ECE 559 NEURAL NETWORKS

NAME: LAKSHMI SRIDEVI **UIN: 668383906**

Question 1: Pseudocode

i) Draw n = 300 real numbers uniformly at random on [0, 1], call them X={x1, . . . , xn}.
ii) Draw n real numbers uniformly at random on [− 1/ 10 , 1/ 10], call them V={ν1, . . . , νn}.
iii) Find $d_i = \sin(20x_i) + 3x_i + \nu_i$, i = 1, . . . , n. Plot points (x_i,d_i).
iv) As given the network is 1x24x1. We have N=24 neurons, one input, one output and 3N+1 weights.
v) Weights and biases from input to 24 neurons, **w0 Є** $\mathbf{R^{24x2}}$.
vi) Weights from 24 neurons to output,**w1 Є** $\mathbf{R^{24x1}}$ and bias is taken separately and labelled as "b".
vii) Weights and biases are initialized at random following a normal distribution.
viii) Define the activation function used in the network: the hidden layer uses the hyperbolic tangent as its activation function given by $\varphi_0(v) = \tanh(v)$ and the output layer uses $\varphi_1(v) = v$.
ix) Define the MSE Function: $E = \frac{1}{n}\sum_{i=0}^{n}(\mathrm{di} - f(xi,w))^2$
x) Perform the forward propagation where,
- The output of the hidden layer is taken as **z Є** $\mathbf{R^{24x1}}$
  - The induced local field is taken as **z0 Є** $\mathbf{R^{24x1}}$
  - Applying the activation function, $\mathbf{z} = \varphi_0(\mathbf{z0}) = \tanh(\mathbf{w0}*\mathbf{X_j})$ where j = 1,2,...,n. and $\mathbf{X_j}$ **Є** $\mathbf{R^{300x2}}$
- The input-output relation for the network:
  - $\boldsymbol{f(X_j, w)} = \mathbf{y} = \mathbf{w1}*\tanh(\mathbf{w0}*\mathbf{X_j})+b$, where j = 1,2,...,n. and $\mathbf{X_j}$ **Є** $\mathbf{R^{300x2}}$**.**

xi) To update weights, we make use of the back propagation technique. In order to perform this technique we need to calculate the gradients with respect to weights and biases at each layer.
xii) Differentiating the MSE with respect to weights and biases at each layer using back propagation algorithm, we get:

$$\frac{\partial E}{\partial w1} = -Xi * \boldsymbol{\varphi}'(\mathbf{z_0}) * \boldsymbol{w_1}^{\boldsymbol{T}} * (-\frac{\partial E}{\partial y})$$

$$\frac{\partial E}{\partial b1} = -\boldsymbol{\varphi}'(\mathbf{z_0}) * \boldsymbol{w_1}^{\boldsymbol{T}} * \left(-\frac{\partial E}{\partial y}\right)$$

$$\frac{\partial E}{\partial w2} = -\boldsymbol{\varphi}(\mathbf{z_0}) * (-\frac{\partial E}{\partial y})$$

$$\frac{\partial E}{\partial b2} = (-\frac{\partial E}{\partial y})$$

The above equations can be simplified as

$$\frac{\partial E}{\partial w1} = -X_i * (\boldsymbol{di} - \boldsymbol{y}) * \boldsymbol{w_1}^{\boldsymbol{T}} * (\mathbf{1} - \boldsymbol{tan}^{\mathbf{2}}\,\mathbf{h}(\mathbf{z_0}))$$

$$\frac{\partial E}{\partial b1} = -(\boldsymbol{di} - \boldsymbol{y}) * \boldsymbol{w_1}^{\boldsymbol{T}} * (\mathbf{1} - \boldsymbol{tan}^{\mathbf{2}}\,\mathbf{h}(\mathbf{z_0}))$$

$$\frac{\partial E}{\partial w2} = -(\boldsymbol{di} - \boldsymbol{y}) * \mathbf{z_1}$$
$$\frac{\partial E}{\partial b2} = -(\boldsymbol{di} - \boldsymbol{y})$$

xiii) The update equation used for the weights and biases is:

$\mathbf{w_{new}}=\mathbf{w_{old}}$ - eta*gradient

xiv)The network is evaluated using the updated weights and biases.

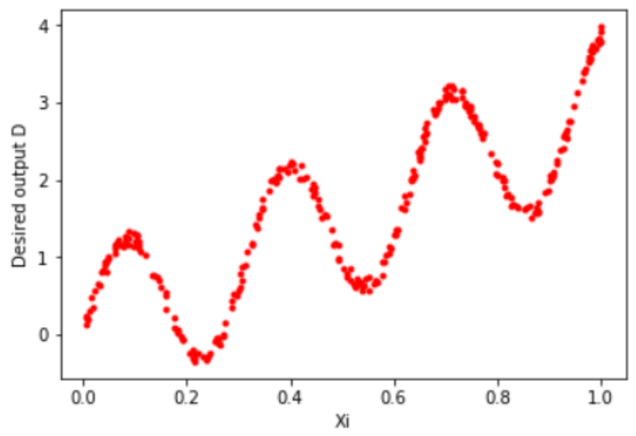
xv)The mean square error is computed for every epoch using the MSE function. A graph between number of epochs and MSE is plotted to the convergence characteristics. If the MSE at a particular epoch tends to increase or remain constant from its previous value, we reduce the value of learning rate by a smaller factor. In this case,

eta=eta - 0.01*eta

xvi) Finally, the graph between the inputs $\mathbf{X_i}$ and the output **y** is plotted to see how well the network fits with the desired output.

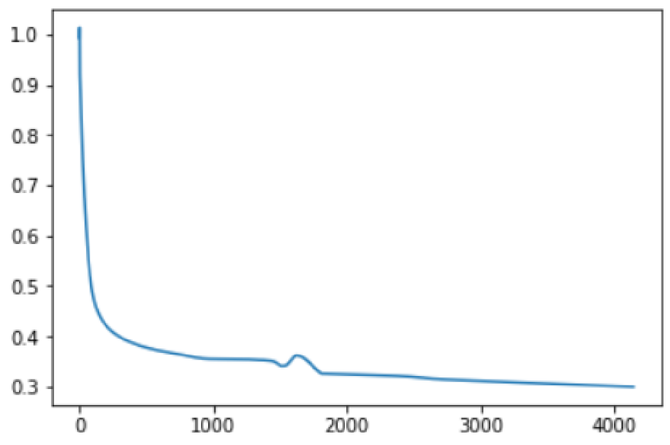
Source Code:

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

xi=np.random.uniform(0,1,300)
o=np.ones(300)
x=np.column_stack((xi,o))
v=np.random.uniform(-0.1,0.1,300)
D=[]
for i in range(300):
    d=math.sin(20*xi[i])+3*xi[i]+v[i]
    D.append(d)

plt.plot(xi,D,'r.')
plt.xlabel('Xi')
plt.ylabel('Desired output D')
plt.show
```



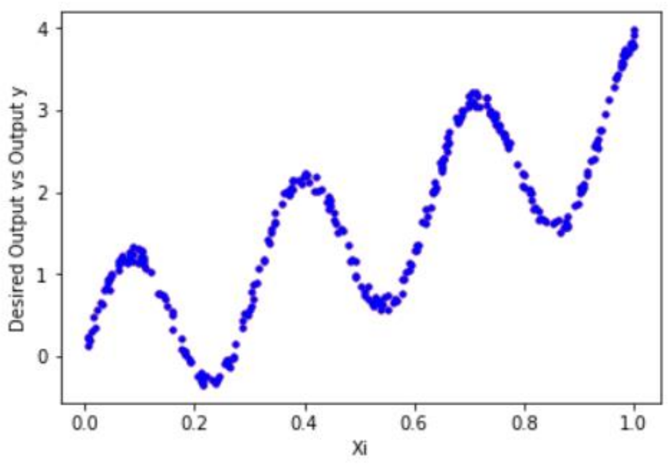
```
w0=np.random.randn(48).reshape(24,2)
w1=np.random.randn(24)
b=np.random.randn()
z0=np.zeros(24)
z=np.zeros(24)
z1=0
y=np.zeros(len(xi))
mse=np.zeros(300)
epoch=0
eta=0.1
MSE=[ ]
Epoch=[ ]

while(epoch<2 or abs(MSE[epoch-1]>=0.3)):

    for i in range(len(xi)):
        for j in range(24):
            for k in range(0,2):
                z0[j]=np.matmul(w0[j],x[i].reshape(2,1))
                z[j]=np.tanh(np.matmul(w0[j],x[i].reshape(2,1)))
                z1=np.matmul(z.reshape(1,24),w1)+b
                y[i]=z1
                g=-(D[i]-y[i])*w1.reshape(24,1)*(1-np.square(np.tanh(z0[j])))
                dg1=g*xi[i]
                db1=g
                dg2=(-(D[i]-y[i])*z).reshape(1,24)
                db2=-(D[i]-y[i])
                G=np.array([dg1,db1]).reshape(2,24)
                w0[j][k]=w0[j][k]-(eta*G[k][j])
                w1[j]=w1[j]-(eta*dg2[0][j])
                b=b-(eta*db2)


    epoch=epoch+1
```

```
     Epoch.append(epoch)
     for i1 in range(len(xi)):
        for j1 in range(24):
           for k1 in range(0,2):


              z[j1]=np.tanh(np.matmul(w0[j1],x[i1].reshape(2,1)))
              z1=np.matmul(z.reshape(1,24),w1)+b
              y[i1]=z1
              mse[i1]=(D[i1]-y[i1])*(D[i1]-y[i1])

     MSE.append(sum(mse)/300)
     if MSE[epoch-1]>=MSE[epoch-2]:
        eta-=0.01*eta
plt.plot(Epoch,MSE)
plt.show
```



Epoch

```
z=np.zeros(24)
z1=0
y=np.zeros(len(xi))
for i in range(len(xi)):
   for j in range(24):
      for k in range(0,2):
         z[j]=np.tanh(np.matmul(w0[j],x[i].reshape(2,1)))
         z1=np.matmul(z.reshape(1,24),w1)+b
         y[i]=z1

plt.plot(xi,D,'r.')
plt.plot(xi,y,'b.')
plt.xlabel('Xi')
plt.ylabel('Desired Output vs Output y')
plt.show
```


Desired Output vs Output y
Xi
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4