

陕西科技大学



随机漫步实验报告

学 生： _____

学 号： _____

学 院： 电子信息与人工智能学院

专 业： 网络工程

指导教师： 丁磊

2022 年 5 月 2 日

实验三 随机漫步

班级： 网络 201 学号： _____

实验预习报告

一、实验目的

- 1、掌握 Numpy 及其随机数模块
- 2、使用 Numpy 进行简单的数据分析
- 3、理解数据可视化，掌握使用 matplotlib 绘图的方法

二、实验要求

- 1、完成随机漫步的模拟，要求如下：

在二维的平面直角坐标系下，向任意方向随机漫步，每一步长度为 0.5 米，总共走 1000 步，使用 numpy 模拟这一过程

- 2、使用 Matplotlib 库来绘制这一次漫步过程的运动轨迹。
- 3、模拟多次随机漫步，并进行统计：
 - (1)、5000 次随机漫步中，距离原点大于 20 米的次数
 - (2)、平均多少次漫步能够运动到距离原点 20 米的地方

三、实验原理

一维随机漫步的基本原理为，使用 numpy 的随机数模块随机生成一定数量的随机数，这样的随机数只需要有两个取值，它们代表着前进或者后退一步的距离。对于多次一维随机漫步，只需要将它们按行逐步求和，就可以知道有哪些地方距离原点超过了指定长度，再对这些数据进行求和、求平均值即可得到需要的结果。

对于本次实验的二维数组，基本原理不变，唯独在随机数的生成上使用了一个变量，也就是运动的方向，使用弧度制表示，接下来，在 x 轴和 y 轴的运动距离都可以通过三角函数求出来

$$x_i = \rho \cos \theta_i, y_i = \rho \sin \theta_i$$

其中, θ_i 表示某一次运动的角度, x_i, y_i 表示在 x 轴和 y 轴上的运动分量。进而可以对元素进行按行累加, 得到每一次运动过后距离原点距离在每一个轴上的分量。最终, 运动的位置到原点的距离就能很方便的通过两点间距离公式求出来

$$s_i = \sqrt{x_i^2 + y_i^2}$$

其中, s_i 表示某一次运动时到原点的距离, x_i, y_i 表示该点分别距离 y 轴和 x 轴的距离。

在使用 Matplotlib 进行可视化绘图的时候, 只需要将 x、y 轴上的数据传入相应函数即可完成绘图。

四、实验预习内容

1、什么是 Numpy 库?

Numpy 库是一个用于科学计算的 Python 扩展库, 它提供了多维数组对象和各种操作数组的函数, 例如数学、逻辑、形状变换、排序、选择、线性代数、傅里叶变换、统计运算和随机模拟等。numpy 库的核心是 ndarray 对象, 它是一个高效的存储和处理多维数组的容器, 支持矢量化和广播等特性。

2、Numpy 的常用方法有哪些?

np.array(): 创建一个数组对象; np.arange(): 创建一个等差数列; np.sum(): 求和数组的元素; np.mean(): 求平均值数组的元素; np.random.randint(): 生成随机整数; np.cumsum() 累加求和, 等。

3、什么是 Matplotlib 库?

matplotlib 库是一个用于绘制图形的 Python 扩展库, 它可以生成各种静态、动态或交互式的图表, 例如折线图、柱状图、饼图、散点图、直方图等。matplotlib 库的核心是 pyplot 模块, 它提供了类似于 MATLAB 的绘图接口, 可以方便地创建和修改图形。

4、Matplotlib 库有哪些常用的方法?

plt.plot(): 绘制折线图或者点图; plt.scatter(): 绘制散点图; plt.title(): 设置图形的标题; plt.xlabel(): 设置 x 轴的标签; plt.ylabel(): 设置 y 轴的标签; plt.legend(): 设置图例; plt.show(): 显示图形, 等。

实验报告

一、实验目的

- 1、掌握 Numpy 及其随机数模块
- 2、使用 Numpy 进行简单的数据分析
- 3、理解数据可视化，掌握使用 matplotlib 绘图的方法

二、实验要求

- 1、完成随机漫步的模拟，要求如下：

在二维的平面直角坐标系下，向任意方向随机漫步，每一步长度为 0.5 米，总共走 1000 步，使用 numpy 模拟这一过程

- 2、使用 Matplotlib 库来绘制这一次漫步过程的运动轨迹。
- 3、模拟多次随机漫步，并进行统计：
 - (1)、5000 次随机漫步中，距离原点大于 20 米的次数
 - (2)、平均多少次漫步能够运动到距离原点 20 米的地方

三、实验原理

一维随机漫步的基本原理为，使用 numpy 的随机数模块随机生成一定数量的随机数，这样的随机数只需要有两个取值，它们代表着前进或者后退一步的距离。对于多次一维随机漫步，只需要将它们按行逐步求和，就可以知道有哪些地方距离原点超过了指定长度，再对这些数据进行求和、求平均值即可得到需要的结果。

对于本次实验的二维数组，基本原理不变，唯独在随机数的生成上使用了一个变量，也就是运动的方向，使用弧度制表示，接下来，在 x 轴和 y 轴的运动距离都可以通过三角函数求出来

$$x_i = \rho \cos \theta_i, \quad y_i = \rho \sin \theta_i$$

其中， θ_i 表示某一次运动的角度， x_i, y_i 表示在 x 轴和 y 轴上的运动分量。进而可以对元素进行按行累加，得到每一次运动过后距离原点距离在每一个轴上的分量。最终，运动的位置到原点的距离就能很方便的通过两点间距离公式求出来

$$s_i = \sqrt{x_i^2 + y_i^2}$$

其中, s_i 表示某一次运动时到原点的距离, x_i, y_i 表示该点分别距离 y 轴和 x 轴的距离。

在使用 Matplotlib 进行可视化绘图的时候, 只需要将 x、y 轴上的数据传入相应函数即可完成绘图。

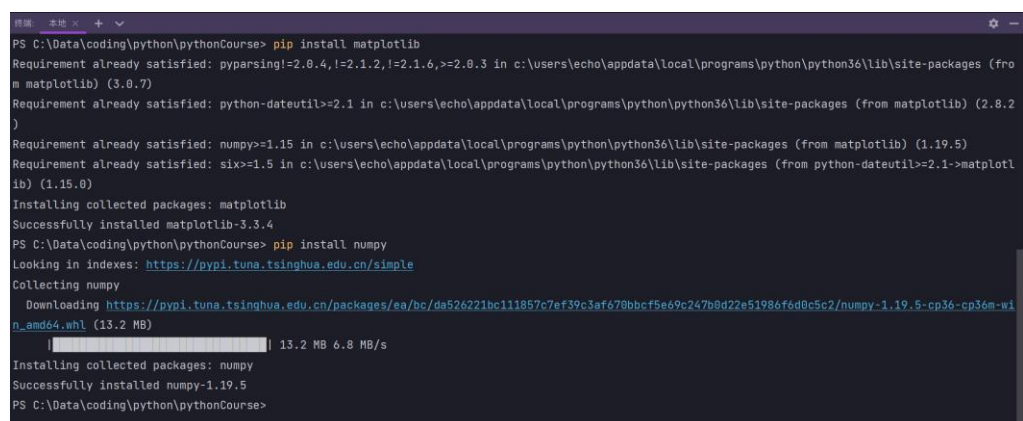
四、实验内容

1、安装所必须的库

使用命令安装 Numpy 和 Matplotlib 库:

pip install numpy

pip install matplotlib



```
PS C:\Data\coding\python\pythonCourse> pip install matplotlib
Requirement already satisfied: pyparsing<=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\echo\appdata\local\programs\python\python36\lib\site-packages (from m
matplotlib) (3.0.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\echo\appdata\local\programs\python\python36\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: numpy>=1.15 in c:\users\echo\appdata\local\programs\python\python36\lib\site-packages (from matplotlib) (1.19.5)
Requirement already satisfied: six>=1.5 in c:\users\echo\appdata\local\programs\python\python36\lib\site-packages (from python-dateutil>=2.1->matplotl
ib) (1.15.0)
Installing collected packages: matplotlib
Successfully installed matplotlib-3.3.4
PS C:\Data\coding\python\pythonCourse> pip install numpy
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting numpy
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/ea/bc/da526221bc11857c7ef39c3af678bbcf5e69c247b8d22e51986f6d8c5c2/numpy-1.19.5-cp36-wi
n_amd64.whl (13.2 MB)
    |#####| 13.2 MB 6.8 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.19.5
PS C:\Data\coding\python\pythonCourse>
```

2、模拟一次的随机漫步

首先, 使用随机数模块生成 1000 个 0 到 359 之间的随机数, 再通过除以 180 乘以 2π 的方式转化为弧度制, 将其作为运动的方向, 接下来, 通过三角函数求出来在 x 轴和 y 轴的运动距离

$$x_i = \rho \cos \theta_i, \quad y_i = \rho \sin \theta_i$$

接下来, 对元素进行累加, 得到每一次运动过后距离原点距离在每一个轴上的分量。最终, 运动的位置到原点的距离就能很方便的通过两点间距离公式求出来

$$s_i = \sqrt{x_i^2 + y_i^2}$$

最后, 将 x、y 轴上的数据传入 plot 函数即可完成绘图。

为了能够有效的组织代码，体现面向对象的编程思维，可以使用一个类把相关的属性组织起来，这样，初始化和调用的时候更加规范和整齐。

最终代码如下：

```
import numpy as np
import matplotlib.pyplot as plt

class Walker:
    def __init__(self, step_length):
        self.step_length = step_length
        self.theta = []
        self.x = [0]
        self.y = [0]
        self.distance = [0]

    def walk(self, steps_num):
        self.theta = np.random.randint(0, 360, (steps_num,)) / 180 * np.pi
        self.x = (np.cos(self.theta) * self.step_length).cumsum(axis=0)
        self.y = (np.sin(self.theta) * self.step_length).cumsum(axis=0)
        self.distance = np.sqrt(self.x ** 2 + self.y ** 2)

    def draw(self):
        plt.plot(self.x, self.y, color='ff0000')
        plt.xlabel('x')
        plt.ylabel('y')
        plt.title('random walk')
        plt.show()

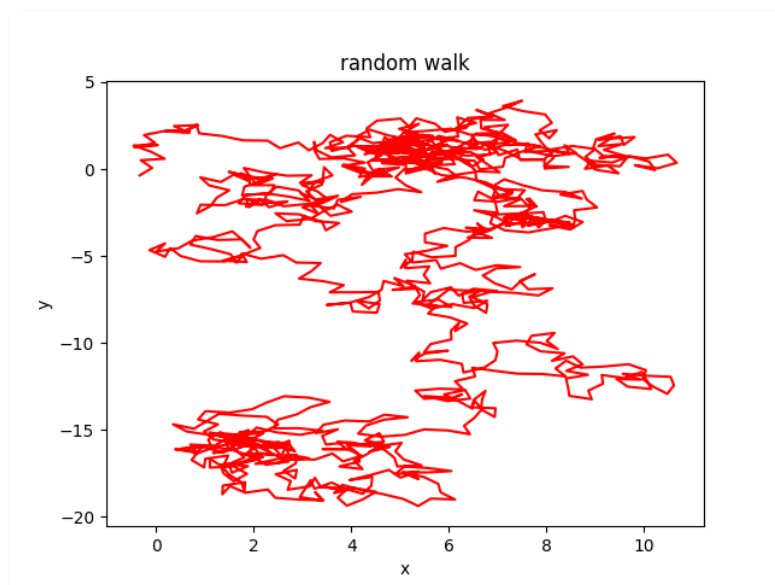
if __name__ == '__main__':
    k = Walker(0.5)
    k.walk(1000)
    k.draw()
```

其中，在初始化类的时候，传入了 0.5 作为一次运动的距离，并且在初始化函数当中将对象的初始化位置定在了原点。

当调用 walk(1000)方法后，首先使用 numpy 的随机数模块生成了 1000 个方向，再由这 1000 个方向计算出了每一次运动时产生在各个方向上的距离，紧接着对他们求累计和，得到了每一次运动后各个方向上到原点的距离，最后使用两点间距离公式得到了原点到每一次运动的点的距离。

调用 draw()方法后，首先使用了 Matplotlib 的 plot()函数绘制了 x、y 上的数据，并且设置了线条为红色，下来又依次的设置了 x、y 轴的标签以及标题，然后开始绘图。

最终绘图结果如下：



随机漫步 1000 次轨迹图

3、模拟多次随机漫步

模拟多次随机漫步基本思想和单次随机漫步相似，只需要把一行随机角度扩展为多行随机角度，剩下的 x 轴、y 轴位移以及到原点距离都可以简单的根据公式计算得到。

多次随机漫步代码如下：

```
import numpy as np

if __name__ == '__main__':
    step = 0.5
    number_of_steps = 1000
    number_of_strolls = 5000

    theta = np.deg2rad(np.random.randint(0, 360, (number_of_strolls, number_of_steps)))
    x = (step * np.cos(theta)).cumsum(1)
    y = (step * np.sin(theta)).cumsum(1)
    dis = np.hypot(x, y)

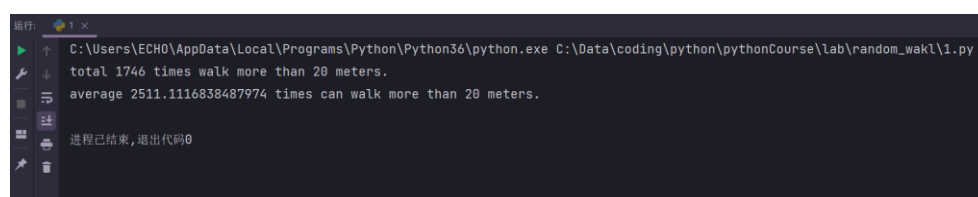
    dis_greater_than_20 = np.where(dis >= 20, 1, 0)
    print(f'total {np.any(dis_greater_than_20 == 1, axis=1).sum()} times walk more than 20 meters.')
    print(f'average {np.mean(np.where(dis_greater_than_20.argmax(axis=1) > 0))} times can walk more than 20 meters.')
```

在这里，并没有对多次随机漫步进行绘图，仅仅是统计了超过 20 米的次数以及到达二十米的平均步数。

首先，使用 `np.deg2rad` 函数将 0 到 360 度之间的随机整数转换为弧度值，表示每次漫步的每一步的方向。然后，使用 `numpy` 中自带的三角函数计算每一步在 x 轴和 y 轴上的位移，并乘以 `step` 得到实际的位移，用 `cumsum` 函数沿着第二

个维度 (axis=1) 累加位移, 得到每次漫步的每一步的 x 坐标和 y 坐标。接着, 使用 `np.hypot` 函数计算每次漫步的每一步距离原点 (0,0) 的距离, 并判断 `dis` 中哪些值大于等于 20。接下来, 沿着第二个维度判断每次漫步是否有任何一步超过了 20 米, 并用进行求和, 得到总共有多少次漫步超过了 20 米, 并输出。最后找出 `dis_greater_than_20` 中第一个为 1 的位置, 并找到返回索引值, 得到每次漫步需要多少步才能超过 20 米, 并用 `mean` 函数求平均值, 输出最终结果。

最终效果:



```
运行: C:\Users\ECHO\AppData\Local\Programs\Python\Python36\python.exe C:\Data\coding\python\pythonCourse\lab\random_walk\1.py
total 1746 times walk more than 20 meters.
average 2511.1116838487974 times can walk more than 20 meters.
进程已结束, 退出代码0
```

五、实验结论

1、Numpy 库可以用来进行大量的数据处理, 亦能用来生成随机数, 是一个全面的数学工具。

2、Matplotlib 库可以十分方便的进行图形绘制, 有利于直观的体现出数据的特征, 提高了数据的可读性。

3、Numpy 和 Matplotlib 库的结合使用已经成为了 python 数据分析当中十分重要的方法, 它们的搭配使用可以极大的提高 python 数据分析的效率以及为用户提供关于数据更加直观的感受。

4、数据分析的过程是一个计算机参与到数学的求解过程, 其核心在于处理数据的时候设计的计算方法, 如: 在随机漫步实验中, 需要使用到平面直角坐标系的距离公式, 如何使用 Numpy 中给定的方法便成为了关键。

5、随机漫步实验是一种经典的随机数处理模型, 不具有太多的复杂度, 有利于直观的体验和感受数据分析的过程。