

```

1 import random
2 import numpy as np
3
4 # CONSTANTS
5 p = np.array([12, 24, 48, 70, 100, 120])
6 N = 120
7 n_trials = 100000
8
9
10 # FUNCTIONS
11 def generate_random_matrix(n):
12     r = np.random.randint(2, size=(n, N))
13     r[r == 0] = -1
14     return r
15
16
17 def perform_one_trial(pattern_n):
18     # Generate weight matrix
19     patterns = generate_random_matrix(pattern_n)
20     W = (1/N) * np.matmul(patterns.T, patterns)
21     # np.fill_diagonal(W, 0)
22
23     # Choose random pattern
24     nu_index = random.randint(0, pattern_n - 1)
25     chosen_pattern = patterns[nu_index]
26     # Choose random neuron
27     neuron_index = random.randint(0, N - 1)
28     target_neuron_value = chosen_pattern[neuron_index]
29
30     # Feed chosen pattern to network
31     selected_weights = np.matrix(W[neuron_index]).T
32     new_neuron_value = np.sign(np.dot(chosen_pattern
33 , selected_weights))
34     new_neuron_value = new_neuron_value.item()
35     if new_neuron_value == 0:
36         new_neuron_value = 1
37     return new_neuron_value == target_neuron_value
38
39 def perform_n_trials(p, n):
40     success = 0
41     for i in range(n):
42         success += perform_one_trial(p)

```

```
43     print("Error probability for " + str(p) + "  
patterns: " + str(1 - success / n))  
44     return 1 - success / n  
45  
46  
47 # MAIN CODE  
48 pnt_vec = np.vectorize(perform_n_trials)  
49 print(pnt_vec(p, n_trials))  
50
```

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # PATTERNS
5 x1 = np.array([ [ -1, -1, -1, -1, -1, -1, -1, -1, -1
  , -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1],[ -1, -1
  , 1, 1, 1, 1, 1, 1, -1, -1],[ -1, 1, 1, 1, -1, -1, 1
  , 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1
  , 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1
  , 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1
  ],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1
  , -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1
  , -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1
  , 1, -1, -1, 1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1
  , -1, -1],[ -1, -1, -1, 1, 1, 1, 1, 1, -1, -1],[ -1
  , -1, -1, -1, -1, -1, -1, -1, -1, -1] ])
6 x2 = np.array([ [ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1
  ],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1
  , 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1
  , -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1
  , -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1
  , 1, 1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1
  ],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1
  , -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1
  , 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1
  ],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1
  , 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1
  , -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1] ])
7 x3 = np.array([ [ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1
  , 1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, -1
  , 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1
  ],[ -1, -1, -1, 1, 1, 1, 1, -1, -1],[ -1, -1, -1, 1, 1
  , 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1
  , 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1
  , -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1, -1, -1
  ],[ 1, 1, 1, -1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1, -1
  , -1, -1, -1, -1, -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1
  , -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1] ])
8 x4 = np.array([ [ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1
  ],[ -1, -1, 1, 1, 1, 1, 1, 1, 1, -1],[ -1, -1, -1, -1
  , -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1
  , 1, 1],[ 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1
  ],[ -1

```

```

8  , -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1
  , -1, -1, 1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1
  , -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, -
  1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1
  , 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1
  ],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1
  , -1, -1, -1, 1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1
  , 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, 1, -1, -1] ])
9 x5 = np.array([ [ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1
  ],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -
  1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1,
  1, 1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1
  , 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1
  , -1, -1, 1, 1, -1],[ -1, 1, 1, 1, 1, 1, 1, 1, 1, -1
  ],[ -1, 1, 1, 1, 1, 1, 1, 1, 1, -1],[ -1, -1, -1, -1
  , -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1
  , 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1
  ],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -
  1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1
  , -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -
  1] ])
10
11 original_shape = x1.shape
12 patterns = [x1, x2, x3, x4, x5]
13 fig = plt.figure()
14 N = original_shape[0]*original_shape[1]
15 rounds_per_iter = 10000
16
17
18 # RESHAPE
19 for i, val in enumerate(patterns):
20     patterns[i] = np.reshape(val, (N))
21
22 # WEIGHT MATRIX
23 pattern_matrix = np.matrix(patterns)
24 W = np.matmul(pattern_matrix.T, pattern_matrix)
25 np.fill_diagonal(W, 0)
26
27 def ms_distance(x,y):
28     return np.sum(np.square((x - y)))
29
30 def draw_digit(x):
31     ax1 = fig.add_subplot(121)
32     ax1.imshow(x, interpolation='nearest')

```

```

33
34 def feed(x):
35     iteration = 0
36     running = True
37
38     while running:
39         old_x = x.copy()
40         for i in range(N):
41             # Update all neurons
42             neuron_index = i
43             selected_weights = np.matrix(W[
neuron_index]).T
44             new_neuron_value = np.sign(np.dot(x,
selected_weights))
45             new_neuron_value = new_neuron_value.item
()
46             if new_neuron_value == 0:
47                 new_neuron_value = 1
48                 x[0][neuron_index] = new_neuron_value
49                 curr_dist = ms_distance(old_x, x)
50                 print("it. finished : " + str(curr_dist))
51                 running = (curr_dist != 0)
52         x = x.reshape(original_shape)
53         draw_digit(x)
54         print(x.tolist())
55
56
57 # MAIN
58 x = np.array([[1, -1, -1, 1, 1, 1, 1, -1, -1, 1], [1
, -1, -1, 1, 1, 1, 1, -1, -1, 1], [1, -1, -1, 1, 1, 1
, 1, -1, -1, 1], [1, -1, -1, 1, 1, 1, 1, -1, -1, 1
], [1, -1, -1, 1, 1, 1, 1, -1, -1, 1], [1, -1, -1, 1
, 1, 1, 1, -1, -1, 1], [1, -1, -1, 1, 1, 1, 1, 1, -1, -1
, 1], [1, -1, -1, -1, -1, -1, -1, -1, -1, 1], [1, -1
, -1, -1, 1, 1, 1, 1, -1], [-1, -1, -1, -1, -1, -1, -
1, -1, 1, 1, -1], [-1, -1, -1, -1, -1, -1, -1, -1, 1, 1
, -1], [-1, -1, -1, -1, -1, -1, -1, 1, 1, -1], [-1, -
1, -1, -1, -1, -1, 1, 1, -1], [-1, -1, -1, -1, -1
, -1, -1, 1, 1, -1], [-1, -1, -1, -1, -1, -1, -1, 1,
1, -1], [-1, -1, -1, -1, -1, -1, -1, 1, 1, -1]])
59 # x = np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1
, 1, -1, -1, -1, -1, 1, 1, 1], [-1, -1, 1, 1, 1, 1, 1,
1, -1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1
, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1

```

```

59 , 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1
    ], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1
    , -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1,
    1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1,
    1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1
    , 1, 1, -1], [-1, -1, 1, 1, 1, 1, 1, 1, -1, -1], [-1
    , -1, -1, 1, 1, 1, 1, -1, -1, -1], [-1, -1, -1, -1
    , -1, -1, -1, -1, -1, -1]])
60 # x = np.array([[1, 1, 1, -1, -1, 1, -1, 1, 1, -1
    ], [1, 1, 1, -1, -1, 1, -1, 1, 1, -1], [1, 1, 1, -1
    , -1, 1, -1, 1, 1, -1], [1, 1, 1, -1, -1, 1, -1, 1,
    1, -1], [1, 1, 1, -1, -1, 1, -1, 1, 1, -1], [1, 1, 1
    , -1, -1, 1, -1, 1, 1, -1], [1, 1, 1, -1, -1, 1, -1
    , 1, 1, -1], [1, 1, 1, 1, 1, -1, 1, 1, 1, -1], [1, 1
    , 1, 1, 1, -1, 1, 1, 1, -1], [1, -1, -1, -1, -1, 1
    , -1, 1, 1, -1], [1, -1, -1, -1, -1, 1, -1, 1, 1, -1
    ], [1, -1, -1, -1, -1, 1, -1, 1, 1, -1], [1, -1, -1
    , -1, -1, 1, -1, 1, 1, -1], [1, -1, -1, -1, -1, 1, -
    1, 1, 1, -1], [1, -1, -1, -1, -1, 1, -1, 1, 1, -1
    ], [1, -1, -1, -1, -1, 1, -1, 1, 1, -1]])
61
62 x = np.reshape(x, (1,N))
63 feed(x)
64 plt.show()

```

```

1 import random
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # CONSTANTS
6 N = 200
7 BETA = 2
8 P = 45
9 T_N = 200000
10 T_REPEAT = 100
11
12
13 # FUNCTIONS
14 def generate_random_matrix(n):
15     r = np.random.randint(2, size=(n, N))
16     r[r == 0] = -1
17     return r
18
19
20 def perform_one_trial():
21     # x1 = np.linspace(0, T_N - 1, T_N)
22     # x2 = np.array([0.] * T_N)
23     # fig = plt.figure()
24     # ax = fig.add_subplot(111)
25     # line1, = ax.plot(x1, x2, 'r-')
26     # plt.ion()
27     # plt.ylim([0., 1.1])
28     # plt.xlim([0., int(T_N / 200) - 1])
29     # plt.show()
30
31     # Generate weight matrix
32     patterns = generate_random_matrix(P)
33     W = (1/N) * np.matmul(patterns.T, patterns)
34     np.fill_diagonal(W, 0)
35
36     # Choose first pattern
37     nu_index = 1
38     chosen_pattern = patterns[nu_index]
39     current_state = chosen_pattern.copy()
40
41     order_sum = 0
42
43     for iteration in range(int(T_N / 200)):
44         for neuron_index in range(N):

```

```

45
46         # Feed chosen pattern to network
47         selected_weights = np.matrix(W[
neuron_index]).T
48         b = np.dot(current_state,
selected_weights).item()
49         # if b == 0:
50         #     b = 1
51
52         b1 = np.exp(-2 * BETA * b)
53
54         p_b = 1 / (1 + b1)
55         new_neuron_value = -1
56         r = random.uniform(0., 1.0)
57
58         if r < p_b:
59             new_neuron_value = 1
60             current_state[neuron_index] =
new_neuron_value
61
62         sum_t = 1/N * np.sum(current_state *
chosen_pattern)
63         order_sum = (iteration*order_sum + sum_t) / (
iteration + 1)
64         #     x2[iteration] = order_sum
65         #     line1.set_ydata(x2)
66         #     fig.canvas.draw()
67         #     fig.canvas.flush_events()
68         #
69         # input()
70         return order_sum
71
72
73 def perform_n_trials():
74     x1 = np.linspace(0, T_REPEAT - 1, T_REPEAT)
75     x2 = np.array([0.] * T_REPEAT)
76
77     fig = plt.figure()
78     ax = fig.add_subplot(111)
79     line1, = ax.plot(x1, x2, 'r-')
80     plt.ion()
81     plt.ylim([0., 1.1])
82     plt.xlim([0., T_REPEAT - 1])
83     plt.show()

```



```
84     order_sum = 0.
85
86     for i in x1:
87         a = perform_one_trial()
88         x2[int(i)] = a
89         line1.set_ydata(x2)
90         fig.canvas.draw()
91         fig.canvas.flush_events()
92         print(a)
93         order_sum += a
94
95     print("<m1> = " + str(order_sum / float(T_REPEAT
96         )))
97     input() # Block afterwards
98 # MAIN CODE
99 perform_n_trials()
100
101
```