Lab 1

Importance Sampling and Monte Carlo Simulations

Lab Objective: *Use importance sampling to reduce the error and variance of Monte Carlo Simulations.*

Introduction

The traditional methods of Monte Carlo integration as discussed in the previous lab are not always the most efficient means to estimate an integral. For example, assume we were trying to find the probability that a randomly chosen variable X from the standard normal distribution is greater than 3. We know that one way to solve this is by solving the following integral:

$$P(X > 3) = \int_3^\infty f_X(t)\,dt = \frac{1}{\sqrt{2\pi}} \int_3^\infty e^{-t^2/2}\,dt \tag{1.1}$$

If we define the function $h : \mathbb{R} \to \mathbb{R}$ as

$$h(t) = \begin{cases} 1 & \text{if } t > 3 \\ 0 & \text{if } t \leq 3 \end{cases},$$

we can rewrite this integral as,

$$\int_3^\infty f_X(t)\,dt = \int_{-\infty}^\infty h(t) f_X(t)\,dt.$$

By the Law of the Unconscious Statistician (see Volume 2 §3.5), we can restate the integral above as,

$$\int_{-\infty}^\infty h(t) f_X(t)\,dt = E[h(X)].$$

Being able to write integrals as expected values is an essential tool in this lab.

Monte Carlo Simulation

In the last section, we expressed the probability of drawing a number greater than 3from the normal distribution as an expected value problem. We can now easily estimate this same probabilty using Monte Carlo simulation. Given a random i.i.d. sample $x_1, x_2, \cdots, x_N$ generated by f_X, we can estimate $E[h(X)]$ using

$$\widehat{E}_n[h(X)] = \frac{1}{N}\sum_{i=1}^{N} h(x_i) \tag{1.2}$$

Now that we have defined the estimator, it is now quite manageable to approximate Equation 1.1. By the Weak Law of Large Numbers (see Volume 2 §3.6), the estimate will get closer and closer to the actual value as we use more and more sample points.

Problem 1. Write a function in Python that estimates the probability that a random draw from the standard normal distribution is greater than 3 using Equation 1.2. Your answer should approach 0.0013499 for sufficiently large samples.

Though this approach gets the job done, it turns out that this isn't very efficient. Since the probability of drawing a number greater than 3 from the standard normal distribution is so unlikely, it turns out we need many sample points to get a good approximation.

Importance Sampling

Importance sampling is one way to make Monte Carlo simulations converge much faster. We chose a different distribution to sample our points to generate more *important* points. With our example, we want to choose a distribution that would generate more numbers around 3 to get a more reliable estimate. The theory behind importance sampling boils down to the following result. In these equations, the random variable X is generated by f_X and the random variable Y is generated by g_X. We X and Y in this way for the remainder of the lab.

$$\begin{aligned} E[h(X)] &= \int_{-\infty}^{\infty} h(t) f_X(t)\, dt \\ &= \int_{-\infty}^{\infty} h(t) f_X(t) \left(\frac{g_X(t)}{g_X(t)}\right) dt \\ &= \int_{-\infty}^{\infty} \left(\frac{h(t) f_X(t)}{g_X(t)}\right) g_X(t)\, dt \\ &= E\left[\frac{h(Y) f_X(Y)}{g_X(Y)}\right] \end{aligned} \tag{1.3}$$

The corresponding estimator is,

$$\begin{aligned}\widehat{E}[h(X)] &= \widehat{E}\left[\frac{h(Y)f_X(Y)}{g_X(Y)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{h(y_i)f_X(y_i)}{g_X(y_i)}\end{aligned}$$

The function f_X is the p.d.f. of the *target distribution*. The function g_X is the p.d.f. of the *importance distribution*. The fraction $\frac{f_X(X)}{g_X(X)}$ is called the *importance weight*. This allows us to draw a sample from any distribution with p.d.f. g_X as long as we multiply $h(X)$ by the importance weight.

Choosing the Importance Distribution

There is no correct choice for the importance distribution. It may be possible to find the distribution that allows the simulation to converge the fastest, but oftentimes, we don't need a perfect answer. Close is to perfect is good enough.

We will solve the same problem as in Problem 1 using importance sampling. We will choose g_X to be the normal distribution with $\mu = 4$ and $\sigma = 1$. We have chosen this distribution for g_X because it will give us more points closer to and greater than 3.

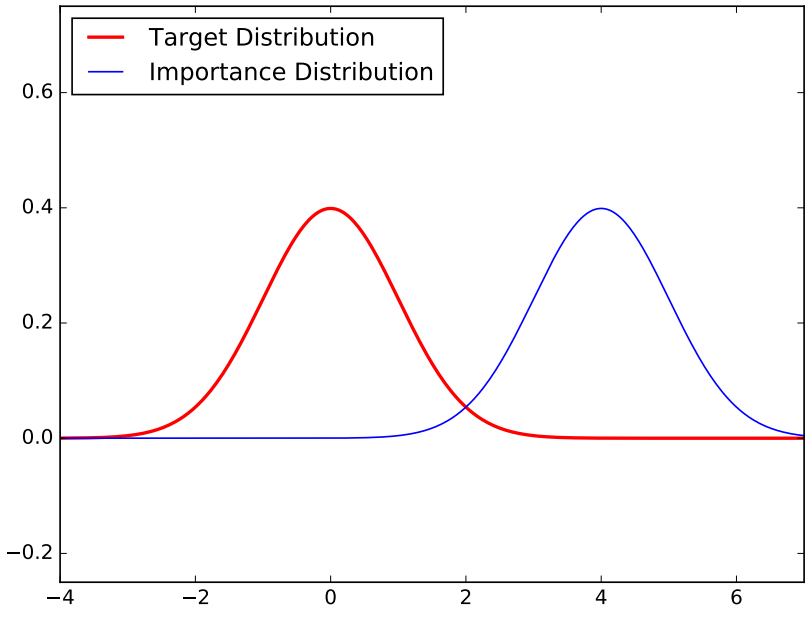

Figure 1.1: In our problem, we choose an importance distribution that will generate more samples that are greater than 3. Though not a perfect choice, choosing a normal distribution with $\mu = 4$ and $\sigma = 1$ will suffice.

```
>>> import scipy.stats as ss
>>> f = lambda x : x > 3
>>> h = lambda x : ss.norm().pdf(x)
>>> g = lambda x : ss.norm(loc=4,scale=1).pdf(x)

# Sample from the N(4,1).
>>> N = 10**7
>>> X = np.random.normal(4,scale=1,size=N)

# Calculate estimate.
>>> 1./N * np.sum(f(X)*h(X)/g(X))
0.00134921134631
```

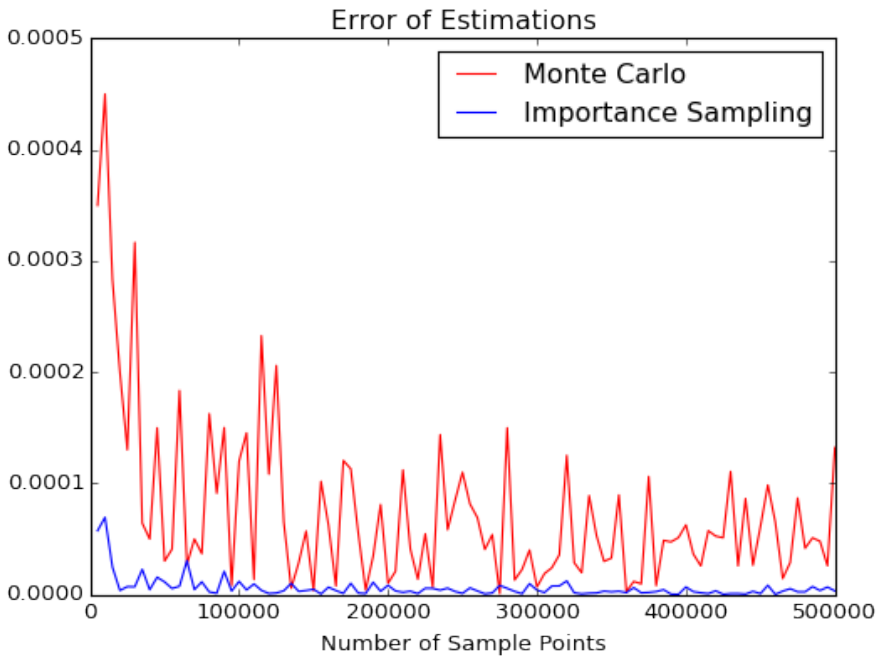

Figure 1.2: Comparison of error between standard method Monte Carlo and Importance Sampling method of Monte Carlo.

Problem 2. A tech support hotline receives an average of 2 calls per minute. What is the probability that they will have to wait at least 10 minutes to receive 9 calls? Implement your estimator using importance sampling. Calculate estimates using 5000, 10000, 15000, $\cdots$, 500000 sample points. Return an array of estimates. Your answers should approach 0.00208725.

Hint: In Volume 2 §3.5, the gamma distribution is defined as,

$$f_X(x) = \frac{b^a x^{a-1} e^{-xb}}{\Gamma(a)}.$$

The version of the gamma distribution in `scipy.stats` is determined by the shape (a) and the scale (θ) of the distribution.

$$f_X(x) = \frac{1}{\Gamma(a)\theta^a} x^{a-1} e^{-x/\theta}$$

You can switch between these representations this with the fact that $\theta = 1/b$.

Problem 3. In this problem, we will visualize the benefits of importance sampling. Create a plot of the error of the traditional methods of Monte Carlo and the importance sampling methods of Monte Carlo. What do you observe? Your answers should resemble Figure 1.2.

Hint: The following code solves Problem 2 using traditional methods of Monte Carlo:

```
h = lambda x : x > 10
MC_estimates = []
for N in xrange(5000,505000,5000):
    X = np.random.gamma(9,scale=0.5,size=N)
    MC = 1./N*np.sum(h(X))
    MC_estimates.append(MC)
MC_estimates = np.array(MC_estimates)
```

Hint: The following code returns the actual value of Equation 1.1:

```
1 - ss.gamma(a=9,scale=0.5).cdf(10)
```

Generalizing the Principles of Importance Sampling

The examples we have explored to this point in the lab were merely educational. Since we have a simple means of calculating the correct answer to Problem 2, it doesn't make much sense to use methods of Monte Carlo in this situation. However, as discussed in the previous lab, there are not always closed-form solutions to the integrals we want to compute.

We can extend the same principles we have discussed thusfar to solve many types of problems. For a more general problem, we can implement importance sampling by doing the following:

1. Define a function h where, $h(t) = \begin{cases} 1 & \text{if condition is met} \\ 0 & \text{otherwise} \end{cases}$.

2. Define a function f_X which is the p.d.f. of the target distribution.

3. Define a function g_X which is the p.d.f. of the importance distribution.

Problem 4. The joint normal distribution of N independent random variables with mean 0 and variance 1 is

$$f_X(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N}} e^{-(\mathbf{x}^T\mathbf{x})/2}.$$

The integral of $f_X(\mathbf{x})$ over a box is the probability that a draw from the distribution will be in the box. However, $f_X(\mathbf{x})$ does not have a symbolic antiderivative.

Use what you have learned about importance sampling to estimate the probability that a given random variable in $\mathbb{R}^2$ generated by f_X will be less than -1.75 in the x-direction and greater than 1.25 in the y-direction.

Treat f_X as the p.d.f. of your target distribution. Use the function `stats.multivariate_normal` to create a multivariate normal distribution to serve as your importance distribution. This function accepts a numpy array `mean` and a numpy array `cov`. The parameter `mean` is an array of the mean of each direction. The parameter `cov` is the covariance matrix. In this case, the covariance matrix will be a diagonal matrix with the variance of each variable down the diagonal.

Unnormalized Target Densities

The methods discussed so far are only applicable if the target density is normalized, or in other words, has an integral of 1. If the target density is not normalized, Equation 1.3 becomes,

$$\begin{aligned}
E[h(X)] &= \frac{\int h(t)f(t)\,dt}{\int f(t)\,dt} \\
&= \frac{\int h(t)f(t)\left(\frac{g_X(t)}{g_X(t)}\right)\,dt}{\int f(t)\left(\frac{g_X(t)}{g_X(t)}\right)\,dt} \\
&= \frac{\int \left(\frac{h(t)f(t)}{g_X(t)}\right) g_X(t)\,dt}{\int \left(\frac{f(t)}{g_X(t)}\right) g_X(t)\,dt} \\
&= \frac{E\left[\frac{h(Y)f(Y)}{g_X(Y)}\right]}{E\left[\frac{f(Y)}{g_X(Y)}\right]}
\end{aligned}$$

The corresponding estimator becomes,

$$\begin{aligned}\widehat{E}_n[h(X)] &= \frac{E\left[\frac{h(Y)f(Y)}{g_X(Y)}\right]}{E\left[\frac{f(Y)}{g_X(Y)}\right]} \\ &= \frac{\frac{1}{N}\sum_{i=1}^{N}\frac{h(y_i)f(y_i)}{g_X(y_i)}}{\frac{1}{N}\sum_{i=1}^{N}\frac{f(y_i)}{g_X(y_i)}}\end{aligned}$$