Home Work 2 - Kohonen algorithm

Students:

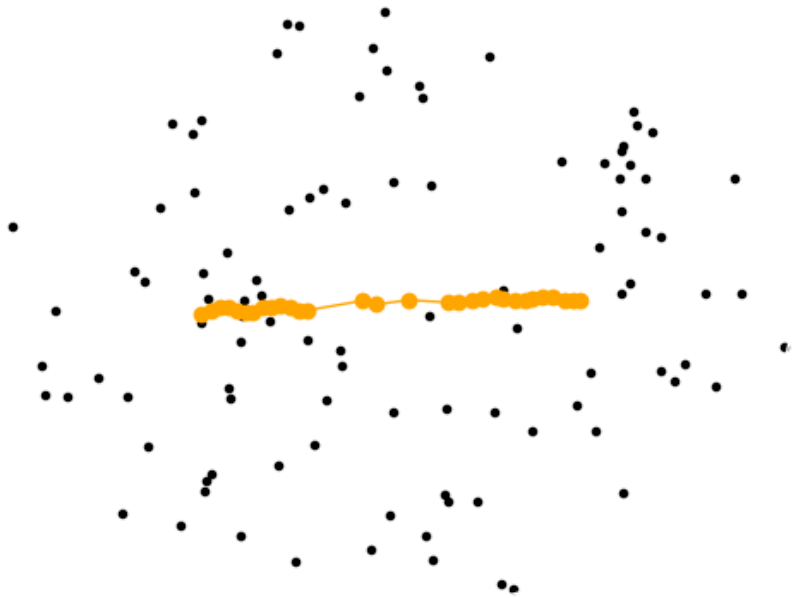
Warda Essa - 208642793

Eli Haimov - 308019306

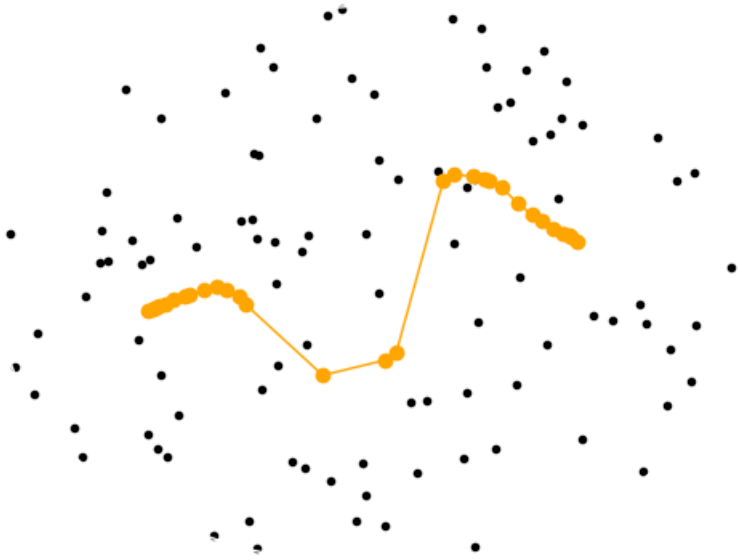
Omar Essa - 315535435

Part A:

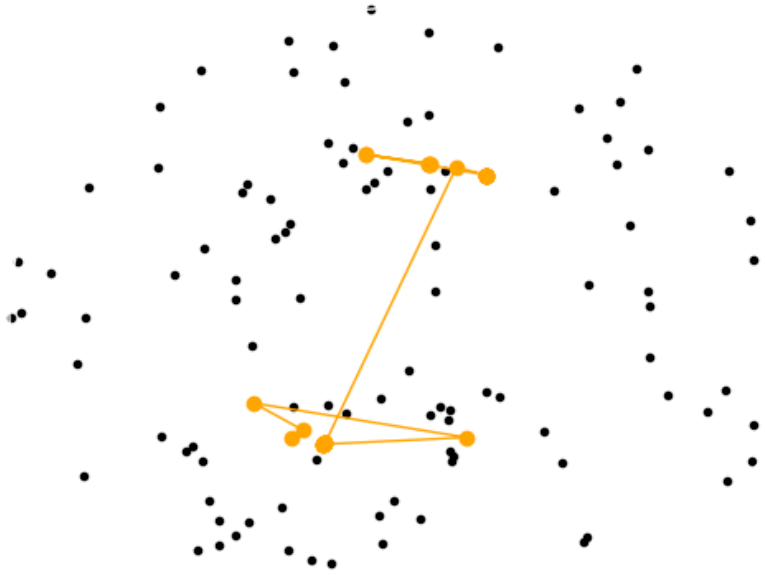
Itr = 2.



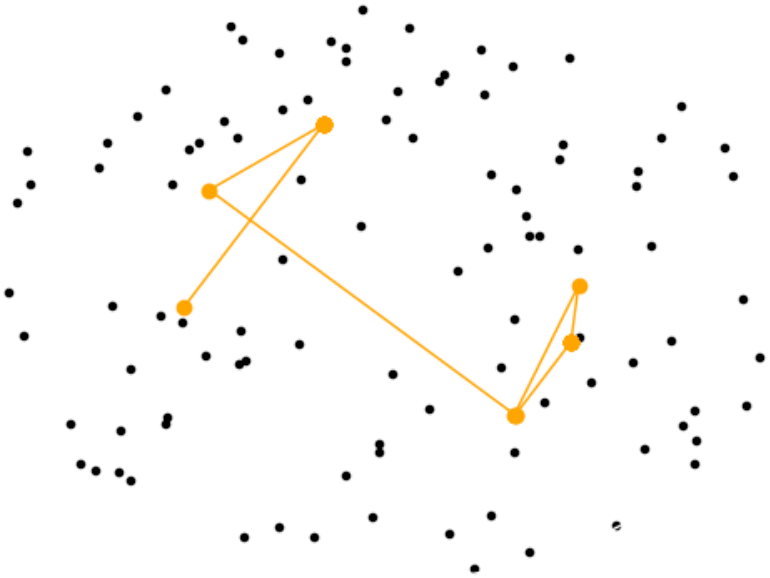
Itr = 10.



Itr = 50.



Itr = 100.



Itr = 1000.

As we can see at first we implanted the neurons in a line shape at random, for each iteration a data point is chosen randomly and then with the algorithms we coded each time we calculate the closest neuron to the data point and that will be our winner neuron.

At fist as you can see after two iterations the neurons are still close to each other the winner neuron did affect some of the neurons to move with him a lot but there is still some who were affected by him less.

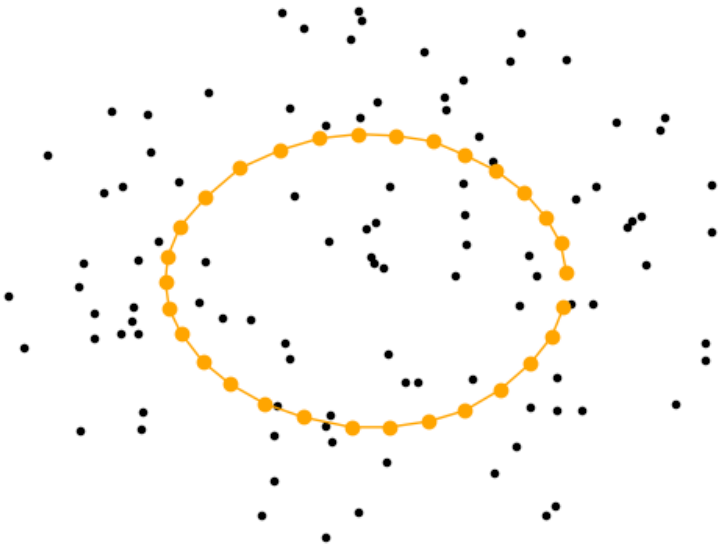
used 30 neurons, the neurons affecting each other less than when there was less neurons connected to each other.

As we do more iterations 10 ,50 ,100 we can see that the winner neuron does move all neuron line to the random data point we chose to each iteration (the neurons are all over the place.

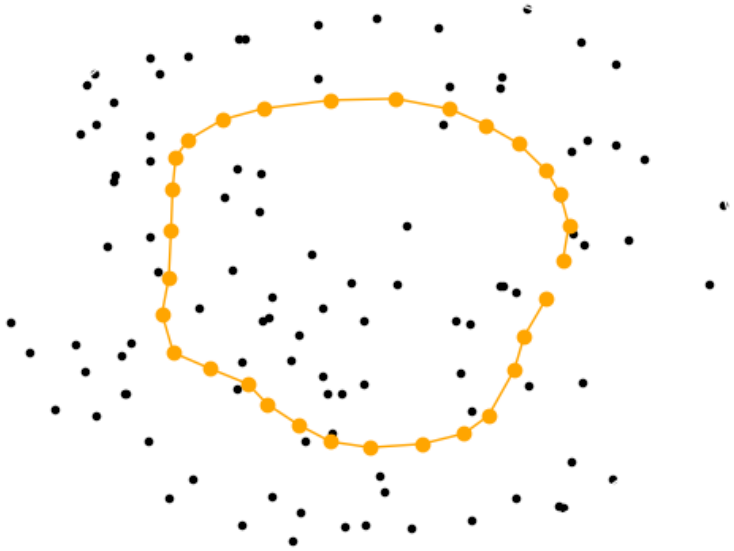
After a lot of iterations we can see that the neurons go farther to the bounds of the circle and looking at 1000 iterations for example we kind of see the neurons takes a shape close to that of a circle.

Part B:

Itr = 2:



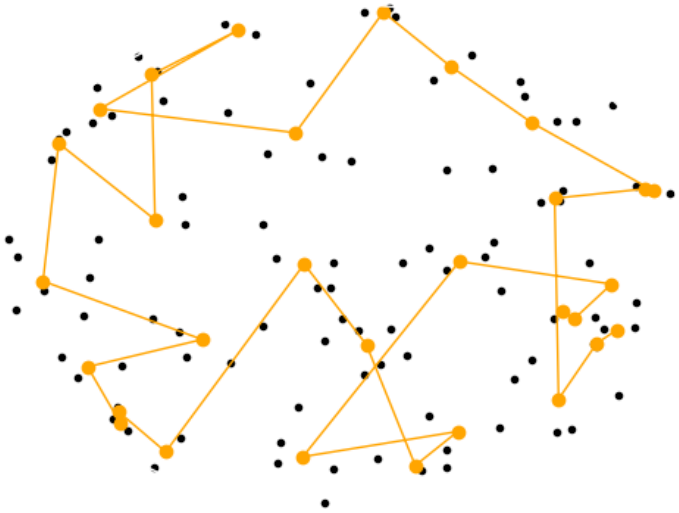
Itr = 10:



Itr = 50:

Itr = 100:

Itr = 1000:



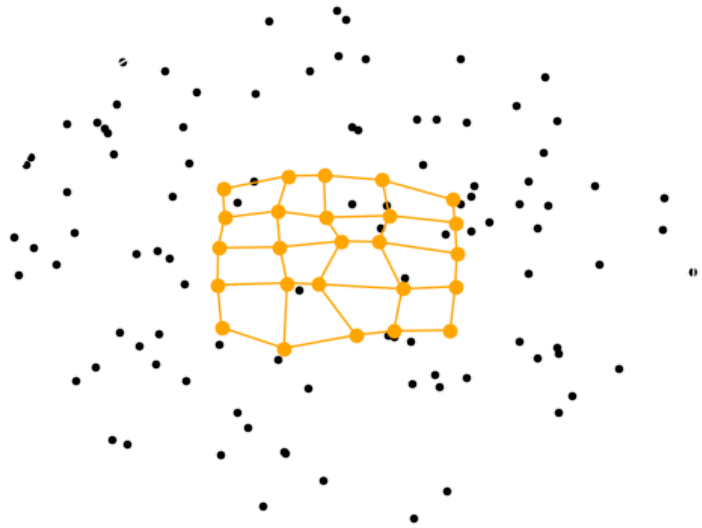
When we use circle shape for the 30 neurons, at first when iteration is 2 we can clearly see that the neurons are forming a perfect circle which is very close to the center of the circle.

As we increase the numbers of iterations we see that the neurons are changing shape and scattering to the sides of the circle. After 100/1000 iteration we see that the most of the neurons moved to the boundaries of the circle and still in the shape of a circle (but not a perfect one).

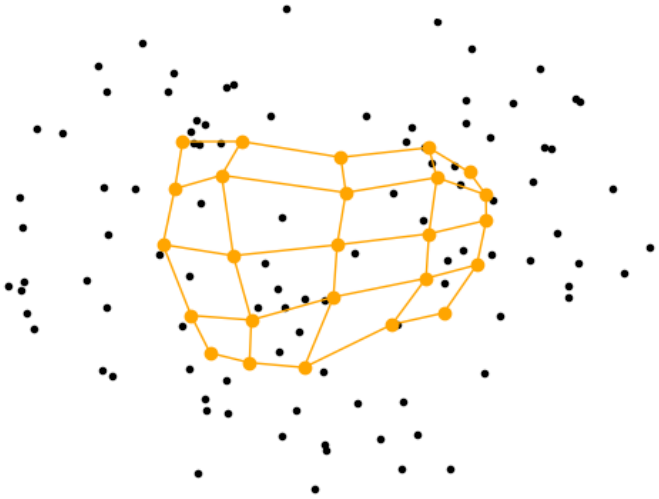
We can clearly see that the neurons in circle shape or in line shape acted almost the same, which is “move to all sides of the circle”.

Part C:

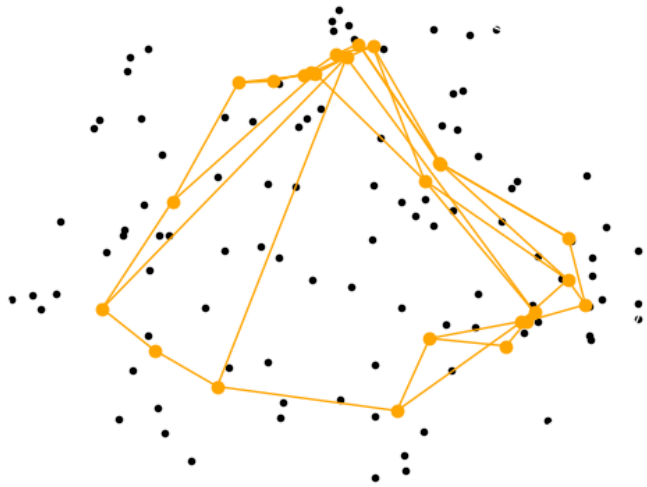
Itr = 2:



Itr = 10:



Itr = 50:



Itr = 100:

Itr = 1000:

5x5 neurons topology is in the shape of square, at first with 2 iteration we can see that the neurons are in the shape of square, close to each other and in the middle. As the number of iterations increase we can see that the neurons are moving all together in the same direction (not all of them). The shape changes to a circle.

At first it may seem that 5x5 topology neurons are doing the same thing as circle and line but in fact there is a different: most of the neurons in 5x5 topology are moving at the same directions to the boundaries of the circle except for one line which is in the other direction and together they form the circle shape.

Part D:

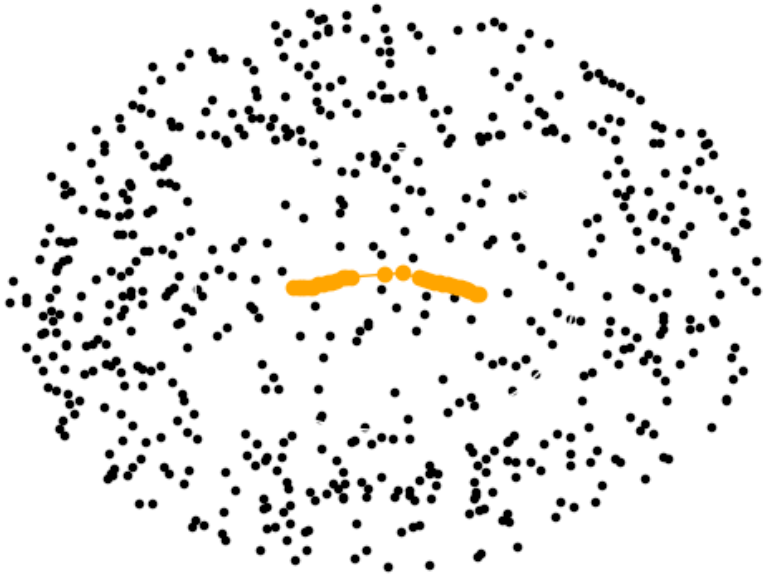
In this part we want the probability to choose a random data point at the center to be less than that everywhere else. So what we increased the number of the data points around the center and by doing that there will be a higher probability to choose data around more than it is to the center.

****

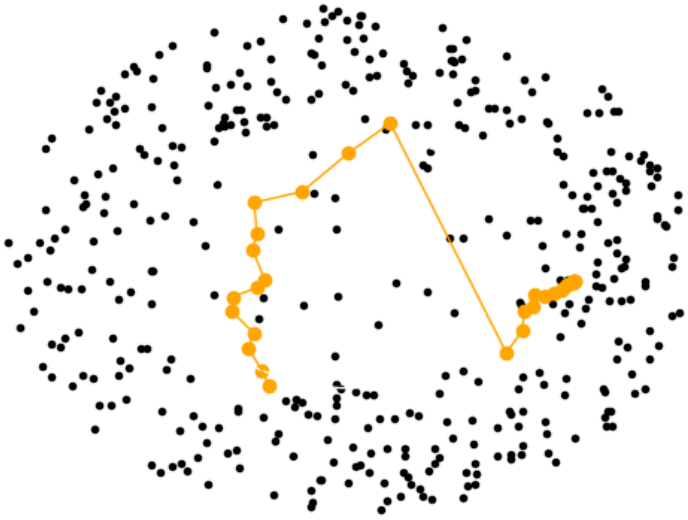
**********LINE**********

****

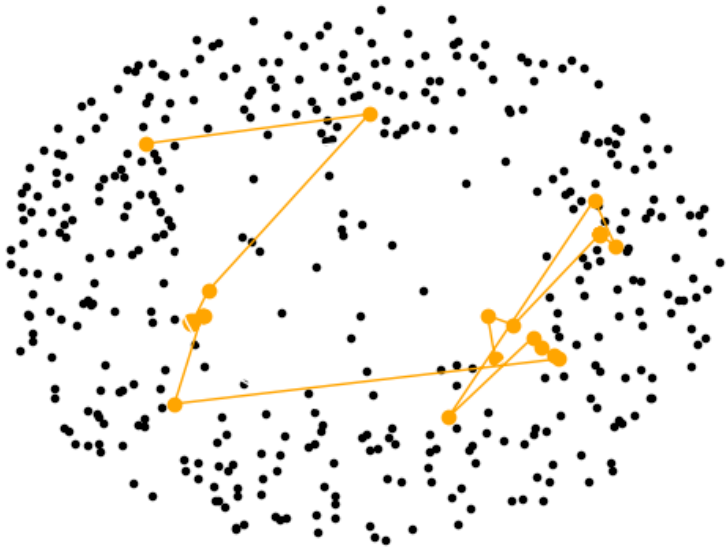
Itr = 2:



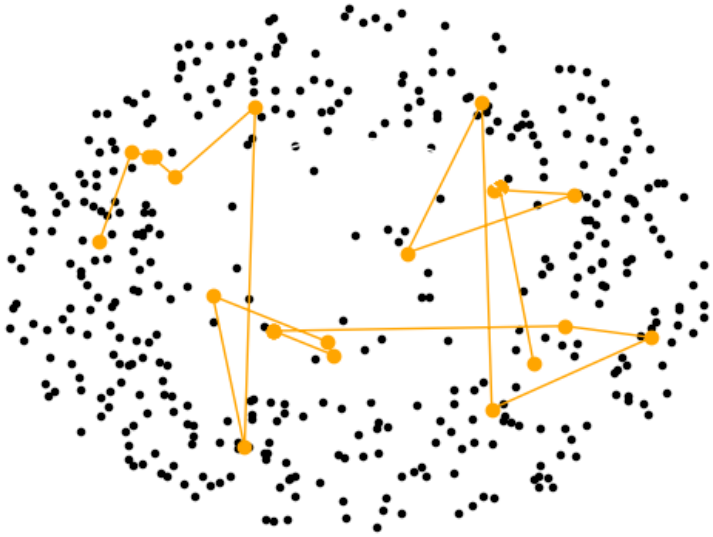
Itr = 10:



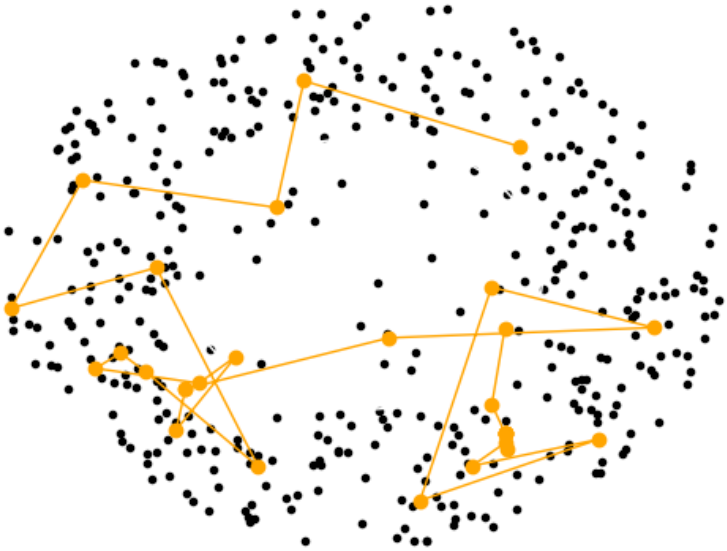
Itr = 50:



Itr = 100:



Itr = 1000:



We can clearly see that even now with the probability to choose a data point in the center is less than that everywhere that the neurons in line topology are acting the same as in part A, the only change now is in the number of iterations.

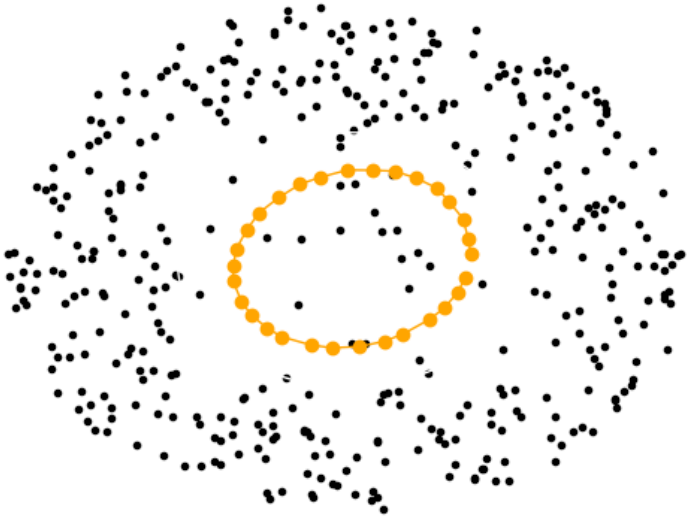
We tested line topology in this part with the same iterations we did in part A and it was clear to us that neurons went to the sides faster than in part A. it took less iterations to get to the same results we had in part A.

****

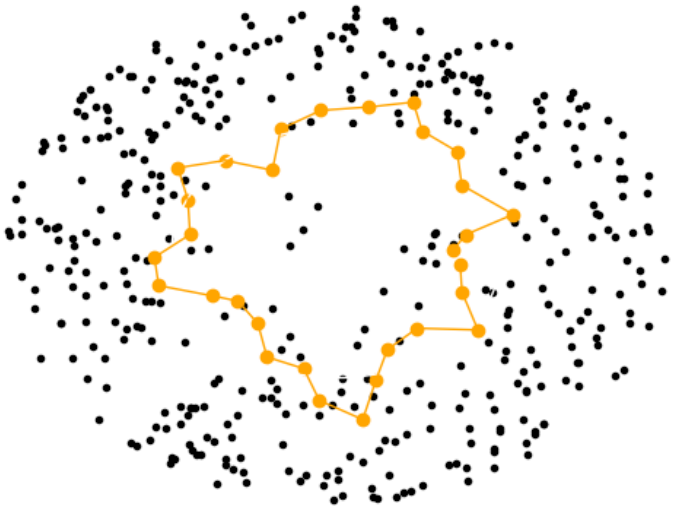
**********CIRCLE**********

****

Itr = 2:



Itr = 10:



Itr = 50:

Itr = 100:

Itr = 1000:

In circle topology in this part we see the exact same thing we have seen in line topology.

The neurons act the same as they did in part A but now the iterations we need to get to the boundaries is less than it was in part A.

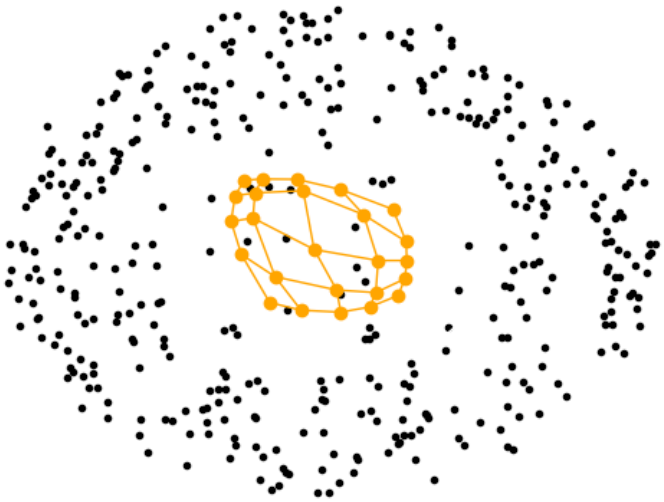
The solo cause of that is the fact that the probability to choose a data point in center is less than that everywhere else so the winner neuron who moves all neurons according to him always be close to a data point far from center.

****

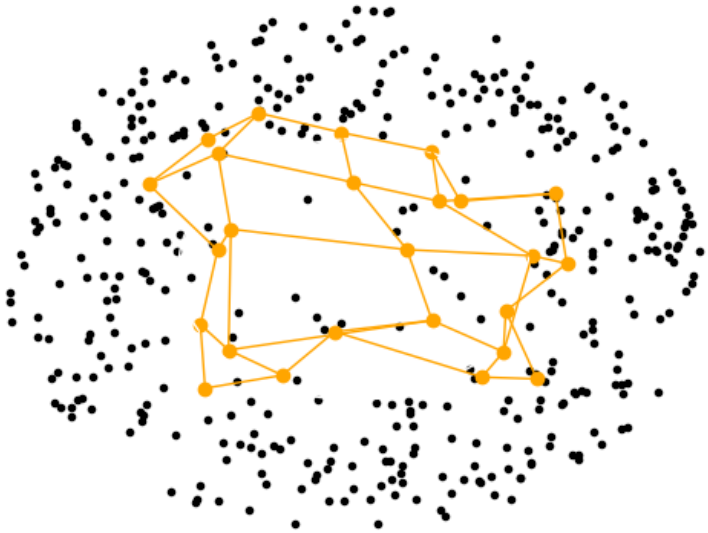
**********5x5 topology**********

****

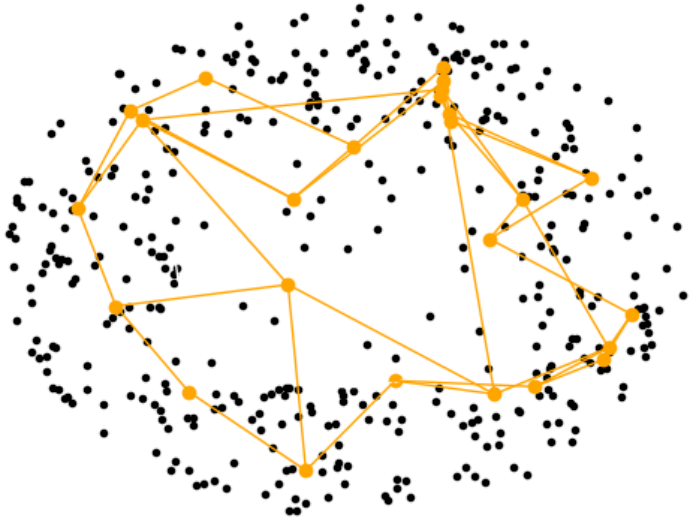
Itr = 2:



Itr = 10:



Itr = 50:



Itr = 100:

Itr = 1000:

5x5 topology in this part is the same as part A neurons go far from the center, same as line and circle in this part, number of iterations is less for neurons to get to the boundaries of the circle. Another change is that neurons scattered all over the circle boundaries in a balanced number. Unlike before (before= most of the neurons are in one direction and the rest on the other) this time it's a balanced diversity all over the circle boundaries.

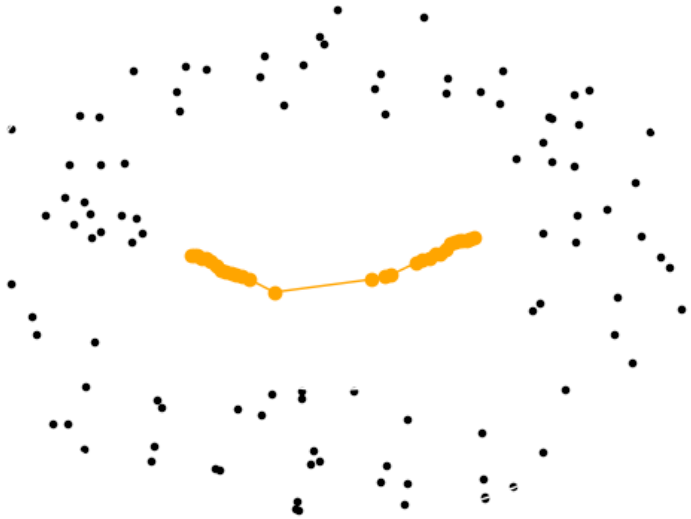
Part E:

****

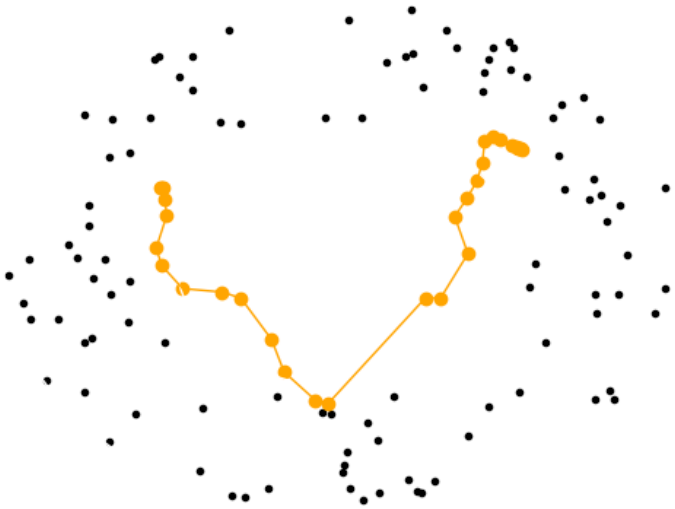
**********LINE**********

****

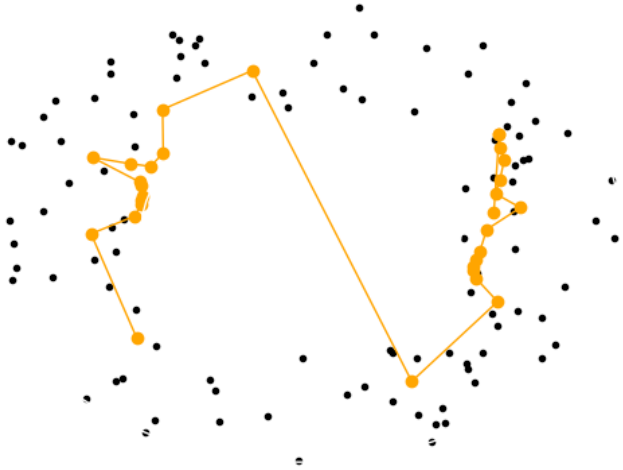
Itr = 2:



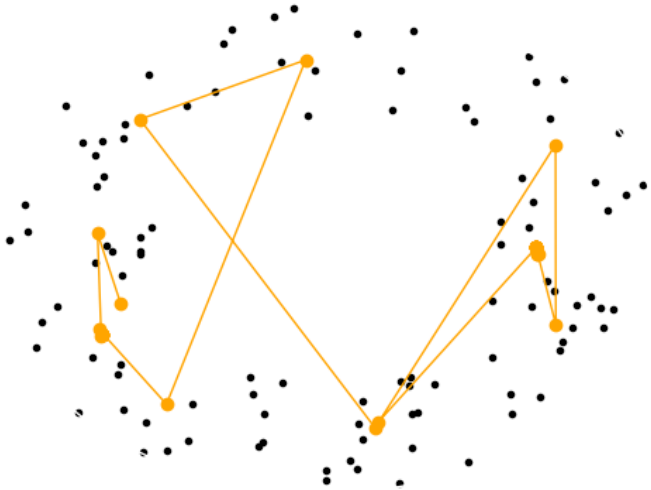
Itr = 10:



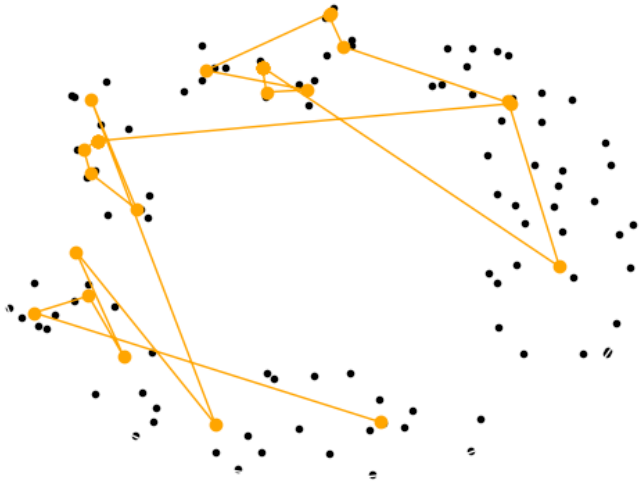
Itr = 50:



Itr = 100:



Itr = 1000:



Now we scattered the data point between to circles and activated kohonen algorithm on 30 neurons in line shape, at first they were in the middle. But as we

increased the iterations number, the neurons starts to get closer to the ring between the two circles.

After much iteration we can see that the neurons start to take the shape of letter C.

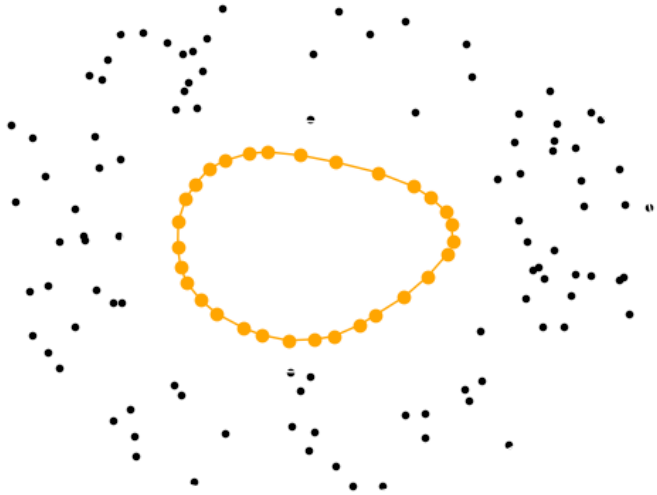
Note: here when data scattered between to circles neurons don't go to the big circle boundaries but instead neurons are scattered all over the ring.

****

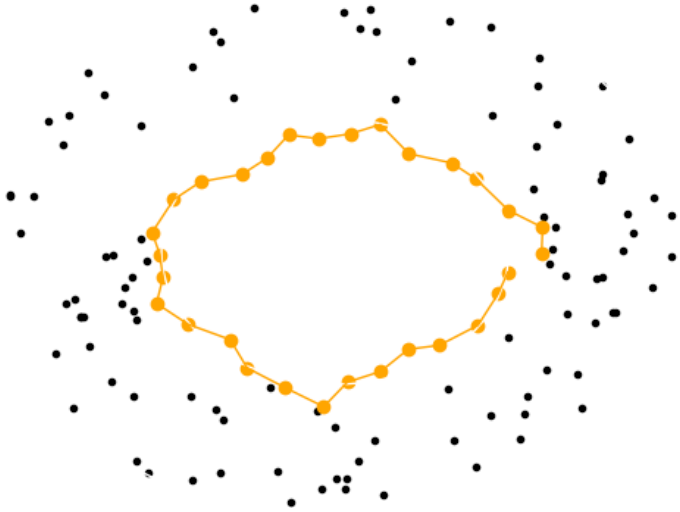
**********circle**********

****

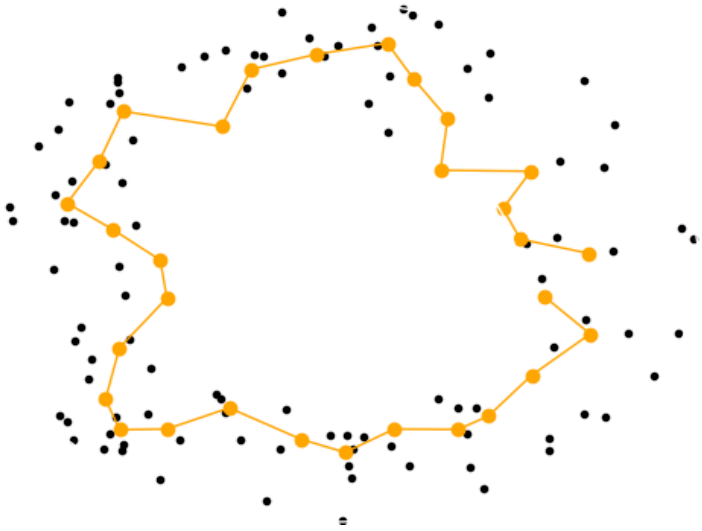
Itr = 2:



Itr = 10:

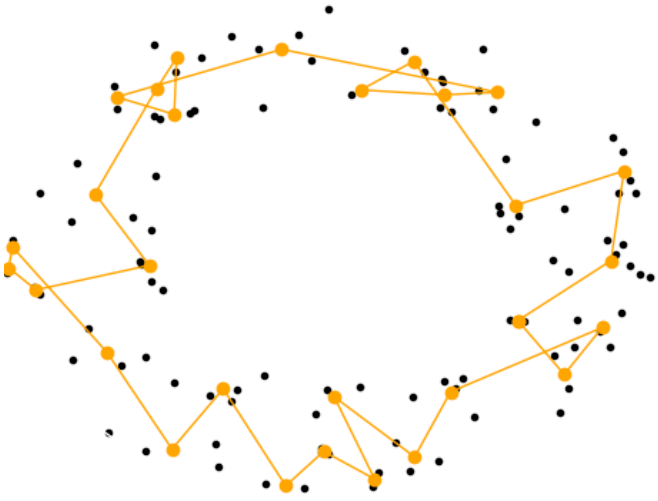


Itr = 50:



Itr = 100:

Itr = 1000:



Same as in line topology in ring, neurons in circle topology also keeps on moving towards the data points scattered between the two circles. After much iteration we can see that the neurons takes the shape of the ring, neurons scattered all over the ring and not sticking to one boundary as in part B.

CODE:

Ex2.py > ...

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



def loadingNeurons(size,sq,r,shape):
    if (shape == "line"):
        neurons = np.zeros((size, 2))
        s = r / len(neurons)
        first = -r / 2
        for i in range(size):
            neurons[i] = np.array([first, 0])
            first = first + s
        return neurons
    elif shape == "circle":
        neurons = np.zeros((size, 2))
        pi = math.pi
        for i in range(len(neurons)):
            neurons[i][0] = math.cos(2 * pi / len(neurons) * i) * r / 2
            neurons[i][1] = math.sin(2 * pi / len(neurons) * i) * r / 2
        return neurons
        neurons = np.zeros((size, 2))
    elif shape =="5x5topology":
        neurons = np.zeros((size, 2))
        s = (r / 2) / (sq - 1)
        Xpoint = -r / 4
        Ypoint = -r / 4
        lines = int(size / sq)
        for i in range(lines):
            for j in range(sq):
                neurons[i * sq + j] = np.array([Xpoint, Ypoint])
                Xpoint = Xpoint + s
```

Ex2.py > ...

```
35                  Xpoint = -r / 4
36                  Ypoint = Ypoint + s
37              return neurons
38
39
40  def coord(arr, shape):
41      if shape == "5x5topology":
42          x = np.array([])
43          y = np.array([])
44          i = 0
45          while i < len(arr):
46              x = np.append(x, np.array(arr[i:i + 5, 0]), 0)
47              y = np.append(y, np.array(arr[i:i + 5, 1]), 0)
48              i = i + 5
49          i = 0
50          while i < 5:
51              j = 0
52              colX = np.array([])
53              colY = np.array([])
54              size = int(len(arr) / 5)
55              while j < size:
56                  colX = np.append(colX, np.array([arr[j * size + i, 0]]), 0)
57                  colY = np.append(colY, np.array([arr[j * size + i, 1]]), 0)
58                  j = j + 1
59              x = np.append(x, colX, 0)
60              y = np.append(y, colY, 0)
61              i = i + 1
62          return x, y
63      elif shape == "circle":
64          x = arr[:, 0]
65          y = arr[:, 1]
66          return x, y
67      elif shape == "line":
68          x = arr[:, 0]
```

Ex2.py > ...

```
            y = arr[:, 1]
            return x, y

def cal_Distance(data, n):
    ans = math.sqrt(math.pow(data[0] - n[0], 2) + math.pow(data[1] - n[1], 2))
    return ans

def winnerNeuronCoord(data, neurons):
    num = math.inf
    neuron_ind =  -1
    for index in range(len(neurons) - 1):
        dis = cal_Distance(data, neurons[index])
        if dis < num:
            num = dis
            neuron_ind = index
    return neuron_ind

def topologicalNeighbor(winner_neuron, n, s):
    distance = cal_Distance(winner_neuron, n)
    ans = math.exp(- (math.pow(distance, 2) / (2 * math.pow(s, 2))))
    return ans

# this function is to:
# update neuron place.
def neuronNewCoord(data, n, a, h):
    new_coord = n + a * h * (data - n)
    return new_coord

def newSigma(s, itr, start):
    ans = s * math.exp(-itr / start)
    return ans

def newAlphe(a, itr, start):
    ans = a * math.exp(-itr / start)
```

Ex2.py > ...

```
103         return ans
104
105     def Algorithm(p, n, ep, r):
106         a = 0.01
107         b = r / 2 + 0.0001
108         start = ep / math.log(ep)
109         for rounds in range(ep):
110             alpha = newAlphe(a, rounds, ep)
111             sigma = newSigma(b, rounds, start)
112             for point in p:
113                 winnerNeuron_ind = winnerNeuronCoord(point, n)
114                 for neurons in range(len(n)):
115                     dis = cal_Distance(n[winnerNeuron_ind],n[neurons])
116                     if dis < sigma:
117                         h = topologicalNeighbor(n[winnerNeuron_ind],n[neurons],sigma)
118                         n[neurons] = neuronNewCoord(point,n[neurons],alpha, h)
119
120
121     r = 2
122     itr = 1000
123     x = 0
124     y = 0
125     sq = 5
126     number_of_Neurons = 30
127     dataShape = "ring"
128     neuronsShape = "5x5topology"
129
130
131     # scattering data points in circle
132     if dataShape == "circle":
133         circle_r = 4
134         circle_amount = 100
135         circle__x =0
136         circle__y = 0
```

Ex2.py > ...

```
        circle__y = 0
        size = circle_amount
        answer = np.zeros((size, 2))
        for i in range(size):
            r = circle_r * math.sqrt(random())
            s = random() * 2 * math.pi
            x = circle__x + r * math.cos(s)
            y = circle__y + r * math.sin(s)
            answer[i] = np.array([x,y])
    #scattering data points in ring
    elif dataShape == "ring":
        size = 100
        answer = np.zeros((size, 2))
        for i in range(size):
            x=0
            y=0
            while (abs(x)<2 and abs(y) < 2):
                    r = 4 * math.sqrt(random())
                    s = random() * 2 * math.pi
                    x= 0 + r * math.cos(s)
                    y = 0 + r*math.sin(s)
            answer[i] = np.array([x,y])
    if neuronsShape == "line":
        neurons =loadingNeurons(number_of_Neurons,0, 2,"line")
    elif neuronsShape == "circle":
        neurons = loadingNeurons(number_of_Neurons,0, 2,"circle")
    elif neuronsShape == "5x5topology":
        neurons = loadingNeurons(25, sq, 2,"5x5topology")
    Algorithm(answer, neurons, itr, 2)
    x_values, y_values = coord(neurons, neuronsShape)
    fig, ax = plt.subplots()
    plt.scatter(answer[:, 0], answer[:, 1], color='black', marker='.', label='points')
    plt.scatter(neurons[:, 0], neurons[:, 1], color='orange', marker='o', label='neurons')
    if neuronsShape == "5x5topology":
        x_values = np.array_split(x_values, 10)
```

Ex2.py > ...

```
    fig, ax = plt.subplots()
    plt.scatter(answer[:, 0], answer[:, 1], color='black', marker='.', label='points')
    plt.scatter(neurons[:, 0], neurons[:, 1], color='orange', marker='o', label='neurons')
    if neuronsShape == "5x5topology":
        x_values = np.array_split(x_values, 10)
        y_values = np.array_split(y_values, 10)
        for val in range(10):
            plt.plot(x_values[val], y_values[val], color='orange', linewidth=1.0)
    else:
        plt.plot(x_values, y_values, color='orange', linewidth=1.0)

    if dataShape == "circle":
        c1 = plt.Circle((x, y), r, color='white', fill=False)
        ax.add_artist(c1)
    elif dataShape == "ring":
        c1 = plt.Circle((x, y), 2, color='white', fill=False)
        c2 = plt.Circle((x, y), 2 * 2, color='white', fill=False)
        ax.add_artist(c1)
        ax.add_artist(c2)
    plt.show()
```