import numpy as np

# McCulloch-Pitts Neuron

def mcculloch\_pitts\_neuron(inputs, weights, threshold=0.5):

    # Calculate the weighted sum

    weighted\_sum = np.dot(inputs, weights)

    # Apply threshold logic

    output = 1 if weighted\_sum > threshold else 0

    return output

# Training the McCulloch-Pitts Neuron for the OR problem

def train\_mcculloch\_pitts\_neuron(inputs, target\_output, weights, threshold=0.5, learning\_rate=0.2, epochs=4):

    for epoch in range(epochs):

        print(f"Epoch {epoch + 1}")

        for input\_values, target in zip(inputs, target\_output):

            # Get the neuron output

            output = mcculloch\_pitts\_neuron(input\_values, weights, threshold)

            # Update the weights based on the error and learning rate

            error = target - output

            weights += learning\_rate \* error \* input\_values

            # Display the updated weights and output

            print("Input:", input\_values, "Target Output:", target, "Predicted Output:", output, "Updated Weights:", weights)

            print("-----------------------")

# Inputs for the OR problem

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

# Target output for OR problem

target\_output = np.array([0, 1, 1, 1])

# Initial weights

initial\_weights = np.array([0.1, 0.3])

# Training the McCulloch-Pitts Neuron for 4 epochs with a threshold of 0.5 and learning rate of 0.2

train\_mcculloch\_pitts\_neuron(inputs, target\_output, initial\_weights, threshold=0.5, learning\_rate=0.2, epochs=4)