机器学习与数据挖掘

Homework 1: Exercises for Monte Carlo Methods

中山大学计算机学院　计算机科学与技术

19335174　施天予

一、Exercise 1

The Monte Carlo method can be used to generate an approximate value of π. 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 π. Let N be the total number of points thrown. When N=20, 50, 100, 200, 300, 500, 1000, 5000, what are the estimated π 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.

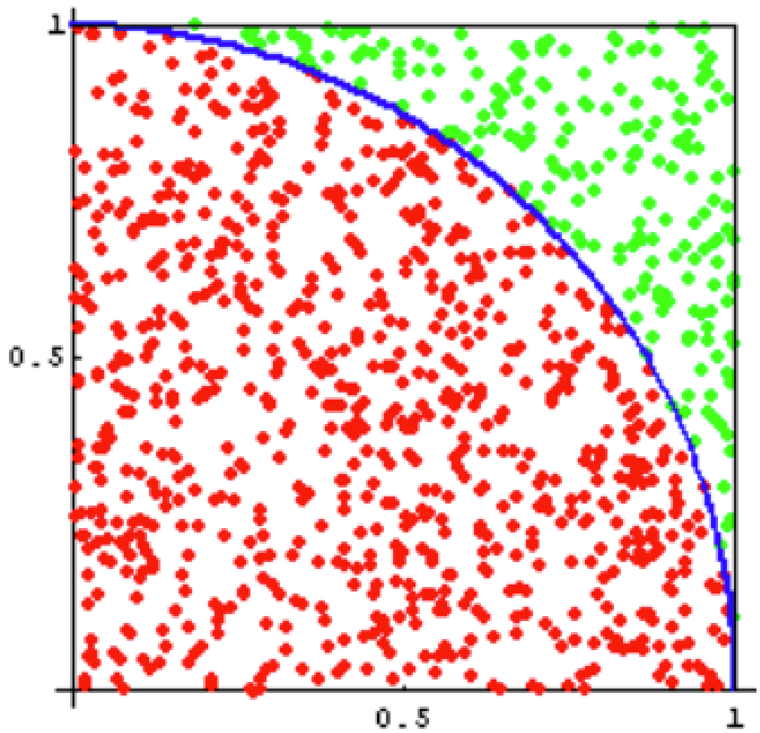

图 1: Monte Carlo method for solving π

解:

点的数量	均值（保留 5 位小数）	方差（保留 7 位小数）
20	3.14000	0.1124000
50	3.13600	0.0599040
100	3.12080	0.0261113
200	3.14579	0.0156023
300	3.14173	0.0082938
500	3.14416	0.0048204
1000	3.14864	0.0020344
5000	3.14162	0.0003446

二、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.

解:

- 可以通过均匀分布采样获得。
- N 越大，与真实值接近的概率更大，方差越小。

采样次数	均值（保留 5 位小数）	方差（保留 7 位小数）
5	0.23600	0.0363039
10	0.27800	0.0181159
20	0.24400	0.0095640
30	0.24833	0.0057194
40	0.23099	0.0040514
50	0.25200	0.0037680
60	0.24666	0.0030722
70	0.24942	0.0028955
80	0.24437	0.0023417
100	0.25020	0.0016859

三、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.

解：

- 难以求解原式积分。
- 可以通过均匀分布采样获得。
- N 越大，结果越接近真实值，方差越小。

采样次数	均值（保留 5 位小数）	方差（保留 7 位小数）
10	105154.61839	11026153884.2830900
20	118750.84635	8097665112.9911380
30	112162.47066	4012623186.1539345
40	109888.15756	2719273469.1368785
50	113178.45292	2319735154.6518600
60	106378.88092	1695133181.4403653
70	118115.57863	1911217325.3331897
80	117888.80739	1366377170.7710676
100	112832.32256	1513035346.3998237
200	112118.34558	580698131.4926296
500	112636.69551	194733607.8384374

四、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).

If P 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).

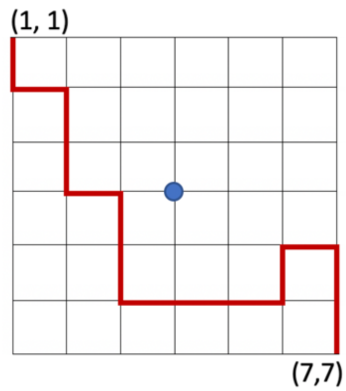

图 2: An ant is trying to get from point A (1,1) to point B (7,7) in a grid

解：

- 采样的次数为 20, 200, 2000, 20000。各重复 100 次。
- N 越大，结果越接近真实值，方差越小。
- 使用蒙特卡洛法模拟得出的最接近 P 的概率约为 0.2562。

采样次数	**均值（保留 4 位小数）**	**方差（保留 7 位小数）**
20	0.2525	0.0080688
200	0.2579	0.0008640
2000	0.2548	0.0001144
20000	0.2562	0.0000009

五、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:

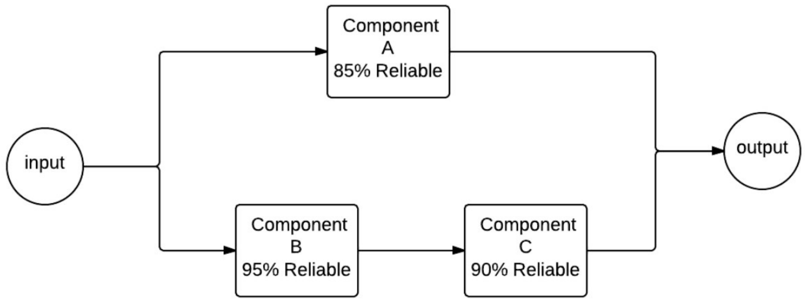

图 3: A system made of discrete components

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:

$$reliability_{BC} = 0.95 * 0.90 = 0.855 reliability_{A} = 0.85$$

And the overall reliability of the system is:

$$reliability_{sys} = 1.0 - [(1.0 - 0.85) * (1.0 - 0.855)] = 0.97825$$

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.

解:

- 采样的次数为 10，20，30，40，50，60，70，80，100，200。各重复 100 次。
- N 越大，结果越接近真实值，方差越小。

如果 A 和 BC 各分一半时间通过，当采样次数为 200 时，结果已经非常接近真实值 $(0.85 + 0.9 * 0.95)/2 = 0.8525$

采样次数	均值（保留 5 位小数）	方差（保留 7 位小数）
10	0.84300	0.0154510
20	0.85300	0.0064409
30	0.84066	0.0041906
40	0.85100	0.0030615
50	0.85519	0.0034009
60	0.85049	0.0018469
70	0.85128	0.0018412
80	0.85187	0.0014855
100	0.85610	0.0011137
200	0.85280	0.0006461

如果 A 和 BC 只要有一个通过就通过，当采样次数为 200 时，结果已经非常接近真实值 0.97825。

采样次数	均值（保留 5 位小数）	方差（保留 7 位小数）
10	0.97200	0.0028159
20	0.97700	0.0009710
30	0.97967	0.0007309
40	0.97950	0.0006172
50	0.97979	0.0003159
60	0.97766	0.0004845
70	0.98114	0.0002689
80	0.97912	0.0002407
100	0.97910	0.0001841
200	0.97825	0.0000871

附录 A. 代码

1. Exercise 1

```
import random
import math
import numpy as np

n = 100
nodes_list = [20, 50, 100, 200, 300, 500, 1000, 5000]

# 随机获取一个点，并判断是否在 1/4 个圆上
def getOneNode():
    x = random.random()
```

```
    y = random.random()
    return math.pow(x, 2)+math.pow(y, 2) <= 1

# 蒙特卡洛法计算pi值
def getPi(nodes):
    circle = 0
    for _ in range(nodes):
        if getOneNode():
            circle += 1
    return 4*(circle/nodes)

# 计算均值和方差
for nodes in nodes_list:
    s = np.empty(n)
    for i in range(n):
        s[i] = getPi(nodes)
    mean = np.mean(s)
    var = np.var(s)
    print(nodes)
    print('mean: ', mean)
    print('var: ', var)
```

2. Exercise 2

```
import random
import math
import numpy as np

n = 100
nodes_list = [5, 10, 20, 30, 40, 50, 60, 70, 80, 100]

# 随机获取一个点，判断是否在函数积分中
def getOneNode():
    x = random.random()
    y = random.random()
    return y <= math.pow(x, 3)

# 蒙特卡洛法计算积分
def calculate(nodes):
    t = 0
    for _ in range(nodes):
        if getOneNode():
            t += 1
    return t/nodes
```

```python
# 计算均值和方差
for nodes in nodes_list:
    s = np.empty(n)
    for i in range(n):
        s[i] = calculate(nodes)
    mean = np.mean(s)
    var = np.var(s)
    print(nodes)
    print('mean: ', mean)
    print('var: ', var)
```

3. Exercise 3

```python
import random
import math
import numpy as np

n = 100
nodes_list = [10, 20, 30, 40, 50, 60, 70, 80, 100, 200, 500]

# 随机获取一个点，计算期望积分
def getOneNode():
    x = random.uniform(2, 4)
    y = random.uniform(-1, 1)
    a = math.pow(y, 2)*math.exp(-math.pow(y, 2))
    b = math.pow(x, 4)*math.exp(-math.pow(x, 2))
    c = x*math.exp(-math.pow(x, 2))
    z = (a+b)/c
    return z*4

# 蒙特卡洛法计算积分
def calculate(nodes):
    t = 0
    tmp = 0
    for _ in range(nodes):
        t = getOneNode()
        tmp += t
    return tmp/nodes

# 计算均值和方差
for nodes in nodes_list:
    s = np.empty(n)
    for i in range(n):
        s[i] = calculate(nodes)
    mean = np.mean(s)
```

```
    var = np.var(s)
    print(nodes)
    print('mean: ', mean)
    print('var: ', var)
```

4. Exercise 4

```
import random
import numpy as np

n = 100
nodes_list = [20, 200, 2000, 20000]

def calculate(num):
    p = 0
    for _ in range(num):
        s = np.zeros((7,7))
        x = 0
        y = 0
        while True:
            if x==6 and y==6:
                p += 1
                break
            s[x][y] += 1
            t = []
            if x-1>=0 and (s[x-1][y]<1 or s[x-1][y]<2 and x-1==3 and y==3): # 向左
                t.append(0)
            if x+1<=6 and (s[x+1][y]<1 or s[x+1][y]<2 and x+1==3 and y==3): # 向右
                t.append(1)
            if y-1>=0 and (s[x][y-1]<1 or s[x][y-1]<2 and y-1==3 and x==3): # 向下
                t.append(2)
            if y+1<=6 and (s[x][y+1]<1 or s[x][y+1]<2 and y+1==3 and x==3): # 向上
                t.append(3)
            if len(t)==0:
                break
            n = t[random.randint(0, len(t)-1)]
            if n == 0:
                x -= 1
            if n == 1:
                x += 1
            if n == 2:
                y -= 1
            if n == 3:
                y += 1
    return p/num
```

```python
# 计算均值和方差
for nodes in nodes_list:
    s = np.empty(n)
    for i in range(n):
        s[i] = calculate(nodes)
    mean = np.mean(s)
    var = np.var(s)
    print(nodes)
    print('mean: ', mean)
    print('var: ', var)
```

5. Exercise 5

A 和 BC 各分一半时间通过

```python
import random
import numpy as np

n = 100
nodes_list = [10, 20, 30, 40, 50, 60, 70, 80, 100, 200]

# 随机采样A
def getA():
    a = random.random()
    return a<=0.85

# 随机采样BC
def getBC():
    b = random.random()
    c = random.random()
    return b<0.95 and c<0.90

# 蒙特卡洛法计算概率
def calculate(nodes):
    t = 0
    for _ in range(nodes//2):
        if getA():
            t += 1
    for _ in range(nodes//2):
        if getBC():
            t += 1
    return t/nodes

# 计算均值和方差
for nodes in nodes_list:
```

```
    s = np.empty(n)
    for i in range(n):
        s[i] = calculate(nodes)
    mean = np.mean(s)
    var = np.var(s)
    print(nodes)
    print('mean: ', mean)
    print('var: ', var)
```

A 和 BC 只要有一个通过就通过

```
import random
import numpy as np

n = 100
nodes_list = [5, 10, 20, 30, 40, 50, 60, 70, 80, 100, 200, 500, 1000, 5000]

# 随机采样一次，判断是否可靠
def getOne():
    a = random.random()
    b = random.random()
    c = random.random()
    return a<=0.85 or b<0.95 and c<0.90

# 蒙特卡洛法计算概率
def calculate(nodes):
    t = 0
    for _ in range(nodes):
        if getOne():
            t += 1
    return t/nodes

# 计算均值和方差
for nodes in nodes_list:
    s = np.empty(n)
    for i in range(n):
        s[i] = calculate(nodes)
    mean = np.mean(s)
    var = np.var(s)
    print(nodes)
    print('mean: ', mean)
    print('var: ', var)
```