INTRODUCTION TO NEURAL NETWORKS (CS 537-01)
ASSIGNMENT 2

SHIVANGI GUPTA
A25266618
sg0097@uah.edu

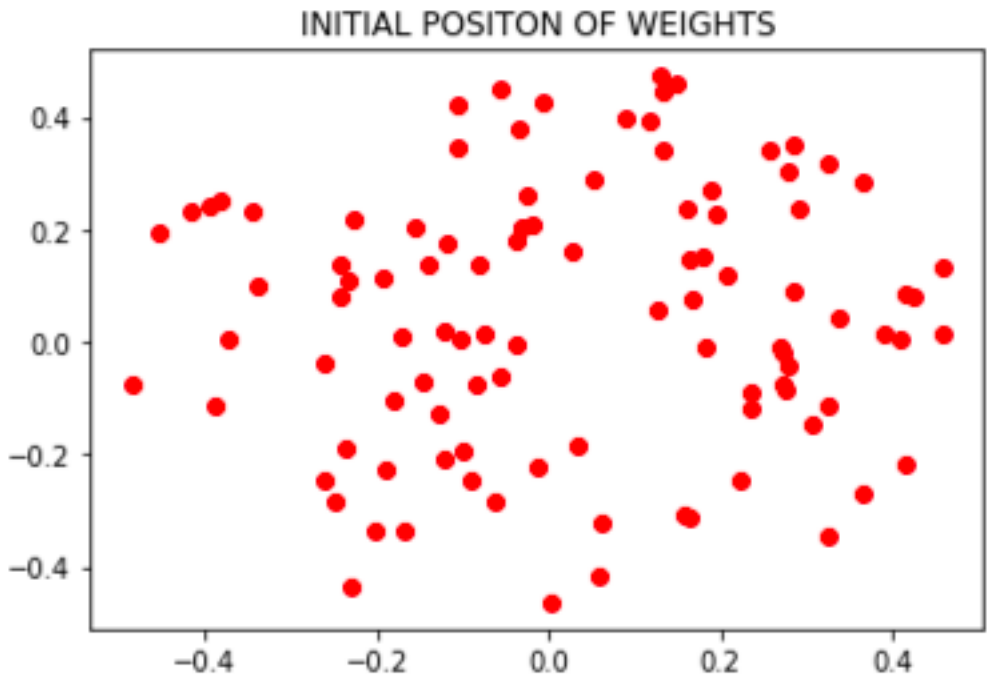
SOURCE CODE

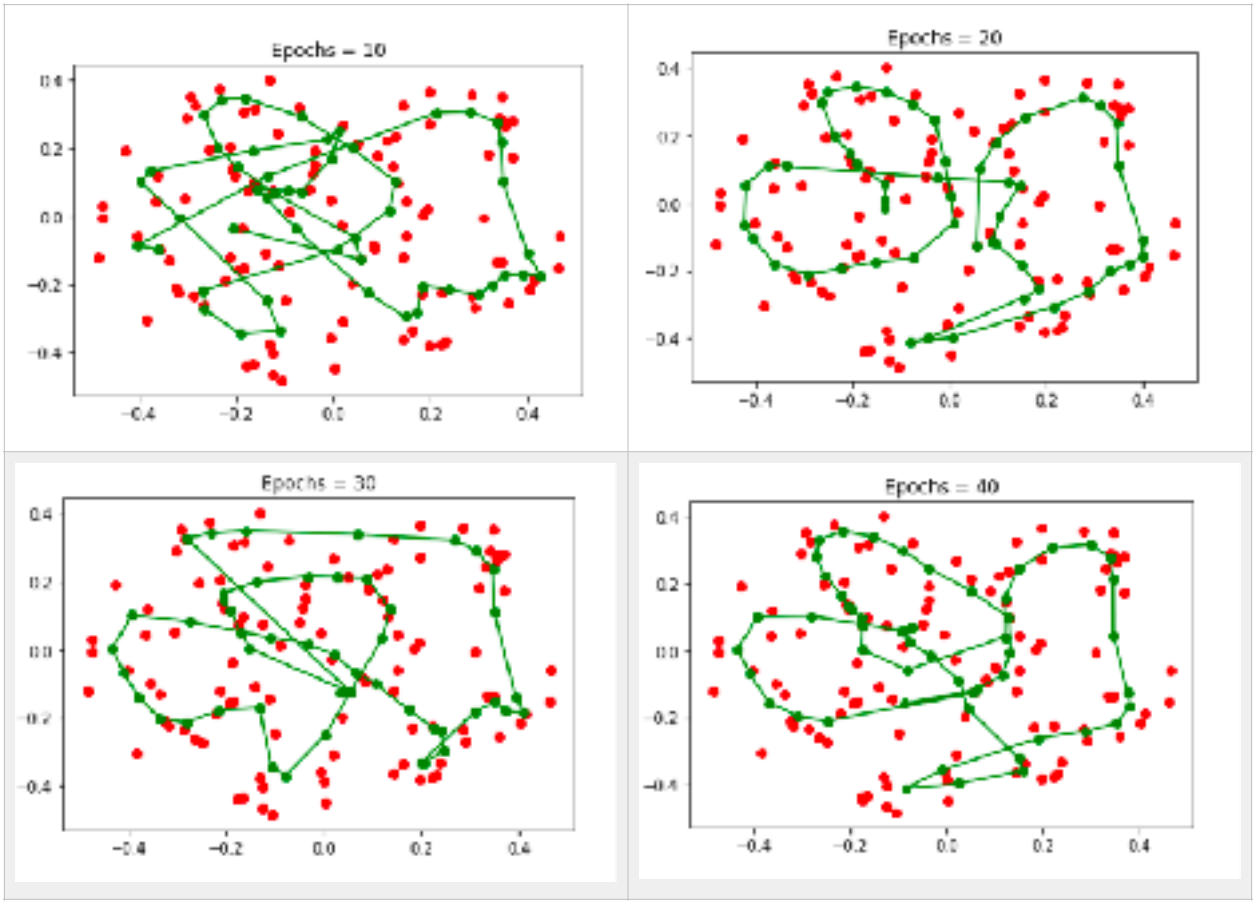
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(20)

X=[]
Y=[]

#Training Points
def Inputs(num_of_iterations):
    for i in range(num_of_iterations):
        x1=np.random.uniform(-0.5,0.5,size=1)
        x2=np.random.uniform(-0.5,0.5,size=1)
        if x1**2+x2**2<0.25 :
            X.append(x1)
            Y.append(x2)
            if len(X)>=100 and len(Y)>=100:
                break
    return X,Y

U=[]
V=[]

#Cluster Units
def Weights(num_of_iterations):
    for i in range(num_of_iterations):
        x1=np.random.uniform(-1,1,size=1)
        x2=np.random.uniform(-1,1,size=1)
        U.append(x1)
        V.append(x2)
        if len(U)>=50 and len(V)>=50:
            break
    return U,V
```

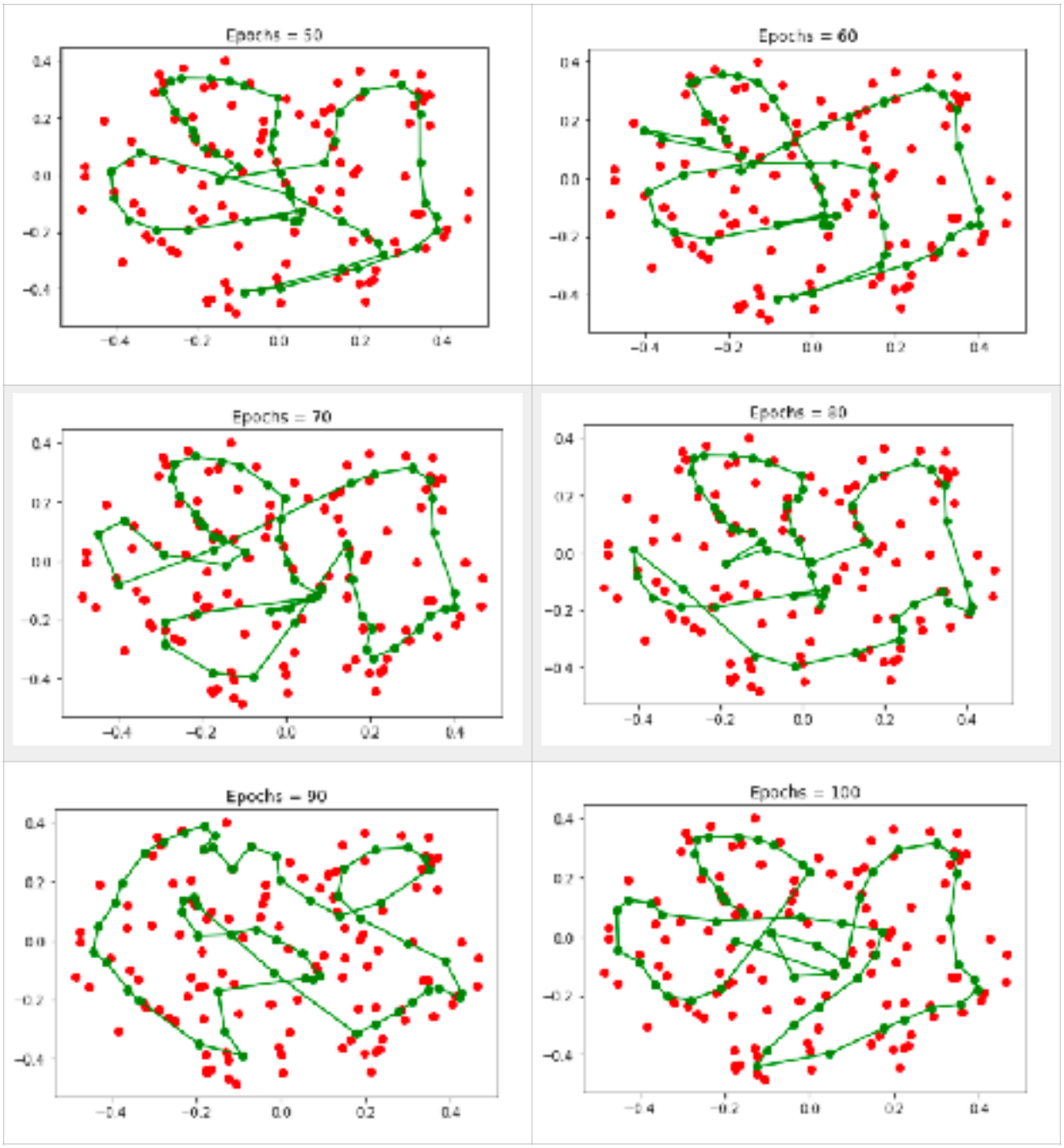
```
#Euclidean Distance
def distance(U,V):
    dist=[]
    for i in range(len(V)):
        dist.append((U-(V[i]))**2)

    return sum(np.transpose(dist)).tolist()

#Kohonen Self-organizing maps
def self_organizing_maps(vector,epochs,alpha,alpha1,weight):
    for i in range(epochs):
        for R in [1,0]:
            if R==1:
                alpha=0.5
                for k in range(100):
                    dist=[]
                    dist.append(distance(U=vector[0][k],V=weight[0]))
                    M=dist[0].index(min(dist[0]))
                    weight[0][M-R]=(weight[0][M-R])+(alpha)*((vector[0][k])-(weight[0][M-R]))
                    weight[0][M]=(weight[0][M])+(alpha)*((vector[0][k])-(weight[0][M]))
                    if M+R>49:
                        weight[0][M]=(weight[0][M])+(alpha)*((vector[0][k])-(weight[0][M]))
                    else:
                      weight[0][M+R]=(weight[0][M+R])+(alpha)*((vector[0][k])-(weight[0][M+R]))
                    alpha-=0.005
                    if(alpha<0.01):
                        break

            else:
                alpha1=0.5
                for k in range(100):
                    dist.append(distance(U=vector[0][k],V=weight[0]))
                    M=dist[0].index(min(dist[0]))
                    weight[0][M]=(weight[0][M])+(alpha1)*((vector[0][k])-(weight[0][M]))
                    alpha1-=0.005
                    if(alpha1<0.01):
                        break
    return weight

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
plt.title('INITIAL POSITON OF WEIGHTS')
plt.show()
```

```
Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=10,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 10')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=20,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 20')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=30,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 30')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=40,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 40')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=50,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 50')
plt.show()
```

```
Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=60,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 60')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=70,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 70')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=80,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 80')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=90,alpha=0.5,alpha1=0
.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 90')
plt.show()

Z=np.transpose(Inputs(200)).transpose()
plt.plot(Z[0],Z[1],'ro')
W=self_organizing_maps(vector=np.transpose(Inputs(200)),epochs=100,alpha=0.5,alpha1=
0.5,weight=np.transpose(Weights(100)))
plt.plot(W.transpose()[0],W.transpose()[1],'-go')
plt.title('Epochs = 100')
plt.show()
```


INITIAL POSITON OF WEIGHTS
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4


Epochs = 10
Epochs = 20
Epochs = 30
Epochs = 40


Epochs = 50
Epochs = 60
Epochs = 70
Epochs = 80
Epochs = 90
Epochs = 100