Name : Uddip Yalamanchili
USN : 1PE16CS170

PROGRAM : 4 - ANN

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)# featues (hrs slept/hrs studied)
y = np.array(([92], [86], [89]), dtype=float)#label marks obtained
X = X/np.amax(X,axis=0) # maximum of X array longitudinally used to normalize
y = y/100

#Sigmoid Function to transfer neuron activation during forward propagation
def sigmoid (x):
    return (1/(1 + np.exp(-x)))
#Derivative of Sigmoid Function to Calculate the derivative of an neuron output
def derivatives_sigmoid(x):
    return x * (1 - x)

#intialize network
epoch=7000 #Setting training iterations
lr=0.1 #Setting learning rate
inputlayer_neurons = 2 #number of features in data set
hiddenlayer_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer

#weight and bias initialization
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))

#Forward Propagation
for i in range(epoch):
        hinp1=np.dot(X,wh)  # this function returns a dot product of two arrays
        hinp=hinp1 + bh
        hlayer_act = sigmoid(hinp)
        outinp1=np.dot(hlayer_act,wout)
        outinp= outinp1+ bout
        output = sigmoid(outinp)

        #Backpropagation
        EO = y-output   # error at the output
        outgrad = derivatives_sigmoid(output)
        d_output = EO* outgrad

        #error at hidden layer
        EH = d_output.dot(wout.T)   #transpose
        hiddengrad = derivatives_sigmoid(hlayer_act)

        #how much hidden layer wts contributed to error
        d_hiddenlayer = EH * hiddengrad
        wout += hlayer_act.T.dot(d_output) *lr

        # dotproduct of nextlayererror and currentlayer
        bout += np.sum(d_output, axis=0,keepdims=True) *lr
        wh += X.T.dot(d_hiddenlayer) *lr

print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
mean_squared =0
mean_abs=0
for op,exp in zip(y,output):
        mean_squared += pow(op-exp,2)
        mean_abs +=abs(op-exp)
mean_squared = mean_squared/len(output)
mean_abs/=len(output)
print("mean squared error :",mean_squared)
print("mean absolute error :",mean_abs)
```

output:

```
uddipyalamanchili@pes:~/programs/ann_back$ python3 ann.py
Input:
[[0.66666667 1.        ]
```

```
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.89443182]
 [0.87830188]
 [0.89608743]]
mean squared error : [0.00034192]
mean absolute error : [0.0166525]
uddipyalamanchili@pes:~/programs/ann_back$
```