**SOURCE CODE**

import numpy as np

def sigmoid(x):

return 1/(1+np.exp(-x))

A = np.array([[1.0,0.0,0.0],[0.0,1.0,0.0],[0.0,0.0,1.0]])

B = np.array([[1.0,0.0,0.0],[0.0,1.0,0.0],[0.0,0.0,1.0]])

#Random weights at first

input = np.array([[0.9],[0.1],[0.8]])

sz = 3

targetOutput = np.array([[0.72630335],[0.70859807],[0.77809706]])

#it should come out when you put the input in the model

for step in range(0,10001):

hiddenLayerInput = np.dot(A,input)

hiddenLayerOutput = hiddenLayerInput.copy()

for idx in range(0,sz):

hiddenLayerOutput[idx][0] = sigmoid(hiddenLayerOutput[idx][0])

outputLayerInput = np.dot(B,hiddenLayerOutput)

outputLayerOutput = outputLayerInput.copy()

for idx in range(0,sz):

outputLayerOutput[idx][0] = sigmoid(outputLayerOutput[idx][0])

#forward done

if step % 500 == 0:

print("After trying "+str(step)+" times forward and backward,The result is")

print(outputLayerOutput)

errorOutput = np.array([[1.0],[1.0],[1.0]]);

for idx in range(0,sz):

errorOutput[idx][0] = targetOutput[idx][0] - outputLayerOutput[idx][0]

sumColumnOfB = B.sum(axis=1)

errorHidden = np.dot(B.T,errorOutput)

for idx in range(0,sz):

errorHidden[idx][0] = errorHidden[idx][0]/sumColumnOfB[idx]

sumColumnOfA = A.sum(axis=1)

errorInput = np.dot(A.T,errorHidden)

for idx in range(0,sz):

errorInput[idx][0] = errorInput[idx][0]/sumColumnOfA[idx]

#error get done

newB = B.copy()

for row in range(0,sz):

for col in range(0,sz):

sigma = 0

for i in range(0,sz):

sigma = sigma + (B.T[i][col] \* outputLayerOutput[i][0])

newB[row][col] = B[row][col] - 0.1 \* (-errorHidden[row][0]\*sigmoid(sigma)\*(1-sigmoid(sigma))\*outputLayerOutput[row][0])

B = newB.copy()

#B update done

newA = A.copy()

for row in range(0,sz):

for col in range(0,sz):

sigma = 0

for i in range(0,sz):

sigma = sigma + (A.T[i][col] \* hiddenLayerOutput[i][0])

newA[row][col] = A[row][col] - 0.1 \* (-errorInput[row][0]\*sigmoid(sigma)\*(1-sigmoid(sigma))\*hiddenLayerOutput[row][0])

A = newA.copy()

#A update done backpropagation done

if step == 10000:

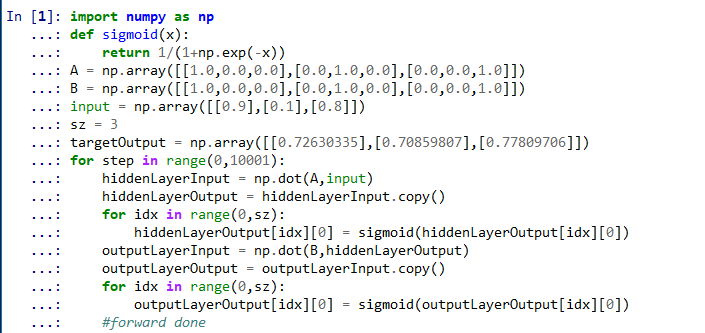
#final result show

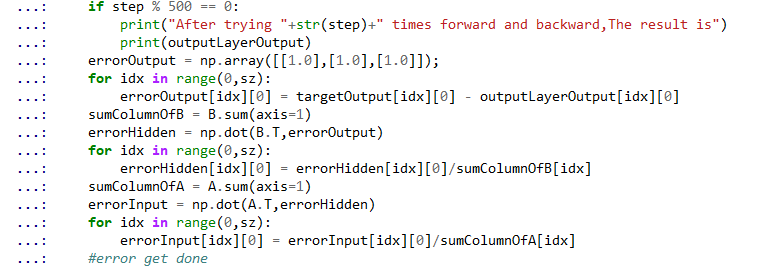
print("new A is")

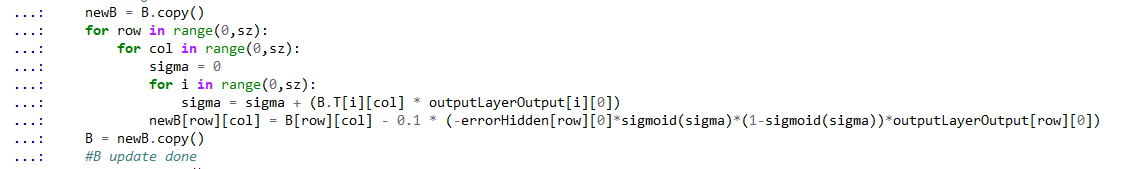
print(A)

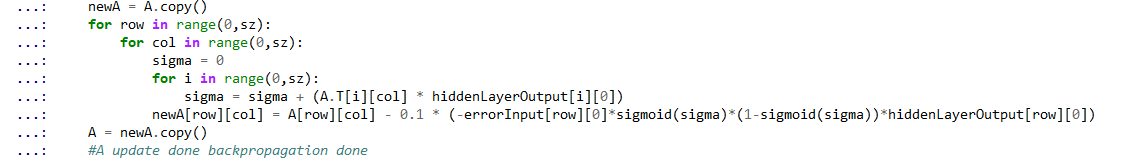
print("new B is")

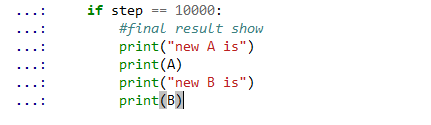
print(B)



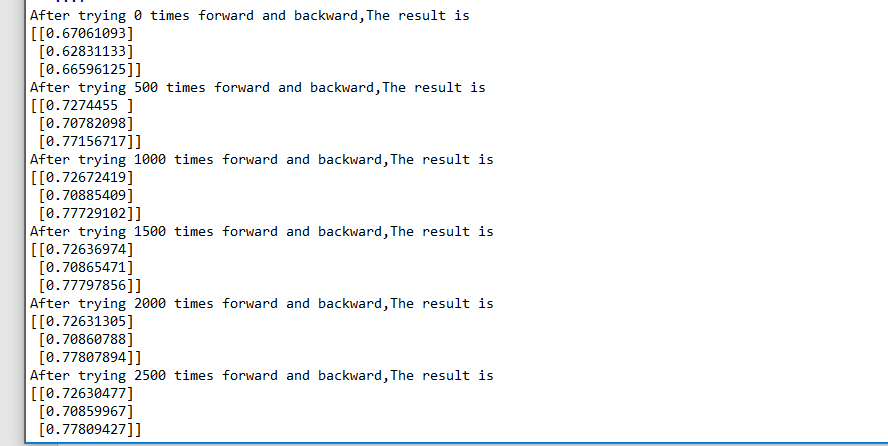




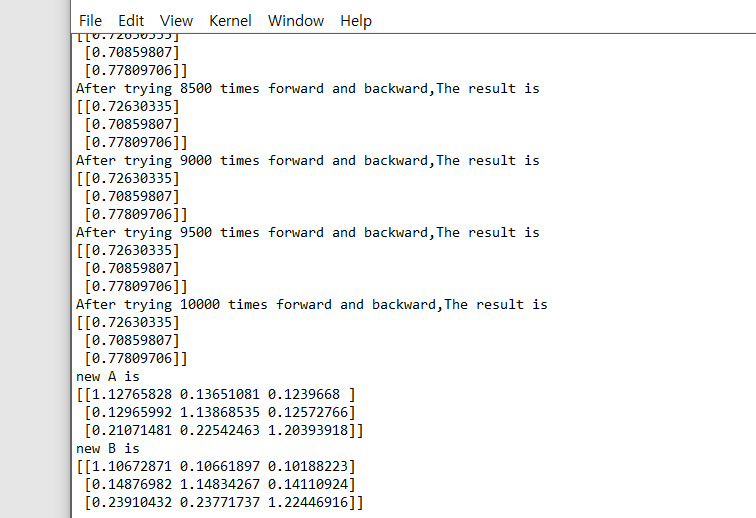




**RESULT**



At first, you can see that it is very different from the target.



After several forward and backpropagation runs, it can be confirmed that it is the same as the target output.