Artificial Intelligence

Probabilistic Inference

Christopher Simpkins

Kennesaw State University

Exact Inference in Bayesian Networks (AIMA 13.3)

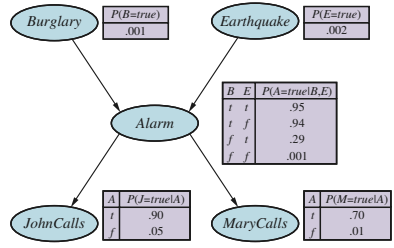
Most common task in probabilistic inference: compute the *posterior probability* of a set of **query variables** given some **event** represented as a set of **evidence variables**.

Notation:

- Query variable: X
- Set of evidence variables: $\boldsymbol{E} = \{E_1, \ldots, E_m\}$
- Particular observed event: $\boldsymbol{e}$
- Hidden (nonevidence, nonquery) variables: $\boldsymbol{Y} = \{Y_1, \ldots, Y_l\}$
- Typical query: $Pr(X \mid e)$

Example:

- X is the boolean random variable $Burglary$
- $\boldsymbol{E} = \{JohnCalls, MaryCalls\}$
- $\boldsymbol{e} = \{JohnCalls = true, MaryCalls = true\}$
- $\boldsymbol{Y} = \{EarthQuake, Alarm\}$



$$Pr(Buglary \mid JohnCalls = true, MaryCalls = true) =< 0.284, 0.716 > .$$

Inference by Enumeration

Recall that we can use the full joint distribution to answer any query:

$$Pr(X|\boldsymbol{e}) = \alpha Pr(X, \boldsymbol{e}) = \alpha \sum_{y} Pr(X, \boldsymbol{e}, \boldsymbol{y}) \tag{12.9}$$

And that a Bayes net completely represents the full joint distribution, so we can reduce the computation of a joint to:

$$Pr(x_1, \ldots, x_n) = \prod_{i=1}^{n} Pr(x_i|parents(X_i)) \tag{13.2}$$

Using these two equations we can enumerate the appropriate probabilities to calculate the answer to any probabilistic query.

- In particular, we can get the answer by computing sums of products of conditional probabilities from a Bayes net.

Example: $Pr(Burglary \mid JohnCalls = true, MaryCalls = true)$.

Using abbreviations and substituting into Eq 12.9 above (e and a are hidden):

$$Pr(B \mid j, m) = \alpha Pr(B, j, m) = \alpha \sum_e \sum_a Pr(B, j, m, e, a)$$

Then we substitute Eq 13.2 for $Pr(B, j, m, e, a)$ to get (onyly showing Burglary=true):

$$Pr(b \mid j, m) = \alpha \sum_e \sum_a Pr(b) Pr(e) Pr(a \mid b, e) Pr(j \mid a) Pr(m \mid a) \quad (1)$$

$$= \alpha Pr(b) \sum_e \sum_a Pr(e) Pr(a \mid b, e) Pr(j \mid a) Pr(m \mid a) \quad (2)$$

$$= \alpha Pr(b) \sum_e Pr(e) \sum_a Pr(a \mid b, e) Pr(j \mid a) Pr(m \mid a) \quad (3)$$

1. Substitute Eq 13.2 for $Pr(B, j, m, e, a)$
2. Pull out $Pr(b)$ from summations because it doesn't depend on the other variable and is thus a constant in all the summation terms.
3. Pull out $Pr(e)$ from the summation over the a values because each value of e doesn't depend on the other variables in the summation over the a values and is thus a constant in the summation terms over the values of a.

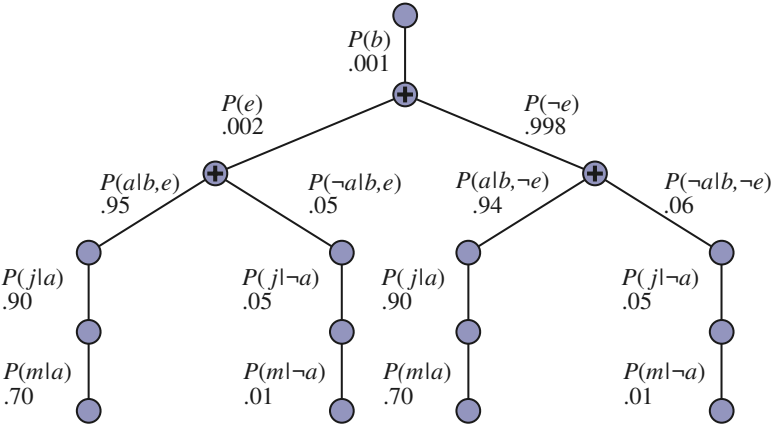
Steps 2 and 3 above reduce the complexity of the computation from $O(n2^n)$ to $O(2^n)$.

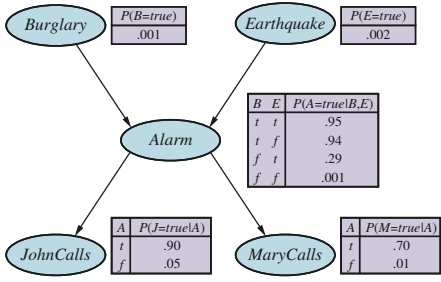
Caclulation of $Pr(b \mid j, m)$

Substiting the values from the CPTs in the Bayes net into

$$\alpha Pr(b) \sum_e Pr(e) \sum_a Pr(a \mid b, e) Pr(j \mid a) Pr(m \mid a)$$

we get the expression tree:





KENNESAW STATE UNIVERSITY

Enumeration Algorithm

The ENUMERATION-ASK algorithm evaluates these expression trees using depth-first, left-to-right recursion.

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

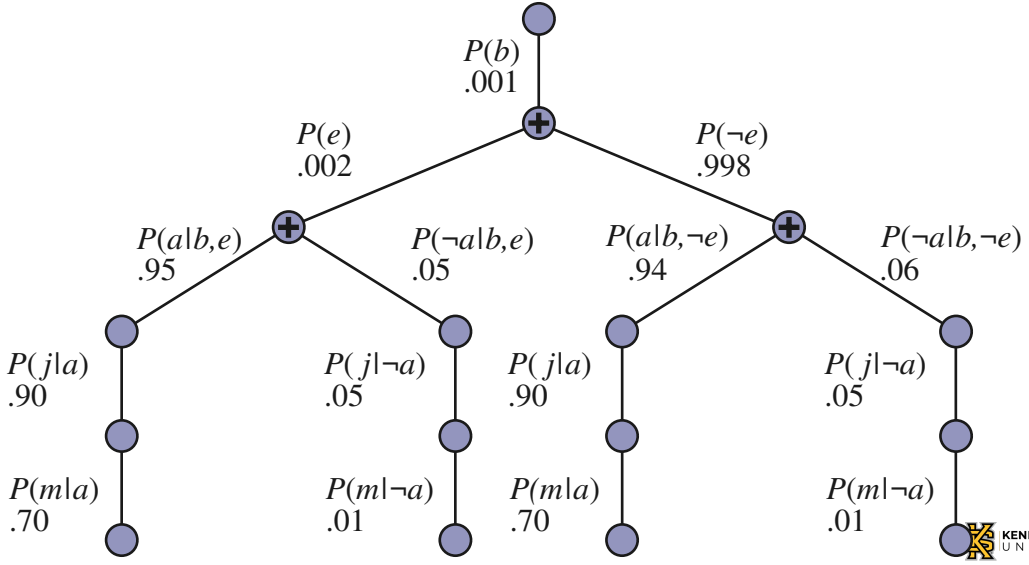
  $\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty
  for each value x_i of X **do**
    $\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL($vars, \mathbf{e}_{x_i}$)
      where $\mathbf{e}_{x_i}$ is $\mathbf{e}$ extended with $X = x_i$
  return NORMALIZE($\mathbf{Q}(X)$)

function ENUMERATE-ALL($vars, \mathbf{e}$) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $V \leftarrow$ FIRST($vars$)
  if V is an evidence variable with value v in $\mathbf{e}$
    then return $P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
    else return $\sum_v P(v \mid parents(V)) \times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_v$)
      where $\mathbf{e}_v$ is $\mathbf{e}$ extended with $V = v$

Unfortunately, its time complexity is $O(s^n)$. But we can improve it ...

Repeated Calculations

Notice that the subexpressions for the products $Pr(j \mid a)Pr(m \mid a)$ and $Pr(j \mid \neg a)Pr(m \mid \neg a)$ are computed twice, once for each value of E.



Variable Elimination

The enumeration algorithm can be improved substantially by eliminating repeated calculations.

- Idea: do the calculation once and save the results for later use.
- This is a form of dynamic programming.
- Several versions of this approach; variable elimination algorithm is simplest.

Variable elimination works by evaluating expressions such as

$$Pr(b \mid j, m) = \alpha Pr(b) \sum_{e} Pr(e) \sum_{a} Pr(a \mid b, e) Pr(j \mid a) Pr(m \mid a) \tag{13.5}$$

in right-to-left order (that is, bottom up in the expression tree), storing intermediate results, and only doing summations for portions of the expression that depend on the variable.

Example: Variable Elimination in Burglary Network

First, annotate the **factor**s in the expression for the network:

$$Pr(B \mid j, m) = \alpha \underbrace{Pr(B)}_{\boldsymbol{f}_1(B)} \sum_e \underbrace{Pr(e)}_{\boldsymbol{f}_2(E)} \sum_a \underbrace{Pr(a \mid B, e)}_{\boldsymbol{f}_3(A,B,E)} \underbrace{Pr(j \mid a)}_{\boldsymbol{f}_4(A)} \underbrace{Pr(m \mid a)}_{\boldsymbol{f}_5(A)}$$

- Each factor is a matrix indexed by the values of its argument variables.
- Notice that the factors for $Pr(j \mid a)$ and $Pr(m|a)$ do not include j and m. This is because the values of j and m ($JohnCalls = true$ and $MaryCalls - true$) are fixed by the query.

So the factors are:

$$\boldsymbol{f}_1(B) = \begin{bmatrix} Pr(b) \\ Pr(\neg b) \end{bmatrix} = \begin{bmatrix} 0.001 \\ 0.999 \end{bmatrix} \qquad \boldsymbol{f}_2(E) = \begin{bmatrix} Pr(e) \\ Pr(\neg e) \end{bmatrix} = \begin{bmatrix} 0.002 \\ 0.998 \end{bmatrix}$$

$$\boldsymbol{f}_4(A) = \begin{bmatrix} Pr(j \mid a) \\ Pr(j \mid \neg a) \end{bmatrix} = \begin{bmatrix} 0.090 \\ 0.05 \end{bmatrix} \qquad \boldsymbol{f}_5(A) = \begin{bmatrix} Pr(m \mid a) \\ Pr(m \mid \neg a) \end{bmatrix} = \begin{bmatrix} 0.070 \\ 0.01 \end{bmatrix}$$

$\boldsymbol{f}_3(A, B, E)$ is a little more complicated ...

KENNESAW STATE UNIVERSITY

$\boldsymbol{f}_3(A, B, E)$

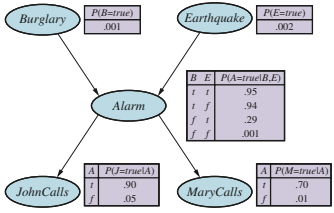
$$Pr(B \mid j, m) = \alpha \underbrace{Pr(B)}_{\boldsymbol{f}_1(B)} \sum_e \underbrace{Pr(e)}_{\boldsymbol{f}_2(E)} \sum_a \underbrace{Pr(a \mid B, e)}_{\boldsymbol{f}_3(A,B,E)} \underbrace{Pr(j \mid a)}_{\boldsymbol{f}_4(A)} \underbrace{Pr(m \mid a)}_{\boldsymbol{f}_5(A)}$$

$\boldsymbol{f}_3(A, B, E)$ is a $2 \times 2 \times 2$ matrix (or a rank-3 tensor). Here's one way to think about it:

- First index with A, yielding two 2×2 submatrices (one for each of the two values of A).
- Rows of each submatrix is indexed by B and columns by E.
- The entries in the submatrices are the values of $Pr(A \mid B, E)$

$$\boldsymbol{f}_3^{(a)}(B, E) = \begin{bmatrix} Pr(a \mid b, e) & Pr(a \mid b, \neg e) \\ Pr(a \mid \neg b, e) & Pr(a \mid \neg b, \neg e) \end{bmatrix} = \begin{bmatrix} 0.95 & 0.94 \\ 0.29 & 0.001 \end{bmatrix}$$

$$\boldsymbol{f}_3^{(\neg a)}(B, E) = \begin{bmatrix} Pr(\neg a \mid b, e) & Pr(\neg a \mid b, \neg e) \\ Pr(\neg a \mid \neg b, e) & Pr(\neg a \mid \neg b, \neg e) \end{bmatrix} = \begin{bmatrix} 0.05 & 0.06 \\ 0.71 & 0.999 \end{bmatrix}$$



Factorized Query

From our original query:

$$Pr(b \mid j, m) = \alpha Pr(b) \sum_{e} Pr(e) \sum_{a} Pr(a \mid b, e) Pr(j \mid a) Pr(m \mid a) \tag{13.5}$$

We annotated the factors:

$$Pr(B \mid j, m) = \alpha \underbrace{Pr(B)}_{\boldsymbol{f}_1(B)} \sum_{e} \underbrace{Pr(e)}_{\boldsymbol{f}_2(E)} \sum_{a} \underbrace{Pr(a \mid B, e)}_{\boldsymbol{f}_3(A,B,E)} \underbrace{Pr(j \mid a)}_{\boldsymbol{f}_4(A)} \underbrace{Pr(m \mid a)}_{\boldsymbol{f}_5(A)}$$

And now we substitute the factor expressions for the original expresions so we can manipulate the factors using the **pointwise product** operation, denoted with $\times$ here:

$$Pr(B \mid j, m) = \alpha \boldsymbol{f}_1(B) \times \sum_{e} \boldsymbol{f}_2(E) \times \sum_{a} \boldsymbol{f}_3(A, B, E) \times \boldsymbol{f}_4(A) \times \boldsymbol{f}_5(A)$$

Now we are ready to evaluate the expression ...

Expression Evaluation

First, sum out A from the pointwise product of $\boldsymbol{f}_3(A, B, E)$, $\boldsymbol{f}_4(A)$, and $\boldsymbol{f}_5(A)$ yielding a new 2×2 factor, $\boldsymbol{f}_6(B, E)$:

$$\begin{aligned}\boldsymbol{f}_6(B, E) &= \sum_a \boldsymbol{f}_3(A, B, E) \times \boldsymbol{f}_4(A) \times \boldsymbol{f}_5(A) \\ &= (\boldsymbol{f}_3(a, B, E) \times \boldsymbol{f}_4(a) \times \boldsymbol{f}_5(a)) + (\boldsymbol{f}_3(\neg a, B, E) \times \boldsymbol{f}_4(\neg a) \times \boldsymbol{f}_5(\neg a))\end{aligned}$$

Now the query expression is $Pr(B \mid j, m) = \alpha \boldsymbol{f}_1(B) \times \sum_e \boldsymbol{f}_2(E) \times \boldsymbol{f}_6(B, E)$

Next, sum out E from the product of $\boldsymbol{f}_2(E)$ and $\boldsymbol{f}_6(B, E)$, yielding a new factor $\boldsymbol{f}_7(B)$:

$$\begin{aligned}\boldsymbol{f}_7(B) &= \sum_e \boldsymbol{f}_2(E) \times \boldsymbol{f}_6(B, E) \\ &= \boldsymbol{f}_2(e) \times \boldsymbol{f}_6(B, e) + \boldsymbol{f}_2(\neg e) \times \boldsymbol{f}_6(B, \neg e)\end{aligned}$$

Which leaves our final form of the query: $Pr(B \mid j, m) = \alpha \boldsymbol{f}_1(B) \times \boldsymbol{f}_7(B)$

This expression can be evaluated by taking the pointwise product and normalizing the result.

Operations on Factors

Two basic operations in variable elimination:

1. the pointwise product operation, and
2. summing out hidden variables from products of factors.

Pointwise Product Example

The pointwise product of two factors $\boldsymbol{f}$ and $\boldsymbol{g}$ yields a new factor $\boldsymbol{h}$ whose variables are the union of the variables in $\boldsymbol{f}$ and $\boldsymbol{g}$ and whose elements are given by the product of the corresponding elements in the two factors.

Given X, Y, Z boolean variables, result of pointwise product $\boldsymbol{f}(X,Y) \times \boldsymbol{g}(Y,Z) = \boldsymbol{h}(X,Y,Z)$ is:

X	Y	$\mathbf{f}(X,Y)$	Y	Z	$\mathbf{g}(Y,Z)$	X	Y	Z	$\mathbf{h}(X,Y,Z)$
t	t	.3	t	t	.2	t	t	t	$.3 \times .2 = .06$
t	f	.7	t	f	.8	t	t	f	$.3 \times .8 = .24$
f	t	.9	f	t	.6	t	f	t	$.7 \times .6 = .42$
f	f	.1	f	f	.4	t	f	f	$.7 \times .4 = .28$
						f	t	t	$.9 \times .2 = .18$
						f	t	f	$.9 \times .8 = .72$
						f	f	t	$.1 \times .6 = .06$
						f	f	f	$.1 \times .4 = .04$

Summing out Variables

Summing out a variable from a product of factors is done by adding up the submatrices formed by fixing the variable to each of its values in turn. For example, to sum out X from $h(X, Y, Z)$, we write

$$
\begin{aligned}
\boldsymbol{h}_2(Y, Z) &= \sum_x \boldsymbol{h}(X, Y, Z) \\
&= \boldsymbol{h}(x, Y, Z) + \boldsymbol{h}(\neg x, Y, Z) \\
&= \begin{bmatrix} .06 & .24 \\ .42 & .28 \end{bmatrix} + \begin{bmatrix} .18 & .72 \\ .06 & .04 \end{bmatrix} \\
&= \begin{bmatrix} .24 & .96 \\ .48 & .32 \end{bmatrix}
\end{aligned}
$$

Variable Elimination Algorithm

With these two basic operations, we can implement the variable elimination algorithm:

function ELIMINATION-ASK(X, **e**, bn) **returns** a distribution over X
  inputs: X, the query variable
    e, observed values for variables **E**
    bn, a Bayesian network with variables $vars$

  $factors \leftarrow []$
  for each V **in** ORDER($vars$) **do**
    $factors \leftarrow [\text{MAKE-FACTOR}(V, \mathbf{e})] + factors$
    if V is a hidden variable **then** $factors \leftarrow \text{SUM-OUT}(V, factors)$
  return NORMALIZE(POINTWISE-PRODUCT($factors$))

Notes about the `order` function:

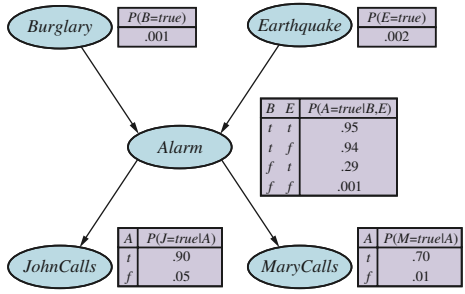
- Any ordering works, some orderings lead to more efficient algorithms.
- No tractable algorithm for determining optimal ordering.
- One heuristic: eliminate whichever variable minimizes the size of the next factor to be contructed.
- General rule: every variable that is not an ancestor of a query variable or evidence variable is irrelevant to the query.

KENNESAW STATE UNIVERSITY

Complexity of Exact Inference in Polytrees

Notice that the Alarm Bayes net is **singly connected**, a.k.a., a **polytree**:

- there is at mose one undirected path between any two nodes in the network.

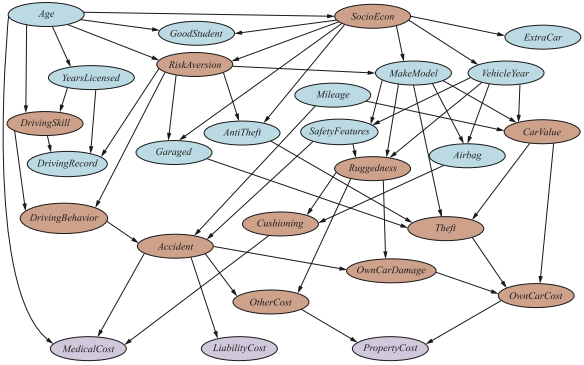


The time and space complexity of polytrees is linear in the size of the network.

- Size of network is defined as number of CPT entries.
- If $|parents(X_i)| \leq c, \forall i \in n$ for some constant c and number of nodes n, then complexity is also linear in number of nodes.

Complexity of Exact Inference in Multiply-connected Networks

Now consider **multiply-connected** networks such as the car insurance network:



- Variable elimination can have exponential worst-case time and space complexity in multiply-connected networks.
- *Since inference in Bayes nets includes inference in propositional logic as a special case, Bayes net inference is* ***NP-hard****.*

KENNESAW STATE UNIVERSITY

Approximate Inference for Bayesian Networks

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

Direct Sampling Methods

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

Feed a sample from U(1, 1) to the inverse CDF of a distribution to sample the distribution.

Prior 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.

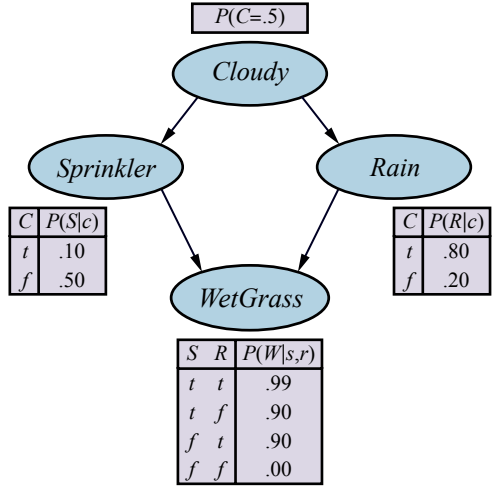
function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

$\mathbf{x} \leftarrow$ 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 $\mathbf{x}$

Example: Generating a Sample from Sprinkler Bayes Net

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

$$S_{PS}(x_1, \ldots, x_n) = \prod_{i=1}^{n} Pr\left(x_i \mid parents(X_i)\right) = Pr(x_1, \ldots, x_n)$$

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:

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

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 speciiied event $x_1, \ldots, x_m$ for $x \leq n$ is:

$$Pr(x_1, \ldots, x_m) \approx \frac{N_{PS}(x_1, \ldots, x_m)}{N} = \hat{Pr}(x_1, \ldots, x_m)$$

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

Rejection Sampling

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

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

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

Likelihood Weighting

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

KENNESAW STATE UNIVERSITY

Likelihood Weighting Example

With the query $Pr(Rain \mid Cloudy = true, WetGrass = true)$ and the ordering $Cloudy, Sprinkler, Rain, WetGrass$, process proceeds as follows:

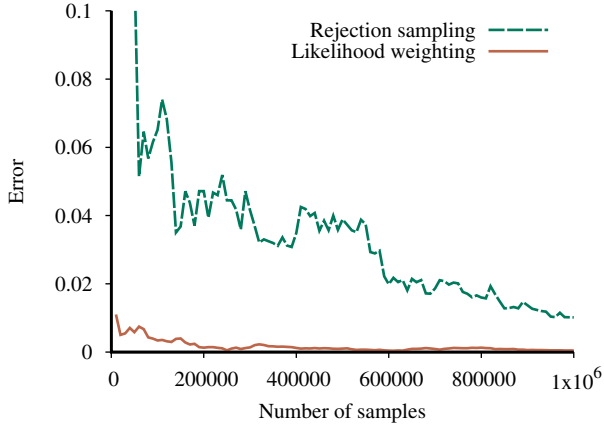
1. $Cloudy$ is an evidence variable with value true. Therefore, we set

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

2. $Sprinkler$ is not an evidence variable, so sample from $Pr(Sprinkler \mod Cloudy = true) = \langle 0.1, 0.9 \rangle$; suppose this returns false.
3. $Rain$ is not an evidence variable, so sample from $Pr(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns true.
4. $WetGrass$ is an evidence variable with value true. Therefore, we set

$$w \leftarrow w \times Pr(WetGrass = true \mid Sprinkler = false, Rain = true) = 0.5 \times 0.9 = 0.45.$$

Rejection vs. Importance Sampling



Markov Chain Monte Carlo (MCMC) Algorithms

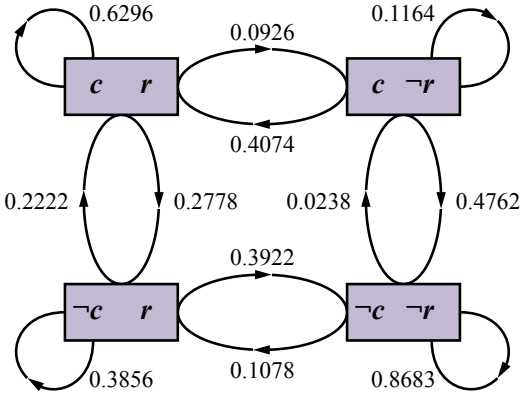
Instead of generating each sample from scratch, MCMC algorithms generate a sample by making a random change to the preceding sample. Think of an MCMC algorithm as being in a particular current state that specifies a value for every variable and generating a next state by making random changes to the current state.

Gibbs Sampling

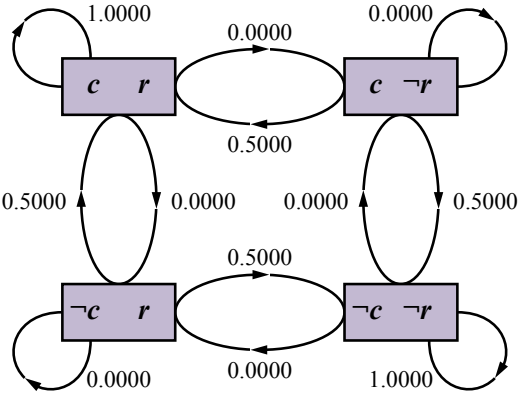
function GIBBS-ASK($X, \mathbf{e}, bn, N$) **returns** an estimate of $\mathbf{P}(X \mid \mathbf{e})$
  local variables: $\mathbf{C}$, a vector of counts for each value of X, initially zero
    $\mathbf{Z}$, the nonevidence variables in bn
    $\mathbf{x}$, the current state of the network, initialized 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 $\mathbf{P}(Z_i \mid mb(Z_i))$
    $\mathbf{C}[j] \leftarrow \mathbf{C}[j] + 1$ where x_j is the value of X in $\mathbf{x}$
  return NORMALIZE($\mathbf{C}$)

Markov Chains

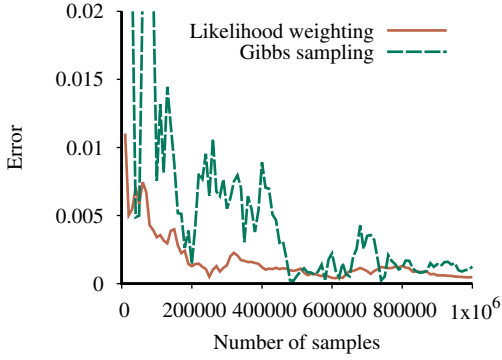


(a)

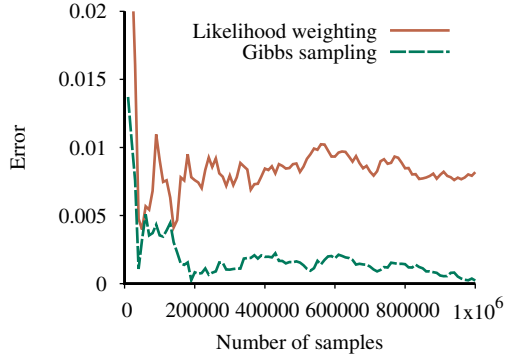


(b)

Gibbs Sampling vs. Importance Sampling



(a)



(b)

Metropolis-Hastings Sampling