Homework 2 : Problem 3.5

Consider two-classes that follow multivariate normal distribution with:

- **mean1 = [1, 1] and mean2 = [0,0]**
- **var = 0.2**

Produce 50 vectors from each class. Use these vectors to design a linear classifier using the perceptron algorithm. After convergence, draw the corresponding decision line.

Note:

To guarantee linear separability of the classes disregard:

- vectors with x1+x2 < 1 for the [1, 1] class.
- vectors with x1+x2 > 1 for the [0, 0] class.

```
# -*- coding: utf-8 -*-
"""
Homework 2 - Problem 3.4

Perceptron Algorithm
Using Punishment and Reward Method
@author: Nazneen Kotwal
"""
import os
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

def decbound(wht):
    """
    To plot the decision boundary
    Parameters
    ----------
    arg1 : wht
         Weights associated witht the feature vector
         Format: [w0,w1,w2]
    """
    x = np.linspace(-2, 2.5, 50)
    y = -(wht[0] + wht[1]*x)/wht[2]
    plt.plot(x, y)

def percepalgo(trainingSet,wht,rho):
    """
    Perceptron Algorithm for Punishment and Reward Method
    ----------
    arg1 : trainingSet
        Feature vector
    arg2 : wht
        Weights associated witht the feature vector
        Format [w0,w1,w2]
    arg3 : rho
        Controls the rate of convergence of the algorithm
    """
    if ((np.dot(wht,trainingSet[0:3]) <= 0) and (trainingSet[-1] == 0)):
        mul =  trainingSet[0:3]
        wht = wht + (rho * mul)
    elif ((np.dot(wht,trainingSet[0:3]) >= 0) and (trainingSet[-1] == 1)):
        mul =  trainingSet[0:3]
```

```
            wht = wht - (rho * mul)
        else:
            wht = wht
        return(wht)

    def main():

        path = "C:/Users/nazne/OneDrive/Documents/1_ECE 759 Pattern Recognition/Homework/Homework2/proble
        print ("The current working directory is", os.getcwd())
        os.chdir(path)

        # Creating a Data Set for Training and Testing
        mean1 = (1, 1)
        cov = ((0.2, 0),(0, 0.2))
        sample1 = np.random.multivariate_normal(mean1, cov, 50)
        mean2 = (0, 0)
        sample2 = np.random.multivariate_normal(mean2, cov, 50)
        sample11=[]
        sample22=[]
        for x in range(50):
            if((sample1[x][0] + sample1[x][1]) > 1):
                sample11.append(sample1[x])
            if((sample2[x][0]+sample2[x][1]) < 1):
                sample22.append(sample2[x])
        print(sample11)
        print(sample22)
        z1 = np.zeros((len(sample11),1))
        z2 = np.ones((len(sample22),1))
        z3 = np.ones((len(sample11),1))
        sample11 = np.hstack((z3,sample11))
        sample22 = np.hstack((z2,sample22))
        a = np.hstack((sample11,z1))
        b = np.hstack((sample22,z2))
        trainingSet = np.concatenate((a,b),axis=0)
        print(trainingSet)
        x1 = trainingSet[:,1]
        x2 = trainingSet[:,2]
        cmap_bold = ListedColormap(['#FF0000', '#00FF00'])
        plt.scatter(x1, x2, c=trainingSet[:,3], cmap=cmap_bold,
                    edgecolor='k', s=20)

        print('No. of Train set: ' + repr(len(trainingSet)))
        iniwht = [2,2,2]
        rho = 0.1
        count = 0
        while True:
            flag = 0
            count += 1
            for x in range(len(trainingSet)):
                wht = percepalgo(trainingSet[x],iniwht,rho)
                print(wht)
                print(iniwht)
                if (np.array_equal(iniwht,wht)==False):
                    print(np.array_equal(iniwht,wht))
                    flag = 1
                    iniwht = wht
                    print(flag)
            if (flag == 0) or (count == 100):
                break
        decbound(wht)
        print('The Update Weight Vector on convergence w = [w0 w1 w2] is %s: ' %(wht))
        print('Number of Loops untill Covergence: %d'  % count)

    if __name__ == '__main__':
        main()
```

Note:

- The Update Weight Vector on convergence w = [w0 w1 w2] is **[-1.3 1.443 1.333]**
- Number of Loops untill Covergence: **4**