**5.** **Build an Artificial Neural Network by implementing the Backpropagation**

**algorithm and test the same using appropriate data sets.**

THEORY:**BACK PROPAGATION**

* Backpropagation  is supervised learning algorithm , for training Neural Networks.
* Every node in Neural Network represent a Neuron, so we can say that Neural Network is a circuit of neurons,
* Neural Network consist an Input layer, an output layer and a hidden layer, let&#39;s see in diagram.

**What is the role of back Propagation?**

1. First of all,if I want to create a neural network, then I have to initialize some weights.
2. Now, whatever values i have selected for weights i do not know how much they are correct.
3. To check that the weight values that ​​I have selected are correct or incorrect I have to calculate the error of the model.
4. Suppose my model error occurred too much
5. Meaning my predicated output is very different from the actual output, so what shall I do? I will try to minimize the error.

**How does back propagation algorithm work?**

Suppose we have a neural network that has an input layer, a hidden layer and an

output layer

step1: First, we give random weights to the model.

step2: Forward propagation (normal neural network calculation)

step3: Calculate total error.

step4: Backward propagation (gradient descent), updating parameters (weights

and bias)

step5: Until the error is minimized (Predicted output to be approximately equal to

original output)

**PROCEDURE/PROGRAM:**

import numpy as np

X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)

y = np.array(([92], [86], [89]), dtype=float)

X = X/np.amax(X,axis=0)

y = y/100

**def** sigmoid (x):

    return 1/(1 + np.exp(-x))

**def** derivatives\_sigmoid(x):

    return x \* (1 - x)

epoch=7000

learning\_rate=0.1

inputlayer\_neurons = 2

hiddenlayer\_neurons = 3

output\_neurons = 1

wh=np.random.uniform(size=(inputlayer\_neurons,hiddenlayer\_neurons))

bh=np.random.uniform(size=(1,hiddenlayer\_neurons))

wo=np.random.uniform(size=(hiddenlayer\_neurons,output\_neurons))

bo=np.random.uniform(size=(1,output\_neurons))

for i in range(epoch):

    net\_h=np.dot(X,wh) + bh

    sigma\_h= sigmoid(net\_h)

    net\_o= np.dot(sigma\_h,wo)+ bo

    output = sigmoid(net\_o)

    deltaK = (y-output)\* derivatives\_sigmoid(output)

    deltaH =  deltaK.dot(wo.T) \* derivatives\_sigmoid(sigma\_h)

    wo = wo + sigma\_h.T.dot(deltaK) \*learning\_rate

    wh = wh + X.T.dot(deltaH) \*learning\_rate

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:

[[0.89423577]

[0.8822454 ]

[0.89371237]]