Lab 1

Importance Sampling and Monte Carlo Simulations

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

Introduction

Using 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 Y from the standard normal distribution is greater than 3. We know that one way to solve this by solving the following integral:

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

By the Law of the Unconscious Statistician, we can restate the integral above as

$$\int_3^\infty f_X(t)dt = \int_{-\infty}^\infty h(t)f_X(t)dt = E[h(X); f_X]$$

where $E[h(X); f_X]$ refers to the expected value of $h(X)$ with the random variable X being generated by f_X, and where

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

Monte Carlo Simulation

We can 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 $E[h(X); f_X]$ using

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

Now that we have defined the estimator, it is now quite manageable to approximate Equation 1.1. By the Weak Law of Large Numbers, 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. What estimate do you get when you use a sample with 10^7 points?

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:

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

The function f_X is the p.d.f of the *desired distribution*. The function g_X is the p.d.f of the *proposal distribution*. The fraction $\frac{f_X(X)}{g_X(X)}$ is called the *likelihood ratio*. This allows us to draw a sample from any distribution as long as we multiply $h(X)$ by the likelihood ratio. 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 because it will give us more points closer to and greater than 3.

```
>>> 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
```

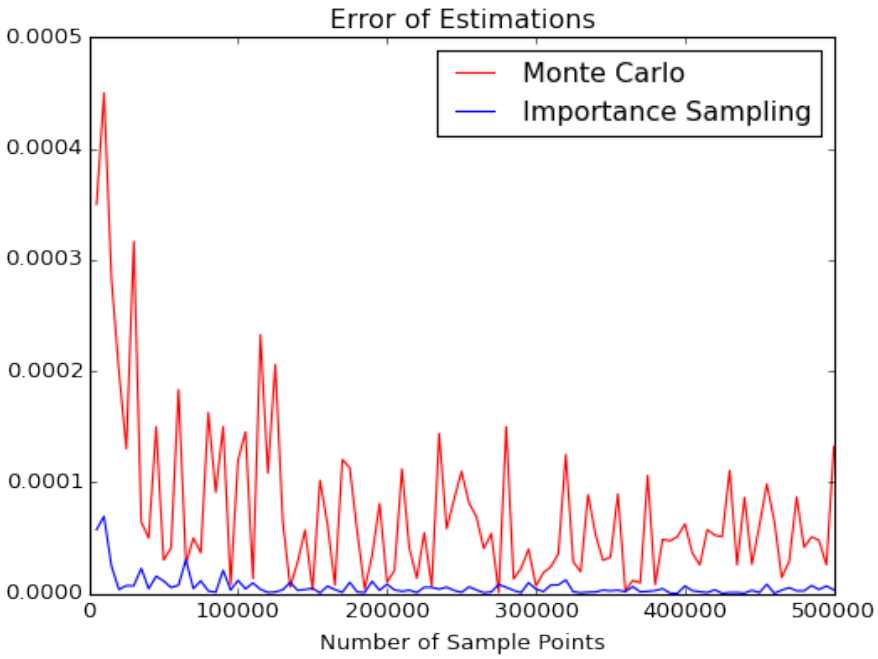

Figure 1.1: Comparison of error between standard method Monte Carlo and Importance Sampling method of Monte Carlo.

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. Your answers should approach 0.00208725.

Hint: The version of the gamma distribution is `scipy.stats` is determined by the shape and the scale (θ) of the distribution. You may be used to the gamma distribution being defined in terms of the shape and the rate (β). You can remedy this with the fact that $\theta = 1/\beta$.

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 1.1.

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, $h(t) = \begin{cases} 1 & \text{if condition is met} \\ 0 & \text{otherwise} \end{cases}$. This condition could be $t > 3$ as it was in our first example. Another condition could be whether a point lies under a curve or not. This would result in Monte Carlo Integration.

2. Define a function f_X which is the p.d.f. of the desired distribution. In the case of integration, this would be the uniform distribution.

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

Problem 4. ***Some problem of integrating something that doesn't have a closed form solution. I was thinking about maybe doing the joint-normal distribution like the last lab, but I thought that might be too repetitive.***