Minimum Distance to Class Mean Classier

Rakib Hossain Rifat
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204099@aust.edu

***Abstract*—In this experiment I will plot and classify given data set using Minimum Distance to Class Mean Classier and calculate the accuracy of this classifier.**

***Index Terms*—Mean classifier, supervised learning.**

I. Introduction

Minimum distance to class mean classier is used to classify unknown data points using mean of given classes. For a given unknown data points, it will use this equation given bellow and classify which class it belongs.

$$g_i(x) = X_i^T \bar{Y} - \frac{1}{2}\bar{Y^T}\bar{Y_i}$$

Here X is the feature input vector and Y is the mean of an individual class.

II. Methodology

In this experiment I need to perform several tasks.

A. Plotting Training Data of Both Classes

for this I need to plot training data of both class from "train.txt", and samples belongs to same class should have same marker and color.

B. Classify Data points of Test Data set

for this I need to classify data points of "test.txt" using class mean classifier using this equation given bellow.

$$g_i(x) = X_i^T \bar{Y} - \frac{1}{2}\bar{Y^T}\bar{Y_i}$$

C. Drawing Decision Boundary

for this I need to draw a decision boundary between two class, and calculate the points of decision boundary.

$$g_1(x) = \bar{w_1}^T X - \frac{1}{2}\bar{w_1}^T\bar{w_1}$$

$$g_1(x) = \bar{w_2}^T X - \frac{1}{2}\bar{w_2}^T\bar{w_2}$$

$$g_1(x) = g_2(x)$$

$$(\bar{w_1}^T - \bar{w_2}^T)X = \frac{1}{2}(\bar{w_1}^T\bar{w_1} - \bar{w_2}^T\bar{w_2})$$

Let,

$$C_1 = \bar{w_1}^T$$

$$C_2 = -\bar{w_2}^T$$

$$Constant = \frac{1}{2}(\bar{w_1}^T\bar{w_1} - \bar{w_2}^T\bar{w_2})$$

So,

$$X_2 = \frac{C_1X_1 + Constant}{C_2}$$

D. Calculating Accuracy

for this I need to calculate accuracy using the equation given bellow .

$$\frac{Matched}{Total} * 100\%$$

III. Result Analysis

A. Plotting Training Data of Both Classes

Here i have plotted data of both class with different marker and color.
Class 1= (2, 2), (3, 1), (3, 3), (-1, -3), (4, 2), (-2, -2)
class 2= (-4, 3), (2, 6), (0, 0), (-2, 2), (-1, -1), (-4, 2)

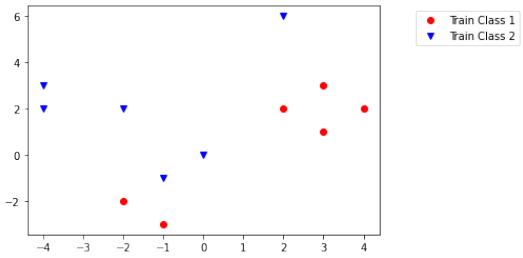

Fig. 1. Training Data of Class One and Class Two

B. Classify Data points of Test Data set

here i have classified test data set and plotted data points with means of each class with different color and marker

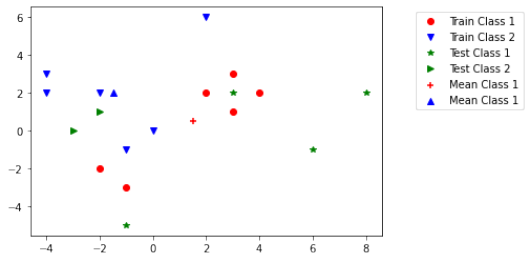

Fig. 2. Test Data of Class One and Class Two with Mean

C. Drawing Decision Boundary

Here I draw The decision boundary between two class, and calculate the points of decision boundary.

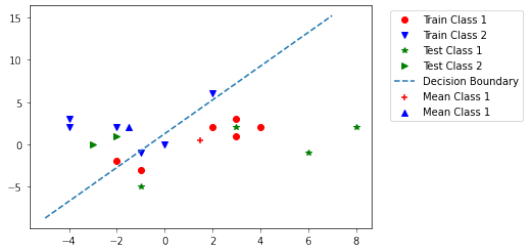

Fig. 3. Test and Training data with Decision Boundary

D. Calculating Accuracy

Here i have calculated the accuracy of this classifier on this data set and the accuracy was 85.71%

IV. Conclusion

from the result it can be said that this classifier is good in classifying data points. Test Data set consists six set of points and five of them was classified correctly. And accuracy of classifier is 85.71 %

V. Code

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Fig. 4. Importing All Libraries

Variables

```
[2]  omega_one=[]
     omega_two=[]
     test_class_one=[]
     test_class_two=[]
     matched=0
```

Fig. 5. Declaring All Variables

```
[3]  train_df=pd.read_csv("/content/train.txt",sep=" " ,names=["x1","x2","Y"])
     test_df=pd.read_csv("/content/test.txt",sep=" " ,names=["x1","x2","Y"])
```

Fig. 6. Path Setup For Data Set

```
def class_separation(df):
  for i in range(0,len(df)):
    li=[]
    if df['Y'][i]==1:
      li=df['x1'][i],df['x2'][i]
      omega_one.append(li)
    elif df['Y'][i]==2:
      li=df['x1'][i],df['x2'][i]
      omega_two.append(li)
```

Fig. 7. Training Data Class Separation

```
def weight_cal(omega):
  temp_li=[]
  temp_li2=[]
  for i in range(0,len(omega)):
    temp_li.append(omega[i][0])
    temp_li2.append(omega[i][1])

  w1_omega=sum(temp_li)/len(temp_li)
  w2_omega=sum(temp_li2)/len(temp_li2)

  w_f=np.array([w1_omega,w2_omega])
  return w_f
```

Fig. 8. Mean Calculation

```
[6]  def belongs(x):
       temp_1=np.dot(np.transpose(w1_f),x)-(.5)*(np.dot(np.transpose(w1_f),w1_f))
       temp_2=np.dot(np.transpose(w2_f),x)-(.5)*(np.dot(np.transpose(w2_f),w2_f))
       if temp_1>temp_2:
         return 1
       else:
         return 2
```

Fig. 9. Test Data Class Calculation

```
[7]  def line_points(min_f,max_f,cons):
       li=[]
       li2=[]
       for i in range(min_f,max_f):

         i_f=np.array(i)
         y=-1*((np.dot(cof1,i_f)+cons)/cof2)
         li.append(y)
         li2.append(i)
         i=i+1


       return li2,li
```

Fig. 10. Decision Boundary Line Points Calculation

```
class_separation(train_df)
w1_f=weight_cal(omega_one)
w2_f=weight_cal(omega_two)
```

Fig. 11. Class Separation And Weight Calculation

```
for i in test_df.index:
      li=[]
      x=np.array([test_df['x1'][i],test_df['x2'][i]])
      li=[test_df['x1'][i],test_df['x2'][i]]
      temp_f=belongs(x)
      if temp_f==test_df['Y'][i]:
        print("matched")
        matched=matched+1
        if test_df['Y'][i]==1:
         test_class_one.append(li)
        elif test_df['Y'][i]==2:
          test_class_two.append(li)

      else:
        print("not matched")
```

Fig. 12. Test Data Verification

```
[10] max_train_x1= max(train_df['x1'])
     max_train_x2= max(train_df['x2'])
     max_test_x1= max(test_df['x1'])
     max_test_x2= max(test_df['x2'])
     min_train_x1= min(train_df['x1'])
     min_train_x2= min(train_df['x2'])
     min_test_x1= min(test_df['x1'])
     min_test_x2= min(test_df['x2'])

     max_f=max(max_train_x1,max_train_x2,max_test_x1,max_test_x2)
     min_f=min(min_train_x1,min_train_x2,min_test_x1,min_test_x2)

     cons=-.5*(np.dot(np.transpose(w1_f),w1_f)-np.dot(np.transpose(w2_f),w2_f))
     w=np.transpose(w1_f)-np.transpose(w2_f)
     cof1=w[0]
     cof2=w[1]
```

Fig. 13. Maximum And Minimum Point And Constant Calculation

```
[12] line_x,line_y=line_points(min_f,max_f,cons)
```

Fig. 14. Decision Boundary Line Points Calculation

```
plt.plot(*zip(*omega_one),'ro',label="Train Class 1")
plt.plot(*zip(*omega_two),"bv",label="Train Class 2")
plt.legend(loc="upper center",bbox_to_anchor=(1.25, 1))
plt.show()
```

Fig. 15. Train data Plotting

```
[18] plt.plot(*zip(*omega_one),'ro',label="Train Class 1")
     plt.plot(*zip(*omega_two),"bv",label="Train Class 2")
     plt.scatter(w1_f[0],w1_f[1],marker='+',c='red',label="Mean Class 1")
     plt.scatter(w2_f[0],w2_f[1],marker='^',c='blue',label="Mean Class 1")
     plt.plot(*zip(*test_class_one),'g*',label="Test Class 1")
     plt.plot(*zip(*test_class_two),"g>",label="Test Class 2")
     plt.legend(loc="upper center",bbox_to_anchor=(1.25, 1))
     plt.show()
```

Fig. 16. Test Data Plotting With Mean

```
[ ]  plt.plot(*zip(*omega_one),'ro',label="Train Class 1")
     plt.plot(*zip(*omega_two),"bv",label="Train Class 2")
     plt.scatter(w1_f[0],w1_f[1],marker='+',c='red',label="Mean Class 1")
     plt.scatter(w2_f[0],w2_f[1],marker='^',c='blue',label="Mean Class 1")
     plt.plot(*zip(*test_class_one),'g*',label="Test Class 1")
     plt.plot(*zip(*test_class_two),"g>",label="Test Class 2")
     plt.plot(line_x,line_y,"--",label="Decision Boundary")
     plt.legend(loc="upper center",bbox_to_anchor=(1.25, 1))
     plt.show()
```

Fig. 17. Decision Boundary Plotting

```
[ ]  print("Accuracy {}".format((matched/len(test_df)*100)))
```

Fig. 18. Accuracy Calculation