Rejection Sampling

To determine a conditional probability \( P(X \mid e) \) we could apply the following simple sampling approach:
1. Generate samples from the prior distribution specified by the network
2. Reject all those that do not match the evidence \( e \)
3. The estimate \( P_{\text{data}}(X = x \mid e) \) is obtained by counting how often \( X = x \) occurs in the remaining samples

The estimated distribution \( P_{\text{data}}(X \mid e) \) that the algorithm returns is, by the definition of the algorithm
\[
\alpha N_{\text{PS}}(X, e) = N_{\text{PS}}(X, e) / N_{\text{PS}}(e)
\]
As an estimate of the probability of a partially specified event it is consistent
\[
P_{\text{data}}(X \mid e) \approx P(X, e) / P(e) = P(X \mid e)
\]

Let us generate 100 samples in order to estimate the distribution \( P(\text{Rain} \mid \text{Sprinkler} = \text{True}) \)
- Suppose that 73 of those that we generate have \( \text{Sprinkler} = \text{False} \) and are rejected
- The remaining 27 have \( \text{Sprinkler} = \text{True} \)
- Out of them 8 have \( \text{Rain} = \text{True} \) and 19 have \( \text{Rain} = \text{False} \)
- Hence, we now have \( P_{\text{data}}(\text{Rain} \mid \text{sprinkler}) \approx [0.296, 0.704] \), while the true distribution is \([0.3, 0.7]\)
- As more samples are collected, the estimate will converge to the true answer
- The standard deviation of the error in each probability will be proportional to \( 1/\sqrt{n} \), where \( n \) is the number of samples used in the estimate
- The large number of rejected samples is a big problem:
  The fraction of samples consistent with the evidence drops exponentially as the number of evidence variables grows
Likelihood weighting

- Rejection sampling is inefficient because it ends up rejecting so many of the generated samples.
- To avoid generating needles samples that anyhow get rejected, let us fix the values for the evidence variables $E$ and sample only the remaining variables $X$ and $Y$.
- Not all events are equal, however.
- Each event is weighted by the likelihood that the event accords to the evidence.
- The likelihood is measured by the product of the conditional probabilities for each evidence variable, given its parents.
- Intuitively, events in which the actual evidence appears unlikely should be given less weight.

To answer the query $P(\text{Rain} \mid \text{sprinkler, wetgrass})$, the weight $w$ is first set to 1.0.

- Sample from $P(\text{Cloudy}) = [0.5, 0.5]$; suppose this returns True.
- Sprinkler is an evidence variable with value True, therefore we update the weight:
  $$w \leftarrow w \times P(\text{sprinkler} \mid \text{cloudy}) = 0.1$$
- Sample from $P(\text{Rain} \mid \text{cloudy}) = [0.8, 0.2]$; suppose this returns True.
- WetGrass is an evidence variable with value True $\Rightarrow$
  $$w \leftarrow w \times P(\text{wetgrass} \mid \text{sprinkler, rain}) = 0.099$$
- Hence, the algorithm returns the event [True, True, True, True] with weight 0.099 and this is tallied under Rain = True.
Let us denote $Z = \{ X \} \cup Y$

The weighted sample algorithm samples each variable in $Z$ given its parent values

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

Parents$(Z_i)$ can include both hidden variables and evidence variables

The sampling distribution $S_{WS}$ pays some attention to the evidence, unlike the prior distribution $P(z)$

In $S_{WS}$, the sampled values for each $Z_i$ will be influenced by evidence among $Z_i$’s ancestors

On the other hand, the true posterior distribution $P(z \mid e)$ also takes non-ancestor evidence into account

The likelihood weight $w$ makes up for the difference between the actual and desired sampling distributions

Let a given sample $x$ be composed from $z$ and $e$, then

$$w(z, e) = \prod_{i=1}^m P(e_i \mid \text{parents}(E_i))$$

The weighted probability of a sample, $S_{WS}(z, e) \cdot w(z, e)$, is

$$\prod_{i=1}^l P(x_i \mid \text{parents}(Z_i)) \cdot \prod_{i=1}^m P(e_i \mid \text{parents}(E_i)) = P(z, e).$$

because the two products cover all the variables in the network
Now it is easy to show that likelihood weighting estimates are consistent:

\[
P_{\text{dist}}(x \mid \mathbf{e}) = \alpha \sum_y N_{\text{WS}}(x, y, \mathbf{e}) \cdot w(x, y, \mathbf{e})
\]

algorithm

\[
\approx \alpha' \sum_y S_{\text{WS}}(x, y, \mathbf{e}) \cdot w(x, y, \mathbf{e})
\]

for large N

\[
= \alpha' \sum_y P(x, y, \mathbf{e})
\]

by prev. slide

\[
= \alpha' P(x, \mathbf{e})
\]

\[
= P(x \mid \mathbf{e})
\]

Because likelihood weighting uses all the samples generated, it can be much more efficient than rejection sampling.

It will, however, suffer a degradation in performance as the number of evidence variables increases.

Because most samples will have very low weights, the weighted estimate will be dominated by a tiny fraction of samples.

**14.5.2 Inference by Markov Chain Simulation**

Markov chain Monte Carlo (MCMC)

A Monte Carlo algorithm is a randomized algorithm, which can give the false answer with a small probability (cf. Las Vegas algorithm).

MCMC generates each event by making a random change to the preceding event.

The next state is generated by randomly sampling a value for one of the nonevidence variables \(X_i\), conditioned on the current values in its Markov blanket.

The Markov blanket of a variable consists of its parents, children, and children’s parents.

MCMC therefore wanders randomly around the state space, flipping one variable at a time, but keeping the evidence variables fixed.
Gibbs sampling

- Consider the query $P(\text{Rain} | \text{sprinkler}, \text{wetgrass})$
- Nonevidence variables $\text{Cloudy}$ and $\text{Rain}$ are initialized randomly to True and False, say
- Thus, the initial state is [True, True, False, True]
- The nonevidence variables are sampled repeatedly in random order
  - $\text{Cloudy}$ is sampled, given the current values of its Markov blanket variables, i.e., $P(\text{Cloudy} | \text{sprinkler}, \neg\text{rain})$. Suppose the result is $\text{Cloudy} = \text{False}$. Then the new state is [False, True, False, True]
  - $\text{Rain}$ is sampled, given the current values in its Markov blanket, i.e., $P(\text{Rain} | \neg\text{cloudy}, \text{sprinkler}, \text{wetgrass})$. Suppose this yields $\text{Rain} = \text{True}$. The new current state is [False, True, True, True]

- If the process visits 20 states where $\text{Rain}$ is True and 60 states in which it is False, then the answer to the query is $\text{Normalize}([20, 60]) = [0.25, 0.75]$

- Gibbs sampling returns consistent estimates for posterior probabilities
- Through Markov chain analysis one can show that the sampling process settles into a “dynamic equilibrium” in which the long run fraction of time spent in each state is exactly proportional to its posterior probability
- This property follows from the specific transition probability with which the process moves from one state to another, as defined by the conditional distribution given the Markov blanket of the variable being sampled
16 MAKING SIMPLE DECISIONS

- Let us associate each state $S$ with a numeric utility $U(S)$, which expresses the desirability of the state.
- A nondeterministic action $a$ will have possible outcome states $\text{Result}(a) = s'$.
- Prior to the execution of $a$ the agent assigns probability $P(\text{Result}(a) = s' \mid a, e)$ to each outcome, where $e$ summarizes the agent’s available evidence of the world.
- The expected utility of $a$ can now be calculated:

$$EU(a \mid e) = \sum_{s'} P(\text{Result}(a) = s' \mid a, e) \cdot U(s')$$

- The principle of maximum expected utility (MEU) says that a rational agent should choose an action that maximizes the agent’s expected utility $\arg\max_a EU(a \mid e)$.
- If we wanted to choose the best sequence of actions using this equation, we would have to enumerate all action sequences, which is clearly infeasible for long sequences.
- If the utility function correctly reflects the performance measure by which the behavior is being judged, using MEU the agent will achieve the highest possible performance score averaged over the environments in which it could be placed.
- Let us model a nondeterministic action with a lottery $L$, where possible outcomes $S_1, \ldots, S_n$ can occur with probabilities $p_1, \ldots, p_n$.

$$L = [p_1, S_1; p_2, S_2; \ldots; p_n, S_n]$$
16.2 The Basis of Utility Theory

$A \succ B$  Agent prefers lottery $A$ over $B$

$A \sim B$  The agent is indifferent between $A$ and $B$

$A \succeq B$  The agent prefers $A$ to $B$ or is indifferent between them

- Deterministic lottery $[1, A] \equiv A$
- Reasonable constraints on the preference relation (in the name of rationality)
  - **Orderability**: given any two states, a rational agent must either prefer one to the other or else rate the two as equally preferable.
    $$(A \succ B) \lor (B \succ A) \lor (A \sim B)$$
  - **Transitivity**:
    $$(A \succ B) \land (B \succ C) \Rightarrow (A \succ C)$$

- **Continuity**:
  $$A \succ B \succ C \Rightarrow \exists p: [p, A; 1-p, C] \sim B$$

- **Substitutability**:
  $$A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$$

- **Monotonicity**:
  $$A \succ B \Rightarrow (p \geq q \Rightarrow [p, A; 1-p, B] \succ [q, A; 1-q, B])$$

- ** Decomposability**: Compound lotteries can be reduced to simpler ones by the laws of probability
  $$[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$$

- Notice that these axioms of utility theory do not say anything about utility
- The existence of a utility function follows from them
Preferences lead to utility

1. Existence of Utility Function:
   If an agent’s preferences follow the axioms of utility, then there exists a real-valued function \( U \) s.t.
   \[
   U(A) > U(B) \iff A \succ B \\
   U(A) = U(B) \iff A \sim B
   \]

2. Expected Utility of a Lottery:
   The utility of a lottery is
   \[
   U([p_1, S_1; \ldots; p_n, S_n]) = \sum_{i=1}^{n} p_i U(S_i)
   \]
   Because the outcome of a nondeterministic action is a lottery, this gives us the MEU decision rule from slide 279.

- The axioms of utility do not specify a unique utility function for an agent.
- For example, we can transform a utility function \( U(S) \) into
  \[
  U'(S) = aU(S) + b
  \]
  where \( b \) is a constant and \( a \) is any positive constant.
- Clearly, this affine transformation leaves the agent’s behavior unchanged.
- In deterministic contexts, where there are states but no lotteries, behavior is unchanged by any monotonic transformation.
- E.g., the cube root of the utility \( \sqrt[3]{U(S)} \).
- Utility function is ordinal — it really provides just rankings of states rather than meaningful numerical values.
16.3 Utility Functions

- Money (or an agent’s total net assets) would appear to be a straightforward utility measure
- The agent exhibits a monotonic preference for definite amounts of money
- We need to determine a model for lotteries involving money
  - We have won a million euros in a TV game show
  - The host offers to flip a coin, if the coin comes up heads, we end up with nothing, but if it comes up tails, we win three million euros
  - Is the only rational choice to accept the offer which has the expected monetary value of 1.5 million euros?
- The true question is maximizing total wealth (not winnings)