HOMEWORK 1: Exercises for Monte Carlo Methods

Student ID: 21307174 Student Name:刘俊杰

Lectured by 梁上松, Sun Yat-sen University

Exercise 1.

The Monte Carlo method can be used to generate an approximate value of pi. The figure below shows a unit square with a quarter of a circle inscribed. The area of the square is 1 and the area of the quarter circle is pi/4. Write a script to generate random points that are distributed uniformly in the unit square. The ratio between the number of points that fall inside the circle (red points) and the total number of points thrown (red and green points) gives an approximation to the value of pi/4. This process is a Monte Carlo simulation approximating pi. Let N be the total number of points thrown. When N=50, 100, 200, 300, 500, 1000, 5000, what are the estimated pi values, respectively? For each N, repeat the throwing process 100 times, and report the mean and variance. Record the means and the corresponding variances in a table.

蒙特卡洛方法可以用于产生接近 pi 的近似值。图 1 显示了一个带有 1/4 内切圆在内的边长为 1 的正方形。正方形的面积是 1，该 1/4 圆的面积为 pi/4。通过编程实现在这个正方形中产生均匀分布的点。落在圈内（红点）的点和总的投在正方形（红和绿点）上的点的比率给出了 pi/4 的近似值。这一过程称为使用蒙特卡洛方法来仿真逼近 pi 实际值。令 N 表示总的投在正方形的点。当投点个数分别是 20, 50, 100, 200, 300, 500, 1000, 5000 时，pi 值分别是多少？对于每个 N，每次实验算出 pi 值，重复这个过程 100 次，并在表中记下均值和方差。

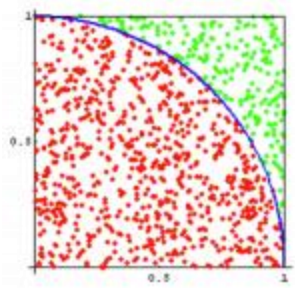

Figure 1 蒙特卡洛方法求解 pi

在第一象限边长为1的正方形中生成随机数坐标进行投点，统计总的投点个数和落在离原点距离<小于等于1的点个数，用落在圆内的点个数除以总的投点个数，得到1/4圆的面积pi/4，最后结果乘以4便是逼近的pi值。

关键函数:

```
import random
# 落点函数(随机产生落在边长为1的正方形中的点的坐标)
def point():
    return [random.uniform(0,1),random.uniform(0,1)]

# 计算pi值
def simulate(n):
    in_circle = 0#统计落在1/4圆内点的个数
    for i in range(n):
        item=point()
        if item[0]**2+item[1]**2<=1:
            in_circle+=1
    pi = in_circle/n*4 #求得落在1/4圆内的概率乘以4得到pi的估计值
    return pi
```

```
# 计算平均值和方差
def calculate(ans,n):
    sum = 0
    for val in ans:
        sum += val
    aver = sum/n
    var = 0
    for val in ans:
        var += (val-aver)**2
    var /= n
    return aver,var
```

进行模拟投点计算:

```
#模拟次数
iterations = [20, 50, 100, 200, 300, 500, 1000, 5000]
#开始模拟
for iter in iterations:
    pi=[]
    for i in range(100):
        pi.append(simulate(iter))
    aver,var = calculate(pi,100)
    print("模拟",iter,"次","均值: ",aver," 方差: ",var)
```

实验结果:

```
PS D:\d_code\MLDM\Lab1> C:\Users\刘俊杰\AppData\Local\Programs\Python\Python39\python.exe "d:\d_code\MLDM\Lab1\ex1.py"
模拟 20 次 均值:  3.1819999999999986  方差:  0.11527600000000005
模拟 50 次 均值:  3.149600000000001  方差:  0.06651584
模拟 100 次 均值:  3.1395999999999993  方差:  0.030191839999999984
模拟 200 次 均值:  3.148599999999999  方差:  0.011954039999999989
模拟 300 次 均值:  3.1337333333333333  方差:  0.009292284444444458
模拟 500 次 均值:  3.134000000000001  方差:  0.005276640000000002
模拟 1000 次 均值:  3.1520799999999998  方差:  0.0026403135999999994
模拟 5000 次 均值:  3.1419120000000005  方差:  0.0005293170559999988
PS D:\d_code\MLDM\Lab1>
```

模拟次数	均值	方差
20	3.1819999999999986	0.11527600000000005
50	3.149600000000001	0.06651584
100	3.1395999999999993	0.030191839999999984
200	3.148599999999999	0.011954039999999989
300	3.1337333333333333	0.009292284444444458
500	3.134000000000001	0.005276640000000002
1000	3.1520799999999998	0.0026403135999999994
5000	3.1419120000000005	0.0005293170559999988

可以从实验结果中看出，随着模拟投点的次数的增大，pi的值越来越逼近pi的实际值。

且随着投点次数的增大，方差越来越小，说明投点次数大估计出的pi值较为接近，说明足够多的投点可以逼近pi的真实值。

Exercise 2.

We are now trying to integrate the another function by Monte Carlo method:

$$\int_0^1 x^3$$

A simple analytic solution exists here: $\int_{x=0}^{1} x^3 = 1/4$. If you compute this integration using Monte Carlo method, what distribution do you use to sample x? How good do you get when N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100, respectively? For each N, repeat the Monte Carlo process 100 times, and report the mean and variance of the integrate in a table.

我们现在尝试通过蒙特卡洛的方法求解如下的积分:

$$\int_0^1 x^3$$

该积分的求解我们可以直接求解，即有$\int_{x=0}^{1} x^3 = 1/4$。如果你用蒙特卡洛的方法求解该积分，你认为 x 可以通过什么分布采样获得 ?如果采样次数是分别是 N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100，积分结果

有多好 ?对于每个采样次数 N，重复蒙特卡洛过程 100 次，求出均值和方差，然后在表格中记录对应的均值和方差。

使用蒙特卡洛法求解一重积分：
在积分区间[a,b]中随机取点 $x_1,x_2,x_3,\cdots\cdots,x_n$。可利用每个点的函数值f($x_i$)X(b-a)估计求得函数曲线下的面积来估计积分的值，取n个点，则估计的积分值为:

$$\int_a^b f(x)dx = \frac{b-a}{n}\sum_{i=1}^{n} f(x_i)$$

关键函数:

```python
import random
import math
# 求x的三次方
def function_x(x):
    x=math.pow(x,3)
    return x
# 返回a到b的随机值
def random_x(a,b):
    return random.uniform(a,b)
# 计算均值和方差
def calculate(ans,n):
    sum = 0
    for val in ans:
        sum += val
    aver = sum/n
    var = 0
    for val in ans:
        var += (val-aver)**2
    var /= n
    return aver,var
```

进行模拟，计算均值和方差:

```python
a,b =  0,1
N = [5, 10, 20, 30, 40, 50, 60, 70, 80, 100]
for n in N:
    ans = []
    for m in range(100):
        total = 0
        for i in range(n):
            x = random_x(a,b)
            total += function_x(x)*(b-a)
        total /= n
        ans.append(total)
    aver,var = calculate(ans,100)
    print("模拟",n,"次","均值: ",aver," 方差: ",var)
```

实验结果:

```
PS D:\d_code\MLDM\Lab1> C:\Users\刘俊然\AppData\Local\Programs\Python\Python39\python.exe d:\d_code\MLDM\Lab1\ex2.py
模拟 5 次 均值:  0.22289920241802935  方差:  0.013862603983146189
模拟 10 次 均值:  0.2557115107647648  方差:  0.009734256927947429
模拟 20 次 均值:  0.2575390174459747  方差:  0.0041676330380882665
模拟 30 次 均值:  0.24572178318759785  方差:  0.0031138319933425597
模拟 40 次 均值:  0.2488016499837002  方差:  0.0016865299228297987
模拟 50 次 均值:  0.2571874205311376  方差:  0.001505772676829946
模拟 60 次 均值:  0.2435003216409662  方差:  0.001265535343037301
模拟 70 次 均值:  0.2533628893043975  方差:  0.001225704134709604
模拟 80 次 均值:  0.2554706192002742  方差:  0.001114543395526967
模拟 100 次 均值:  0.2551214290876304  方差:  0.0006702462336641791
PS D:\d_code\MLDM\Lab1>
```

模拟采样次数	均值	方差
5	0.22289920241802935	0.013862603983146189
10	0.2557115107647648	0.009734256927947429
20	0.2575390174459747	0.0041676330380882665
30	0.24572178318759785	0.0031138319933425597
40	0.2488016499837002	0.0016865299228297987
50	0.2571874205311376	0.001505772676829946
60	0.2435003216409662	0.001265535343037301
70	0.2533628893043975	0.001225704134709604
80	0.2554706192002742	0.001114543395526967
100	0.2551214290876304	0.0006702462336641791

可以从实验结果中看出，随着模拟采样的次数的增大，求得积分的均值值越来越逼近积分的实际值0.25。且随着采样次数的增大，方差越来越小，说明积分值较为接近，说明足够多的采样可以逼近积分的真实值。

Exercise 3.

We are now trying to integrate a more difficult function by Monte Carlo method that may not be analytically computed:

$$\int_{x=2}^{4}\int_{y=-1}^{1} f(x,y) = \frac{y^2 * e^{-y^2} + x^4 * e^{-x^2}}{x * e^{-x^2}}$$

Can you compute the above integration analytically? If you compute this integration using Monte Carlo method, what distribution do you use to sample (x,y)? How good do you get when the sample sizes are N = 5, 10, 20, 30, 40, 50, 60, 70, 80, 100, 200 respectively? For each N, repeat the Monte Carlo process 100 times, and report the mean and variance of the integrate.

我们现在尝试通过蒙特卡洛的方法求解如下的更复杂的积分：

$$\int_{x=2}^{4}\int_{y=-1}^{1} f(x,y) = \frac{y^2*e^{-y^2}+x^4*e^{-x^2}}{x*e^{-x^2}}$$

你能够通过公式直接求解上述的积分吗？如果你用蒙特卡洛的方法求解该积分，你认为(x, y)可以通过什么分布采样获得？如果点（x, y）的采样次数是分别是 N = 10, 20, 30, 40, 50, 60, 70, 80, 100, 200, 500, 积分结果有多好？对于每个采样次数 N，重复蒙特卡洛过程 100 次，求出均值和方差，然后在表格中记录对应的均值和方差 。

使用蒙特卡洛法求解一重积分：

在x积分区间$[a_x,b_x]$中随机取点x，在y积分区间$[a_y,b_y]$中随机取点y ，(x_1,y_1)，(x_2,y_2),……， (x_n,y_n)。可利用每个点的函数值f(x_i,y_i)X$[b_x - a_x]$X$[b_y - a_y]$估计求得函数曲面下的体积来估计积分的值，取n个点，则估计的积分值为:

$$\int_{a_y}^{b_y}\int_{a_x}^{b_x} f(x,y)dxdy = \frac{(b_x - a_x)X(b_y - a_y)}{n}\sum_{i=1}^{n} f(x_i,y_i)$$

关键函数:

```
import random
import math
def function_xy(x,y):#计算函数值
    e=math.exp(1)
    x2=math.pow(x,2)
    y2=math.pow(y,2)
    xy = (x2**2*math.pow(e,-x2)+y2*math.pow(e,-y2))/(x*math.pow(e,-x2))
    return xy
def random_xy(a,b):#产生随机数
    return random.uniform(a,b)
def calculate(ans,n):#计算均值和方差
    sum = 0
    for val in ans:
        sum += val
    aver = sum/n
    var = 0
    for val in ans:
        var += (val-aver)**2
    var /= n
    return aver,var
```

进行试验计算均值和方差:

```
ax,bx =  2,4
ay,by =  -1,1
N = [10, 20, 30, 40, 50, 60, 70, 80, 100, 200, 500]#模拟次数
area_D = math.fabs(ax-bx)* math.fabs(ay-by)
for n in N:
    ans = []
    for m in range(100):
        total = 0
        for i in range(n):
            x = random_xy(ax,bx)
            y = random_xy(ay,by)
            total += function_xy(x,y)*area_D
        total /= n
        ans.append(total)
    aver,var = calculate(ans,100)
    print("模拟",n,"次","均值: ",aver," 方差: ",var)
```

实验结果:

```
PS D:\d_code\MLDM\Lab1> C:\Users\刘俊杰\AppData\Local\Programs\Python\Python39\python.exe "d:\d_code\MLDM\Lab1\ex3.py"
模拟 10 次 均值： 117967.59578337282 方差： 10073986643.63734
模拟 20 次 均值： 112462.44030439557 方差： 5257709795.15385
模拟 30 次 均值： 120961.3266901963 方差： 5497784811.913334
模拟 40 次 均值： 102672.38210588814 方差： 3205367363.6630235
模拟 50 次 均值： 117170.1883853467 方差： 2438581533.3738456
模拟 60 次 均值： 112988.51764158819 方差： 2046589283.4134588
模拟 70 次 均值： 116029.50263871184 方差： 2035289645.0119581
模拟 80 次 均值： 105484.74255320849 方差： 1263437484.79846
模拟 100 次 均值： 115157.87663063737 方差： 1194334179.1666288
模拟 200 次 均值： 113991.48111217182 方差： 522105604.4410331
模拟 500 次 均值： 113367.30375398157 方差： 289329812.1839975
PS D:\d_code\MLDM\Lab1>
```

模拟采样次数	均值	方差
10	117967.59578337282	10073986643.63734
20	112462.44030439557	5257709795.15385
30	120961.3266901963	5497784811.913334
40	102672.38210588814	3205367363.6630235
50	117170.1883853467	2438581533.3738456
60	112988.51764158819	2046589283.4134588
70	116029.50263871184	2035289645.0119581
80	105484.74255320849	1263437484.79846
100	115157.87663063737	1194334179.1666288
200	113991.48111217182	522105604.4410331
500	113367.30375398157	289329812.1839975

可以从实验结果中看出，随着模拟采样的次数的增大，求得积分的均值值越来越逼近积分的实际值。且随着采样次数的增大，方差越来越小，说明积分值较为接近，说明足够多的采样可以逼近积分的真实值。

Exercise 4.

An ant is trying to get from point A to point B in a grid. The coordinates of point A is (1,1) (this is top left corner), and the coordinates of point B is (n,n) (this is bottom right corner, n is the size of the grid).

Once the ant starts moving, there are four options, it can go left, right, up or down (no diagonal movement allowed). If any of these four options satisfy the following:

(a) The new point should still be within the boundaries of the n×n grid

(b) Only the center point (4, 4) is allowed to be visited zero, one or two times, while the remainder points should not be visited previously (are allowed to be visited zero or one time).

IfP is the probability of the ant reaching point B for a 7×7 grid, use Monte Carlo simulation to compute P. Pick the answer closest to P in value (assume 20,000 simulations are sufficient enough to compute P).

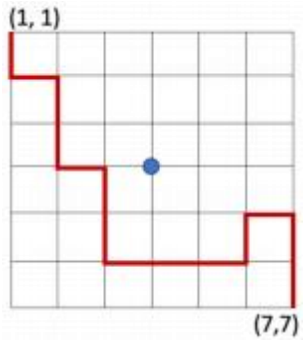

Figure 2 An ant is trying to get from point A (1,1) to point B (7,7) in a grid.

蚂蚁除(4,4)可以经过两次外，其他点只能经过一次，当蚂蚁四个方向都无法移动时则蚂蚁的移动失败。每次的移动都是随机选择一个可移动的方向进行移动，计算能到达(7,7)的成功率。

关键函数:

```python
import random
import math
dir = [[0,1],[1,0],[-1,0],[0,-1]]#移动方向
def init_visited():#初始化每个点可被访问的次数
    visited = [[1 for i in range(8)] for j in range(8)]
    visited[4][4]=2
    visited[1][1]=0
    return visited
def if_fail(visited,x,y):#判断是否四个方向都无法移动，并返回可移动的方向
    global dir
```

```
    flag = False
    index=[]
    for i in range(4):
        xt = x+dir[i][0]
        yt = y+dir[i][1]
        if xt<1 or yt<1 or xt>7 or yt >7 or visited[xt][yt]<1:
            continue
        if visited[xt][yt]>0:
            flag = True
            index.append(i)
    return flag,index
```

进行试验，计算成功率:

```
succeed=0#成功的次数
for iter in range(20000):#进行20000次模拟
    visited=init_visited()#初始化visited,visited储存了每个点可访问的次数
    x,y =1,1#x,y当前位置
    while True:
        if  x==7 and y==7:#到达目的地
            succeed+=1
            break
        flag,step=if_fail(visited,x,y)#flag判断是否还能继续,step返回可移动的方向
        if flag==False:#无法移动，此次模拟失败
            break
        index=random.randint(0,len(step)-1)#随机选择可移动方向
        xt = x+dir[step[index]][0]
        yt = y+dir[step[index]][1]
        x = xt
        y = yt
        visited[x][y]-=1#该点访问次数减1
print(succeed/20000)
```

实验结果:

```
PS D:\d_code\MLDM\Lab1> C:\Users\刘俊杰\AppData\Local\Programs\Python\Python39\python.exe "d:\d_code\MLDM\Lab1\ex4.py"
0.2561
PS D:\d_code\MLDM\Lab1>
```

通过多次实验，能成功到达目的地的成功率为0.25左右。

Exercise 5

Given a system made of discrete components with known reliability, what is the reliability of the overall system? For example, suppose we have a system that can be described with the following high-level diagram:

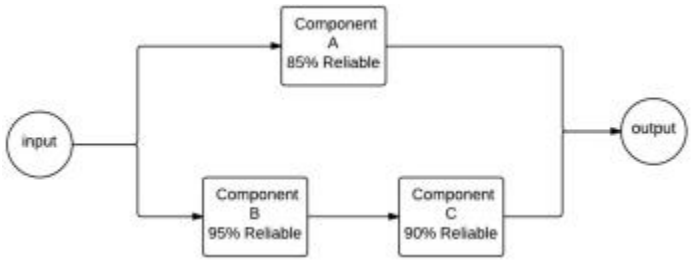

Figure 3 A system made of discrete components.

When given an input to the system, that input flows through component A or through components B and C, each of which has a certain reliability of correctness. Probability theory tells us the following:

$$reliability_{BC} = 0.95 * 0.90 = 0.855$$

reliality$A = 0.85$

And the overall reliability of the system is:

$$\begin{aligned} reliability_{sys} &= 1.0 - [(1.0 - 0.85) * (1.0 - 0.855)] \\ &= 0.97825 \end{aligned}$$

Create a simulation of this system where half the time the input travels through component A. To simulate its reliability, generate a number between 0 and 1. If the number is 0.85 or below, component A succeeded, and the system works. The other half of the time, the input would travel on the lower half of the diagram. To simulate this, you will generate two numbers between 0 and 1. If the number for component B is less than 0.95 and the number for component C is less than 0.90, then the system also succeeds. Run many trials to see if you converge on the same reliability as predicted by probability theory.

根据题目对系统的假设模拟为一半的输入经过A,另一半的输入经过B和C，这种假设模拟通过的成功率理论值为0.855*0.5+0.85*0.5=0.8525

而若该系统为并联系统，即A和BC只要有一条路径可通过，则本次输入可通过，并联的成功率理论值为:
进行试验,计算成功率:

$$\begin{aligned} reliability_{sys} &= 1.0 - [(1.0 - 0.85) * (1.0 - 0.855)] \\ &= 0.97825 \end{aligned}$$

```
import random
N = [5, 10, 20, 30, 40, 50, 60, 70, 80, 100,500,1000,10000,100000]#模拟次数
for n in N:
    succeeds = 0#题目假设成功次数
    succeeds2 = 0#并联成功次数
    for i in range(n):
        #模拟题目假设
        if i<n//2:
            a=random.uniform(0,1)
            if a<=0.85:
                succeeds+=1
        else:
            b=random.uniform(0,1)
            c=random.uniform(0,1)
            if b<=0.95 and c<0.90:
                succeeds+=1
        #模拟并联
        a,b,c=random.uniform(0,1),random.uniform(0,1),random.uniform(0,1)
        if a<0.85 or(b<0.95 and c<0.90):
            succeeds2+=1
    print("模拟次数: ",n," 题目假设的成功率: ",succeeds/n," 并联的成功率: ",succeeds2/n)
```

实验结果:

```
PS D:\d_code\MLDM\Lab1> C:\Users\刘俊杰\AppData\Local\Programs\Python\Python39\python.exe "d:\d_code\MLDM\Lab1\ex5.py"
模拟次数： 5 题目假设的成功率： 0.6 并联的成功率： 1.0
模拟次数： 40 题目假设的成功率： 0.775 并联的成功率： 0.975
模拟次数： 50 题目假设的成功率： 0.74 并联的成功率： 0.98
模拟次数： 60 题目假设的成功率： 0.8833333333333333 并联的成功率： 1.0
模拟次数： 70 题目假设的成功率： 0.8571428571428571 并联的成功率： 1.0
模拟次数： 80 题目假设的成功率： 0.8125 并联的成功率： 0.9625
模拟次数： 100 题目假设的成功率： 0.85 并联的成功率： 0.99
模拟次数： 500 题目假设的成功率： 0.874 并联的成功率： 0.974
模拟次数： 1000 题目假设的成功率： 0.846 并联的成功率： 0.977
模拟次数： 10000 题目假设的成功率： 0.8476 并联的成功率： 0.9796
模拟次数： 100000 题目假设的成功率： 0.85451 并联的成功率： 0.97737
PS D:\d_code\MLDM\Lab1>
```

模拟次数	题目假设成功率	并联成功率
5	0.6	1.0
40	0.775	0.975
50	0.74	0.98
60	0.8833333333333333	1.0
70	0.8571428571428571	1.0
80	0.8125	0.9625
100	0.85	0.99
500	0.874	0.974
1000	0.846	0.977
10000	0.8476	0.9796
100000	0.85451	0.97737

可以看出经过多次实验，模拟的次数越多，题目假设的成功率接近理论值0.8525，并联的成功率均值越接近理论值0.97825。