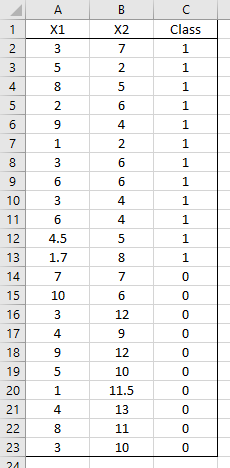
***HW1: Viết chương trình sử dụng thuật toán Perceptron Learning Algorithm***

1. Lý thuyết:

Perceptron là một thuật toán Classification cho trường hợp đơn giản nhất: chỉ có 2 class và cũng chỉ hoạt động được trong một trường hợp rất cụ thể.Bài toán Perceptron được phát biểu như sau: Cho 2 class được gán nhãn, hãy tìm một đường thẳng sao cho toàn bộ các điểm thuộc class 1 nằm về 1 phía, toàn bộ các điểm thuộc class 2 nằm về phía còn lại của đường thẳng đó. Với giả định rằng tồn tại một đường thẳng như thế.

Giả sử ta có 2 tập dữ liệu là X1 và X2 đã được gán nhãn theo thứ tự là “green” và “red”.



1. Mô phỏng huấn luyện Perceptron bằng Python:

* Đọc dữ liệu từ file CSV và biểu diễn trên đồ thị:

def Read\_Data():

    df = pd.read\_csv('D:\HCMUT\AI\HW\PLA.csv')

    plt.plot(df['X1'][1:12], df['X2'][1:12], 'go', color='green')

    plt.plot(df['X1'][12:], df['X2'][12:], 'gs', color='red')

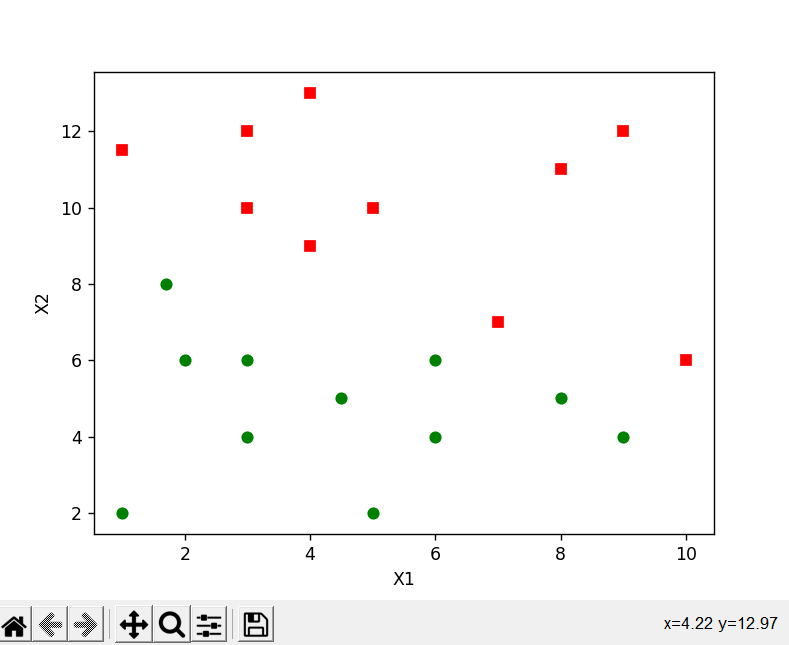
    plt.xlabel('X1') *# label x*

    plt.ylabel('X2') *# label y*

    plt.show()

    return df

* + Hàm Read\_Data(): trả về df để lưu dữ liệu.
  + Kết qua sau khi plot:



* Tiến hành Perceptron Learning Algorithm:
  + Khởi tạo mảng X =[], thêm vào dữ liệu các cột X1, X2 và X có dạng: [ [1, X1, X2] ]

def PLA(data):

    X = []

    for i in range(0, 22):

        X.append([1, data['X1'][i], data['X2'][i]])

* + Tạo W = [0, 0, 0]

W = np.zeros(len(X[0]))

* + Chọn miền hội tụ là 400. Và tiến hành giải thuật PLA

for i in range(400):

        for i in range(len(X)):

            if data['Class'][i] == 1 and (W @ np.array(X[i])) < 0:

                W += X[i]

            if data['Class'][i] == 0 and (W @ np.array(X[i])) >= 0:

                W -= X[i]

print("W =", W)

    sampleX = np.linspace(min(data['X1']) - 3, max(data['X1']) + 3, int((max(data['X1']) - min(data['X1']))\*100))

    sampleY = (-W[0] - W[1]\*sampleX)/W[2]

    plt.plot(sampleX, sampleY, color='black')

    plt.plot(data['X1'][1:12], data['X2'][1:12], 'go', color='green')

    plt.plot(data['X1'][12:], data['X2'][12:], 'gs', color='red')

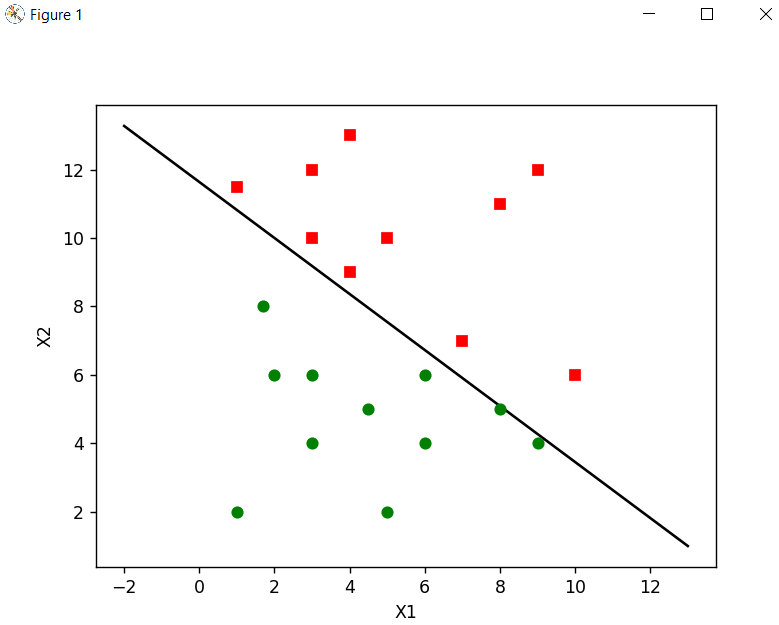
    plt.xlabel('X1') *# label x*

    plt.ylabel('X2') *# label y*

    plt.show()

    return W

* + Sau đó in ra màn hình giá trị của W và vẽ đường phân tách 2 class:
  + Kết quả sau khi chạy:



Và W = [128 -9 -11]

* Dùng Perceptron để phân loại các điểm dữ liệu:
  + Tạo các điểm chưa phân loại bằng hàm random và được biểu diễn bằng hình tam giác:

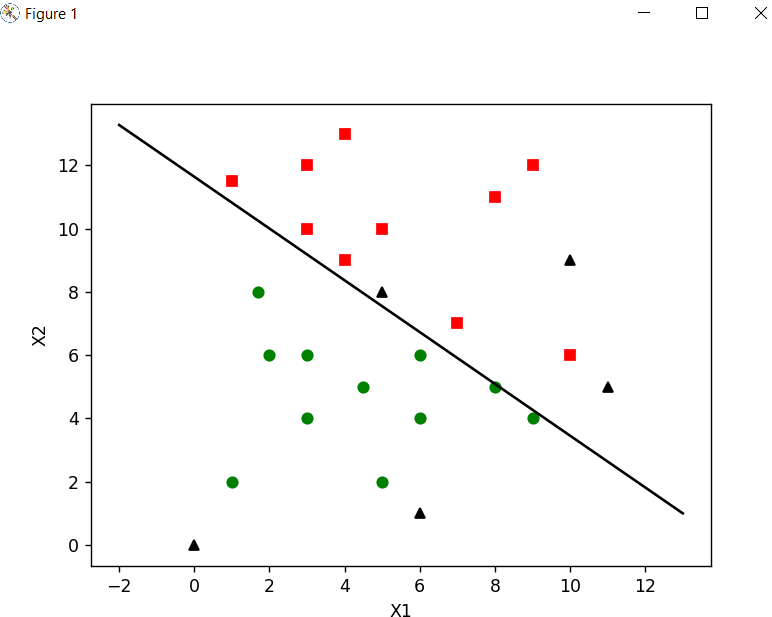
x\_test = []

    for i in range(5):

        x\_test.append([1, random.randint(0, 11), random.randint(0, 11)])

        plt.plot(x\_test[i][1], x\_test[i][2], 'b^', color='black')

* + Kết quả:



* + Hàm phân loại:

for i in range(len(x\_test)):

        if W @ np.array(x\_test[i]) < 0:

            plt.plot(x\_test[i][1], x\_test[i][2], 'b^', color='red')

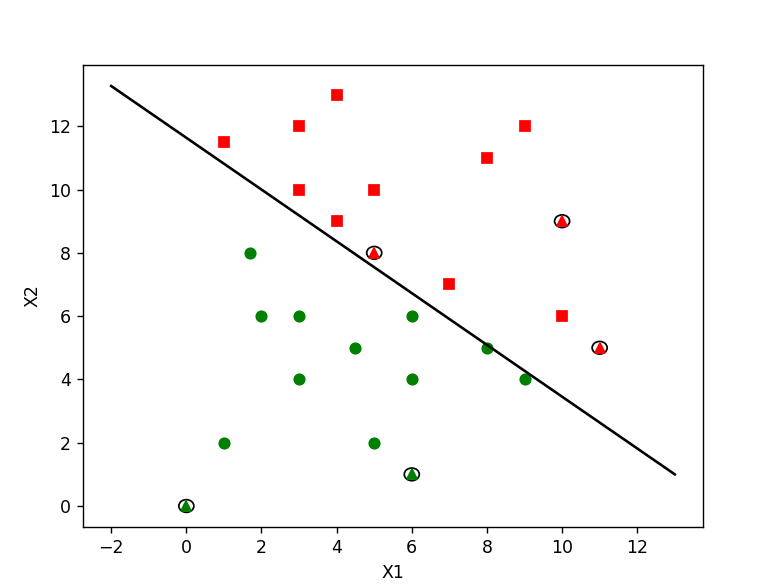
        else:

            plt.plot(x\_test[i][1], x\_test[i][2], 'b^', color='green')

        circle1 = plt.Circle((x\_test[i][1], x\_test[i][2]), .2, color='black', fill = False)

        plt.gcf().gca().add\_artist(circle1)

* + Kết quả sau khi phân loại:



* Full code:

from os import X\_OK

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import random

*#Input*

def Read\_Data():

    df = pd.read\_csv('D:\HCMUT\AI\HW\PLA.csv')

    plt.plot(df['X1'][1:12], df['X2'][1:12], 'go', color='green')

    plt.plot(df['X1'][12:], df['X2'][12:], 'gs', color='red')

    plt.xlabel('X1') *# label x*

    plt.ylabel('X2') *# label y*

    plt.show()

    return df

*#Perceptron Learning Algorithm*

def PLA(data):

    X = []

    for i in range(0, 22):

        X.append([1, data['X1'][i], data['X2'][i]])

    W = np.zeros(len(X[0]))

    for i in range(400):

        for i in range(len(X)):

            if data['Class'][i] == 1 and (W @ np.array(X[i])) < 0:

                W += X[i]

            if data['Class'][i] == 0 and (W @ np.array(X[i])) >= 0:

                W -= X[i]

    print("W =", W)

    sampleX = np.linspace(min(data['X1']) - 3, max(data['X1']) + 3, int((max(data['X1']) - min(data['X1']))\*100))

    sampleY = (-W[0] - W[1]\*sampleX)/W[2]

    plt.plot(sampleX, sampleY, color='black')

    plt.plot(data['X1'][1:12], data['X2'][1:12], 'go', color='green')

    plt.plot(data['X1'][12:], data['X2'][12:], 'gs', color='red')

    plt.xlabel('X1') *# label x*

    plt.ylabel('X2') *# label y*

    plt.show()

    return W

*#Main*

if \_\_name\_\_ == "\_\_main\_\_":

    data = Read\_Data()

    W = PLA(data)

    x\_test = []

    for i in range(5):

        x\_test.append([1, random.randint(0, 11), random.randint(0, 11)])

        plt.plot(x\_test[i][1], x\_test[i][2], 'b^', color='black')

    plt.plot(data['X1'][1:12], data['X2'][1:12], 'go', color='green')

    plt.plot(data['X1'][12:], data['X2'][12:], 'gs', color='red')

    sampleX = np.linspace(min(data['X1']) - 3, max(data['X1']) + 3, int((max(data['X1']) - min(data['X1']))\*100))

    sampleY = (-W[0] - W[1]\*sampleX)/W[2]

    plt.plot(sampleX, sampleY, color='black')

    plt.xlabel('X1') *# label x*

    plt.ylabel('X2') *# label y*

    plt.show()

    for i in range(len(x\_test)):

        if W @ np.array(x\_test[i]) < 0:

            plt.plot(x\_test[i][1], x\_test[i][2], 'b^', color='red')

        else:

            plt.plot(x\_test[i][1], x\_test[i][2], 'b^', color='green')

        circle1 = plt.Circle((x\_test[i][1], x\_test[i][2]), .2, color='black', fill = False)

        plt.gcf().gca().add\_artist(circle1)

    plt.plot(data['X1'][1:12], data['X2'][1:12], 'go', color='green')

    plt.plot(data['X1'][12:], data['X2'][12:], 'gs', color='red')

    sampleX = np.linspace(min(data['X1']) - 3, max(data['X1']) + 3, int((max(data['X1']) - min(data['X1']))\*100))

    sampleY = (-W[0] - W[1]\*sampleX)/W[2]

    plt.plot(sampleX, sampleY, color='black')

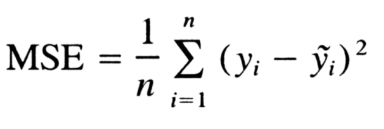
    plt.xlabel('X1') *# label x*

    plt.ylabel('X2') *# label y*

    plt.show()

***HW2: Compare mean squared error and mean absolute error:***

* MSE:
  + Là số liệu phổ biến nhất được sử dụng cho các bài toán Regression. Về cơ bản, nó tìm bình phương sai số trung bình giữa các giá trị được dự đoán và thực tế. MSE là thước đo chất lượng của một công cụ ước tính – nó luôn không âm và các giá trị càng gần 0 càng tốt.
  + Khi vẽ biểu đồ của toàn bộ dữ liệu, MSE cho biết mức độ gần của đường hồi quy với một tập hợp các điểm. Để giảm thiểu MSE, mô hình có thể chính xác hơn, có nghĩa là mô hình gần với dữ liệu thực tế hơn. MSE càng thấp thì dự báo càng tốt.
  + Công thức của MSE là:



(Với yi là biến quan sát ; là giá trị dự đoán)

* MAE: sai số trung bình tuyệt đối, đo độ lớn trung bình các lỗi trong 1 tập hợp các dự đoán mà không cần xem xét hướng của chúng.
  + Công thức của MAE là:



(Với yi là giá trị dự đoán ; xi là giá trị thực)

* Chính vì vậy, sử dụng MSE nếu các giá trị ngoại lệ ( các giá trị có sai số cao hơn hẳn các mẫu khác) là quan trọng. Lý do chính là trong MSE có bình phương sai số, các mẫu ngoại lệ sẽ được chú ý nhiều hơn. Và nếu các ngoại lệ không quan trọng thì sử dụng MAE.