Project 4

Wenjun Li

February 26, 2020

Question 1

Approximate the following integrals using a Monte Carlo simulation. Compare your estimates with the exact values (if known):

a. $\int_{-2}^{2} e^{x^2+x}\,dx$.

b. $\int_{-\infty}^{+\infty} e^{-x^2}\,dx$.

c. $\int_0^1 \int_0^1 e^{-(x+y)^2}\,dy\,dx$.

Results and Analysis

One of the earliest applications of random numbers was in the computation of integrals. For example, if we want to compute θ where

$$\theta = \int_0^1 g(x)\,dx$$

The Monte Carlo approach approximates this integrals by generating a large number of random numbers u_i and taking their average value as the approximation value. We can express θ as

$$\theta = E[g(U)]$$

if $U = u_1, u_2, ..., u_n$ are i.i.d. uniform (0, 1) random variables, then according to the Strong Law of Large Numbers (SLLN), we can approximate the integral value with probability 1.

Part (a)
For part (a), we can substitute x with t as x = 4t - 2, i.e. t = (x+2)/4 and $dt = \frac{dx}{4}$. In this way, the upper bound of integration will be 1 and the lower bound of the integration will be 0. Then, let t be a standard uniformly distributed random variable.

$$\theta = \int_{-2}^{2} e^{x^2+x}\,dx = \int_0^1 e^{(4t-2)^2+4t-2}\,4dt = 4\int_0^1 e^{16t^2-12t+2}\,dt$$

for this problem, I randomly sample t for N = 10000 times and repeat the simulation for M = 10 and M = 100 times. The results are as below:

Theoretical Value: $\theta = 93.16$
Approximation Value: $\theta = 92.83 (M = 10), \theta = 92.93 (M = 100)$

Part (b)
For part (b), we can substitute x with t as x = -1 + 1/t, i.e. y = 1/(1+x) and dy = -y^2dx.

$$\theta = \int_{-\infty}^{+\infty} e^{-x^2}\,dx = 2\int_0^{+\infty} e^{-x^2}\,dt = 4\int_0^1 \frac{e^{-\frac{t^2}{(1-t)^2}}}{(1-t)^2}\,dt$$

for this part, I again simulate for N = 10000 times and repeat it for M = 10 and M = 100 times. The results are as below:

$$\textbf{Theoretical Value: } \theta = 1.7725$$
$$\textbf{Approximation Value: } \theta = 1.7689(M = 10), \theta = 1.7718(M = 100)$$

Part (c)
For part (c), we can directly sample x and y from standard uniform distribution and compute the approximation. The results are as below:

$$\textbf{Theoretical Value: } \theta = 0.4118$$
$$\textbf{Approximation Value: } \theta = 0.4131(M = 10), \theta = 0.4121(M = 100)$$

Summary In conclusion, the approximated values are very close to theoretical values and Monte Carlo can approximate the integrals very well.

Python Code

Listing 1: Code of Question 1

```python
import numpy as np


N = 10000       # simulation times
M = 100         # number of repeat

# part (a)
theta_a = []
for j in range(M):
    for i in range(N):
        x = np.random.uniform()
        x = 4 * np.exp(16*(x**2) - 12*x + 2)
        theta_a.append(x)
print('The result of Question1 part (a) is: {}'.format(np.mean(theta_a)))


# part (b)
theta_b = []
for j in range(M):
    for i in range(N):
        x = np.random.uniform()
        x = 2 * (np.exp(-(1/x - 1)**2) / (x**2))
        theta_b.append(x)
print('The result of Question1 part (a) is: {}'.format(np.mean(theta_b)))
```

```python


# part (c)
theta_c = []
for j in range(M):
    for i in range(N):
        x = np.random.uniform()
        y = np.random.uniform()
        t = np.exp(-(x+y)**2)
        theta_c.append(t)
print('The result of Question1 part (a) is: {}'.format(np.mean(theta_c)))


# Results when M = 10
# The result of Question1 part (a) is: 92.82611069576404
# The result of Question1 part (a) is: 1.7689143095766002
# The result of Question1 part (a) is: 0.41310349770717636


# Results when M = 100
# The result of Question1 part (a) is: 92.92954773287994
# The result of Question1 part (a) is: 1.7718544002728807
# The result of Question1 part (a) is: 0.41212223311226553
```

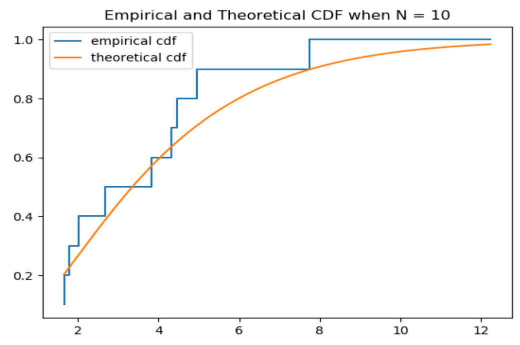
Question 2

Define the random variable $X = Z_1^2+Z_2^2+Z_3^2+Z_4^2$ where Z_k $N(0,1)$. Then X $\chi^2(4)$. Generate 10 samples from X by first sampling Z_i for $i = 1,2,3,4$ and then computing X. Plot the empirical distribution $F^*_{10}(x)$ for your samples and overlay the theoretical distribution $F(x)$. Estimate a lower bound for $\|F^*_{10}(x) - F(x)\|_{+\infty}$ by computing the maximum difference at each of your samples: . Then find the 25th, 50th, and 90th percentiles using your empirical distribution and compare the value to the theoretical percentile values for $\chi^2(4)$. Repeat the above using 100 and 1000 samples from X.

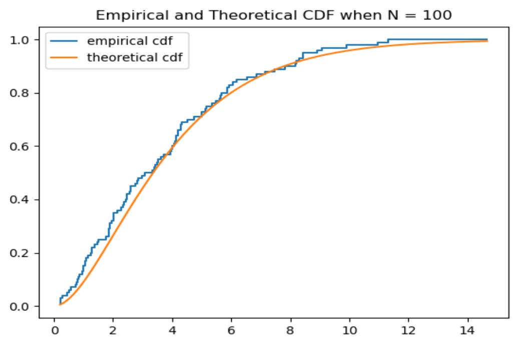
Results and Analysis

In this question, we need to obtain the CDF given n samples $x_1, x_2, ..., x_n$ of X. We can generate empirical distribution function which is a step function. In this problem, first I generate N = 10 samples Z_1, Z_2, Z_3, Z_4 from standard normal distribution N(0, 1). Then I use them to obtain $\chi^2(4) = Z_1^2 + Z_2^2 + Z_3^2 + Z_4^2$. Finally, I plot this empirical distribution and the theoretical $\chi^2(4)$ CDF in the same plot. To approximate the theoretical CDF, I improve the N and tried N = 100, 1000. The plots are shown in Figure 1. As you can see from the figure, maximum difference and the percentiles, we can conclude that the empirical distribution function approximates the theoretical CDF very well. This question also asks the maximum difference at each of the samples and the 25th, 50th, and 90th percentiles. The results are displayed below:

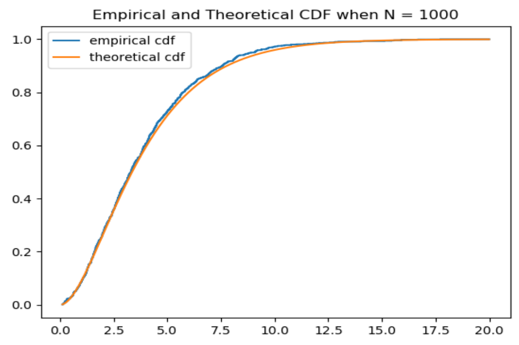
when n = 10: max difference = 0.2263395573321131
the 0.25th percentile of empirical is 1.4566; the 0.25th percentile of theoretical is 1.9225
the 0.50th percentile of empirical is 2.9102; the 0.5th percentile of theoretical is 3.3566
the 0.90th percentile of empirical is 7.1764; the 0.9th percentile of theoretical is 7.7794
when n = 100: max difference = 0.07252711828734393
the 0.25th percentile of empirical is 1.8216; the 0.25th percentile of theoretical is 1.9225
the 0.50th percentile of empirical is 3.4965; the 0.5th percentile of theoretical is 3.3566
the 0.90th percentile of empirical is 7.6821; the 0.9th percentile of theoretical is 7.7794
when n = 1000: max difference = 0.02119088123190105
the 0.25th percentile of empirical is 1.9559; the 0.25th percentile of theoretical is 1.9225
the 0.50th percentile of empirical is 3.2994; the 0.5th percentile of theoretical is 3.3566
the 0.90th percentile of empirical is 7.8475; the 0.9th percentile of theoretical is 7.7794



(a)



(b)



(c)

Figure 1: Empirical Distribution Function and Theoretical CDF.

Python Code

Listing 2: Code of Question 2

```python
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt


N = 1000
x = []

for i in range(N):
    z1 = np.random.randn()
    z2 = np.random.randn()
    z3 = np.random.randn()
    z4 = np.random.randn()
    x.append(z1**2 + z2**2 + z3**2 + z4**2)

x = sorted(x)
t = np.arange(min(x), max(x), 0.01)
theo_cdf = stats.chi2.cdf(t, 4)

# plot
plt.figure()
plt.step(x, np.arange(1/N, 1+1/N, 1/N), label='empirical cdf')
plt.plot(t, theo_cdf, label='theoretical cdf')
plt.title('Empirical and Theoretical CDF when N = {}'.format(N))
plt.legend()
plt.show()


print('when n = {}'.format(N))
# maximum difference
max_diff = max(abs(stats.chi2.cdf(x, 4) - np.arange(1/N, 1+1/N, 1/N)))
print('max difference = {}'.format(max_diff))


# percentile
percentile = [0.25, 0.50, 0.9]
for p in percentile:
    p_emp = []
    for i in range(N):
        if (i+1)/N >= p:
            p_emp = x[i]
            break
    print('the {}th percentile of empirical is {}'.format(p, p_emp))

    p_theo = stats.chi2.ppf(p, 4)
    print('the {}th percentile of theoretical is {}'.format(p, p_theo))


# when n = 10
# max difference = 0.2263395573321131
# the 0.25th percentile of empirical is 1.4566224439688158
# the 0.25th percentile of theoretical is 1.922557526229554
# the 0.5th percentile of empirical is 2.910229780077728
# the 0.5th percentile of theoretical is 3.3566939800333224
# the 0.9th percentile of empirical is 7.176424566950743
# the 0.9th percentile of theoretical is 7.779440339734858


# when n = 100
# max difference = 0.07252711828734393
# the 0.25th percentile of empirical is 1.821696497714953
# the 0.25th percentile of theoretical is 1.922557526229554
# the 0.5th percentile of empirical is 3.496553346505567
# the 0.5th percentile of theoretical is 3.3566939800333224
# the 0.9th percentile of empirical is 7.682103514367712
# the 0.9th percentile of theoretical is 7.779440339734858


# when n = 1000
# max difference = 0.02119088123190105
# the 0.25th percentile of empirical is 1.9559444628799594
# the 0.25th percentile of theoretical is 1.922557526229554
# the 0.5th percentile of empirical is 3.299465137514017
# the 0.5th percentile of theoretical is 3.3566939800333224
# the 0.9th percentile of empirical is 7.847506922588133
# the 0.9th percentile of theoretical is 7.779440339734858
```

Question 3

A geyser is a hot spring characterized by an intermittent discharge of water and steam. Old Faithful is a famous cone geyser in Yellowstone National Park, Wyoming. It has a predictable geothermal discharge and since 2000 it has erupted every 44 to 125 minutes. Refer to the addendum data file that contains waiting times and the duration for 272 eruptions. Compute a 95% statistical confidence interval for the mean waiting time using data from only the first

15 eruptions. Compare this to a 95% bootstrap confidence interval using the same 15 data samples. Repeat these calculations using all the data samples. Comment on the relative width of the confidence intervals when using only 15 samples vs using all samples.

Results and Analysis

In this problem, we need to compute the statistical Confidence Interval (C.I.) and the bootstrap Confidence Interval. To do this, I first load the data into array and then transfer them into lists. To compute the statistical C.I., I wrote two functions for 15 data and all data, respectively. For 95% C.I., the $Z_{0.025}$ is 1.96, so the lower bound is $mean - Z_{0.025} * \frac{std}{\sqrt{n}}$ and the upper bound is $mean + Z_{0.025} * \frac{std}{\sqrt{n}}$. To compute the Bootstrap C.I., we need to sample from the data with replacement. The simulation results are shown below.

From the results, we can say that the difference between the bootstrap C.I. and the statistical C.I. is very small and the difference becomes smaller as the data amount increases.

95% Statistical C.I. for 15 data: lower bound: 63.28; upper bound: 78.59
95% Bootstrap C.I. for 15 data: lower bound: 63.66; upper bound: 77.53
95% Statistical C.I. for all data: lower bound: 69.28; upper bound: 72.51
95% Bootstrap C.I. for all data: lower bound: 69.20; upper bound: 72.47

Python Code

Listing 3: Code of Question 3

```python
import numpy as np
import random


# load data
data = np.loadtxt('faithful.dat.txt', skiprows=26)
data = data[:, 2]
data_15 = data[0:15]

data = list(data)
data_15 = list(data_15)


def statistical_CI(data_):
    mean = np.mean(data_)
    std = np.std(data_, ddof=1)
    z = 1.96
    return [mean - z*(std/np.sqrt(len(data_))), mean + z*(std/np.sqrt(len(data_)))
        ]
```

```


def bootstrap_CI(data_):
    sample_mean_history = []
    for i in range(1000):
        samples = []
        for j in range(len(data_)):
            samples.append(random.sample(data, 1))
        sample_mean_history.append(np.mean(samples))
    return [np.percentile(sample_mean_history, 2.5), np.percentile(
       sample_mean_history, 97.5)]


def run_15(data_):
    lower, upper = statistical_CI(data_)
    print('95% Statistical C.I. for 15 data: lower bound: {}; upper bound: {}'.
       format(lower, upper))
    lower, upper = bootstrap_CI(data_)
    print('95% Bootstrap C.I. for 15 data: lower bound: {}; upper bound: {}'.
       format(lower, upper))


def run_all(data_):
    lower, upper = statistical_CI(data_)
    print('95% Statistical C.I. for all data: lower bound: {}; upper bound: {}'.
       format(lower, upper))
    lower, upper = bootstrap_CI(data_)
    print('95% Bootstrap C.I. for all data: lower bound: {}; upper bound: {}'.
       format(lower, upper))


# run
run_15(data_15)
run_all(data)


# output
# 95% Statistical C.I. for 15 data: lower bound: 63.278770830413194; upper bound:
   78.58789583625348
# 95% Bootstrap C.I. for 15 data: lower bound: 63.666666666666664; upper bound:
   77.535
# 95% Statistical C.I. for all data: lower bound: 69.28139874542947; upper bound:
   72.51271890162934
```

```
# 95% Bootstrap C.I. for all data: lower bound: 69.19806985294117; upper bound:
    72.46700367647058
```