Lecture 10: Approximate inference: Particle-based methods

- Types of approximate inference methods
- Sampling from a Bayesian network
  - Forward sampling
  - Rejection sampling
  - Likelihood weighting
- More generally: Importance sampling

Approximate inference methods

- Instead of computing (conditional) probabilities directly, compute an *approximately correct answer*
- The answer only needs to be good enough to let us do the real task (which is most often finding the most likely value for the query, or decision making)
- *"Good enough"* can be expressed in terms of:
  - **Absolute error**: $|p(Y|e) - \hat{p}(Y|e)| \leq \epsilon$
  - **Relative error**: $\frac{1}{1+\epsilon} \leq \frac{p(Y|e)}{\hat{p}(Y|e)} \leq (1+\epsilon)$

  where Y are the query variables and the evidence variables E have value e
- We will discuss this more later, as similar error measures are used in learning

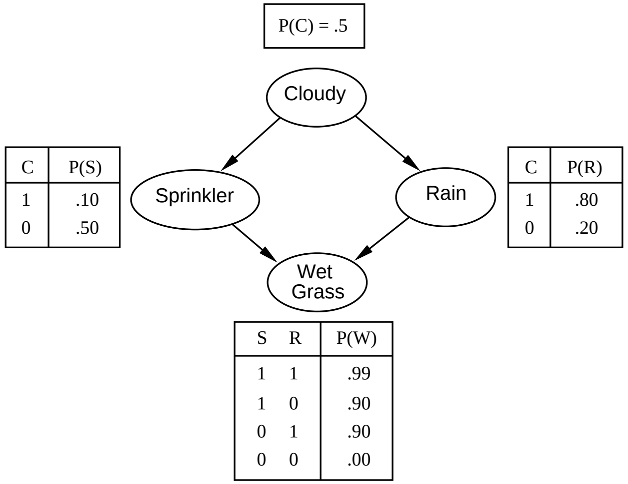
Two classes of approximate inference methods

1. **Particle-based methods**: use the model to *generate instances* (particles), from the distribution, then compute sufficient statistics for the distribution
   - A particle could have values for all variables, or only for the ones that are necessary to answer the query (based on conditional independence)
   - The capacity to use the model to generate data is *key* for probabilistic models (often called **generative models**), not only for inference but also to understand the model
2. **Optimization-based (variational) methods**: use exact inference, but on a model which is simpler than the real model

Particle-based methods

- How can particles (instances) be generated?
  - **Random sampling**:
    - ∗ **Rejection sampling**: Sample directly from the desired distribution
    - ∗ **Likelihoood weighting**: Sample from a different distribution but then apply a correction
    - ∗ **Gibbs sampling**: Sample from distributions that are increasingly closer to the desired distribution
  - **Direct search**: Deterministically generate particles so that the cases forming most of the probability mass are covered
- If possible, only some of the variables are sampled

Example: Sprinkler network



P(C) = .5

C	P(S)
1	.10
0	.50

C	P(R)
1	.80
0	.20

S	R	P(W)
1	1	.99
1	0	.90
0	1	.90
0	0	.00

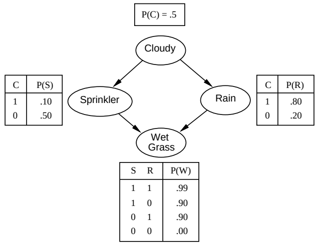
Approximate the marginal probability $p(W = 1)$

Main idea of forward (logic) sampling

- Traverse the network, in the direction of the arcs
- At each node, *sample* a value for the corresponding random variable from the CPD

  Constraint: Parents must already have values
- After we got N samples, *count* how many have the desired value for the query variables, and divide by N (of course, assuming discrete variables)

Example: Forward sampling



P(C) = .5

C	P(S)
1	.10
0	.50

C	P(R)
1	.80
0	.20

S	R	P(W)
1	1	.99
1	0	.90
0	1	.90
0	0	.00

1. Sample C according to its probability distribution. Say $C = 1$.
2. Sample R according to $p(R|C = 1)$. Say $R = 1$.
3. Sample S according to $p(S|C = 1)$. Say $S = 0$.
4. Sample W according to $p(W|R = 1, S = 0)$. Say $W = 1$.

Now we have a complete sample: $\langle C = 1, R = 1, S = 0, W = 1\rangle$

We repeat the steps above as much as needed.

Example: Computing marginal probabilities from samples

Suppose we generate N samples using the above technique. How do we estimate $p(W = 1)$?

$$p(W = 1) \approx \frac{N(W = 1)}{N}$$

Analyzing the error

- We would like to know how many particles we need to generate in order to get a good approximation of the marginal probability $p(Y = y)$.
- First tool: **Hoeffding bound**: Given a sequence of N independent Bernoulli trials with probability of success θ, let $\hat{\theta} = \frac{N(X=1)}{N}$. Then:

$$p(|\theta - \hat{\theta}| > \epsilon) \leq 2e^{-2N\epsilon^2}$$

So, with very high probability, the absolute error is smaller than ϵ

- Second tool: **Chernoff bound**: Morevover, we have:

$$p(\hat{\theta} > \theta(1+\epsilon)) \leq e^{-N\theta\epsilon^2/3}$$

Applying the bounds to forward sampling

- Define an auxiliary random variable: $X = 1$ if we got a sample with $Y = y$, $X = 0$ otherwise
- X is binomially distributed, and its probability is $p(Y = y)$!
- So the bounds can be applied
- For instance, if we want the probability of absolute error greater than ϵ to be less than δ, we need:

$$N \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta}$$

- Similarly, for the relative error to be within ϵ, we need at least:

$$N \geq \frac{3}{p(Y=y)\epsilon^2} \frac{2}{\delta}$$

This is pretty useless, as it depends on $p(Y = y)$ (unknown)

Example: Computing conditional probabilities

- How do we estimate $p(W = 1|C = 1)$?

Example: Computing conditional probabilities

- How do we estimate $p(W = 1|C = 1)$?

$$\begin{aligned} p(W=1|C=1) &= \frac{p(C=1,W=1)}{p(C=1)} \\ &\approx \frac{N(C=1,W=1)}{N} \frac{N}{N(C=1)} = \frac{N(C=1,W=1)}{N(C=1)} \end{aligned}$$

- Note that we did not use all the samples in this computation! Only the samples in which $C = 1$ were used.
- One can show that if we have good estimates for both joint probabilities, the estimate for the ratio will also be good.

Rejection sampling

- Generate samples by forward sampling of the network:
  - Let $X_1, \ldots X_n$ be an ordering of the variables consistent with the arc direction in the Bayes net structure, and so that each variable comes after its parents
  - For $i = 1, \ldots, n$, sample X_i from $p(X_i | X_{\pi_i})$.

  Note that all the parents of X_i are surely instantiated when we get to sample X_i.
- Throw away the samples inconsistent with the evidence

Rejection sampling

- Generate samples by forward sampling of the network:
  - Let $X_1, \ldots X_n$ be an ordering of the variables consistent with the arc direction in the Bayes net structure, and so that each variable comes after its parents
  - For $i = 1, \ldots, n$, sample X_i from $p(X_i | \pi_{X_i})$.

  Note that all the parents of X_i are surely instantiated when we get to sample X_i.
- Throw away the samples inconsistent with the evidence

Problem: If the evidence is unlikely, then we will throw away most samples, and it takes a long time to gather enough data for a reliable estimate.

Becoming more efficient

- Instead of generating samples in which $C = 0$ and throwing them away, do not generate them at all!
- *Idea:* Fix the values for the evidence variables, sample only the other variables. Then we can use all the samples.
- In our case, set $C = 1$, then:
  1. Sample R from $p(R|C = 1)$
  2. Sample S from $p(S|C = 1)$
  3. Sample W from $p(W|R, S)$

  Now we approximate $p(W = 1|C = 1) \approx \frac{N(W=1)}{N}$

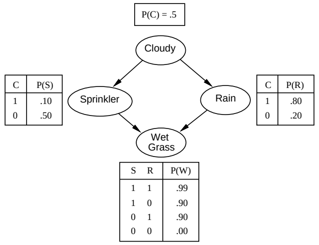
Downstream evidence

Suppose we want to compute $p(C|W = 1)$. We fix $W = 1$ and we need to sample C, R, S.

- We would like to sample R from $p(R|W = 1)$.

  But we do not have these probabilities! We could do arc reversal on the network, but this is actually quite expensive.
- *Idea:* sample the network top-down like before, but fix the values of the evidence variables.

Example



C	P(S)
1	.10
0	.50

C	P(R)
1	.80
0	.20

S	R	P(W)
1	1	.99
1	0	.90
0	1	.90
0	0	.00

1. Sample C according to $p(C)$. Say $C = 0$.
2. Sample R according to $p(R|C = 0)$. Say $R = 0$
3. Sample S according to $p(S|C = 0)$. Say $S = 0$.
4. $W = 1$ (since it is the evidence)

Is this a good instance?

A simple case

- Consider a very simple network: $X \rightarrow Y$.
- We want to compute $p(X|Y = 1)$.
  1. Sample X from $p(X)$
  2. Set $Y = 1$
- Problem: These samples come from $p(X)$, not $p(X, Y = 1)$.

  So we have:

$$\frac{N(X = 1, Y = 1)}{N} \approx p(X = 1), \text{ not } p(X = 1, Y = 1)$$

A simple case (continued)

- To see the fix to this problem, let us consider how we would compute $p(X = 1, Y = 1)$ exactly:

$$p(X = 1, Y = 1) = p(Y = 1|X = 1)p(X = 1)$$

- Since our sample count approximates $p(X = 1)$, all we have to do is multiply the estimate by the **weight** $p(Y = 1|X = 1)$.
- We do the same thing to estimate $p(Y = 1, X = 0)$. Then we can approximate the conditional as usual.
- This algorithm is called **likelihood weighting**

Likelihood weighting

Let $X_1, \ldots X_n$ be an ordering of the variables consistent with the arc direction in the Bayes net structure

1. Repeat for $i = 1, \ldots, N$ times:
   (a) $w = 1$
   (b) For $j = 1, \ldots, n$ do:
   - If X_j has been observed as evidence ($X_j \in E$), $w \leftarrow w \cdot p(X_j = x_j|X_{\pi_j})$
   - Else sample X_j from its CPD, $p(X_j|X_{\pi_j})$
2. $\hat{p}(Y = y|E = e) = \frac{\sum_{i=1}^{N} w_i \delta_i(Y=y)}{\sum_{i=1}^{N} w_i}$ where $\delta_i(Y = y)$ is an indicator variable equal to 1 if $Y = y$ in the ith sample.

Importance sampling

Likelihood weighting is a special case of a more general procedure, called **importance sampling**

- Suppose we want to estimate the expected value of a function f depending on a random variable X drawn according to the **target** probability distribution $p(X)$.
- If we had N samples x_i drawn from $p(X)$, we could estimate the expectation using the empirical mean:

$$E_p[f] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

- But instead, we have only samples drawn according to a different **proposal** or **sampling** distribution $q(X)$.
- How can we do the estimation?

Unnormalized importance sampling

- We do a simple trick:

$$\begin{aligned} E_p[f] &= \sum_x f(x)p(X=x) \\ &= \sum_x f(x)q(X=x)\frac{p(X=x)}{q(X=x)} = E_q\left[f\frac{p}{q}\right] \end{aligned}$$

- Only requirement: if $p(x) > 0$ then $q(x) > 0$
- So for an estimator, we should average each sample of the function, $f(x_i)$ **weighted** by the ratio of its probability under the target and the sampling distribution:

$$E_p[f] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)\frac{p(x_i)}{q(x_i)}$$

A problem

- The previous estimator makes the assumption that we know the target distribution p. But this seems restrictive
- E.g., in a Markov network, we know unnormalized clique potentials. These are *proportional* to p. But to compute p exactly requires computing the partition function, which is expensive
- Let $p' = \alpha p$ be known (α is an arbitrary constant) .
- In this case, just plugging p' into the importance sampling expectation directly does not work correctly:

$$E_q\left[f\frac{p'}{q}\right] = \sum_x q(x)f(x)\frac{p'(x)}{q(x)} = \sum_x f(x)\alpha p(x) = \alpha E_p[f]$$

A solution!

- The previous estimate is off by a factor of α. So if we knew α, we could correct it.
- An interesting observation:

$$E_q\left[\frac{p'}{q}\right] = \sum_x q(x)\frac{p'(x)}{q(x)} = \sum_x \alpha p(x) = \alpha$$

Hence, we can divide the two expectations and get the correct answer!

- If we estimate the expectations from samples, we get:

$$E_p[f] \approx \frac{\sum_{i=1}^{n} f(x_i)\frac{p'(x_i)}{q(x_i)}}{\sum_{i=1}^{n}\frac{p'(x_i)}{q(x_i)}}$$

This is called **normalized importance sampling**

Properties of statistical estimators

- Suppose that we have a data sample of size N
- If, for any N, the expected value of the estimator (over multiple samples drawn from the same distribution) is correct, the estimator is called **unbiased**
- If, in the limit of $N \to \infty$, the estimator has the correct expected value, it is called **consistent**
- The **variance** of the estimator tells us how much variability to expect based on different samples.
  Recall that for a random variable, the variance is defined as:

$$E\left[(X - E[X])^2\right] = E[X^2] - (E[X])^2$$

Bias and variance of importance sampling

- Unnormalized importance sampling is unbiased, consistent, but has potentially high variance
- The variance depends on how different the target and proposal distributions are, as well as on the function f
- The normalized importance sampling estimator is biased but consistent
- The theoretical variance is not comparable to the unnormalized estimator, but in practice it tends to be much lower
- The bias-variance trade-off is a constant issue in statistical estimation and machine learning

Applying importance sampling to approximate inference

- Suppose we are interested in a set of variables Z having particular values z (because they are evidence or query variables)
- Consider a **mutilated** Bayesian network in which the nodes Z have no parents and are just set to the desired value. All other nodes stay the same
- This will be the *proposal distribution*
- It is easy to show that the weights computed by likelihood weighting are exactly importance sampling weights under this proposal distribution (and the desired target)
- The function f is just the indicator function

Additional algorithms

- Computing the marginal probability $p(Z = z)$
- Normalized likelihood weighting (based on normalized importance sampling)
- Ratio likelihood weighting (similar, but we set the values for the query too, and usually use different numbers of samples for the top and the bottom estimator)
- In all cases, if the values of the variables are unusual, we may need a lot of samples to get a good estimate