计算机模拟 Homework 5

汪奕晨 3180105843

数学科学学院数学与应用数学专业

1 问题描述

我们知道三维格点的布朗运动是非常返的。

本文通过数值模拟计算返回概率。空间为 (x_i, y_i, z_i)，其中 x_i, y_i, z_i 均为整数。初始位置为原点。每一步布朗运动，都等概地向上、下、左、右、前和后移动一格。

2 设计思路

2.1 实验设计思路

同时考虑 S 个初始位于原点的点，独立抽取在三维空间中游走，各自进行 N 步，同时记录是否曾回到原点，可以得到 N 步后曾回到原点的比例 p。

在实现的过程中，还可以记录每一步时的比例 $p(N)$，绘制 $p(N) - N$ 的图像。在一个实例 (即 S 个点) 中，$p(N)$ 关于 N 是严格单调递增的，这可能会使得结果存在偶然因素。故我们设置 M 个实例后取 p 的均值，来判断 $p(N)$ 关于 N 的收敛情况

2.2 代码设计思路

这里给出一些代码设计的技巧

快速一次性抽取给定步数的运动结果。

可以使用 np.random.choice 快速抽取，使用逻辑索引对需要对 x, y, z 坐标进行 $+1$ 或 -1 操作处向量化处理，而后使用 np.cumsum 方法快速获得累计值，即累计运动的偏移量，再加上当前的位置，既可以得到每一步运动后所处的位置。

```
walk = np.random.choice(np.arange(6), steps, True)
new_path = np.zeros((3,steps))
new_path[0][walk==0] += 1
new_path[0][walk==1] -= 1
new_path[1][walk==2] += 1
new_path[1][walk==3] -= 1
```

```
  new_path[2][walk==4] += 1
  new_path[2][walk==5] -= 1
  new_path = np.cumsum(new_path, axis=1) + self.position.reshape(3,1)
```

快速生成 latex 表格，当不同参数的运行结果存在 pd.DataFrame 中时，可以使用 df.to_latex 快速生成 latex 代码

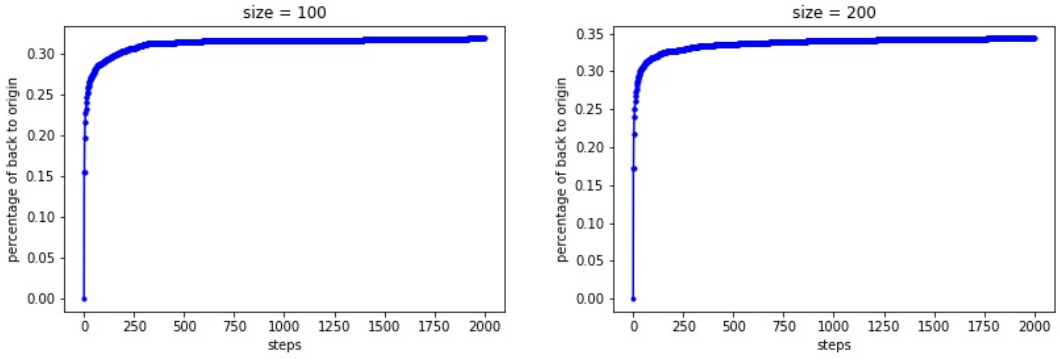
```
with open('df_latex.txt','w') as tf:
  tf.write(df.to_latex())
```

3 模拟结果与分析

固定 $M = 20$, 对不同规模的 S, N 求解，得到的结果如 Table (3), 不同参数下 $p(N)$ 随 N 的变化图如 Figure (1)

S \ N	400	800	1200	1600	2000
50	0.307000	0.311000	0.313000	0.313000	0.313000
100	0.312500	0.315000	0.315500	0.317500	0.318000
150	0.332667	0.336667	0.338333	0.339667	0.340667
200	0.334500	0.339250	0.341500	0.342750	0.343500

Table 1: Results Table $M = 10$



(a) $S = 100, N = 2000, M = 20$ $p - N$ Graph (b) $S = 200, N = 2000, M = 20$ $p - N$ Graph

Figure 1: Performance of Simulation

可以看到，随着 N 增大，p 会收敛到 0.34 左右的数值, 且 S 越大，收敛地越快。

附录

代码

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import time
import pandas as pd

class rand_move_3d():
  def __init__(self):
      self.position = np.array([0,0,0])
      self.path = self.position.reshape(3,1)
      self.step = 0
      self.back = np.array([])

  def rand_walk(self, steps):
      walk = np.random.choice(np.arange(6), steps, True)
      new_path = np.zeros((3,steps))
      new_path[0][walk==0] += 1
      new_path[0][walk==1] -= 1
      new_path[1][walk==2] += 1
      new_path[1][walk==3] -= 1
      new_path[2][walk==4] += 1
      new_path[2][walk==5] -= 1
      new_path = np.cumsum(new_path, axis=1) + self.position.reshape(3,1)

      self.path = np.hstack((self.path, new_path))
      self.position = self.path[:,-1]
      self.step += steps
      new_back = np.zeros(steps)

      if self.back.shape[0] > 0 and self.back[-1] == 1:
          new_back = 1
      else:
          origin = np.array([0,0,0])
          for i in range(self.step):
              if ( self.path[:, i+1] == origin ).all():
                  new_back[i:] = 1
                  break
```

```
38
39          self.back = np.hstack((self.back, new_back))
40
41
42      def plot_path(self, savepath=None):
43          fig = plt.figure()
44          ax = Axes3D(fig)
45
46          ax.plot(self.path[0], self.path[1], self.path[2], 'b.-')
47          if savepath:
48              fig.savefig(savepath)
49          fig.show()
50
51
52  class batch_walk():
53      def __init__(self, size):
54          self.size = size
55          self.persons = [rand_move_3d() for i in range(size)] if size else None
56
57      def test(self, N):
58          '''
59          N: the number of steps
60          '''
61          self.n = N
62          self.p = np.zeros(N)
63          count = np.zeros(N)
64          start_time = time.time()
65          for i in range(self.size):
66              self.persons[i].rand_walk(N)
67              count += self.persons[i].back
68
69          self.p = count/self.size
70          end_time = time.time()
71          # print('run time = {}'.format(end_time - start_time))
72          # print('final percentage = {:.3f}'.format(self.p[-1]))
73          return self.p
74
75
76      def plot_p(self, p=None, savepath=None):
77          plt.figure()
78          x = np.arange(self.n)
```

```
        if p is None:
            p = self.p
        plt.plot(x, p, 'b.-')
        plt.xlabel('steps')
        plt.ylabel('percentage of back to origin')
        plt.title('size = {:}'.format(self.size))
        if savepath:
            plt.savefig(savepath)
        plt.show()
```

Listing 1: 随机游走类与批量游走类

```
def para_test(size, step, N):
  p = np.zeros(step)
  for _ in range(N):
      B = batch_walk(size)
      p += B.test(step)
  p /= N
  B.plot_p(p=p, savepath='rand_move_{:}_{:}_{:}.jpg'.format(size,step,N))
  return p

def paras_test(size_list, step_list, N):
  df = pd.DataFrame(np.zeros((len(size_list), len(step_list))))
  for i,size in enumerate(size_list):
      # for N in N_list:
      p = para_test(size, step_list[-1], N)
      df.loc[i,:] = p[step_list-1]
      print('size = {} is done'.format(size))
  df.index = size_list
  df.columns = step_list
  return df

size_list  = np.array([50*i for i in range(1, 5)])
step_list = np.array([400 * i for i in range(1, 6)])
N = 10

df = paras_test(size_list, step_list, N)
with open('df_latex.txt','w') as tf:
  tf.write(df.to_latex())
```

Listing 2: 测试模块