**Name: Gaurang Vaghela**

**Rollno: TEAD-22561**

**ANN (SL-II)**

**Practical 7  
Problem Statement:** Implement to show Back Propagation Network for XOR function with Binary Input and Output

**Code:**

import numpy as np

# Define the parameters of the network

input\_neuron = 2 # Number of input neurons

hidden\_neuron = 4 # Number of hidden neurons

output\_neuron = 1 # Number of output neurons

learning\_rate = 0.1

epochs = 10000

# Define the training data

X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

Y = np.array([[0], [1], [1], [0]])

# Weights from input to hidden layer

W1 = np.random.uniform(size=(input\_neuron, hidden\_neuron))

# Weights from hidden to output layer

W2 = np.random.uniform(size=(hidden\_neuron, output\_neuron))

# Bias for hidden layer

b1 = np.random.uniform(size=(1, hidden\_neuron))

# Bias for output layer

b2 = np.random.uniform(size=(1, output\_neuron))

# Define the sigmoid activation function and its derivative

def sigmoid(x):

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

def sigmoid\_derivative(x):

return x \* (1 - x)

# Train the network using backpropagation

for i in range(epochs):

# Forward pass

hidden\_layer\_input = np.dot(X, W1) + b1

hidden\_layer\_output = sigmoid(hidden\_layer\_input)

output\_layer\_input = np.dot(hidden\_layer\_output, W2) + b2

output\_layer\_output = sigmoid(output\_layer\_input)

# Backward pass

output\_error = Y - output\_layer\_output

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

hidden\_error = output\_delta.dot(W2.T)

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

# Update weights and biases

W2 += np.dot(hidden\_layer\_output.T, output\_delta) \* learning\_rate

b2 += np.sum(output\_delta, axis=0, keepdims=True) \* learning\_rate

W1 += np.dot(X.T, hidden\_delta) \* learning\_rate

b1 += np.sum(hidden\_delta, axis=0, keepdims=True) \* learning\_rate

# Test the network with some example inputs

x\_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

y\_test = np.array([[0], [1], [1], [0]])

hidden\_layer\_input = np.dot(x\_test, W1) + b1

hidden\_layer\_output = sigmoid(hidden\_layer\_input)

output\_layer\_input = np.dot(hidden\_layer\_output, W2) + b2

output\_layer\_output = sigmoid(output\_layer\_input)

print("Input:")

print(x\_test)

print("Output:")

print(output\_layer\_output)

print("Expected Output:")

print(y\_test)

**Output:**

