```
#########################################
#    Harikrishna Prabhu              ##
#    3333077042                    ##
#########################################
import numpy as np
import matplotlib.pyplot as plt
import math
import plotDecisionBound as pB
import random

op=input("Enter the data points with which you want to run the Perceptron
Algorithm:\n(1)Synthetic 1\n(2)Synthetic 2\n(3)Synthetic 3\n")
print("\n\n")
print("For Synthetic:"),;print(op),;print('\n')
error_RATE_train=0
error_RATE_test=0
mis_class=0
#Data input
if(op==1):
        given_Data_TRAIN=np.genfromtxt("synthetic1_train.csv", delimiter=',')
        feature_train=np.copy(given_Data_TRAIN)
        class_Label_train=given_Data_TRAIN[:,2]
        length_train=len(class_Label_train)
        ####
        given_Data_TEST=np.genfromtxt("synthetic1_test.csv", delimiter=',')
        feature_test=np.copy(given_Data_TEST)
        class_Label_test=given_Data_TEST[:,2]
        length_test=len(class_Label_test)
        #
        augmented=np.ones(length_train)
        for i in range(0,length_train):
                if(class_Label_train[i]!=1):
                        given_Data_TRAIN[i,0]=-given_Data_TRAIN[i,0]
                        given_Data_TRAIN[i,1]=-given_Data_TRAIN[i,1]
                        augmented[i]=-1 #augmented
if(op==2):
        given_Data_TRAIN=np.genfromtxt("synthetic2_train.csv", delimiter=',')
        feature_train=np.copy(given_Data_TRAIN)
        class_Label_train=given_Data_TRAIN[:,2]
        length_train=len(class_Label_train)
        ####
        given_Data_TEST=np.genfromtxt("synthetic2_test.csv", delimiter=',')
        feature_test=np.copy(given_Data_TEST)
        class_Label_test=given_Data_TEST[:,2]
```

```
    length_test=len(class_Label_test)
    #
    augmented=np.ones(length_train)
    for i in range(0,length_train):
        if(class_Label_train[i]!=1):
            given_Data_TRAIN[i,0]=-given_Data_TRAIN[i,0]
            given_Data_TRAIN[i,1]=-given_Data_TRAIN[i,1]
            augmented[i]=-1 #augmented
    #print(augmented)
if(op==3):

    given_Data_TRAIN=np.genfromtxt("feature_train.csv", delimiter=',')
    class_Label_train=np.genfromtxt("label_train.csv", delimiter=',')
    feature_train=np.copy(given_Data_TRAIN)
    length_train=len(class_Label_train)
    augmented=np.ones(length_train)
    ####
    given_Data_TEST=np.genfromtxt("feature_test.csv", delimiter=',')
    class_Label_test=np.genfromtxt("label_test.csv", delimiter=',')
    feature_test=np.copy(given_Data_TEST)
    length_test=len(class_Label_test)
    #
    for i in range(0,length_train):
        if(class_Label_train[i]!=1):
            given_Data_TRAIN[i,0]=-given_Data_TRAIN[i,0]
            given_Data_TRAIN[i,1]=-given_Data_TRAIN[i,1]
            augmented[i]=-1 #augmented
#####
epoch_length_train=1000
sample_length_train=length_train #max=num of data points
sample_data=np.zeros(3)
done=False
weight_vector=np.array([0.1, 0.1, 0.1])
kount=0
fount=0
w0=0
w1=0
w2=0
ww=0
min_w=np.zeros(3)
min_val=11110
slope=0
intercept=0
```

```
w_final=np.zeros(3)
#print("\tweight_vector:"),;print(weight_vector)
for i in range(0,epoch_length_train):
        #print("#"),;print(i)
        myList=random.sample(xrange(0,length_train),sample_length_train)
        a=weight_vector
        #print(myList)
        for j in range(0,sample_length_train):
                sample_data[0]=given_Data_TRAIN[myList[j],0]
                sample_data[1]=given_Data_TRAIN[myList[j],1]
                sample_data[2]=augmented[j]
                #
                w0=sample_data[0]*weight_vector[0]
                w1=sample_data[1]*weight_vector[1]
                w2=augmented[j]*weight_vector[2]
                ww=w0+w1+w2

        #print("\tweight_vector:"),;print(weight_vector),;print("\t\tWW:"),;print(ww),;print("\t\
tData Points:"),;print(sample_data)
                if(ww<=0):

        weight_vector=([sample_data[0]+weight_vector[0],sample_data[1]+weight_vector[1],sa
mple_data[2]+weight_vector[2]])

        #print("\tweight_vector:"),;print(weight_vector),;print("\t\tWW:"),;print(ww),;print("\t\
tData Points:"),;print(sample_data)
                        mis_class+=1
                        t=mis_class
        if(a[0] == weight_vector[0] and a[1] == weight_vector[1] and a[2] == weight_vector[2]):
                w_final=np.copy(weight_vector)
                done = True
                break
        mis_class=0
        if(t<min_val):
                #print(t)
                min_w=np.copy(weight_vector) #np.array([w0,w1,w2])
                min_val=t

if done:
        error_RATE_train=0

        print("Final:"),;print(w_final)
        for j in range(0,length_train):
                w0=given_Data_TRAIN[j,0]*w_final[0]
                w1=given_Data_TRAIN[j,1]*w_final[1]
                w2=w_final[2]
                ww=w0+w1+w2
                if(ww<0):
                        error_RATE_train+=1
        for i in range(0,length_test):
                error_RATE_test=0
                ww0=given_Data_TEST[i,0]*w_final[0]
                ww1=given_Data_TEST[i,1]*w_final[1]
                ww2=w_final[2]
                www=ww0+ww1+ww2
                if(www<0):
                        error_RATE_test+=1
else:
        error_RATE_train=0
        error_RATE_test=0
        print("Min_w/Final_w:"),;print(min_w)
        w_final=np.copy(min_w)
        for j in range(0,length_train):

                w0=given_Data_TRAIN[j,0]*w_final[0]
                w1=given_Data_TRAIN[j,1]*w_final[1]
                w2=w_final[2]
                ww=w0+w1+w2
                if(ww<0):
                        error_RATE_train+=1

        for i in range(0,length_test):
                error_RATE_test=0
                ww0=given_Data_TEST[i,0]*w_final[0]
                ww1=given_Data_TEST[i,1]*w_final[1]
                ww2=w_final[2]
                www=ww0+ww1+ww2
                if(www<0):
                        error_RATE_test+=1
print("Error Rate for Training data:"),;print(error_RATE_train)
print("Error Rate for Test data:"),;print(error_RATE_test)

plt.plot(feature_train[class_Label_train==1,0],feature_train[class_Label_train==1,1],'rs')
plt.plot(feature_train[class_Label_train==2,0],feature_train[class_Label_train==2,1],'g^')
plt.autoscale(enable=True)
w0=weight_vector[0]
w1=weight_vector[1]
w2=weight_vector[2]
```

```
slope=-1*(w0/w1)
intercept=-1*(w2/w1)
xc=np.arange(-10,10,0.1)
plt.plot(xc,xc*slope + intercept,'b.')
plt.legend(('Class 1','Class 2','Decision Boundary' ), loc=2)


plt.show()
```