Ex No : 6 ANN BY BACK PROPAGATION ALGORITHM

Aim:

To implement multi layer artificial neural network using back propagation algorithm.

Equipments Required:

1. Hardware – PCs
2. Anaconda – Python 3.7 Installation / Moodle-Code Runner /Google Colab

Related Theory Concept:

Algorithm for ANN Backpropagation:

• Weight initialization: Set all weights and node thresholds to small random numbers. Note that the node threshold is the negative of the weight from the bias unit (whose activation level is fixed at 1).

• Calculation of Activation:

1.The activation level of an input is determined by the instance presented to the network.

2.The activation level oj of a hidden and output unit is determined. • Weight training:

3.Start at the output units and work backward to the hidden layer recursively and adjust weights.

4.The weight change is completed.

5.The error gradient is given by:

a. For the output units.

b. For the hidden units.

4.Repeat iterations until convergence in term of the selected error criterion. An iteration includes presenting an instance, calculating activation and modifying weights.

Algorithm:

1.Import packages

2.Defining Sigmoid Function for output

3.Derivative of Sigmoid Function

4.Initialize variables for training iterations and learning rate

5.Defining weight and biases for hidden and output layer

6.Updating Weights

Program:

```
/*
Program to implement ANN by back propagation algorithm.
Developed by  : S.SUDHAKARAN
RegisterNumber : 212219040155
*/

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
lr=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))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))


for i in range(epoch):
```

```
    hinp1=np.dot(X,wh)
hinp=hinp1+bh
hlayer_act=sigmoid(hinp)
outinp1=np.dot(hlayer_act,wout)
outinp=outinp1+bout
output=sigmoid(outinp)


EO=y-output
outgrad=derivatives_sigmoid(output)
d_output=EO*outgrad
EH=d_output.dot(wout.T)
hiddengrad=derivatives_sigmoid(hlayer_act)
d_hiddenlayer=EH*hiddengrad
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"+str(output))
```

Output:

```
Input:
[[0.66666667 1.         ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
[[0.75947941]
 [0.74376953]
 [0.75837003]]
```

Result:

Thus the python program successully implemented multi layer artificial neural network using back propagation algorithm.