Where are we?

- Last time . . .
- Inference in Bayesian Networks $P(X|e)$
- Exact methods: enumeration, variable elimination algorithm involved evaluation large sums of products.
- Computationally intractable in the worst case
- Today . . . Approximate Inference in Bayesian Networks

Approximate Inference in BNs

- Exact inference computationally very hard
- Approximate methods important, here randomised sampling algorithms
- Monte Carlo algorithms
- We will talk about two types of MC algorithms:
  1. Direct sampling methods
  2. Markov chain sampling

Direct sampling methods

- Basic idea: generate samples from a known probability distribution
- Consider an unbiased coin as a random variable – sampling from the distribution is like flipping the coin
- It is possible to sample any distribution on a single variable given a set of random numbers from $[0,1]$ - biased coin
- Simplest method: generate events from network without evidence
  - Sample each variable in ‘topological order’
  - Probability distribution for sampled value is conditioned on values assigned to parents
Example

- Consider the following BN and ordering

  \[ \text{[Cloudy, Sprinkler, Rain, WetGrass]} \]

Example

- Direct sampling process:
  - Sample from \( P(\text{Cloudy}) = <0.5, 0.5> \), suppose this returns true
  - Sample from \( P(\text{Sprinkler}|\text{Cloudy} = \text{true}) = <0.1, 0.9> \), suppose this returns false.
  - Sample from \( P(\text{Rain}|\text{Cloudy} = \text{true}) = <0.8, 0.2> \), suppose this returns true.
  - Sample from \( P(\text{WetGrass}|\text{Sprinkler} = \text{false}, \text{Rain} = \text{true}) = <0.9, 0.1> \), suppose this returns true.
  - Event returned = \([\text{true, false, true, true}]\).
  - This is a sample from the joint.
  - Repeat.

Direct Sampling Methods

- Generates samples with probability \( S(x_1, \ldots, x_n) \)
  \[
  S(x_1, \ldots, x_n) = P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\text{parents}(X_i))
  \]

- Answers are computed by counting the number \( N(x_1, \ldots, x_n) \) of the times event \( x_1, \ldots, x_n \) was generated and dividing by total number \( N \) of all samples

- In the limit, we should get
  \[
  \lim_{n \to \infty} \frac{N(x_1, \ldots, x_n)}{N} = S(x_1, \ldots, x_n) = P(x_1, \ldots, x_n)
  \]

  When the estimated probability becomes exact in the limit we call the estimate consistent and we write \( \approx \) in this sense,

  \[
  P(x_1, \ldots, x_n) \approx \frac{N(x_1, \ldots, x_n)}{N}
  \]

Example

- \( P_{\text{est}}(\text{Rain}=\text{true})? \)
  - Repeat 100 times:
    - Sample from \( P(\text{Cloudy}) = <0.5, 0.5> \).
    - Given the result, Sample from \( P(\text{Rain}|\text{Cloudy}) \)
  - \( \text{Count the number of samples where Rain} = 1 \) and divide by 100 samples
  - \( P_{\text{est}}(\text{Rain}=\text{true}) \approx 0.511 \)
Rejection Sampling

- What if you wanted to compute P(X|e) — with some evidence?
- Rejection Sampling: A general method to produce samples for hard-to-sample distribution from easy-to-sample distribution
- To determine P(X|e) generate samples from the prior distribution specified by the BN first
- Then reject those that do not match the evidence
- The estimate P(X = x|e) is obtained by counting how often X = x occurs in the remaining samples
- Rejection sampling is consistent because, by definition:

\[ \hat{P}(X|e) = \frac{N(X,e)}{N(e)} = \frac{P(X,e)}{P(e)} - P(X|e) \]

Back to our example

- Assume we want to estimate P(Rain|Sprinkler = true), using 100 samples
- 73 have Sprinkler = false (rejected), 27 have Sprinkler = true
- Of these 27, 8 have Rain = true and 19 have Rain = false
- P(Rain|Sprinkler = true) = \( \alpha_{8,19} = 0.296, 0.704 \)
- True answer would be \( 0.3, 0.7 \)
- But the procedure rejects too many samples that are not consistent with e (exponential in number of variables)
- Not really usable (similar to naively estimating conditional probabilities from observation)

Likelihood weighting

- Avoids inefficiency of rejection sampling by generating only samples consistent with evidence
- Fixes the values for evidence variables E and samples only the remaining variables X and Y
- Since not all events are equally probable, each event has to be weighted by its likelihood that it accords to the evidence
- Likelihood is measured by product of conditional probabilities for each evidence variable, given its parents.

Consider query P(Rain|Sprinkler = true, WetGrass = true) in our example;
- Initially set weight \( w = 1 \), then event is generated:
  - Sample from P(Cloudy) = \( 0.5, 0.5 \), suppose this returns true
  - Sprinkler is evidence variable with value true, we set \( w \leftarrow w \times P(Sprinkler = true|Cloudy = true) = 0.1 \)
  - Sample from P(Rain|Cloudy = true) = \( 0.8, 0.2 \), suppose this returns true
  - WetGrass is evidence variable with value true, we set \( w \leftarrow w \times P(WetGrass = true|Sprinkler = true, Rain = true) = w\times0.99=0.099 \)
  - Sample returned=[true,true,true,true] with weight 0.099 tallied under Rain = true
Likelihood weighting: why it works

- Define all the non-evidence variables $Z = X \cup Y$ (X the query, Y the hidden) and consider sampling distribution $S$

  $$S(z,e) = \prod_{i=1}^{j} P(z|\text{parents}(Z))$$

- $S$'s sample values for each $Z_i$ is influenced by the evidence among $Z_i$'s ancestors
- But $S$ pays no attention when sampling $Z_i$'s value to evidence from $Z_i$'s non-ancestors; so it's not sampling from the true posterior probability distribution!
- But the likelihood weight $w$ makes up for the difference

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

Markov Chain Monte-Carlo (MCMC) Algorithm

- MCMC algorithm: create an event from a previous event, rather than generate all events from scratch
- Helpful to think of the BN as having a current state specifying a value for each variable
- Consecutive state is generated by sampling a value for one of the non-evidence variables $X_i$ conditioned on the current values of variables in the Markov blanket of $X_i$
- Recall that Markov blanket consists of parents, children, and children’s parents
- Algorithm randomly wanders around state space flipping one variable at a time and keeping evidence variables fixed

Likelihood weighting: why it works

- Since two products cover all the variables in the network, we can write:

  $$P(z,e) = \prod_{i=1}^{j} P(z_i|\text{parents}(Z_i)) \prod_{i=1}^{m} P(e_i|\text{parents}(E_i))$$

- With this, it is easy to derive that likelihood weighting is consistent (tutorial exercise)
- much more efficient than rejection sampling but efficiency gets worse when number of evidence variables increases
- Problem: most samples will have very small weights as the number of evidence variables increases
- The weighted estimate will be dominated by tiny fraction of samples that give more than infinitesimal likelihood to the evidence

MCMC/Gibbs Algorithm

- Consider query $P(\text{Rain}|\text{Sprinkler} = \text{true}, \text{WetGrass} = \text{true})$ once more
- Sprinkler and WetGrass (evidence variables) are fixed to their observed values, hidden variables Cloudy and Rain are initialised randomly (e.g. true and false)
- Assume initial state is [true, true, false, true]
- Execute repeatedly:
  - Sample Cloudy given values of Markov blanket, i.e. sample from $P(\text{Cloudy}|\text{Sprinkler} = \text{true}, \text{Rain} = \text{false})$
  - Suppose result is false, new state is [false, true, false, true]
  - Sample Rain given values of Markov blanket, i.e. sample from $P(\text{Rain}|\text{Sprinkler} = \text{true}, \text{Cloudy} = \text{false}, \text{WetGrass} = \text{true})$
  - Suppose we obtain Rain = true, new state [false, true, true, true]
MCMC Algorithm

- Each visited state is a sample
- If there was 20 states where Rain=1 and 60 states where Rain=0,
  \[ P_{est}(Rain|Sprinkler = true, WetGrass = true) = \frac{20}{80}, \frac{60}{80} \]
- Basic idea of proof that MCMC is consistent:
  The sampling process settles into a “dynamic equilibrium” in which the long-term fraction of time spent in each state is exactly proportional to its posterior probability
- MCMC is a very powerful method used for all kinds of things involving probabilities

Summary

- Exact inference is in general intractable
- Stochastic approximation techniques such as likelihood weighting and MCMC can give reasonable estimates of the true posterior probabilities in a network and can cope with much larger networks than can exact algorithms
- There are 2 important families of approximation methods that were not covered here: variational approximations and belief propagation.
- Next time: Time and Uncertainty I