Simulation Of Two Dimensional Random Walk using python

Jennifer Okereke

March 10th 2023

1 Introduction

The purpose of this my code is to simulate 2 Dimensional random walks, both fair and biased (I am going to explain what i meant my fair and biased walks later). The code defines the number of steps and walks, initializes arrays for storing position and displacement data, generates random numbers based on a given seed, and fills the arrays with coordinates determined by the categorical distribution. the code then calculates and plots the final positions of the random walks. finally, the code plots the fair and biased random walks seperately. At the end I will demonstrate the difference between my data and Raleigh distribution.

Random walks have many applications in Physics, Mathematics and other fields. The experimental/theoritical application of this code is that this code can be use to model the behavior of molecules in a gas where the molecules have a higher probability of moving in one direction than the other as simulated by the "Biased" Random walk.

Also, this code can be used to model the brownian motion of particles in a fluid. As I know that Brownian motion is caused by zig-zag pattern. The "Fair" random walk simulated in this my code can be use to model Brownian motion in a fluid where the particle has an equal chance of moving in any direction.

This background introduction has explained what the simulation is all about and what questions the code can answer.

This paper is organized as follows: Sec. 2 Explains the two different walks. Algorithm analysis Sec. 3, with an analysis of the outputs included in Sec. 4. Finally, conclusions are presented in Sec. 5.

2 What the code is all about

The given code simulates a two-dimensional random walk using the numpy, matplotlib, and Random modules in Python. The simulation consists of two parts: a "fair" random walk and a "biased" random walk. In the "fair" random walk, each step can be taken in one of six directions (up, down, left, right, forward, or backward), each with an equal probability. In the "biased" random walk, the probabilities of taking steps in each direction are determined by an exponential distribution (which means the probability is not equal that is why it is exponential and hence a biased random walk).

3 Code and Experimental Simulation

This simulation is separated into two separate codes. The first code is a random class which contains all of the code used to generate the distribution. The second code is the abalsyes code, which collects and reads our data, and displays it in a graph/figure.

This version of this code is described by a categorical distribution with equal probability because this best describe the model of the fair walk this is chosen because it is normally use to explain or describe the model of a walker or particle that can move with equal probabilities in any of the direction.

Again, this code also uses truncated exponential distribution. kindly read the statement below to know how i use the exponential distribution in this code.

This is a Python code for simulating 2D random walks. It simulates both a "fair" random walk, where each step has an equal probability of moving in any of the six directions (up, down, left, or right,backward or forward), and a "biased" random walk, where the probabilities of each direction are determined by an exponential distribution.

The code generates two sets of random walks, each consisting of 2400 walks of 400 steps each. For each walk, the x and y coordinates of each step are stored in arrays, and the final x and y coordinates and the total distance traveled are stored separately. The final positions are plotted in 2D space, with the fair random walk plotted and the biased also plotted.

The code also uses a custom Random class, which provides methods for generating random numbers with specific distributions. The TruncExp method is used to generate the probabilities for the biased random walk, which is a truncated exponential distribution with a mean of 1 and a lower bound of 0 and upper bound of 1. The Categorical method is used to select a direction for each step, based on the probabilities generated by TruncExp.

The probability density function (PDF) of the truncated exponential distribution can be represented as follows:

$$f(x;\lambda,a,b) = \{\ \lambda e^{-\lambda x} 1 - e^{-\lambda(b-a)}, for\ a \leq x \leq b,\ 0, otherwise, \tag{1}$$

where λ is the rate parameter of the exponential distribution, and a and b are the lower and upper bounds of the distribution.

The cumulative distribution function (CDF) of the truncated exponential distribution can be represented as follows:

$$F(x;\lambda,a,b) = \{\ 1 - e^{-\lambda(x-a)} 1 - e^{-\lambda(b-a)}, for\ a \leq x \leq b,\ 0, otherwise. \tag{2}$$

The mean and variance of the truncated exponential distribution can be calculated as follows: $\mathrm{mean} = \mathrm{e}^{-\lambda a} - e^{-\lambda b} \lambda(1 - e^{-\lambda(b-a)}),\ variance = e^{-2\lambda a} - 2e^{-\lambda(a+b)} + e^{-2\lambda b}\lambda^2(1 - e^{-\lambda(b-a)})$

Note that the mean of the truncated exponential distribution with lower bound a and upper bound b is given by $(e^{-\lambda a} - e^{-\lambda b})/\lambda$, which is equivalent to the mean of the standard exponential distribution multiplied by the ratio of the area under the PDF between a and b and the total area under the PDF.

As required in this project prompt. This code is defining a fair walk by randomly assigning one of six possible directions (up, down, left, right,forward and backward) to each step of the walk. The line val = random.Categorical(0.15,0.15,0.15,0.15,0.15,0.15) uses the random.Categorical function to randomly select one of the six directions with equal probability (0.15 for each direction). This means that the walker is equally likely to move in any of the six possible directions at each step, resulting in a fair (or unbiased) walk.

In this code, the Categorical function is also used to generate random numbers, but the probabilities of each possible outcome are now being determined by the TruncExp function which generates random numbers from a truncated exponential distribution.

This function TruncExp generates a truncated exponential random variable. It takes three parameters: beta is the rate parameter of the exponential distribution, bottom is the lower bound of the truncated distribution, and top is the upper bound of the truncated distribution.

The function first generates a random number a from an exponential distribution with rate parameter beta using the Exponential method. It then checks if the value of a falls within the range [bottom, top] using the inequality (bottom ¡= a ¡= top) == False. If the value of a falls outside of the range, the function generates another random number a until it falls within the range.

Once a value of a is generated that falls within the truncated range, the function returns the value. This ensures that the returned value is always within the truncated range [bottom, top].

The TruncExp function has three parameters: the mean of the exponential distribution (1. in this case), the lower bound of the truncation (0. in this case), and the upper bound of the truncation (1. in this case).

The output of the TruncExp function will be a positive number between 0 and 1. These numbers are then used as probabilities for each of the possible outcomes in the Categorical distribution.

From this code, we can observe that the probabilities are being determined by an exponential distribution, the resulting random walk is biased towards certain directions. Specifically, the probabilities of moving in one direction versus another will depend on the values generated by the TruncExp function. Therefore, the resulting walk is a biased random walk.

4 Analysis

Running this code will result to four figure which i will explain what each of the figures is depicting in this experiment as the output analaysis.

In this code, the data variable likely contains a one-dimensional array or a Pandas Series object with the data I want to analyze and compare to the

Rayleigh distribution. The code then fits a Rayleigh distribution to this data and generates a histogram of the data along with the corresponding probability density function (PDF) of the fitted distribution. This allows you to visually compare the data to the Rayleigh distribution.

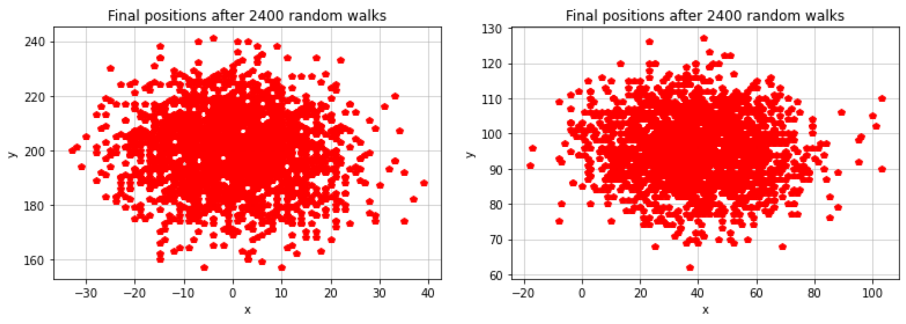

Figure 1: Simulations of the fair Random Walk after 2400 walks. (Left) (rate parameter from categorical distribution of 0.15). (Right) Simulation of the Biased Random Walk after 2400 walks. (rate parameter from an exponential distribution). Each entry corresponds to 400 steps and 2400 walk. Shown in the figures are the X and y displacement of the particle

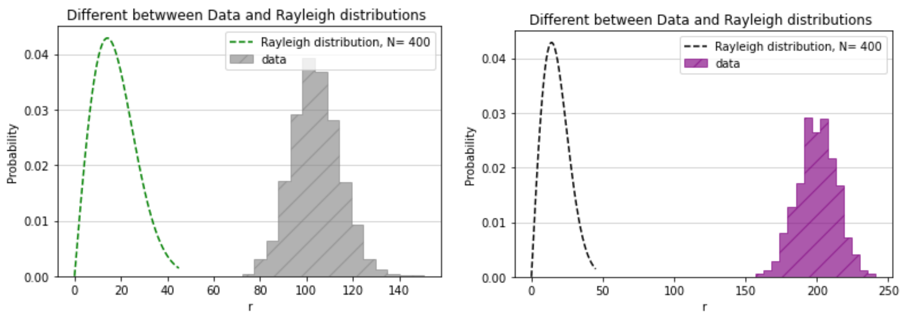

Figure 2: Simulations of the fair Random Walk after 2400 walks. (Left) (rate parameter from categorical distribution of 0.15). (Right) Simulation of the Biased Random Walk after 2400 walks. (rate parameter from an exponential distribution). Each entry corresponds to 400 steps and 2400 walk. Shown in the figures are the comparison between the data and the Raleigh distribution probability density distribution

This is more like the posterior probability. here the curve is normalized.

5 Conclusion

From the result of the code. I can conclude that the gap between the data and the Raleigh distribution indicates significant differences between the two distributions.However, in general, the plot can be used to visually compare the shape and distribution of the data to the Rayleigh distribution. If the two distributions match closely, the plot will show a relatively smooth and consistent curve for both the data and Rayleigh distribution but in this case there is a gap between the two distributions.

My refernces are Stackoverflow.com Prof.Rogan's Code and typing format on overleaf. colleagues, Above all Prof. Rogan's lectures, materials and office hour.