Designing a Minimum Distance to Class Mean Classifier

Anika Tanzim
Department of Computer Science of Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204072@aust.edu

***Abstract*—The objective of the this experiment is to design a minimum distance class mean classifier and observe the result of the classifier. By implementing this, it will be able to know the basics of pattern recognition. This is a simple linear classifier. The model is trained with the train dataset at first and then it classifies the test dataset. It gives good accuracy mostly but in scattered data , it is not able to do the classification perfectly.**

***Index Terms*—Linear Discriminant, Class Mean Classifier, Decision Boundary.**

I. Introduction

A classifier sorts data into classes or categories based on some set of rules. A Minimum Distance to Class Mean Classifier is an algorithm that sorts data into classes based on the distances between the data and the mean of the classes.

For example, if there are some sample of data already classified into two classes, the class means are Y_1 and Y_2 respectively. Now the distance between the test data (x) and the means of the classes will be $D_1(x)$ and $D_2(x)$.

Then the decision rule is,

1) If $D_1(x) < D_2(x)$, then x is in the class 1
2) If $D_1(x) > D_2(x)$, then x is in the class 2

Here, Euclidean distance is used to calculate the distance. If they are equal then the data is on the Decision Boundary.

Minimum distance to mean classifier is a linear classifier. It does not give a good result for scattered data. It only gives a good result when the sample points of one class is far from the other.

II. Experimental Design / Methodology

A. *Plotting all sample points (train data) from both classes*

A train dataset is given. All the points are classified into two classes - Train Class 1 and Train Class 2. Using Matplotlib, all the points from Train class 1 are plotted using blue dot marker and the points from train class 2 are plotted using green square marker in figure 1.

B. *Using a minimum distance classifier with respect to 'class mean', classifying and plotting the test data point and the class means*

The Linear Discriminant function i.e. used here is,

$$g_i(X) = X^T\overline{Y}_i - \frac{1}{2}\overline{Y}_i^T\overline{Y}_i$$

X - the feature vector and $\overline{Y}_i$ - the mean vector of class i. T denotes the transpose of the vector. The function can also be written like this,

$$g_i(X) = \overline{Y}_{ix}X + \overline{Y}_{iy}Y - \frac{1}{2}(\overline{Y}_{ix}\overline{Y}_{ix} + \overline{Y}_{iy}\overline{Y}_{iy})$$

if $X^T = [X \;\; Y]$ and $\overline{Y}_i^T = [\overline{Y}_{ix} \;\; \overline{Y}_{iy}]$

So, the decision rule for the given two classes is,

1) If $g_1(X) > g_2(X)$, then X is in the class 1
2) If $g_1(X) < g_2(X)$, then X is in the class 2

So,the mean vectors for each classes are calculated and plotted shown in figure 2. Then the discriminant functions for each test points are calculated and the class for each test point is determined using the decision rule. The test data are plotted in figure 2. The same class colors are used but with different marker.

C. *Drawing the decision boundary between the two classes*

Decision boundary between the two classes is to be drawn now. The decision boundary should be in a form of y = mx + c. Again, we know that all points on the decision boundary will be at equal distance from Class 1 mean and Class 2 mean. So,

$$g_1(X) = g_2(X)$$
$$\text{or, } g_1(X) - g_2(X) = 0$$

After substituting,

$$(\overline{Y}_1 - \overline{Y}_2)X^T - \frac{1}{2}(\overline{Y}_1^T\overline{Y}_1 - \overline{Y}_2^T\overline{Y}_2) = 0$$

Simplifying the values we get the y = mx + c form ,

$$Y = \frac{(\overline{Y}_{2x} - \overline{Y}_{1x})X + \frac{1}{2}(\overline{Y}_{1x}\overline{Y}_{1x} + \overline{Y}_{1y}\overline{Y}_{1y} - \overline{Y}_{2x}\overline{Y}_{2x} - \overline{Y}_{2y}\overline{Y}_{2y})}{\overline{Y}_{1y} - \overline{Y}_{2y}}$$

By plotting some x in range of minimum and maximum value of train data, where step size is 1, in this equation, the decision boundary is plotted in the figure 4 as a red line.

D. *Finding the accuracy*

Accuracy is the number of correctly classified points among the total number of points.

$$Accuracy = \frac{no.\ of\ correctly\ classified\ points}{Total\ no.\ of\ points} \quad (1)$$

III. Result Analysis

We have experimented the given dataset for result analysis. In the dataset, we were given 12 data in train dataset which were already classified into Class 1 and 2. Class means

were calculated from the train data. Then the test data were classified depending on the class means. There were 7 data in the test dataset with their actual class number in which they should belong to. After classiying the test data, we have obtained accuracy of 85.714%

The results of the four tasks are given below:

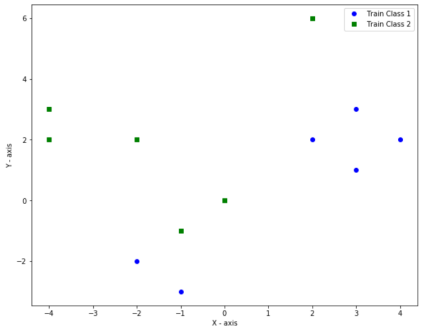

Fig. 1. Sample points of train dataset (Task 1 Output)

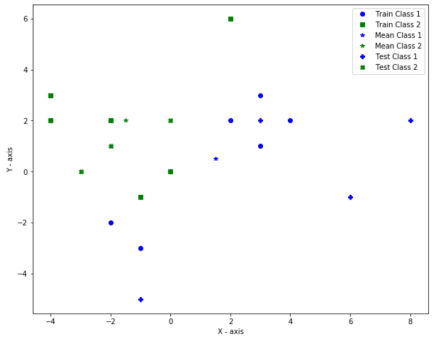

Fig. 2. Classified test dataset points (Task 2 Output)

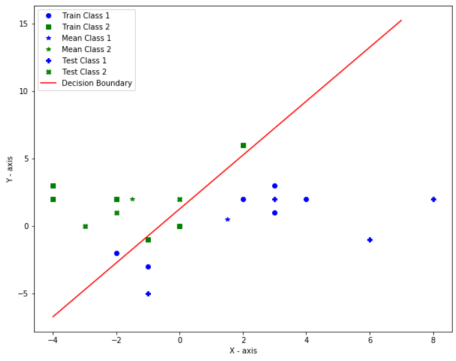

Fig. 3. Decision boundary between the two classes (Task 3 Output)

TABLE I
TABLE OF THE MISCLASSIFIED DATA

X	Y	*Actual Class*	*Predicted Class*
0	2	1	2

Based on the given data, one data is not correctly classified from the test dataset. So, the accuracy obtained 85.714%

IV. Conclusion

The minimum distance to class mean classify is a simple linear classifier. So, the decision boundary is linear. Thus the chances to misclassify by this classifier is high to some extent.

V. Algorithm Implementation / Code

```
import matplotlib.pyplot as pit

#1. plotting train data
#train_data
trx1=[]
try1=[]
trx2=[]
try2=[]
with open('train.txt','r') as f:
    for line in f:
        x,y,c = line.split(" ")
        if(int(c) ==1):
            trx1.append(int(x))
            try1.append(int(y))
        else:
            trx2.append(int(x))
            try2.append(int(y))


fig_1 = pit.figure(1, figsize=(10,8))
pit.plot(trx1, try1,'ob', label='Train Class 1')
pit.plot(trx2,try2,'sg', label='Train Class 2')
pit.xlabel('X - axis')
pit.ylabel('Y - axis')
pit.legend()


#2. classifying test data and plotting test data and mean value

meanX1=sum(trx1)/len(trx1)
meanY1=sum(try1)/len(try1)
meanX2=sum(trx2)/len(trx2)
meanY2=sum(try2)/len(try2)

#plotting mean of train data
pit.plot(meanX1, meanY1,'*b', label='Mean Class 1')
pit.plot(meanX2, meanY2,'*g', label='Mean Class 2')
pit.legend()


def g_x(x,y,meanX,meanY):
    return (meanX*x + meanY*y) - (0.5*(meanX*meanX +meanY*meanY))

#test data
testX=[]
testY=[]
testC=[]

with open('test.txt','r') as f:
    for line in f:
        x,y,c = line.split(" ")
        testX.append(int(x))
        testY.append(int(y))
        testC.append(int(c))

teX1=[]
teY1=[]
teX2=[]
teY2=[]
count=0
res= [-1]*len(testC)

for i in range(len(testX)):
    g1_x = g_x(testX[i],testY[i],meanX1,meanY1)
    g2_x = g_x(testX[i],testY[i],meanX2,meanY2)

    if g1_x > g2_x:
        res[i] = 1 #result class for test data
        teX1.append(testX[i])
        teY1.append(testY[i])
    else:
        res[i] = 2
        teX2.append(testX[i])
        teY2.append(testY[i])

    if res[i] ==testC[i]:
        count=count+1

#plotting test data according to our result
pit.plot(teX1, teY1,'Pb', label='Test Class 1')
pit.plot(teX2,teY2,'Xg', label='Test Class 2')
pit.legend()

#3. decision boundary
def db(x):
    y = ((meanX2 - meanX1) * x + 0.5* (meanX1*meanX1 + meanY1* meanY1
                - meanX2*meanX2 - meanY2*meanY2)) /(meanY1 - meanY2)
    return y

x=[]
y=[]
for i in range(-4,8):
    x.append(i)
    y.append(db(i))

print(x,y)

#plotting decision boundary
pit.plot(x, y,'r',label='Decision Boundary')
pit.legend()

#4.accuracy
accuracy = (count/len(testC))*100
print("Accuracy= " + str(accuracy) + "%")
```