*

Network Structure

- It is important that the network edges represent the causal direction
- We may end up with a complicated network otherwise
- For example, if we consider the topological order ($MaryCalls$, $JohnCalls$, $Alarm$, $Burglary$, $Earthquake$), we may get the following complicated network

Network Structure

- It is important that the network edges represent the causal direction
- We may end up with a complicated network otherwise
- For example, if we consider the topological order ($MaryCalls$, $JohnCalls$, $Alarm$, $Burglary$, $Earthquake$), we may get the following complicated network



Network Structure

- Representing a domain with a wrong topological order may give bad results
- For example, consider the topological order $(MaryCalls, JohnCalls, Earthquake, Burglary, Alarm)$

Network Structure

- Representing a domain with a wrong topological order may give bad results
- For example, consider the topological order ($MaryCalls$, $JohnCalls$, $Earthquake$, $Burglary$, $Alarm$)

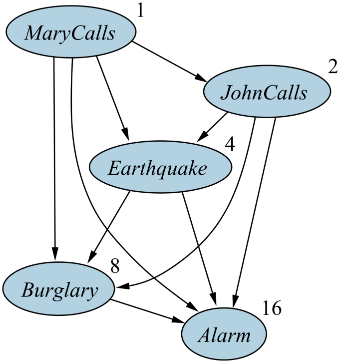


Figure: Bad Bayesian Network Structure

Network Properties

- Non-descendants property: Each variable is conditionally independent of its non-descendants, given its parents

- Non-descendants property: Each variable is conditionally independent of its non-descendants, given its parents

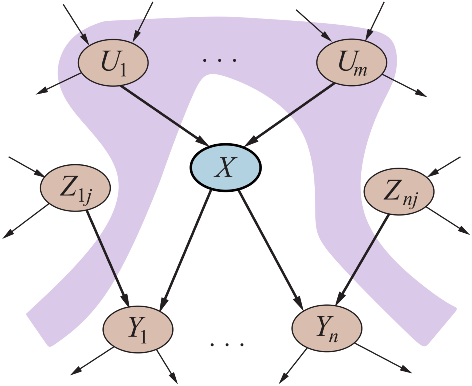


Figure: Conditional Independence Property

Network Properties

- 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*

Network Properties

- 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*

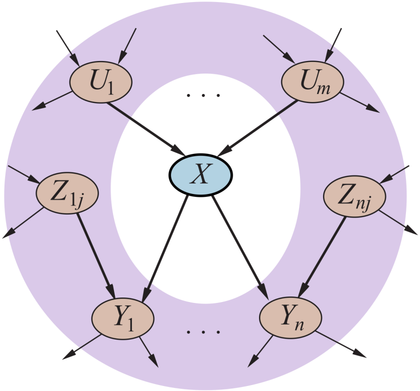


Figure: Conditional Independence Property

- We have seen how to calculate joint probability from a Bayesian Network

Inferences in Bayesian Networks

- We have seen how to calculate joint probability from a Bayesian Network
- A typical inference task requires us to calculate probability distribution of a query variable X, given a set of evidence variables $\mathbf{E} = \{E_1, \ldots, E_m\}$

Inferences in Bayesian Networks

- We have seen how to calculate joint probability from a Bayesian Network
- A typical inference task requires us to calculate probability distribution of a query variable X, given a set of evidence variables $\mathbf{E} = \{E_1, \ldots, E_m\}$
- Let $\mathbf{e}$ denote a particular evidence and $\mathbf{Y}$ denote the hidden variables $Y_1, \ldots, Y_l$

Inferences in Bayesian Networks

- We have seen how to calculate joint probability from a Bayesian Network
- A typical inference task requires us to calculate probability distribution of a query variable X, given a set of evidence variables $\mathbf{E} = \{E_1, \ldots, E_m\}$
- Let $\mathbf{e}$ denote a particular evidence and $\mathbf{Y}$ denote the hidden variables $Y_1, \ldots, Y_l$
- Now, the query needs to be answered is $\mathbf{P}(X|\mathbf{e})$

Inferences in Bayesian Networks

- We have seen how to calculate joint probability from a Bayesian Network
- A typical inference task requires us to calculate probability distribution of a query variable X, given a set of evidence variables $\mathbf{E} = \{E_1, \ldots, E_m\}$
- Let $\mathbf{e}$ denote a particular evidence and $\mathbf{Y}$ denote the hidden variables $Y_1, \ldots, Y_l$
- Now, the query needs to be answered is $\mathbf{P}(X|\mathbf{e})$
- For example, compute the probability distribution of *Burglary*, given $JohnCalls = True \wedge MaryCalls = True$

Inferences in Bayesian Networks

- Recall that given full joint distributions, the query can be answered using

$$\mathbf{P}(X|\mathbf{e}) = \alpha \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$

- Recall that given full joint distributions, the query can be answered using
$$\mathbf{P}(X|\mathbf{e}) = \alpha\mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$
- As we have already seen, terms such as $P(x, \mathbf{e}, \mathbf{y})$ can be written as products of conditional probabilities from the network

Inferences in Bayesian Networks

- Recall that given full joint distributions, the query can be answered using

$$\mathbf{P}(X|\mathbf{e}) = \alpha\mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$

- As we have already seen, terms such as $P(x, \mathbf{e}, \mathbf{y})$ can be written as products of conditional probabilities from the network
- Therefore, a query can be answered using a Bayes net by computing sums of products of conditional probabilities

Inference Example

- Consider the query $\mathbf{P}(B|j \wedge m)$. The hidden variables are E and A

Inference Example

- Consider the query $\mathbf{P}(B|j \wedge m)$. The hidden variables are E and A
- This can be computed as

$$\mathbf{P}(B|j, m) = \alpha \mathbf{P}(B, j, m) = \alpha \sum_{e} \sum_{a} \mathbf{P}(B, j, m, e, a)$$

Inference Example

- Consider the query $\mathbf{P}(B|j \wedge m)$. The hidden variables are E and A
- This can be computed as

$$\mathbf{P}(B|j,m) = \alpha\mathbf{P}(B,j,m) = \alpha \sum_{e} \sum_{a} \mathbf{P}(B,j,m,e,a)$$

- This gets simplified as (for the case $B = True$)

$$P(b|j,m) = \alpha \sum_{e} \sum_{a} P(b)P(e)P(a|b,e)P(j|a)P(m|a)$$

Inference Example

- Consider the query $\mathbf{P}(B|j \wedge m)$. The hidden variables are E and A
- This can be computed as

$$\mathbf{P}(B|j, m) = \alpha \mathbf{P}(B, j, m) = \alpha \sum_{e} \sum_{a} \mathbf{P}(B, j, m, e, a)$$

- This gets simplified as (for the case $B = True$)

$$P(b|j, m) = \alpha \sum_{e} \sum_{a} P(b)P(e)P(a|b, e)P(j|a)P(m|a)$$

- To compute this, we need to add four terms, each computed by multiplying five numbers

Inference Example

- Consider the query $\mathbf{P}(B|j \wedge m)$. The hidden variables are E and A
- This can be computed as

$$\mathbf{P}(B|j,m) = \alpha \mathbf{P}(B,j,m) = \alpha \sum_e \sum_a \mathbf{P}(B,j,m,e,a)$$

- This gets simplified as (for the case $B = True$)

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

- To compute this, we need to add four terms, each computed by multiplying five numbers
- In the worst case, we can have $O(2^n)$ terms in the sum, each a product of $O(n)$ probability values. So, the complexity is $O(n2^n)$

Inference Example

- Consider the query $\mathbf{P}(B|j \wedge m)$. The hidden variables are E and A
- This can be computed as

$$\mathbf{P}(B|j, m) = \alpha \mathbf{P}(B, j, m) = \alpha \sum_e \sum_a \mathbf{P}(B, j, m, e, a)$$

- This gets simplified as (for the case $B = True$)

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

- To compute this, we need to add four terms, each computed by multiplying five numbers
- In the worst case, we can have $O(2^n)$ terms in the sum, each a product of $O(n)$ probability values. So, the complexity is $O(n2^n)$
- A small optimization is possible

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

Structure of Computation

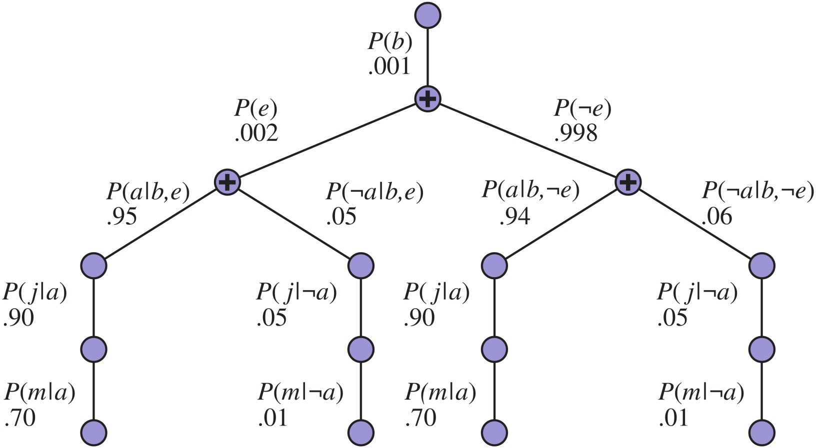


Figure: Structure of computation

Exact Inference Procedure

```
function ENUMERATION-ASK(X, e, bn) returns a distribution over X
  inputs: X,  the query variable
          e, observed values for variables E
          bn, a Bayes net with variables vars

  Q(X) ← a distribution over X, initially empty
  for each value x_i of X do
      Q(x_i) ← ENUMERATE-ALL(vars, e_{x_i})
          where e_{x_i} is e extended with X = x_i
  return NORMALIZE(Q(X))

function ENUMERATE-ALL(vars, e) returns a real number
  if EMPTY?(vars) then return 1.0
  V ← FIRST(vars)
  if V is an evidence variable with value v in e
      then return P(v | parents(V)) × ENUMERATE-ALL(REST(vars), e)
      else return ∑_v P(v | parents(V)) × ENUMERATE-ALL(REST(vars), e_v)
          where e_v is e extended with V = v
```

Figure: Exact inference in Bayes Bet

Inference in BN is *NP-hard*

- Inference in BN is *NP-hard*, since 3-SAT problems can be encoded as inference problem in BN

Inference in BN is *NP-hard*

- Inference in BN is *NP-hard*, since 3-SAT problems can be encoded as inference problem in BN
- For example, $(W \vee X \vee Y) \wedge (\neg W \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z)$ can be encoded as follows

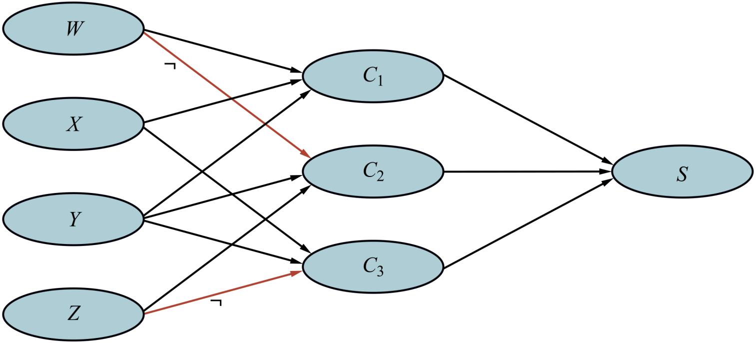


Figure: BN that represents a 3-SAT problem

Node clustering

- Related nodes may be merged to form *meganodes*

Node clustering

- Related nodes may be merged to form *meganodes*

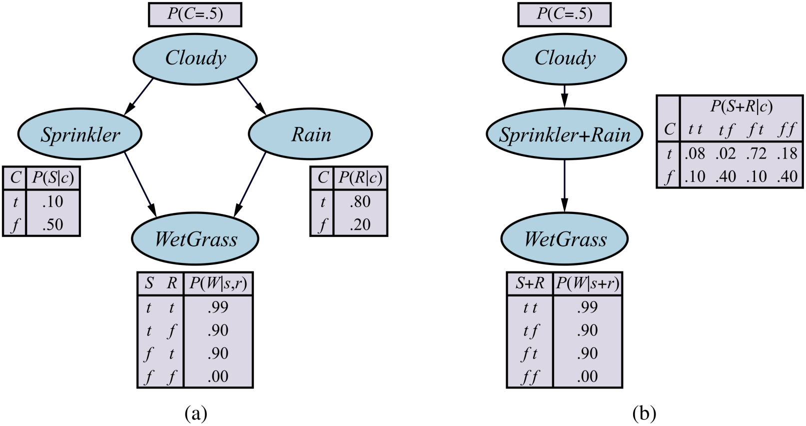


Figure: Clustering of related nodes into a meganode

Approximate inference in BN

- Approximate inferences from Bayesian Networks can be carried out using randomized sampling algorithms, also called as *Monte Carlo* algorithms

Approximate inference in BN

- Approximate inferences from Bayesian Networks can be carried out using randomized sampling algorithms, also called as *Monte Carlo* algorithms
- Accuracy of such algorithms depends on the number of samples generated

Approximate inference in BN

- Approximate inferences from Bayesian Networks can be carried out using randomized sampling algorithms, also called as *Monte Carlo* algorithms
- Accuracy of such algorithms depends on the number of samples generated
- Idea: To find probability of a proposision ϕ, generate N samples and return the ratio of samples in which ϕ holds to total samples N

Approximate inference in BN

- Approximate inferences from Bayesian Networks can be carried out using randomized sampling algorithms, also called as *Monte Carlo* algorithms
- Accuracy of such algorithms depends on the number of samples generated
- Idea: To find probability of a proposision ϕ, generate N samples and return the ratio of samples in which ϕ holds to total samples N
- We will look at two kinds of sampling methods: Direct sampling, and Markov Chain sampling

Generating random samples

- How to generate random samples from known probability distributions?

Generating random samples

- How to generate random samples from known probability distributions?
- Consider the random variable *Weather* with values $\langle \mathit{sun}, \mathit{rain}, \mathit{cloud}, \mathit{snow} \rangle$, with probability distribution $\langle 0.6, 0.1, 0.29, 0.01 \rangle$

Generating random samples

- How to generate random samples from known probability distributions?
- Consider the random variable *Weather* with values $\langle sun, rain, cloud, snow \rangle$, with probability distribution $\langle 0.6, 0.1, 0.29, 0.01 \rangle$
- Construct cumulative distribution: $\langle 0.6, 0.7, 0.99, 1.0 \rangle$

Generating random samples

- How to generate random samples from known probability distributions?
- Consider the random variable *Weather* with values $\langle sun, rain, cloud, snow \rangle$, with probability distribution $\langle 0.6, 0.1, 0.29, 0.01 \rangle$
- Construct cumulative distribution: $\langle 0.6, 0.7, 0.99, 1.0 \rangle$
- Generate a random number r uniformly distributed in the range $[0, 1]$

Generating random samples

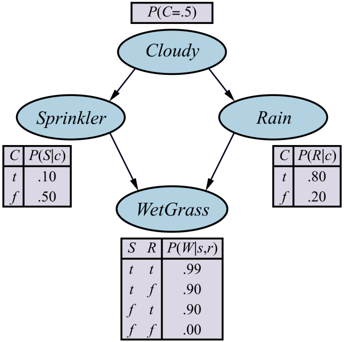
- How to generate random samples from known probability distributions?
- Consider the random variable *Weather* with values $\langle sun, rain, cloud, snow \rangle$, with probability distribution $\langle 0.6, 0.1, 0.29, 0.01 \rangle$
- Construct cumulative distribution: $\langle 0.6, 0.7, 0.99, 1.0 \rangle$
- Generate a random number r uniformly distributed in the range $[0, 1]$
- Return the first value whose cumulative probability exceeds r

Generating random samples

- How to generate random samples from known probability distributions?
- Consider the random variable *Weather* with values $\langle sun, rain, cloud, snow \rangle$, with probability distribution $\langle 0.6, 0.1, 0.29, 0.01 \rangle$
- Construct cumulative distribution: $\langle 0.6, 0.7, 0.99, 1.0 \rangle$
- Generate a random number r uniformly distributed in the range $[0, 1]$
- Return the first value whose cumulative probability exceeds r
- For example, if $r = 0.77$, we return *cloud*, and if $r = 0.46$, we return *sun*

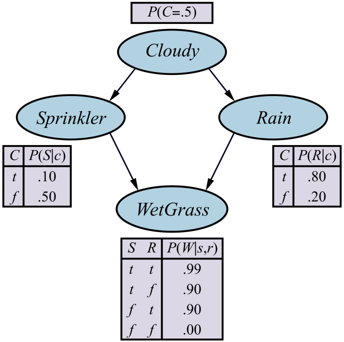
Direct sampling

- Let us directly generate random sample from the following BN
- The random variables will be sampled in a topological order



Direct sampling

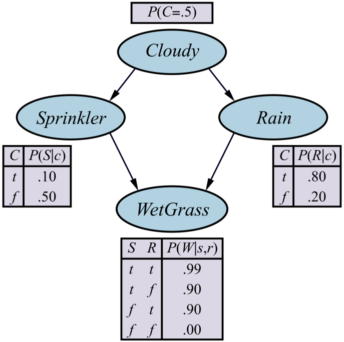
- Let us directly generate random sample from the following BN
- The random variables will be sampled in a topological order



- $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$, $r = 0.4$, value is *true*

Direct sampling

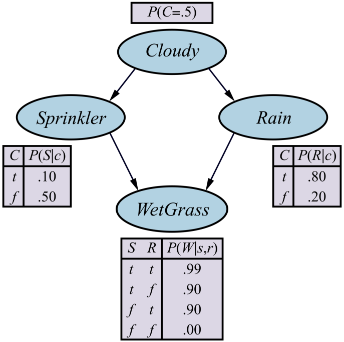
- Let us directly generate random sample from the following BN
- The random variables will be sampled in a topological order



- $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$, $r = 0.4$, value is *true*
- $\mathbf{P}(Sprinkler | Cloudy = true) = \langle 0.1, 0.9 \rangle$, $r = 0.75$, value is *false*

Direct sampling

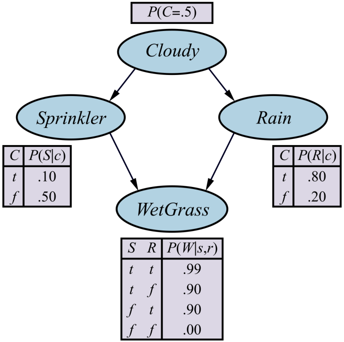
- Let us directly generate random sample from the following BN
- The random variables will be sampled in a topological order



- $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$, $r = 0.4$, value is *true*
- $\mathbf{P}(Sprinkler | Cloudy = true) = \langle 0.1, 0.9 \rangle$, $r = 0.75$, value is *false*
- $\mathbf{P}(Rain | Cloudy = true) = \langle 0.8, 0.2 \rangle$, $r = 0.28$, value is *true*

Direct sampling

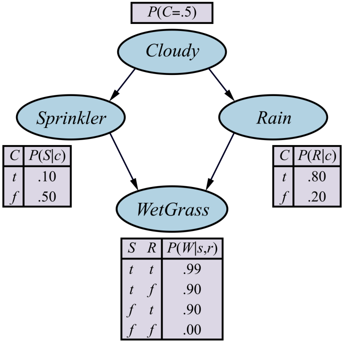
- Let us directly generate random sample from the following BN
- The random variables will be sampled in a topological order



- $\mathbf{P}(Cloudy) = \langle 0.5, 0.5\rangle$, $r = 0.4$, value is *true*
- $\mathbf{P}(Sprinkler|Cloudy = true) = \langle 0.1, 0.9\rangle$, $r = 0.75$, value is *false*
- $\mathbf{P}(Rain|Cloudy = true) = \langle 0.8, 0.2\rangle$, $r = 0.28$, value is *true*
- $\mathbf{P}(WetGrass|Sprinkler = false, Rain = true) = \langle 0.9, 0.1\rangle$, $r = 0.56$, value is *true*

Direct sampling

- Let us directly generate random sample from the following BN
- The random variables will be sampled in a topological order



- $\mathbf{P}(\mathit{Cloudy}) = \langle 0.5, 0.5 \rangle$, $r = 0.4$, value is *true*
- $\mathbf{P}(\mathit{Sprinkler} | \mathit{Cloudy} = \mathit{true}) = \langle 0.1, 0.9 \rangle$, $r = 0.75$, value is *false*
- $\mathbf{P}(\mathit{Rain} | \mathit{Cloudy} = \mathit{true}) = \langle 0.8, 0.2 \rangle$, $r = 0.28$, value is *true*
- $\mathbf{P}(\mathit{WetGrass} | \mathit{Sprinkler} = \mathit{false}, \mathit{Rain} = \mathit{true}) = \langle 0.9, 0.1 \rangle$, $r = 0.56$, value is *true*

- The random event sampled is $[\mathit{true}, \mathit{false}, \mathit{true}, \mathit{true}]$

Prior Sampling Algorithm

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

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

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

Direct Sampling is Consistent

- Let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by prior sampling algorithm

Direct Sampling is Consistent

- Let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by prior sampling algorithm
- By the construct of the algorithm, we have

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

Direct Sampling is Consistent

- Let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by prior sampling algorithm
- By the construct of the algorithm, we have

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

- From the properties of BN, we have

$$S_{PS}(x_1, \ldots, x_n) = P(x_1, \ldots, x_n)$$

Direct Sampling is Consistent

- Let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by prior sampling algorithm
- By the construct of the algorithm, we have

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

- From the properties of BN, we have

$$S_{PS}(x_1, \ldots, x_n) = P(x_1, \ldots, x_n)$$

- Suppose there are N total samples and let $N_{PS}(x_1, \ldots, x_n)$ be the number of times a specific event has occurred, we expect

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

Direct Sampling is Consistent

- Let $S_{PS}(x_1, \ldots, x_n)$ be the probability that a specific event is generated by prior sampling algorithm
- By the construct of the algorithm, we have

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

- From the properties of BN, we have

$$S_{PS}(x_1, \ldots, x_n) = P(x_1, \ldots, x_n)$$

- Suppose there are N total samples and let $N_{PS}(x_1, \ldots, x_n)$ be the number of times a specific event has occurred, we expect

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

- Hence, this probability estimate is *consistent*

Rejection Sampling

- How do we estimate conditional probabilities such as $\mathbf{P}(X|\mathbf{e})$?

Rejection Sampling

- How do we estimate conditional probabilities such as $\mathbf{P}(X|\mathbf{e})$?
- Generate random samples and reject all those that do not match the evidence, and estimate $\hat{P}(X = x|\mathbf{e})$ by counting how often $X = x$ occurs in the remaining samples

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

Rejection Sampling

- How do we estimate conditional probabilities such as $\mathbf{P}(X|\mathbf{e})$?
- Generate random samples and reject all those that do not match the evidence, and estimate $\hat{P}(X = x|\mathbf{e})$ by counting how often $X = x$ occurs in the remaining samples

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

- As argued in the previous slide,

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

Rejection Sampling

- How do we estimate conditional probabilities such as $\mathbf{P}(X|\mathbf{e})$?
- Generate random samples and reject all those that do not match the evidence, and estimate $\hat{P}(X = x|\mathbf{e})$ by counting how often $X = x$ occurs in the remaining samples

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

- As argued in the previous slide,

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

- Hence, rejection sampling produces consistent estimates

Rejection Sampling

- How do we estimate conditional probabilities such as $\mathbf{P}(X|\mathbf{e})$?
- Generate random samples and reject all those that do not match the evidence, and estimate $\hat{P}(X = x|\mathbf{e})$ by counting how often $X = x$ occurs in the remaining samples

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

- As argued in the previous slide,

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

- Hence, rejection sampling produces consistent estimates
- Issue: When does the algorithm converge? Samples that are not rejected depends on $P(\mathbf{e})$ which could be extremely small fraction

Rejection Sampling Algorithm

function REJECTION-SAMPLING(X, $\mathbf{e}$, bn, N) **returns** an estimate of $\mathbf{P}(X \mid \mathbf{e})$
  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: $\mathbf{C}$, 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 $\mathbf{e}$ **then**
      $\mathbf{C}[j] \leftarrow \mathbf{C}[j]+1$ where x_j is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{C}$)

Importance Sampling

- How do generate random samples that match the evidence and still consistent?

Importance Sampling

- How do generate random samples that match the evidence and still consistent?
- Sample from $P(\mathbf{z}|\mathbf{e})$ where $\mathbf{z}$ stands for all the non-evidence variables (including the query variable X) — but that is not easy!

Importance Sampling

- How do generate random samples that match the evidence and still consistent?
- Sample from $P(\mathbf{z}|\mathbf{e})$ where $\mathbf{z}$ stands for all the non-evidence variables (including the query variable X) — but that is not easy!
- Sample from $Q(\mathbf{z})$ and apply *weight* to each sample
- Weight (correction factor) in this case will be $\frac{P(\mathbf{z}|\mathbf{e})}{Q(\mathbf{z})}$

Importance Sampling

- How do generate random samples that match the evidence and still consistent?
- Sample from $P(\mathbf{z}|\mathbf{e})$ where $\mathbf{z}$ stands for all the non-evidence variables (including the query variable X) — but that is not easy!
- Sample from $Q(\mathbf{z})$ and apply *weight* to each sample
- Weight (correction factor) in this case will be $\frac{P(\mathbf{z}|\mathbf{e})}{Q(\mathbf{z})}$
- The estimate can be easily shown to be consistent

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

Likelihood Weighting

- We will select a Q which is close to the posterior that we want to estimate. Let all the nonevidence variables be $\mathbf{Z} = \{Z_1, \ldots, Z_l\}$. We sample the following

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

Likelihood Weighting

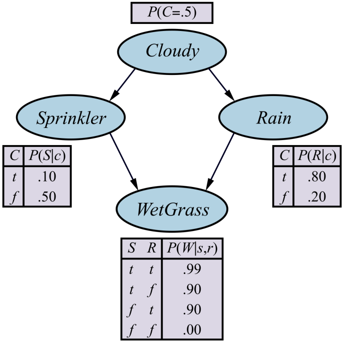
- We will select a Q which is close to the posterior that we want to estimate. Let all the nonevidence variables be $\mathbf{Z} = \{Z_1, \ldots, Z_l\}$. We sample the following

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

- From the properties of BN, the weights can be worked out as

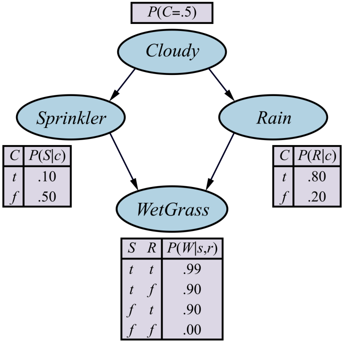
$$w(\mathbf{z}) = \alpha \prod_{i=1}^{m} P(e_i | parents(E_i))$$

Likelihood Weighting: Example



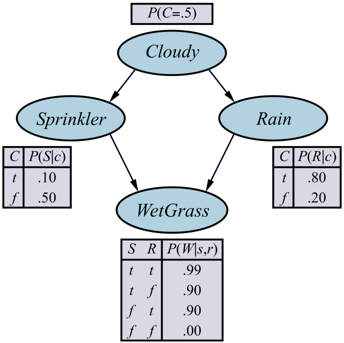
- Let the query be $\mathbf{P}(Rain|Cloudy = true, WetGrass = true)$

Likelihood Weighting: Example



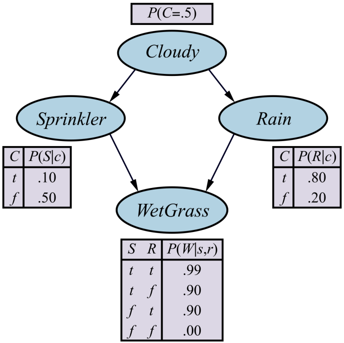
- Let the query be $\mathbf{P}(Rain \mid Cloudy = true, WetGrass = true)$
- Initially weight w is set to 1.0

Likelihood Weighting: Example



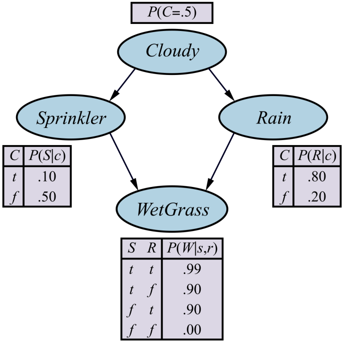
- Let the query be $\mathbf{P}(Rain|Cloudy = true, WetGrass = true)$
- Initially weight w is set to 1.0
- We will sample the variables in topological order: C, S, R, W

Likelihood Weighting: Example



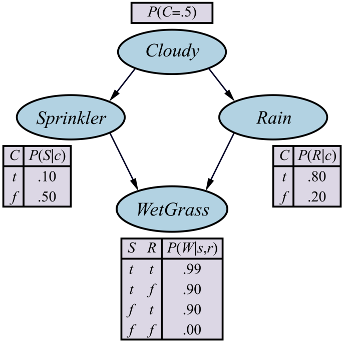
- Let the query be $\mathbf{P}(Rain | Cloudy = true, WetGrass = true)$
- Initially weight w is set to 1.0
- We will sample the variables in topological order: C, S, R, W
- *Cloudy* is evidence variable set to *true*. Weight is adjusted: $w = w \times P(Cloudy = true) = 0.5$

Likelihood Weighting: Example



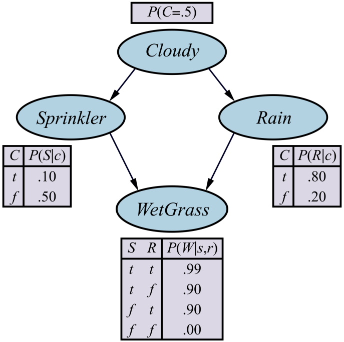
- Let the query be $\mathbf{P}(Rain|Cloudy = true, WetGrass = true)$
- Initially weight w is set to 1.0
- We will sample the variables in topological order: C, S, R, W
- *Cloudy* is evidence variable set to *true*. Weight is adjusted: $w = w \times P(Cloudy = true) = 0.5$
- *Sprinkler* is not an evidence variable; sample from $\mathbf{P}(Sprinkler|Cloudy = true) = \langle 0.1, 0.9\rangle$; suppose it returns *false*

Likelihood Weighting: Example



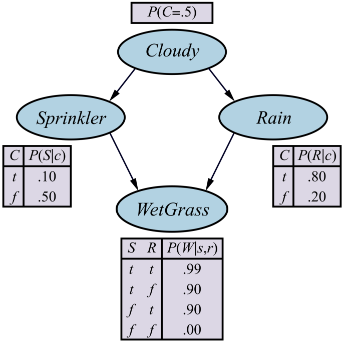
- *Rain* is not an evidence variable; sample from $\mathbf{P}(Rain|Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*

Likelihood Weighting: Example



- *Rain* is not an evidence variable; sample from $\mathbf{P}(Rain|Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*
- *WetGrass* is an evidence variable set to *true*. So $w = w \times P(w|\neg s, r) = 0.5 \times 0.9 = 0.45$

Likelihood Weighting: Example



- *Rain* is not an evidence variable; sample from $\mathbf{P}(Rain|Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns *true*
- *WetGrass* is an evidence variable set to *true*. So $w = w \times P(w|\neg s, r) = 0.5 \times 0.9 = 0.45$
- The sampled event is [*true*, *false*, *true*, *true*] with weight 0.45 and is counted under $Rain = true$

Likelihood-weighted Importance Sampling Algorithm

function LIKELIHOOD-WEIGHTING($X, \mathbf{e}, bn, N$) **returns** an estimate of $\mathbf{P}(X \mid \mathbf{e})$
  inputs: X, the query variable
    $\mathbf{e}$, observed values for variables $\mathbf{E}$
    bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$
    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($bn, \mathbf{e}$)
    $\mathbf{W}[j] \leftarrow \mathbf{W}[j] + w$ 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 $w \leftarrow w \times P(X_i = x_{ij} \mid parents(X_i))$
      else $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return $\mathbf{x}$, w

Summary

- We have discussed how knowledge under uncertainty can be represented using Bayesian Networks

What Next?

- Read chapter 13 of the text book!