import numpy as np

# Activation function: Sigmoid and its derivative

def sigmoid(x):

    return 1 / (1 + np.exp(-x))

# Inputs

x1 = 1

x2 = 3

inputs = np.array([x1, x2])

# Target output

target\_output = 0.95

# Learning rate (eta)

eta = 0.1

# Initial weights based on the diagram

# weights between input layer and hidden layer

w11 = 0.8

w12 = 0.5

w21 = 0.1

w22 = 0.2

# weights between hidden layer and output layer

w31 = 0.2

w32 = 0.7

# Organize the weights into numpy arrays for simplicity

hidden\_weights = np.array([[w11, w12], [w21, w22]])  # shape (2, 2)

output\_weights = np.array([w31, w32])  # shape (2,)

# Number of iterations for FFBP

iterations = 1

# Feed-forward and backpropagation (FFBP)

for i in range(iterations):

    # ---- Feed-forward step ----

    # Input layer -> Hidden layer

    activity\_hidden = np.dot(inputs, hidden\_weights)  # weighted sum for hidden layer

    activation\_hidden = sigmoid(activity\_hidden)  # apply sigmoid activation for hidden layer

    print(f"Question 1 + 2. This is Activation of the Hidden Layer: {activation\_hidden}")

    # Hidden layer -> Output layer

    activity\_output = np.dot(activation\_hidden, output\_weights)  # weighted sum for output layer

    activation\_output = sigmoid(activity\_output)  # apply sigmoid activation for output layer

    print(f"Question 3. This is the Activation of the Output Layer: {activation\_output}")

    # Calculate error at the output

    error\_output = activation\_output - target\_output

    # ---- Backpropagation step ----

    # Calculate delta for output layer

    delta\_output = error\_output \* activation\_output \* (1 - activation\_output)

    # Update weights between hidden and output layers

    output\_weights -= eta \* delta\_output \* activation\_output

    print(f"Question 4 + 5. Here are output weights: {output\_weights}")

    # Calculate delta for hidden layer

    delta\_hidden = delta\_output \* output\_weights \* activation\_hidden \* (1 - activation\_hidden)

    # Update weights between input and hidden layers

    hidden\_weights -= eta \* np.outer(inputs, delta\_hidden)

    print(f"Question 6. Here are hidden weights: {hidden\_weights}")

    print(f"This is Question 6: {hidden\_weights[0, 1] - w12}")