# Assignment 5: Write a program to demonstrate the perceptron learning law with its decision region

# using python. Give the output in graphical form

# Name:

# Class: TE-AIML

# Roll No:

import numpy as np

import matplotlib.pyplot as plt

class Perceptron:

    def \_\_init\_\_(self, num\_inputs):

        self.weights = np.zeros(num\_inputs)

        self.bias = 0

    def predict(self, input):

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

        return 1 if activation >= 0 else -1

    def train(self, inputs, labels, learning\_rate=0.1, num\_epochs=10):

        for epoch in range(num\_epochs):

            for input, label in zip(inputs, labels):

                prediction = self.predict(input)

                if prediction != label:

                    self.weights += learning\_rate \* label \* input

                    self.bias += learning\_rate \* label

    def plot\_decision\_region(self, inputs, labels):

        x\_min, x\_max = inputs[:, 0].min() - 0.5, inputs[:, 0].max() + 0.5

        y\_min, y\_max = inputs[:, 1].min() - 0.5, inputs[:, 1].max() + 0.5

        xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.02),

                             np.arange(y\_min, y\_max, 0.02))

        Z = np.array([self.predict([x1, x2]) for x1, x2 in np.c\_[xx.ravel(), yy.ravel()]])

        Z = Z.reshape(xx.shape)

        plt.contourf(xx, yy, Z, cmap=plt.cm.RdBu, alpha=0.6)

        plt.scatter(inputs[:, 0], inputs[:, 1], c=labels, cmap=plt.cm.RdBu, edgecolors='black')

# create some example data

inputs = np.array([[1, 2], [2, 1], [-1, -2], [-2, -1]])

labels = np.array([1, 1, -1, -1])

# create a perceptron with 2 input nodes

perceptron = Perceptron(2)

# train the perceptron on the data

perceptron.train(inputs, labels)

# plot the decision region of the perceptron

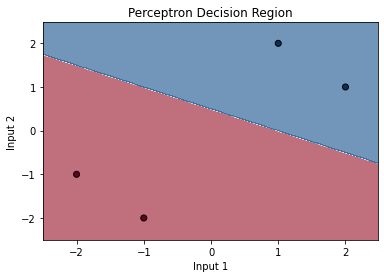
perceptron.plot\_decision\_region(inputs, labels)

plt.title("Perceptron Decision Region")

plt.xlabel("Input 1")

plt.ylabel("Input 2")

plt.show()



# The code defines a class named Perceptron, which is a linear binary classifier used for supervised learning.

# The Perceptron class has three methods:

# \_\_init\_\_: initializes the weights and bias to zero.

# predict: computes the activation of the perceptron for an input and returns +1 or -1 based on the sign of the activation.

# train: updates the weights and bias based on the training data to minimize the error.

# plot\_decision\_region: plots the decision region of the perceptron on a two-dimensional input space.

# The plot\_decision\_region method takes the input data and the corresponding labels as input arguments and uses them to plot

# the decision boundary of the perceptron.

# The input data is a 2D numpy array with two columns representing the features, and the labels are a 1D numpy array

# with the corresponding class labels (+1 or -1).

# The decision boundary is plotted using the contourf function from the matplotlib library, which creates a filled contour

# plot of the predicted class labels on a grid of input values.

# The scatter function is used to plot the training data points with different colors based on their class labels.

# The plt.show() function displays the plot.