HOMEWORK 1: Exercises for Monte Carlo Methods

March 5, 2019

16340305 郑先淇

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.

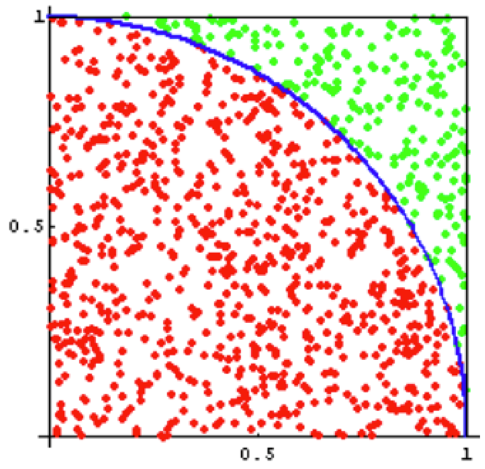


Figure 1 蒙特卡洛方法求解 pi

Solution

Analysis：我使用 python3.7 + pycharm 求解这个问题。首先使用 random 函数随机生成单位正方形之间的点，记录其分布情况，再求出 1/4 圆内的点与单位正方形内点的比值，将该值乘以 4 即为 pi 值，重复上述过程 20 次，利用 numpy.mean 函数和 numpy.var 函数求出均值 mean 和方差 variance。改变投点个数，记录对应的 mean 和 variance，用 PrettyTable 绘制结果表格。

Code：

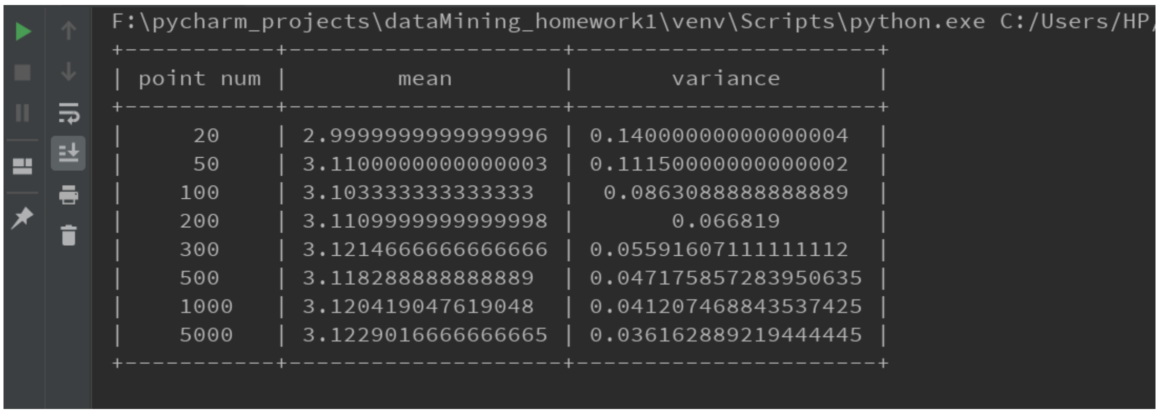
```python
import random
import numpy as np
from prettytable import PrettyTable


def getPi():
    arr = []
    table = PrettyTable(["point num", "mean", "variance"])
    time = [20, 50, 100, 200, 300, 500, 1000, 5000]
    for t in time:
        # print(t)
        for j in range(20):
            cnt = 0
            for i in range(t):
                x = random.uniform(0, 1)  # 从一个均匀分布中随机采样，区间为左闭右开
                y = random.uniform(0, 1)

                if (x * x + y * y) < 1:
                    cnt += 1
                    # 点分布在圆内的数量,因为正方形的面积为 1，所以这个同时也是圆内的点与正方形内点的比值
            vpi = 4.0 * (cnt / t)  # 4 * pi/4 = pi
            arr.append(vpi)
        mean = np.mean(arr)
        variance = np.var(arr)
        table.add_row([t, mean, variance])
    print(table)


getPi()
```

Result:



```
F:\pycharm_projects\dataMining_homework1\venv\Scripts\python.exe C:/Users/HP/
+-----------+--------------------+----------------------+
| point num |        mean        |       variance       |
+-----------+--------------------+----------------------+
|     20    | 2.9999999999999996 | 0.14000000000000004  |
|     50    | 3.1100000000000003 | 0.11150000000000002  |
|    100    | 3.103333333333333  |  0.08630888888888889 |
|    200    | 3.1109999999999998 |       0.066819       |
|    300    | 3.1214666666666666 | 0.05591607111111112  |
|    500    | 3.118288888888889  | 0.047175857283950635 |
|    1000   | 3.120419047619048  | 0.041207468843537425 |
|    5000   | 3.1229016666666665 | 0.036162889219444445 |
+-----------+--------------------+----------------------+
```

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: $\int_{x=0}^{1} x^3 = \frac{1}{4}$. 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 20 times, and report the mean and variance of the integrate in a table.

Solution

Analysis：使用蒙特卡洛方法，x 应服从均匀分布，求解这个积分，可以通过在[0, 1]之间随机选取采样点，使用以下公式近似求解积分值：

$$\bar{I} = (b-a)\frac{1}{N}\sum_{i=0}^{N-1} f(X_i) = \frac{1}{N}\sum_{i=0}^{N-1}\frac{f(X_i)}{\frac{1}{b-a}}$$

程序编写思路为，首先使用 random 函数随机选取 x，求出相应的 y 值，然后根据所有采样点所取得的 y 值，根据以上的公式，计算近似积分值 y。重复上述过程 100 从，求解均值 mean 和方差 variance。最后根据题意设置不同的采样点数，使用 prettytable 函数画出所有结果即可。

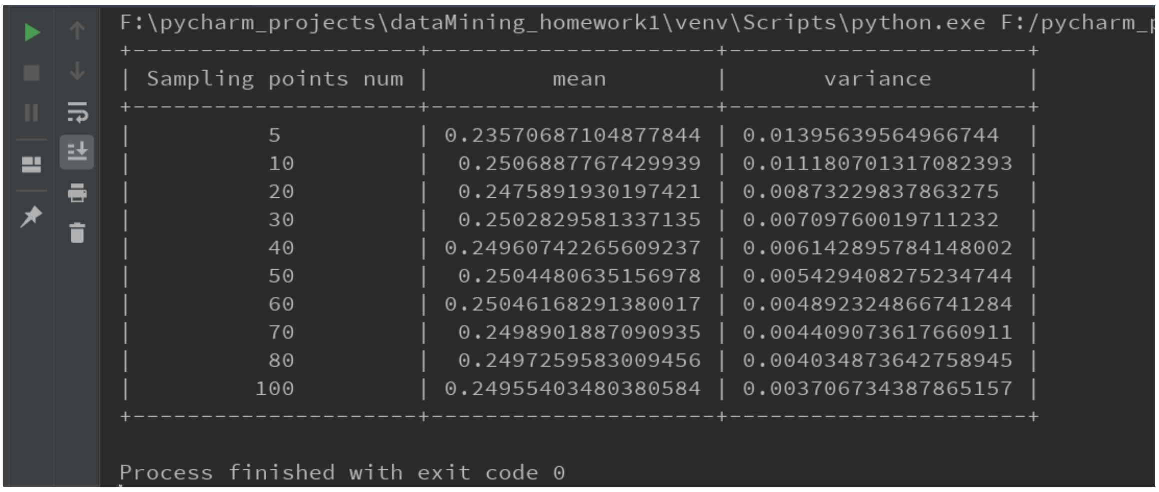
Code:

```python
import random
import numpy as np
from prettytable import PrettyTable


def calculate_integral():
    arr = []
    table = PrettyTable(["Sampling points num", "mean", "variance"])
    time = [5, 10, 20, 30, 40, 50, 60, 70, 80, 100]
    for t in time:
        for j in range(100):  # 重复蒙特卡洛方法 100 次
            sum_y = 0
            for i in range(t):
                x = random.uniform(0, 1)  # x 服从均匀分布，在 0-1 之间随机取值
                y = x*x*x
                sum_y += y
            Y = sum_y / t  # 求出单次的积分值
            arr.append(Y)
        mean = np.mean(arr)  # 重复 100 次求均值
        variance = np.var(arr) # 重复 100 次求方差
        table.add_row([t, mean, variance])
    print(table)


calculate_integral()
```

Result：



```
F:\pycharm_projects\dataMining_homework1\venv\Scripts\python.exe F:/pycharm_
+---------------------+---------------------+-----------------------+
| Sampling points num |         mean        |        variance       |
+---------------------+---------------------+-----------------------+
|          5          | 0.23570687104877844 |  0.013956395649667 44 |
|          10         |  0.2506887767429939 | 0.011180701317082393 |
|          20         |  0.2475891930197421 | 0.00873229837863275  |
|          30         |  0.2502829581337135 | 0.00709760019711232  |
|          40         | 0.24960742265609237 | 0.006142895784148002 |
|          50         |  0.2504480635156978 | 0.005429408275234744 |
|          60         | 0.25046168291380017 | 0.004892324866741284 |
|          70         |  0.2498901887090935 | 0.004409073617660911 |
|          80         |  0.2497259583009456 | 0.004034873642758945 |
|         100         | 0.24955403480380584 | 0.003706734387865157 |
+---------------------+---------------------+-----------------------+


Process finished with exit code 0
```

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.

Solution

Analysis: 求解这个积分，我们采取和 exercise 2 类似的做法，我们把二重积分近似为随机数函数的期望。二重积分可以理解求一个不规则三维物体的体积，我们任意取定 x 和 y 值，我们可以得到 f(x,y)的值，该三维不规则物体的体积可近似为以((4-2) * (1+1))为底，f(x,y)为高的规则物体的体积。增加采样点的个数，我们可以得到

$$\mathrm{E(x,y)} = \frac{1}{4} * \frac{1}{\mathrm{N}}\sum f(xi, yi)$$

程序编写思路为：首先使用 random 函数随机选取 x,y 值，然后套公式计算出 f(x,y)。根据采样点的个数求其所有采样点测期望作为二重积分的近似值。然后重复蒙特卡洛方法 100 次 ，求解均值 mean 和方差 variance。最后是设置不同的采样点，重复上述步骤。

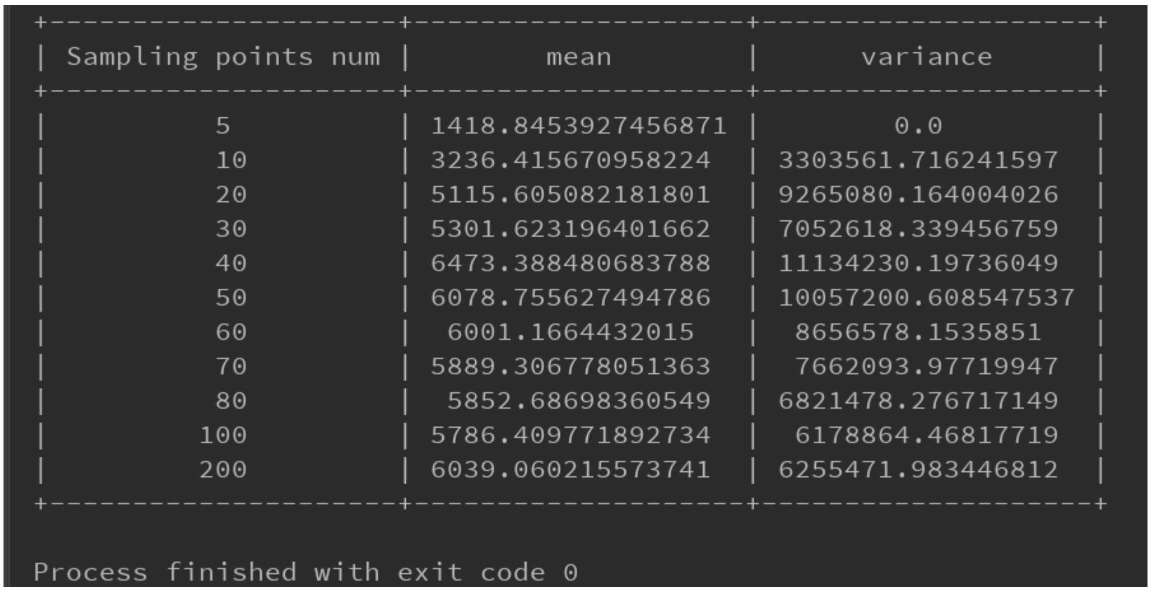
Code：

```python
import random
import numpy as np
import math
from prettytable import PrettyTable


def calculate_complicated_integral():
    arr = []
    table = PrettyTable(["Sampling points num", "mean", "variance"])
    time = [5, 10, 20, 30, 40, 50, 60, 70, 80, 100, 200]
    for t in time:
        for j in range(1):  # 重复蒙特卡洛方法 100 次
            sum = 0
            for i in range(t):
                x = random.uniform(2, 4)  # 在 2-4 之间随机取值
                y = random.uniform(-1, 1)  # 在-1 到 1 之间随机取值
                f_xy = ((y*y) * math.exp(-y*y) + (x*x*x*x) * math.exp(-x*x))
 / (x * math.exp(-x*x))
                # 直接套公式
                sum += f_xy
            Y = 1/4 * (sum / t)  # 以随机数函数的期望作为近似的积分值
            arr.append(Y)
        mean = np.mean(arr)  # 重复 100 次求均值
        variance = np.var(arr)  # 重复 100 次求方差
        table.add_row([t, mean, variance])
    print(table)


calculate_complicated_integral()
```

Result：



```
+--------------------+-------------------+-------------------+
| Sampling points num |        mean       |      variance     |
+--------------------+-------------------+-------------------+
|          5         | 1418.8453927456871 |        0.0        |
|         10         | 3236.415670958224 | 3303561.716241597 |
|         20         | 5115.605082181801 | 9265080.164004026 |
|         30         | 5301.623196401662 | 7052618.339456759 |
|         40         | 6473.388480683788 | 11134230.19736049 |
|         50         | 6078.755627494786 | 10057200.608547537 |
|         60         |  6001.1664432015  |  8656578.1535851  |
|         70         | 5889.306778051363 |  7662093.97719947 |
|         80         |  5852.68698360549 | 6821478.276717149 |
|        100         | 5786.409771892734 |  6178864.46817719 |
|        200         | 6039.060215573741 | 6255471.983446812 |
+--------------------+-------------------+-------------------+

Process finished with exit code 0
```