**Lab Assignment Group 8**

**Code:**

import numpy as np  
  
#Activation sigmoid  
def sigmoid(x):  
 return 1 / (1 + np.exp(-x))  
  
#Derivative of sigmoid  
def sigmoid\_derivative(x):  
 return x \* (1 - x)  
  
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  
y = np.array([[0], [1], [1], [0]])  
  
np.random.seed(42)  
  
input\_dim = 2  
hidden\_dim = 2  
output\_dim = 1  
  
weights\_input\_hidden = 2 \* np.random.random((input\_dim, hidden\_dim)) - 1  
weights\_hidden\_output = 2 \* np.random.random((hidden\_dim, output\_dim)) - 1  
  
biases\_hidden = np.zeros((1, hidden\_dim))  
biases\_output = np.zeros((1, output\_dim))  
  
learning\_rate = 0.1  
num\_epochs = 10000  
  
for epoch in range(num\_epochs):  
 # Forward propagation  
 hidden\_layer\_input = np.dot(X, weights\_input\_hidden) + biases\_hidden  
 hidden\_layer\_activation = sigmoid(hidden\_layer\_input)  
  
 output\_layer\_input = np.dot(hidden\_layer\_activation, weights\_hidden\_output) + biases\_output  
 output\_layer\_activation = sigmoid(output\_layer\_input)  
  
 # Backpropagation  
 error = y - output\_layer\_activation  
 output\_layer\_delta = error \* sigmoid\_derivative(output\_layer\_activation)  
  
 hidden\_layer\_error = output\_layer\_delta.dot(weights\_hidden\_output.T)  
 hidden\_layer\_delta = hidden\_layer\_error \* sigmoid\_derivative(hidden\_layer\_activation)  
  
 weights\_hidden\_output += hidden\_layer\_activation.T.dot(output\_layer\_delta) \* learning\_rate  
 biases\_output += np.sum(output\_layer\_delta, axis=0, keepdims=True) \* learning\_rate  
  
 weights\_input\_hidden += X.T.dot(hidden\_layer\_delta) \* learning\_rate  
 biases\_hidden += np.sum(hidden\_layer\_delta, axis=0, keepdims=True) \* learning\_rate  
  
test\_input = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])  
hidden\_layer\_output = sigmoid(np.dot(test\_input, weights\_input\_hidden) + biases\_hidden)  
predicted\_output = sigmoid(np.dot(hidden\_layer\_output, weights\_hidden\_output) + biases\_output)  
  
print("Predicted Output:")  
print(predicted\_output)

Predicted Output:  
[[0.0961913 ]  
 [0.89393519]  
 [0.89410922]  
 [0.08557778]]

test\_input

array([[0, 0],  
 [0, 1],  
 [1, 0],  
 [1, 1]])

y

array([[0],  
 [1],  
 [1],  
 [0]])

final\_preditions = [1 if predict >= 0.5 else 0 for predict in predicted\_output]

**Output:**

final\_preditions

[0, 1, 1, 0]