import numpy as np

# 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)

# Define the neural network class

class NeuralNetwork:

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

# Initialize weights with random values

self.weights\_input\_hidden = np.random.uniform(-1, 1, (input\_size, hidden\_size))

self.weights\_hidden\_output = np.random.uniform(-1, 1, (hidden\_size, output\_size))

def forward(self, inputs):

# Forward propagation

self.hidden\_input = np.dot(inputs, self.weights\_input\_hidden)

self.hidden\_output = sigmoid(self.hidden\_input)

self.output\_input = np.dot(self.hidden\_output, self.weights\_hidden\_output)

self.predicted\_output = sigmoid(self.output\_input)

return self.predicted\_output

def backward(self, inputs, target, learning\_rate):

# Backpropagation

error = target - self.predicted\_output

delta\_output = error \* sigmoid\_derivative(self.predicted\_output)

error\_hidden = delta\_output.dot(self.weights\_hidden\_output.T)

delta\_hidden = error\_hidden \* sigmoid\_derivative(self.hidden\_output)

# Update weights

self.weights\_hidden\_output += np.outer(self.hidden\_output, delta\_output) \* learning\_rate

self.weights\_input\_hidden += np.outer(inputs, delta\_hidden) \* learning\_rate

def train(self, training\_data, targets, epochs, learning\_rate):

for epoch in range(epochs):

for i in range(len(training\_data)):

inputs = training\_data[i]

target = targets[i]

self.forward(inputs)

self.backward(inputs, target, learning\_rate)

def predict(self, inputs):

return self.forward(inputs)

# Define XOR dataset

training\_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])

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

# Create and train the neural network

input\_size = 2

hidden\_size = 4

output\_size = 1

learning\_rate = 0.1

epochs = 10000

nn = NeuralNetwork(input\_size, hidden\_size, output\_size)

nn.train(training\_data, targets, epochs, learning\_rate)

# Test the trained network

for i in range(len(training\_data)):

inputs = training\_data[i]

prediction = nn.predict(inputs)

print(f"Input: {inputs}, Predicted Output: {prediction}")