Designing a Minimum Distance to Class Mean Classifier

Devopriya Tirtho
16.02.04.033
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh

***Abstract*—The field of study named ‘Machine Learning’ tends to learn from appropriate datasets and tries to interpret results from that learning. Classification is a part of Machine Learning which tries to use those Machine Learning algorithms which learn from ‘Train Data’ to find the accurate class for ‘Test Data’. ‘Minimum Distance to Class Mean Classifier’ is one of the forms of Classification Technique which is very accurate for predicting appropriate class with the help linear discriminant function.**

***Index Terms*—Machine Learning, Classification, Algorithm, Dataset.**

I. Introduction

‘Minimum Distance to Class Mean Classifier’ is a classification technique which trains itself with the training vectors to fix the classified mean point and based on that classified mean point calculates the appropriate class of unclassified vectors of more than one class. Training dataset suggests that there are ‘n’ number of classified samples with proper features’ values. With the help of a **‘Linear Discriminant Function’** and some basic criterions unknown vectors are classified. So, this method works on finding the appropriate class of unknown sample of vector where classes are of more than one class.

II. Task

There are two datasets named ‘train’ and ‘test’. From the ‘train’ dataset we can find the sample prototypes of two classes which are classified by number ‘1’ and ‘2’. Here are the datapoints of ‘train’ Dataset:
W1= {(2, 2), (3, 1), (3, 3), (-1, -3), (4, 2), (-2, -2)}
W2= {(-4, 3), (2, 6), (0, 0), (-2, 2), (-1, -1), (-4, 2)}

- Firstly, we have to plot all the data points of each class with corresponding marker in graph.
- After plotting, we have to find the mean value of **‘Class 1’s’** points and **‘Class 2’s’** points.
- With the help of the following discriminant function we have to specify the appropriate class of the ‘test’ dataset.
- We have to draw the decision boundary to visualize the boundary of classification.
- Finally, we have to find the accuracy of the model for our given ‘test’ dataset.

III. Experimental Design

- **Plotting All Training Points:** In our given dataset which is named as ‘train’, we need to pick the values which represent X-coordinate values and Y-coordinate values. Then with appropriate marker and distinct color we have to plot the points of **‘Class 1’** and **‘Class 2’**.
- **Calculate the Mean Point:** Then we have to find the mean point of **‘Class 1’** and **‘Class 2**’ by taking the mean of appropriate vectors and plot the mean points of ‘Class 1’ and ‘Class 2’ with appropriate marker in the graph.
- **Classify the Test Data:** After finding the mean vectors of **‘Class 1’** and **‘Class 2’** we take the discriminant function for two classes as g1 and g2.
  Here,
  $g_1(x) = w_1^T x - 1/2 w_1^T w_1$

  $g_2(x) = w_2^T x - 1/2 w_2^T w_2$

  By doing appropriate calculation for each data points of ‘test’ dataset we compare the values of g1 and g2 to find the appropriate class of each data point.

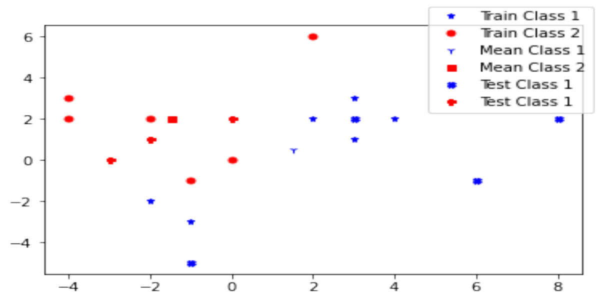


Fig. 1. After classification of all testing points

- **Drawing Decision Boundary:** To distinguish the area of each class we need to draw the decision boundary. The mathematical form the decision boundary is given below:

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

$$> w_1^T x - w_2^T x - 1/2(w_1^T w_1 - 1/2 w_2^T) = 0$$

$$> (w_1^T - w_2^T)x - 1/2(w_1^T w_1 - 1/2 w_2^T) = 0$$

$$> (COEF_1 \quad COEF_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + CONSTANT = 0$$

Here,

$$CONSTANT = -1/2(w_1^T w_1 - 1/2 w_2^T)$$

$$> (COEF_1 x_1 + COEF_2 x_2) + CONSTANT = 0$$

$$> x_2 = (COEF_1 x_1 + CONSTANT)/-COEF_2$$

By taking equidistance values for x1 for X-coordinate, we get the values for x2 for Y-Coordinate and plot the decision boundary with corresponding (x1, x2) point.

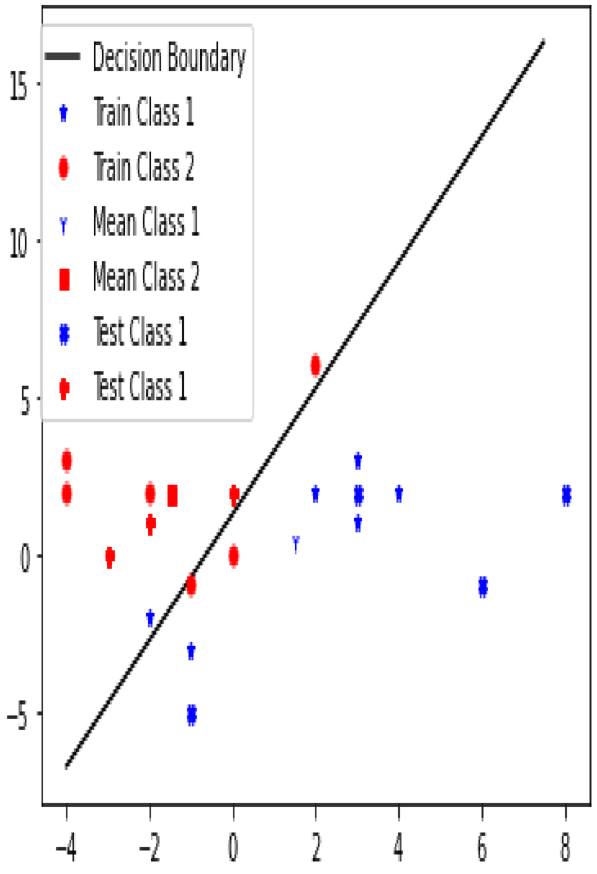
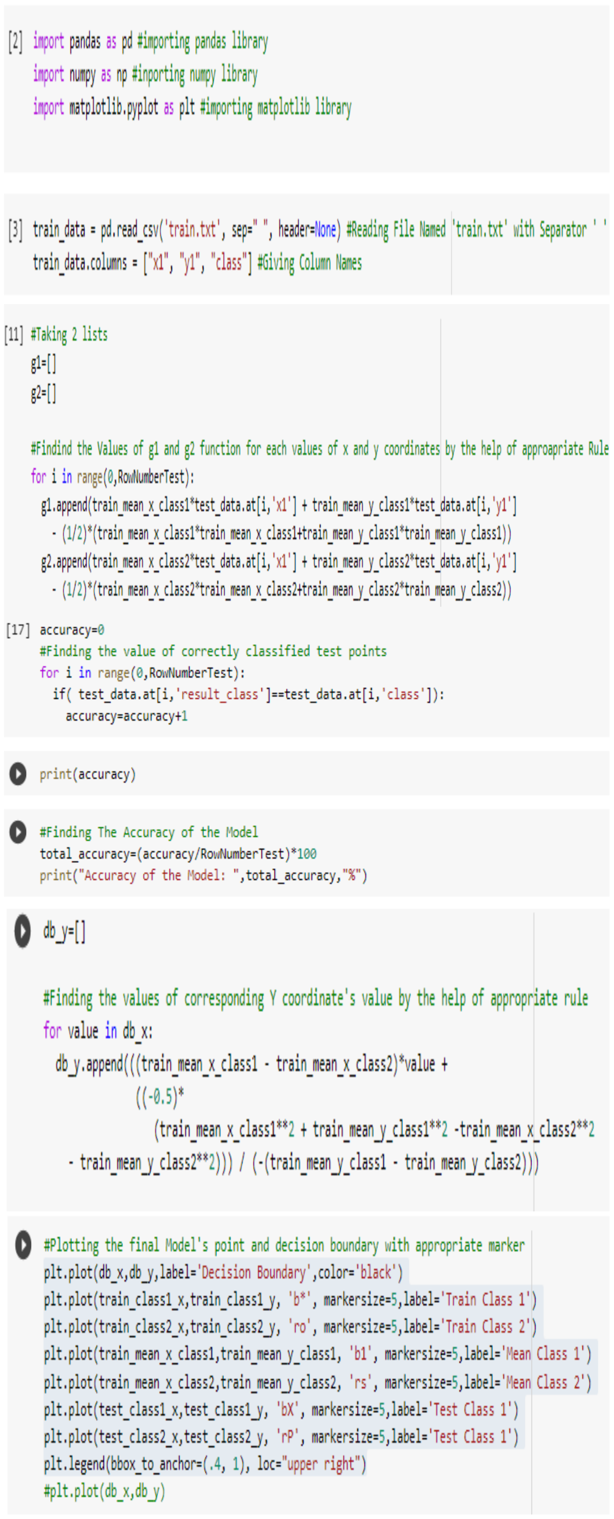


Fig. 2. Drawing of decision boundary

IV. Python Code



```
[2] import pandas as pd #importing pandas library
    import numpy as np #importing numpy library
    import matplotlib.pyplot as plt #importing matplotlib library
```

```
[3] train_data = pd.read_csv('train.txt', sep=" ", header=None) #Reading File Named 'train.txt' with Separator ' '
    train_data.columns = ["x1", "y1", "class"] #Giving Column Names
```

```
[11] #Taking 2 lists
     g1=[]
     g2=[]


     #Findind the Values of g1 and g2 function for each values of x and y coordinates by the help of approapriate Rule
     for i in range(0,RowNumberTest):
       g1.append(train_mean_x_class1*test_data.at[i,'x1'] + train_mean_y_class1*test_data.at[i,'y1']
         - (1/2)*(train_mean_x_class1*train_mean_x_class1+train_mean_y_class1*train_mean_y_class1))
       g2.append(train_mean_x_class2*test_data.at[i,'x1'] + train_mean_y_class2*test_data.at[i,'y1']
         - (1/2)*(train_mean_x_class2*train_mean_x_class2+train_mean_y_class2*train_mean_y_class2))

[17] accuracy=0
     #Finding the value of correctly classified test points
     for i in range(0,RowNumberTest):
       if( test_data.at[i,'result_class']==test_data.at[i,'class']):
         accuracy=accuracy+1
```

```
print(accuracy)
```

```
#Finding The Accuracy of the Model
total_accuracy=(accuracy/RowNumberTest)*100
print("Accuracy of the Model: ",total_accuracy,"%")
```

```
db_y=[]


#Finding the values of corresponding Y coordinate's value by the help of appropriate rule
for value in db_x:
  db_y.append(((train_mean_x_class1 - train_mean_x_class2)*value +
               ((-0.5)*
                 (train_mean_x_class1**2 + train_mean_y_class1**2 -train_mean_x_class2**2
    - train_mean_y_class2**2))) / (-(train_mean_y_class1 - train_mean_y_class2)))
```

```
#Plotting the final Model's point and decision boundary with appropriate marker
plt.plot(db_x,db_y,label='Decision Boundary',color='black')
plt.plot(train_class1_x,train_class1_y, 'b*', markersize=5,label='Train Class 1')
plt.plot(train_class2_x,train_class2_y, 'ro', markersize=5,label='Train Class 2')
plt.plot(train_mean_x_class1,train_mean_y_class1, 'b1', markersize=5,label='Mean Class 1')
plt.plot(train_mean_x_class2,train_mean_y_class2, 'rs', markersize=5,label='Mean Class 2')
plt.plot(test_class1_x,test_class1_y, 'bX', markersize=5,label='Test Class 1')
plt.plot(test_class2_x,test_class2_y, 'rP', markersize=5,label='Test Class 1')
plt.legend(bbox_to_anchor=(.4, 1), loc="upper right")
#plt.plot(db_x,db_y)
```

Fig. 3. Snapshot of python code

V. Result Analysis

The implemented model gives us the accuracy of **85.714%** for the 'test' data where some of the results were misclassified.

VI. Conclusion

‘Minimum Distance to Class Mean Classifier’ is a simple classification technique which uses linear discriminant function to classify unknown classes. This model is not perfectly accurate as it gives us some misclassified result. Further improvement can be done using non-linear decision boundary to classify all the unknown data perfectly.