Neural Network

Assignment - 01

**Q. Implement Neural Network in Python/R.**

**Initial Values:**

**Target Outputs:**

**Activation Function:** Sigmoid Function

**Loss Function:** Mean Squared Error

**Answer.**

**Code : -**

import numpy as np

def sigmoid(x):

    return 1 / (1 + np.exp(-x))

def sigmoid\_derivative(x):

    return x \* (1 - x)

# Initial values

I = np.array([0.50, 0.10])

W1 = np.array([[0.15, 0.25], [0.20, 0.30]])

W2 = np.array([[0.35, 0.45], [0.40, 0.50]])

B1 = np.array([0.35, 0.35])

B2 = np.array([0.60, 0.60])

target\_outputs = np.array([0.01, 0.99])

# Forward propagation

h\_input = np.dot(I, W1.T) + B1

h\_output = sigmoid(h\_input)

o\_input = np.dot(h\_output, W2.T) + B2

network\_output = sigmoid(o\_input)

# Calculate mean squared error loss

loss = np.mean((target\_outputs - network\_output) \*\* 2)

# Backpropagation

# Calculate output layer errors and deltas

output\_error = target\_outputs - network\_output

output\_delta = output\_error \* sigmoid\_derivative(network\_output)

# Calculate hidden layer errors and deltas

hidden\_error = np.dot(output\_delta, W2)

hidden\_delta = hidden\_error \* sigmoid\_derivative(h\_output)

# Update weights and biases

learning\_rate = 0.5

W2 += learning\_rate \* np.outer(output\_delta, h\_output)

B2 += learning\_rate \* output\_delta

W1 += learning\_rate \* np.outer(hidden\_delta, I)

B1 += learning\_rate \* hidden\_delta

# Print the updated weights and biases

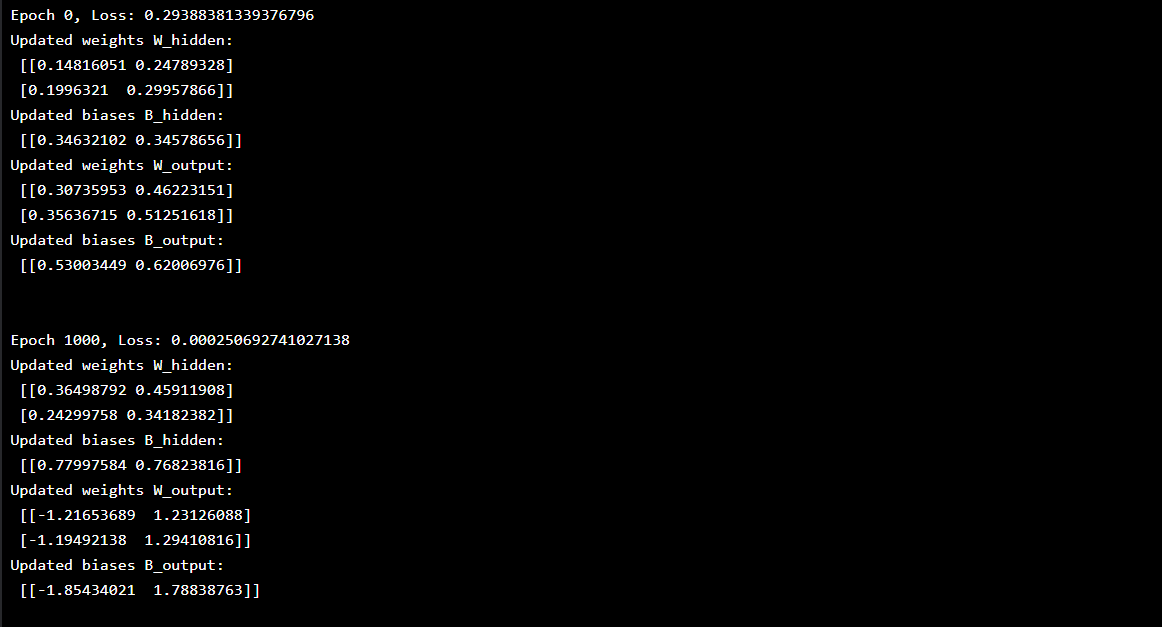
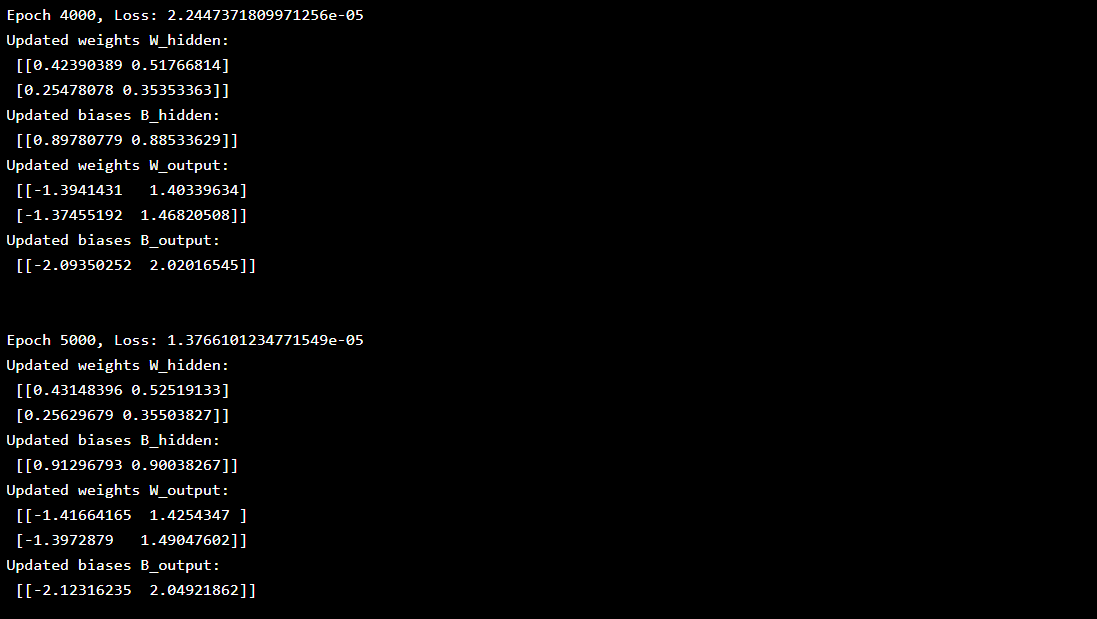
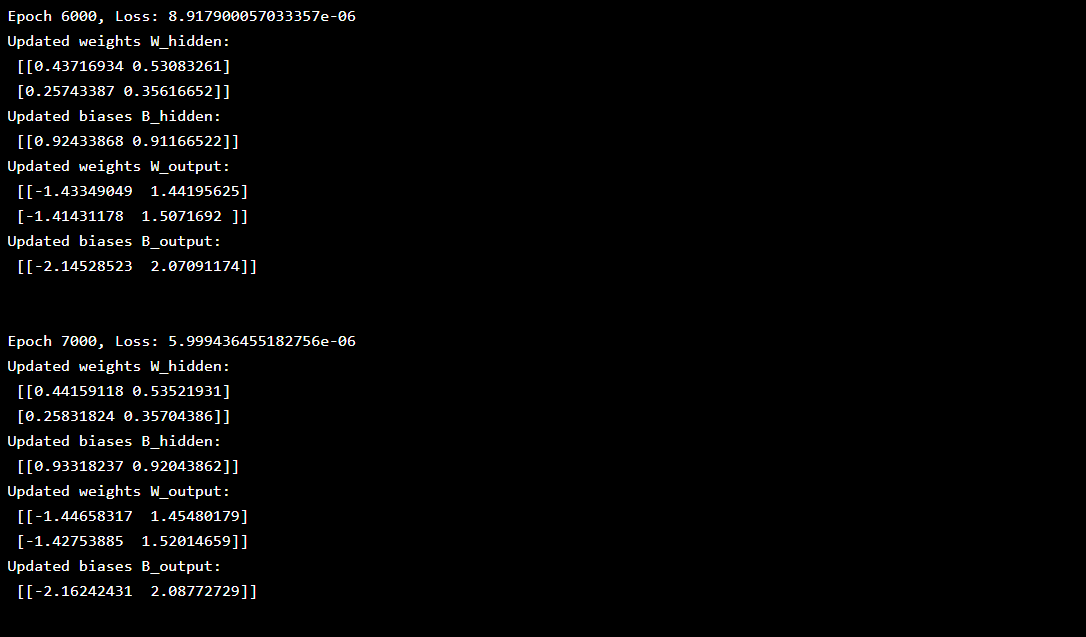
print("Updated weights W1:\n", W1)

print("Updated biases B1:\n", B1)

print("Updated weights W2:\n", W2)

print("Updated biases B2:\n", B2)

**Output : -**

** ** **** 