Pourna Sengupta — March 3, 2021

Po

Simulation of Random Walks

Line[7:27]: Problem 2

Line [55 → 55:64]:

Problem 1 (a, b, c)

```
1  import matplotlib.pyplot as plt
2  import math
3  import numpy as np
4  import random
5  import pylab
6
7  def navEff():
8      print("PROBLEM 2 \n")
9      w = np.arange(0.0, 1.0, 0.0001)
10     totalDist = 0.0
11     ne = list()
12     for i in range(1000):
13         #x = v(w[i] * math.cos(thetai + thetaB) + (1.0 - w[i])* math.cos(thetai + thetaC))
14         x = (w[i] * math.cos(math.pi/3) + (1.0 - w[i])* math.cos(math.pi/30))
15         #y = v(w[i] * math.sin(thetai + thetaB) + (1.0 - w[i])* math.sin(thetai + thetaC))
16         y = (w[i] * math.sin(math.pi/3) + (1.0 - w[i])* math.sin(math.pi/30))
17         #Navigational Efficiency
18         #NE = Net distance in theta initial direction / total distance
19         d = math.sqrt(math.pow(x, 2) + math.pow(y, 2))
20         totalDist = totalDist + d
21         ne.append(d / totalDist)
22
23     #print(ne)
24     ne.remove(1.0)
25     maxN = max(ne)
26     maxW = ne.index(maxN)
27     print("Maximum Navigational Efficiency (for theta(*CRW) = pi/30 and theta(*BRW) = pi/3): " , maxN, " when W = ", maxW + 1, "\n ")
28
29 class random_walks_python():
30     print("PROBLEM 1 \n")
31     def random_walks(self):
32         N = 500  # no of steps per trajectory
33         realizations = 50  # number of trajectories
34         v = 1.0  # velocity (step size)
35         theta_s_array = [round(math.pi / 24, 4), round(math.pi / 12, 4), round(math.pi / 3, 4)]  # the width of the random walk turning angle
36         w_array = [0.0, 0.5,
37                    1.0]  # w is the weighting given to the directional bias (and hence (1-w) is the weighting given to correlated motion)
38         ratio_theta_s_brw_crw = 1
39         plot_walks = 1
40         count = 0
41
42         efficiency_array = np.zeros([len(theta_s_array), len(w_array)])
43         for w_i in range(len(w_array)):
44             w = w_array[w_i]
45             for theta_s_i in range(len(theta_s_array)):
46                 theta_s_crw = np.multiply(ratio_theta_s_brw
47                 theta_s_brw = theta_s_array[theta_s_i]
48                 x, y = self.BCRW(N, realizations, v, theta_
49                 if plot_walks == 1:
50                     count += 1
51                     plt.figure(count)
52                     plt.plot(x.T, y.T)
53                     plt.axis('equal')
54                 efficiency_array[theta_s_i, w_i] = np.divic
55         #FIGURE 1 CODE ADDED HERE
56
57             # plt.show()
58         plt.figure()
59         legend_array = []
60         w_array_i = np.repeat(w_array, len(efficiency_array))
61         for theta_s_i in range(0, len(theta_s_array)):
62             legend_array.append(
63                 ["$\theta^{*CRW}=$", (ratio_theta_s_brw_crw * theta_s_array[theta_s_i]), "$\theta^{*BRW}=$",
64                  (theta_s_array[theta_s_i])])
65
66         plt.xlabel('w')
67         plt.ylabel('navigational efficiency')
68         wVals = list([0, 20, 30])
69         plt.plot(wVals, efficiency_array[0], 'bo', label=legend_array[0])
70         plt.plot(wVals, efficiency_array[1], 'go', label=legend_array[1])
71         plt.plot(wVals, efficiency_array[2], 'ro', label=legend_array[2])
72         plt.legend(loc='best', prop={'size': 5.2})
73         plt.show()
```

```
55 b0, b05, b1 = efficiency_array[0]
56 g0, g05, g1 = efficiency_array[1]
57 r0, r05, r1 = efficiency_array[2]
58 b = (b1 + g1 + r1) / 3
59 c = (b0 + g0 + r0) / 3
60 bc = (b05 + g05 + r05) / 3
61 print("(a) Biased Random Walks (BRW): ", b, "\n")
62 print("(b) Correlated Random Walks (CRW): ", c, "\n")
63 print("(a) Equally Balanced Random Walks (BRW & CRW): ", bc, "\n")
64 navEff()
```

Figure 1

```python
    # The function generates 2D-biased correlated random walks
    def BCRW(self, N, realizations, v, theta_s_crw, theta_s_brw, w):
        X = np.zeros([realizations, N])
        Y = np.zeros([realizations, N])
        theta = np.zeros([realizations, N])
        X[:, 0] = 0
        Y[:, 0] = 0
        theta[:, 0] = 0

        for realization_i in range(realizations):
            for step_i in range(1, N):
                theta_crw = theta[realization_i][step_i - 1] + (theta_s_crw * 2.0 * (np.random.rand(1, 1) - 0.5))
                theta_brw = (theta_s_brw * 2.0 * (np.random.rand(1, 1) - 0.5))

                X[realization_i, step_i] = X[realization_i][step_i - 1] + (v * (w * math.cos(theta_brw))) + (
                            (1 - w) * math.cos(theta_crw))
                Y[realization_i, step_i] = Y[realization_i][step_i - 1] + (v * (w * math.sin(theta_brw))) + (
                            (1 - w) * math.sin(theta_crw))

                current_x_disp = X[realization_i][step_i] - X[realization_i][step_i - 1]
                current_y_disp = Y[realization_i][step_i] - Y[realization_i][step_i - 1]
                current_direction = math.atan2(current_y_disp, current_x_disp)

                theta[realization_i, step_i] = current_direction

        return X, Y

rdm_plt = random_walks_python()
rdm_plt.random_walks()
```

Simulation of Random Walks Code Outputs

PROBLEM 1

(a) Biased Random Walks (BRW): 0.9357885607863011

(b) Correlated Random Walks (CRW): 0.23581187210099

(a) Equally Balanced Random Walks (BRW & CRW): 0.9175197856865527

PROBLEM 2

Maximum Navigational Efficiency (for theta(*CRW) = pi/30 and theta(*BRW) = pi/3): 0.4999896952371048 when W = 1

Paper Review

(a) What do you feel the main contribution of this paper is?

The papers main contribution to the field of science is their research into movement patterns and finding strong evidence showing that multiphasic movement strategy is a better model than one-behavior strategies. The research data analysis also identifies movement behaviors that distinguish movement patterns: a directed extensive phase and tortuous intensive phase. This finding is incredibly useful to researches in the field as is allows for even more accurate analysis and understanding of animal movement patterns.

(b) What's the essential principle that the paper exploits?

The essential principle that the paper exploits is the comparison of multiphasic movement strategy models to singular phase models. The paper clearly proves the superiority of multiphasic models by showing that 98% of the movement paths studied used CCRWs as the best model. The researchers emphasize the need to compare Lévy walking models to stronger multiphasic models such as CCRWs.

(c) Describe one major strength of the paper.

The paper clearly proves that multiphasic models are better suited to analyze movement patterns in searching behaviors and movement patterns. A strength of this paper is their discussion of potential error in their use of the local turn method. The writers clearly explain the effects of this potential error and how it can manipulate data analysis. With this, they also discussed the benefit of using the model that could cause manipulation. I find this to be very interesting and necessary in papers such as this one. Discussing methods and areas where there are significant findings but maybe a few errors that are unresolved can allow for discussion of possible ways to resolve the errors and potentially be able to work past any manipulation of data.

(d) Describe one weakness of the paper.

Something I think may be useful (though may not be standard in research journal articles) is figures, graphs, tables, etc. in the discussion to show some of the analyses made. This could really help in helping readers understand the data analysis better.

(e) Describe one future work direction you think should be followed.

The writers mentioned memory-based movement models, which I see as a very interesting direction to move in for research in searching behaviors and movement patterns. Especially for animals that migrate annually, it may be very interesting to see long term if migration patterns differ over time due to changing generations and varied familiarity with environments and migration patterns.