Designing a Minimum Distance to Class Mean Classifier

Maliha Mahjabin Chowdhury
Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204043@aust.edu

***Abstract*—Classification techniques are widely used in modern days. It is a process of predicting the class of an unknown data point, by using a set of correctly classified data points. For instance, determining whether a mail is a spam or not is a great application of classification models. In this report, we thoroughly discuss about one of the simplest classifiers, Minimum Distance to Class Mean Classifier. At first, we design the classifier and following that we use the model to predict classes for some unknown data points.**

***Index Terms*—Mean, Linear Classifier, Eucildean Distance.**

I. Introduction

Minimum Distance to Class Mean Classifier is one of the easiest, yet a very effective classifier. As the name suggests, the model uses the class mean and predicts the target by calculating the Euclidean distance of a data point from the class mean. Suppose, there are two classes labelled Class-1 and Class-2 and the means are respectively Mean-1 and Mean-2. When an unknown data point comes, we calculate the Euclidean distance of that data point from both Mean-1 and Mean-2, and whichever distance is found to be lesser, the data point is predicted to belong to that class. This is the principle of Minimum Distance to Class Mean Classifier. The rest of the report is organised in the following manner: section 2 discusses our experimental design. In section 3, we illustrate the results and present our findings. Finally, section 4 summarizes the report and section 5 demonstrates the algorithm implementation.

II. Methodology

In this section, we illustrate the basic algorithm of Minimum Distance to Class Mean Classifier. Additionally, we demonstrate the method of obtaining the equation of Decision Boundary and the implementation of the algorithm.

A. Algorithm

1) Read the train dataset and separate dependent and independent variables
2) Calculate the mean vector $\overline{Y_i}$ for each class i
3) Read the test dataset
4) Compute for each data point X for every class i

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

5) if $g_1(X) > g_2(X)$ class = 1 $else$ class = 2
6) Plot the data points with appropriate marker
7) Draw the Decision Boundary
8) Calculate accuracy

B. Equation of Decision Boundary

For a data point $X(x, y)$ on the decision boundary, we can write,

$$g_1(X) = g_2(X)$$

$$\implies X\overline{Y_1^T} - \frac{1}{2}\overline{Y_1^T}\overline{Y_1} = X\overline{Y_2^T} - \frac{1}{2}\overline{Y_2^T}\overline{Y_2}$$

$$\implies X\overline{Y_1^T} - X\overline{Y_2^T} = \frac{1}{2}\overline{Y_1^T}\overline{Y_1} - \frac{1}{2}\overline{Y_2^T}\overline{Y_2}$$

$$\implies X(\overline{Y_1^T} - \overline{Y_2^T}) = \frac{1}{2}(\overline{Y_1^T}\overline{Y_1} - \overline{Y_2^T}\overline{Y_2})$$

$$\implies [X_x\ X_y][\overline{Y_x}\ \overline{Y_y}]^T = \frac{1}{2}(\overline{Y_1^T}\overline{Y_1} - \overline{Y_2^T}\overline{Y_2})$$

$$\implies \overline{Y_x}X_x + \overline{Y_y}X_y = \frac{1}{2}(\overline{Y_1^T}\overline{Y_1} - \overline{Y_2^T}\overline{Y_2})$$

$$\implies X_y = -\frac{\frac{1}{2}(\overline{Y_1^T}\overline{Y_1} - \overline{Y_2^T}\overline{Y_2}) + \overline{Y_x}X_x}{\overline{Y_y}} \tag{2}$$

C. Implementation Process

We design the classifier in Python. This is accomplished using the most popular libraries: NumPy [1], pandas [2] and Matplotlib [3]. NumPy and pandas help processing the dataset and performing necessary operations and calculations respectively. We visualize the results and the decision boundary using Matplotlib. We classify the data points using the equation 1 and draw the decision boundary using the equation 2. We use different colors and markers for different classes as well as for train set and test set. Figure 1 presents classified data points on a plot.

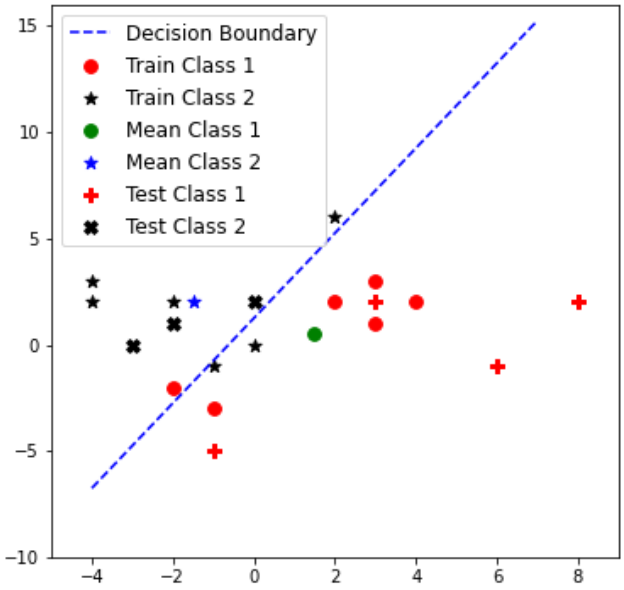

Fig. 1. Classified data points and Decision Boundary

III. Result Analysis

We use Accuracy as an evaluation metric to analyze the performance of the classifier. It defines the overall effectiveness.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} * 100\% \quad (3)$$

From our Experimentation, we obtain accuracy of 85.71% and have only 1 data point misclassified out of 7 data points. The decision boundary is equally distanced from the means of the both classes.

IV. Conclusion

For the given dataset, our model performs satisfactorily, achieving an accuracy around 86%. However, this model is subjected to perform poorly with a linearly inseparable dataset. In our experimentation, since the given data are two dimensional, we can visualize the data points and the decision boundary; which is one dimensional. It is note worthy that designing the model is simple and it also requires low computational power.

V. Algorithm Implementation

The classifier is designed in Python. The implementation is given below.

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

dataset = pd.read_csv("train.txt",
sep = "␣", header = None)

train = dataset.iloc[:, :].values

dataset = pd.read_csv("test.txt",
sep = "␣", header = None)

test = dataset.iloc[:, :].values

myFig = plt.figure(figsize=[14,6])

plt.scatter(list(train[i][0]
for i in range(len(train))
if train[i][2] == 1),
list(train[i][1] for i in range(len(train))
if train[i][2] == 1),
color = "red", marker = 'o',
label = "Train␣Class␣1", s = 60)


plt.scatter(list(train[i][0]
for i in range(len(train))
if train[i][2] == 2),
list(train[i][1] for i in range(len(train))
if train[i][2] == 2),
color = "black", marker = '*',
label = "Train␣Class␣2", s = 60)


mean_1 = np.array([np.mean(list(train[i][0]
for i in range(len(train))
if train[i][2] == 1)),
np.mean(list(train[i][1]
for i in range(len(train))
if train[i][2] == 1))])


plt.scatter(mean_1[0], mean_1[1],
color = "green", marker = 'o',
label = "Mean␣Class␣1", s = 60)


mean_2 = np.array([np.mean(list(train[i][0]
for i in range(len(train))
if train[i][2] == 2)),
np.mean(list(train[i][1]
for i in range(len(train))
if train[i][2] == 2))])


plt.scatter(mean_2[0], mean_2[1],
color = "blue", marker = '*',
label = "Mean␣Class␣2", s = 60)


plt.xlim(-5, 9)
plt.ylim(-10, 16)
```

```
prediction = []


for i in range(len(test)):
  x1 = test[i][0]
  x2 = test[i][1]

  g1_x = (x1*mean_1[0] + x2*mean_1[1])
  - 0.5 * (mean_1[0]*mean_1[0] +
  mean_1[1]*mean_1[1])

  g2_x = (x1*mean_2[0] + x2*mean_2[1])
  - 0.5 * (mean_2[0]*mean_2[0] +
  mean_2[1]*mean_2[1])

  if g1_x > g2_x:
    prediction.append(1)
  else:
    prediction.append(2)


plt.scatter(list(test[i][0]
for i in range(len(test))
if prediction[i] == 1),
list(test[i][1]
for i in range(len(test))
if prediction[i] == 1),
color = "red", marker = "P",
label = "Test Class 1", s = 60)


plt.scatter(list(test[i][0]
for i in range(len(test))
if prediction[i] == 2),
list(test[i][1]
for i in range(len(test))
if prediction[i] == 2),
color = "black", marker = "X",
label = "Test Class 2", s = 60)


m = mean_1 - mean_2
c = 0.5 * ((mean_1[0]*mean_1[0] +
mean_1[1]*mean_1[1]) -
(mean_2[0]*mean_2[0] +
mean_2[1]*mean_2[1]))


plt.plot(list(i for i in range(-4, 8)),
list((-m[0]*j + c)/m[1]
for j in range(-4, 8)),
color = "blue", linestyle = "dashed",
label = "Decision Boundary",
markersize = 16)

plt.legend(loc = "best", fontsize = 12)
plt.show()

accuracy = 0
for i in range(len(test)):
  if test[i][2] == prediction[i]:
    accuracy = accuracy + 1
accuracy = (accuracy / len(test)) * 100
print("Accuracy : ", accuracy)
```

REFERENCES

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[2] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.