Lab 2 Report

二维随机游走

王世炟

PB20151796

April 22, 2023

Part 1: 实验要求

本次实验要求使用 python 模拟二维随机游走过程，处理不确定性问题。
具体内容如下：
一个人站在原点，每一秒都会随机选择上下左右的一个方向迈出一步。

1. 经过 x 秒之后，他与原点的期望距离是多少?
2. 在这 x 秒内，他回到原点的期望次数是多少?
3. (选做) 他第一次回到原点的期望时间是多少?

基本要求：完成上述模拟过程的代码，并绘制 $10 \leq x \leq 10000$ 范围内到原点的期望距离曲线以及回到原点的期望次数曲线。
拓展要求（选做）：在基本要求的基础上，对设定场景进行改变，并绘制相应的运行曲线。

1. 有偏随机游走（选择不同方向时每一秒的步长不同）
2. 选择不同方向的概率不相同，具有倾向性
3. 选择的方向不局限于上下左右，可以是其他方向（如对角线）
4. 其他

Part 2: 实验环境

本次实验的实验环境为：

- Windows10 64 位
- IDE：VS Code

- python==3.9.5
- jupyter==1.0.0
- matplotlib==3.5.1
- numpy==1.23.0

Part 3: 实验内容

3.1 实验设计

基本要求的实现比较简单，只需要记录人的位置坐标 (x, y), 每一次移动前在 $[(1,0),(0,1),(-1,0),(0,-1)]$ 内随机选择一个方向进行移动即可。$x=0, y=0$ 是返回原点的条件。至于期望，我们只需要多次重复上述过程，取估计期望。

拓展要求的实现相对复杂一些。**首先对于要求一**，我们仍可以在上面的集合内随机选择一个方向，只不过在移动之前要乘上一个步长 d，d 是由区间 $[a, b]$ 随机产生的一个实数。**对于要求二**，python 提供了一些接口可以做到以不同的概率选择一个集合中的某个值。**对于要求三**，我们可以在每次移动之前随机选取一个 $[0, 2\pi]$ 中的一个角 α，然后将移动方向设置为 $(\sin\alpha, \cos\alpha)$, 就可以做到随机选取方向。

3.2 核心代码解释

导入库:

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

选择方向:

```
if directions:  # if directions is not None
                if directions == 'random':  # if directions is 'random'
                    # generate a random angle
                    angle = random.uniform(0, 2*math.pi)
                    # convert angle to direction
                    direction = (math.cos(angle), math.sin(angle))
                else:  # if directions is a list, choose a direction
                    from the list
                    direction = random.choices(directions, weight)[0]

```

```
                else: # if directions is None, choose a direction from [(1, 0), (-1, 0), (0, 1), (0, -1)]
                    direction = random.choices([(1, 0), (-1, 0), (0, 1), (0, -1)], weight)[0]
```

选择是否有偏：

```
    if bias: # if bias is not None, add bias to the direction
                    x, y = self.x[-1] + direction[0] * (bias[i-1] if i-1 < len(bias) else 1), self.y[-1] + direction[1] * (bias[i-1] if i-1 < len(bias) else 1)
                else: # if bias is None, do not add bias to the direction
                    x, y = self.x[-1] + direction[0], self.y[-1] + direction[1]
```

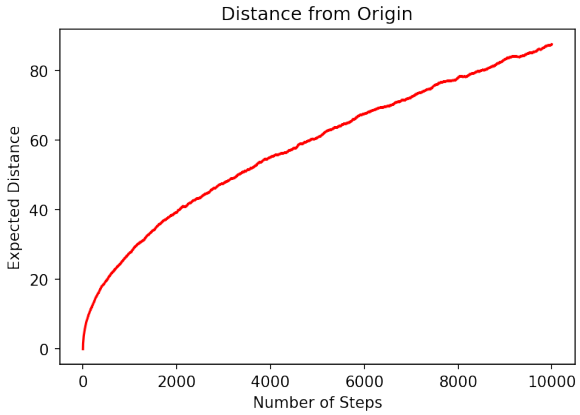
计算期望：

```
    def simulate(self, bias=None, directions=None, weight=None):
        k = self.num
        # Simulate num random walks
        for i in range(self.num):
            model = RandomWalk(self.steps)
            model.simulate(bias, directions, weight)
            self.distances += np.array(model.distances)
            self.returns += np.array(model.returns)
            if model.first_return != None:
                self.first_return += model.first_return
                k -= 1
        self.distances = self.distances / self.num
        self.returns = self.returns / self.num
        self.first_return = self.first_return / (self.num - k)
```

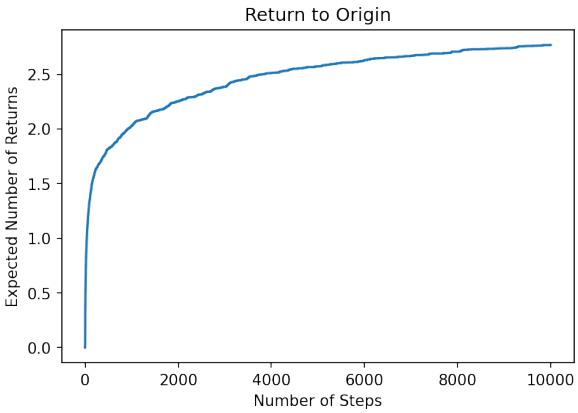
Part 4: 实验结果

4.1 基本要求

我们对 $10 \leq x \leq 10000$ 范围内的随机游走进行了 1000 次重复实现，并取数据的平均值以估计期望值。



(a) 期望距离曲线

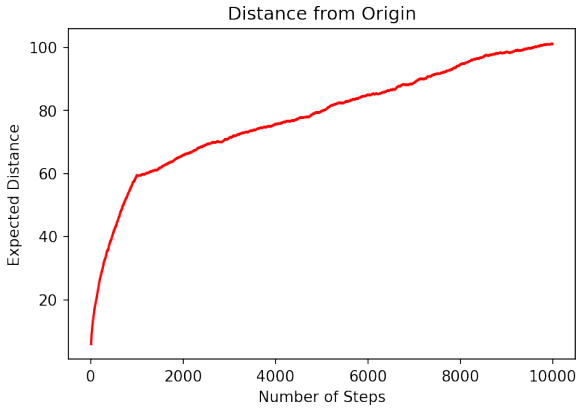


(b) 回到原点的期望次数曲线

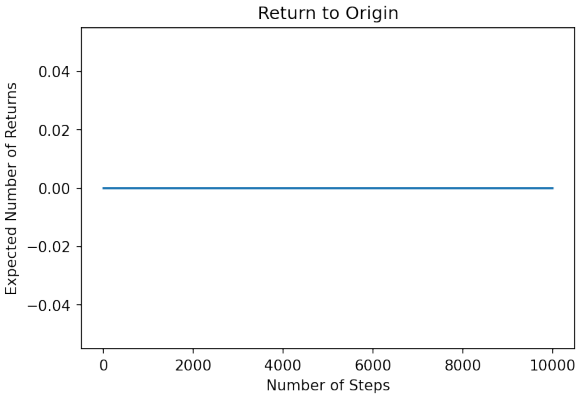
由上图可以观察猜测得到经过 x 秒之后，他与原点的期望距离为 $\sqrt{x}$。在这 x 秒内，他回到原点的期望次数为 3 **次**。
他第一次回到原点的期望时间: 264.379s

4.2 拓展要求

有偏随机游走:
每一步步长随机，平均值为 2:

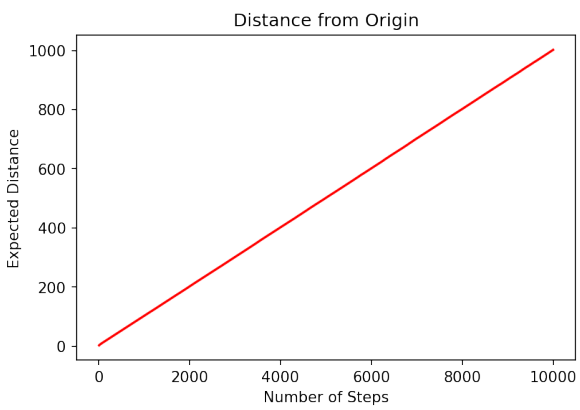


(a) 期望距离曲线

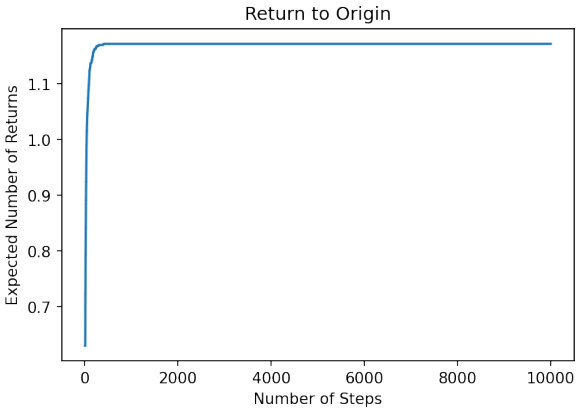


(b) 回到原点的期望次数曲线

从未返回原点。
以不同概率选择不同方向:
以不同概率选择不同方向，选择四个方向的概率分别为 0.3,0.2,0.25,0.25



(a) 期望距离曲线



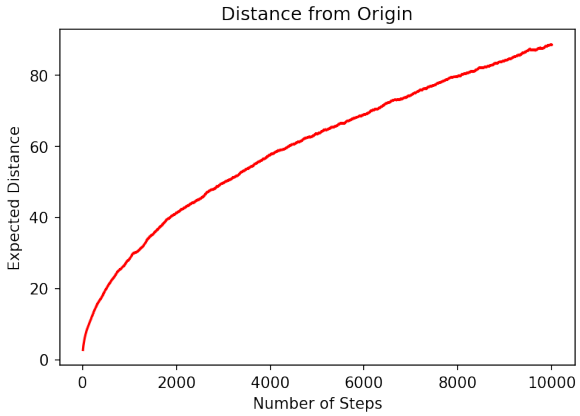
(b) 回到原点的期望次数曲线

由上图可以观察猜测得到经过 x 秒之后，他与原点的期望距离呈线性逐渐增大。在这 x 秒内，他回到原点的期望次数为 1.2 **次**。

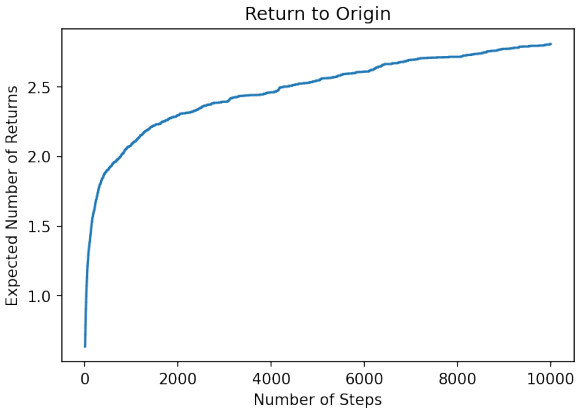
他第一次回到原点的期望时间: 6.647s

选择的方向不局限于上下左右:

首先是只有对角线方向:



(a) 期望距离曲线



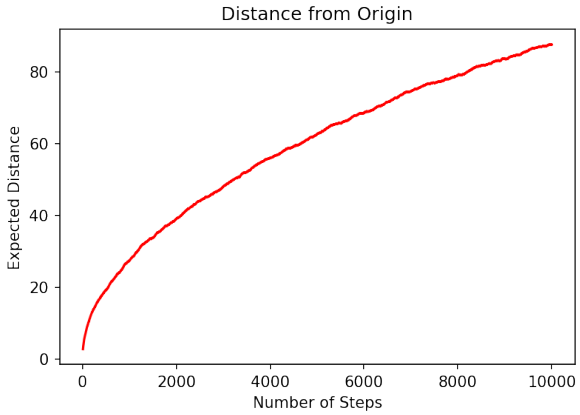
(b) 回到原点的期望次数曲线

仅有对角线方向与上下左右四个方向是相同的，只是旋转了坐标轴。

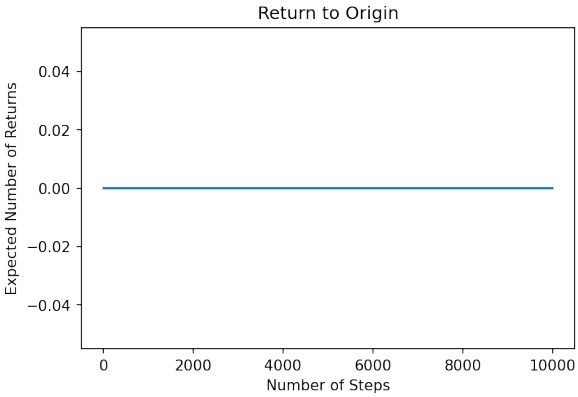
由上图可以观察猜测得到经过 x 秒之后，他与原点的期望距离为 $\sqrt{x}$。在这 x 秒内，他回到原点的期望次数为 3 **次**。

他第一次回到原点的期望时间: 245.575s

然后是随机方向:



(a) 期望距离曲线



(b) 回到原点的期望次数曲线

从未返回原点。

Part 5: 实验总结

通过本次实验，用 python 模拟了二维随机游走，并从实验数据中获得一般性结论。

1. 随着时间的增加，人与原点的期望距离也会增加，但增加的速度会逐渐减缓。
2. 回到原点的期望次数随着时间的增加而增加，但增加的速度会逐渐减缓。
3. 在有偏随机游走的情况下，人与原点的期望距离和回到原点的期望次数都会受到步长不同的影响，但总体趋势与随机游走相似。

总的来说，本次实验通过模拟随机游走过程，让我们更好地理解了不确定性问题，并且通过拓展要求，让我们更深入地了解了随机游走的不同变体。