

CS 3750 Machine Learning Lecture 10

Monte Carlo inference

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

CS 3750 Advanced Machine Learning

Importance Sampling

- an approach for estimating the expectation of a function $f(x)$ relative to some distribution $P(X)$ (target distribution)
- generally, we can estimate this expectation by generating samples $x[1], \dots, x[M]$ from P , and then estimating

$$E_p[f] = \frac{1}{M} \sum_{m=1}^M f(x[m])$$

- However, we might prefer to generate samples from a different distribution Q (proposal or sampling distribution) instead, since it might be impossible or computationally very expensive to generate samples directly from P .
- Q can be arbitrary, but it should dominate P , i.e. $Q(x) > 0$ whenever $P(x) > 0$

CS 3750 Advanced Machine Learning

Unnormalized Importance Sampling

- Since we generate samples from Q instead of P ,
- we need to adjust our estimator to compensate for the incorrect sampling distribution.

$$E_{p(x)}[f(X)] = E_{Q(x)}\left[f(x) \frac{P(x)}{Q(x)}\right]$$

- So we can use standard estimator for expectations relative to Q .
- **Method:** We generate a set of M samples $D = \{x[1], \dots, x[M]\}$ from Q , and estimate:

$$\hat{E}_D(f) = \frac{1}{M} \sum_{m=1}^M f(x[m]) \frac{P(x[m])}{Q(x[m])}$$

Importance sampling

- This is an unbiased estimator: its mean for any data set is precisely the desired value

$$w(x) = P(x) / Q(x) \quad \text{- a weighting function, or a correction weight}$$

- We can estimate the distribution of the estimator around its mean: as $M \rightarrow \infty$

$$E_{Q(x)}[f(X)w(X)] - E_p[f(X)] \propto N(0; \sigma_Q^2 / M)$$

$$\text{where } \sigma_Q^2 = [E_{Q(x)}[(f(X)w(X))^2]] - (E_{Q(x)}[f(X)w(X)])^2$$

$$\sigma_Q^2 = [E_{Q(x)}[(f(X)w(X))^2]] - (E_{P(x)}[f(X)])^2$$

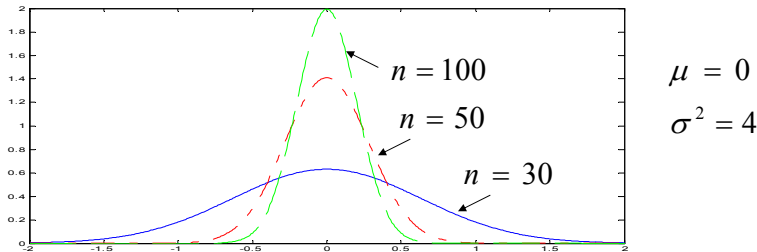
Central limit theorem

- **Central limit theorem:**

Let random variables X_1, X_2, \dots, X_n form a random sample from a distribution with mean μ and variance σ^2 , then if the sample n is large, the distribution

$$\sum_{i=1}^n X_i \approx N(n\mu, n\sigma^2) \quad \text{or} \quad \frac{1}{n} \sum_{i=1}^n X_i \approx N(\mu, \sigma^2 / n)$$

Effect of increasing the sample size n on the sample mean:



CS 3750 Advanced Machine Learning

Importance sampling

- When $f(X)=1$, the variance is simply the variance of the weighting function $P(X)/Q(X)$. Thus, the more different Q is from P , the higher is the variance of the estimator.
- In general, the lowest variance is achieved when

$$Q(X) \propto |f(X)| P(X)$$

- We should avoid cases where our sampling probability $Q(X) \ll P(X)f(X)$ in any part of the space, as these cases can lead to very large or even infinite variance.
- Problem with unnormalized IS: P is assumed to be known

CS 3750 Advanced Machine Learning

Normalized Importance Sampling

- When P is only known up to a normalizing constant α
- We have access to a function $P'(X)$, such that P' is not a normalized distribution, but $P'(X) = \alpha P(X)$
- In this context, we cannot define the weights relative to P , so we define:

$$w(X) = \frac{P'(X)}{Q(X)}$$

$$\begin{aligned} E_{P(X)}[f(X)] &= \sum_x P(x) f(x) = \sum_x Q(x) f(x) \frac{P'(x)}{Q(x)} = \frac{1}{\alpha} \sum_x Q(x) f(x) \frac{P'(x)}{Q(x)} \\ &= \frac{1}{\alpha} E_{Q(x)}[f(X) w(X)] = \frac{E_{Q(x)}[f(X) w(X)]}{E_{Q(x)}[w(X)]} \end{aligned}$$

Why?
$$E_{Q(x)}[w(X)] = \sum_x Q(x) \frac{P'(x)}{Q(x)} = \sum_x P'(x) = \alpha$$

CS 3750 Advanced Machine Learning

Importance sampling

- Using an empirical estimator for both the numerator and denominator, we can estimate:

$$\hat{E}_D(f) = \frac{\sum_{m=1}^M f(x[m]) w(x[m])}{\sum_{m=1}^M w(x[m])}$$

- The normalized estimator is biased
- Its variance is typically lower than that of the unnormalized estimator. This reduction in variance often outweighs the bias term.
- So normalized estimator is often used in place of the unnormalized estimator, even in cases where P is known and we can sample from it effectively.

CS 3750 Advanced Machine Learning

Importance sampling for BBNs

- Our goal is to generate particles (sample instances) from the BBN efficiently. Avoid the problem of rejection sampling.
- The approach is equivalent to the **likelihood weighting** procedure discussed earlier
- We define a proposal distribution that “sets” the value of $Z_i \in Z$ to take the prespecified value in a way that influences the sampling process only for its descendants. (z is a particular event)

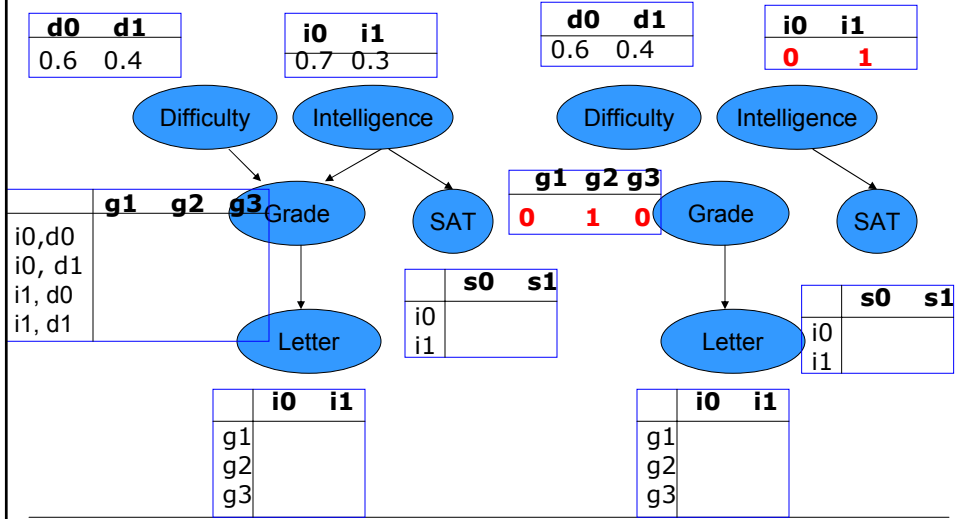
Importance sampling in BBNs

- Let B be a network, and $Z_i = z_i$, an instantiation of some of the variables. We define **the mutilated network** $B_{Z=z}$ as follows:
- Each node $Z_i = z_i$ has no parents in $B_{Z=z}$;
- The CPD of Z_i in $B_{Z=z}$ gives probability 1 to $Z_i \in Z$ and probability 0 to all other values $z_i \in \text{Dom}(Z_i)$
- The parents and CPDs of all other nodes $X \notin Z$ are unchanged

Importance sampling in BBN

Original BN

Mutilated network $B_{I=i^1, G=g^2}$



CS 3750 Advanced Machine Learning

Unconditional probability of an event

- We can use unnormalized importance sampling to estimate the value of joint on $P(X)$.
- $Q(X)$ is the one defined by the mutilated network:

$$\hat{P}_D(z) = \frac{1}{M} \sum_{m=1}^M 1\{\xi[m] < Z \} w(\xi[m]) = \frac{1}{M} \sum_{m=1}^M w[m]$$
- $Q(X)$ is easy to sample !
- But we still need $w(X)$...

$$P(X) = \prod_{X_i} P(X_i | pa(X_i)) \quad Q(X) = \prod_{X_i} P_B(X_i | pa(X_i))$$

$$w(X) = \frac{\prod_{X_i} P(X_i | pa(X_i))}{\prod_{X_i} P'(X_i | pa(X_i))} = \prod_{Z_i \in Z} P(Z_i | pa(Z_i))$$

That is easy ...

CS 3750 Advanced Machine Learning

Data-Dependent Likelihood Weighting

- Adding bells and whistles ...
- The intuition is not every samples contribute equally to the quality of the estimate. A sample with high weight is more compatible with the evidence e , and may provide us with more information.
- A more sophisticated approach is to pre-define a total weight. We stop sampling when the total weight of the generated particles reaches our pre-defined value.
- It allows early stopping in cases where we were lucky in our random choice of samples.

CS 3750 Advanced Machine Learning

Ratio likelihood weighting

- What is we have to calculate conditional probability:
- Estimate the conditional probability $P(y|e)$ in two phases: use likelihood weighting to estimate $P(e)$ and $P(y,e)$ separately.
- Use LW algorithm M times with the argument $E=e$ to generate a set D of weighted samples $(\xi[1], w[1]), \dots, (\xi[M], w[M])$
use the same algorithm M' times with argument $Y=y, E=e$ to generate another set D' of weighted samples
 $(\xi'[1], w'[1]), \dots, (\xi'[M], w'[M])$
- Then we can estimate:

$$\hat{P}_D(y|e) = \frac{\hat{P}_{D'}(y, e)}{\hat{P}_D(e)} = \frac{1/M' \sum_{m=1}^{M'} w'[m]}{1/M \sum_{m=1}^M w[m]}$$

CS 3750 Advanced Machine Learning

Normalized likelihood weighting

- Estimating the conditional probability $P(y|e)$ in two phases: estimation of $P(e)$ and estimation of $P(y,e)$ may be expensive
- Idea: use normalized sampling
use the algorithm M times with argument $E=e$ to generate another set D of weighted samples
 $(\xi[1], w[1]), \dots, (\xi[M], w[M])$

- Then we can estimate:

$$\hat{P}_D(y|e) = \frac{\sum_{m=1}^M w[m] 1_{y_m}}{\sum_{m=1}^M w[m]}$$

- This is exactly the method we used earlier ...