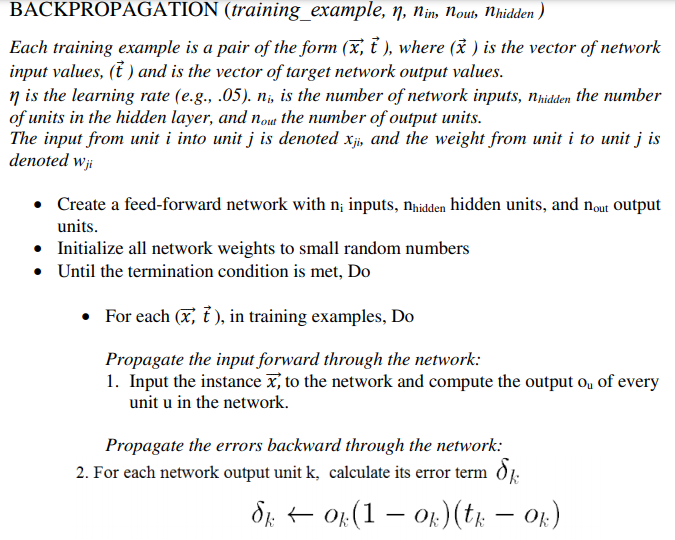
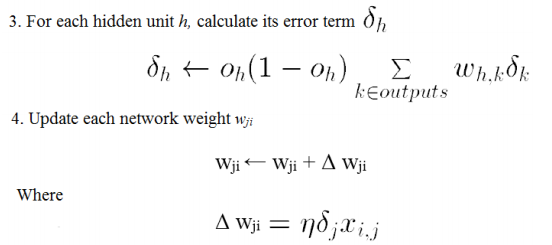
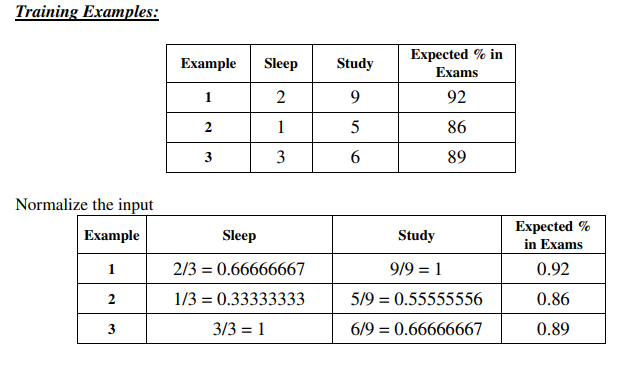
## WEEK 8:

## Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate datasets.







Program:

import numpy as np

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float) # two inputs [sleep,study]

y = np.array(([92], [86], [89]), dtype=float) # one output [Expected % in Exams]

X = X/np.amax(X,axis=0) # maximum of X array longitudinally

y = y/100

#Sigmoid Function

def sigmoid (x):

    return 1/(1 + np.exp(-x))

#Derivative of Sigmoid Function

def derivatives\_sigmoid(x):

    return x \* (1 - x)

#Variable initialization

epoch=5000  #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)) #weight of the link from input node to hidden node

bh=np.random.uniform(size=(1,hiddenlayer\_neurons)) # bias of the link from input node to hidden node

wout=np.random.uniform(size=(hiddenlayer\_neurons,output\_neurons)) #weight of the link from hidden node to output node

bout=np.random.uniform(size=(1,output\_neurons)) #bias of the link from hidden node to output node

#draws a random range of numbers uniformly of dim x\*y

for i in range(epoch):

  #Forward Propogation

    hinp1=np.dot(X,wh)

    hinp=hinp1 + bh

    hlayer\_act = sigmoid(hinp)

    outinp1=np.dot(hlayer\_act,wout)

    outinp= outinp1+ bout

    output = sigmoid(outinp)

#Backpropagation

    EO = y-output

    outgrad = derivatives\_sigmoid(output)

    d\_output = EO\* outgrad

    EH = d\_output.dot(wout.T)

#how much hidden layer weights contributed to error

    hiddengrad = derivatives\_sigmoid(hlayer\_act)

    d\_hiddenlayer = EH \* hiddengrad

# dotproduct of nextlayererror and currentlayerop

wout += hlayer\_act.T.dot(d\_output) \*lr

wh += X.T.dot(d\_hiddenlayer) \*lr

print("Input: \n" + str(X))

print("Actual Output: \n" + str(y))

print("Predicted Output: \n" ,output)

Input:

[[0.66666667 1. ]

[0.33333333 0.55555556]

[1. 0.66666667]]

Actual Output:

[[0.92]

[0.86]

[0.89]]

Predicted Output:

[[0.76912718]

[0.75738895]

[0.77014643]]

## Discussion Topics:

## 1. What are the four main steps in back propagation algorithm?

## Step – 1: Forward Propagation

## Step – 2: Calculate Error

## Step – 3: Back Propagation

## Step – 4: Updating weights (Reducing Error)

## 2. What are general limitation of back propagation rule?

## a) local minima problem b) slow convergence c) scaling

## 3. Define feed forward neural network.

It is a directed acyclic Graph which means that there are no feedback connections or loops in the network. It has an input layer, an output layer, and a hidden layer. In general, there can be multiple hidden layers.