import numpy as np

class Perceptron:

    def \_\_init\_\_(self, weights, desired\_output, bias, eta):

        self.weights = np.array(weights)

        self.bias = bias

        self.desired\_output = desired\_output

        self.eta = eta

        self.delta\_weights = np.zeros(len(weights))

        self.delta\_bias = 0.0

        self.activity = 0.0

        self.activation = 0.0

        self.delta = 0.0

    # Calculate activity

    def calc\_activity(self, input):

        input = np.array(input)

        self.activity = self.bias + np.dot(self.weights, input)

    # Sigmoid activation function

    def calc\_activation(self):

        self.activation = 1 / (1 + np.exp(-self.activity))

    # Calculate delta and set delta weights and bias

    def set\_delta\_weights(self, input):

        input = np.array(input)

        activation\_derivative = self.activation \* (1 - self.activation)  # Derivative of sigmoid

        self.delta = (self.activation - self.desired\_output) \* activation\_derivative

        self.delta\_weights = self.eta \* self.delta \* input

        self.delta\_bias = self.eta \* self.delta

    # Update weights and bias using delta

    def update\_weights(self):

        self.weights = self.weights - self.delta\_weights

        self.bias = self.bias - self.delta\_bias

    # Train the Perceptron for a certain number of iterations

    def train(self, input, iterations):

        for i in range(iterations):

            # Calculate activity and activation. Set delta weights & bias. Update weights and bias. Print each iteration.

            self.calc\_activity(input)

            self.calc\_activation()

            self.set\_delta\_weights(input)

            self.update\_weights()

            print(f"Iteration {i+1}: Weights = {self.weights}, Bias = {self.bias}, Activation = {self.activation}")

# Question 1

weights = [0.24, 0.88]

bias = 0.0

eta = 5.0

desired\_output = 0.95

iterations = 75

# Set Perceptron

perceptron = Perceptron(weights, desired\_output, bias, eta) # type: ignore

# Set Inputs and Calculate Acitivity; Calculate Activation

inputs = [0.8, 0.9]

perceptron.calc\_activity(inputs)

perceptron.calc\_activation()

print(perceptron.activation)

# Question 2

perceptron.train(inputs, iterations)

# Question 3

weights = [0.24, 0.88]

bias = 0.0

eta = 5.0

desired\_output = 0.15

iterations = 30

# Set Perceptron

perceptron = Perceptron(weights, desired\_output, bias, eta) # type: ignore

# Set Inputs and Calculate Acitivity; Calculate Activation

inputs = [0.8, 0.9]

perceptron.train(inputs, iterations)