Designing A Minimum Distance to Class Mean Classifier

Author: Humaira Zahin Mauni
ID: 16.01.04.012

Abstract
A minimum distance to class mean classifier is a linear classifier which uses each class's mean as the prototype of said class. Training the classifier involves calculating the mean of each class and partitioning the feature space according to the nearest mean. The classification itself is done by assigning unlabeled patterns to the nearest class mean in the feature space.

Keywords
Linear Classification, class mean, Linear Discriminant Function

Introduction
The minimum distance to class mean classifier is used to classify unknown data to classes that minimize the distance between the data and the class in the feature space. This is a simple and easy to implement a classifier that offers us a glimpse at how classifiers map multi-featured data to a feature space.

Experimental Design
Two-class set of prototypes have to be taken from "train.txt" and "test.txt" files.

1. Plot all sample points (train data) from both classes, but samples from the same class should have the same color and marker.
2. Using a minimum distance classifier with respect to 'class mean', classify the test data points by plotting them with the designated class-color but a different marker. Use the Linear Discriminant Function given below. Also, plot the class means.

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

3. Draw the decision boundary between the two classes.
4. Find accuracy.

Algorithm Implementation

```
import numpy as np
import pandas as pd
import io
import matplotlib.pyplot as plt
from google.colab import files

### Q1 ###
```

```
uploaded = files.upload() #upload train.txt

df_train = pd.read_csv(io.BytesIO(uploaded['train.txt'] ), sep=" " ,  header = None, d
type = 'Int64')
df_train.columns = ['x', 'y', 'label']
print(df_train)

groups = df_train.groupby('label')

fig, ax = plt.subplots()
for name, group in groups:
    name = 'Train Class ' + str(name)
    ax.plot(group.x, group.y, marker='o', linestyle='', ms=12, label=name)
ax.legend()

plt.show()

### Q2 ###
uploaded = files.upload() #upload test.txt

df_test = pd.read_csv(io.BytesIO(uploaded['test.txt'] ), sep=" " ,  header = None, dty
pe = 'Int64')
X_train = df_train.iloc[:, 0:2].values
y_train = df_train.iloc[:, 2].values
y_train = y_train.astype('int')
X_test = df_test.iloc[:, 0:2].values
y_test = df_test.iloc[:, 2].values
y_test = y_test.astype('int')

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)

y_pred = lda.predict(X_test)

df_test.columns = ['x', 'y', 'actual_label']
df_test['predicted label'] = y_pred
print(df_test)


### Q3 ###

groups = df_train.groupby('label')
groups2 =  df_test.groupby('predicted label')
mean = df_train.groupby('label').mean()
x = np.arange(start=-4, stop=8, step=1)
m1 = mean.iloc[0]['x']-mean.iloc[1]['x']
m2 = mean.iloc[0]['y']-mean.iloc[1]['y']
```

```
c = -
0.5* (((mean.iloc[0]['x']**2) + (mean.iloc[0]['y']**2)) - ((mean.iloc[1]['x']**2) + (m
ean.iloc[1]['y']**2)))
y = (-m1/m2)*x +c

### Plotting the results to Q2 and Q3 ###
fig, ax = plt.subplots()
for name, group in groups:
    name = 'Train Class ' + str(name)
    ax.plot(group.x, group.y, marker='o', linestyle='', ms=12, label=name)
for name, group in groups2:
    name = 'Test Class ' + str(name)
    ax.plot(group.x, group.y, marker='x', linestyle='', ms=12, label=name)
ax.plot(mean.iloc[0]['x'], mean.iloc[0]['y'], 'sb', label='Mean Class 1')
ax.plot(mean.iloc[1]['x'], mean.iloc[1]['y'], 'sr', label='Mean Class 2')
ax.plot(x,y, ':', label = 'Decision Boundary')
ax.legend()

plt.show()

### Q4 ###
from sklearn.metrics import accuracy_score
print('Accuracy = ', accuracy_score(y_test, y_pred)*100, '%')
```

Result Analysis

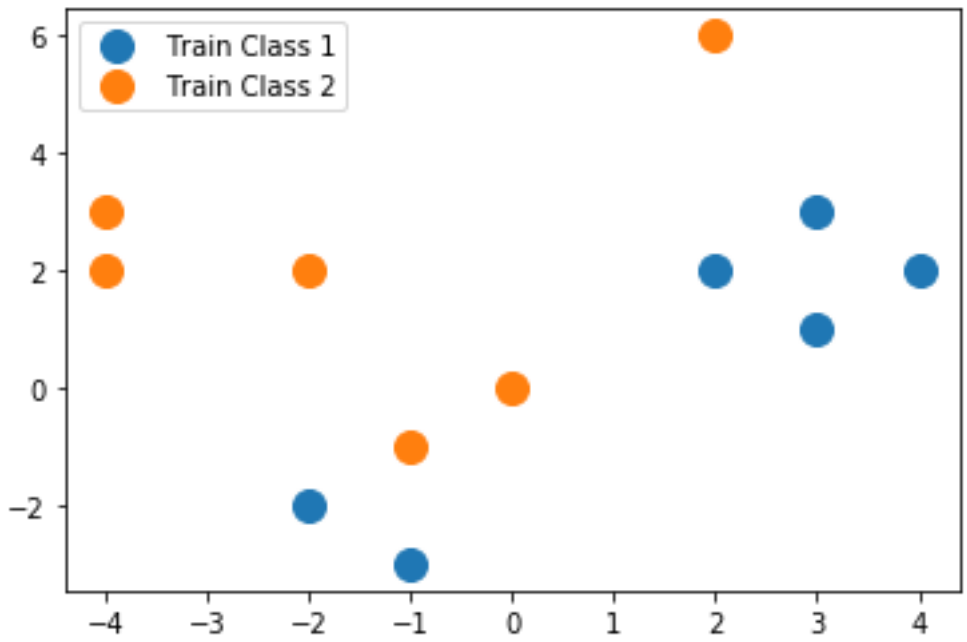

Fig 1: - Graphical representation of the training data in the feature space

Here, we can see the training data mapped onto the feature space. Data form each class are distinguished by color.

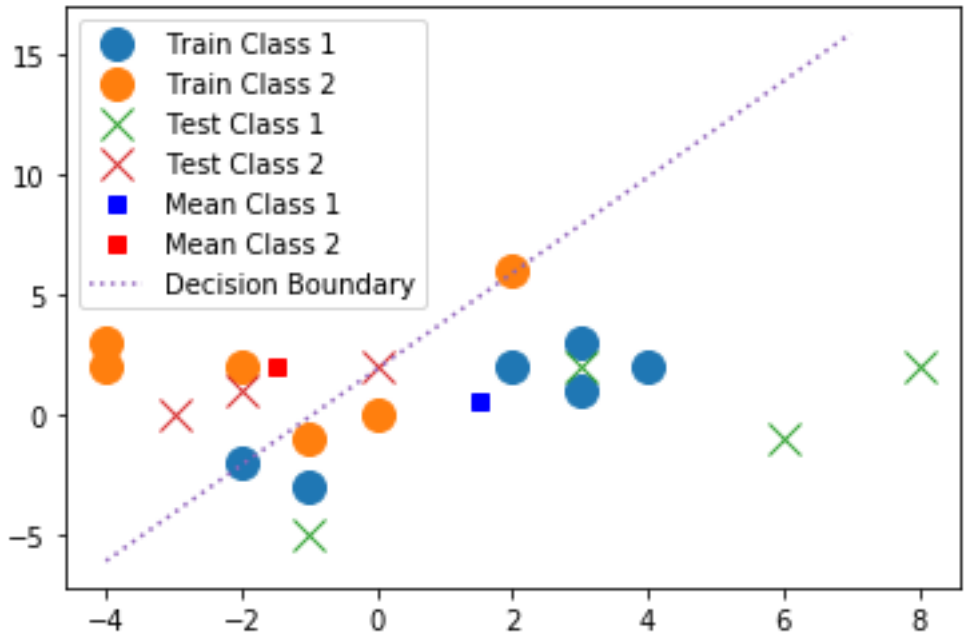

Fig 2: - Graphical representation of the feature space and all it's necessary components after classification has taken place

Here, we observe the decision boundary generated by the linear discriminant function applied, and thus the separation created which provides us with the basis for our classification model. The tested data points and the class means are also plotted for overall visual representation for analysis.

The accuracy of this model is found to be **85.71%**, with 1 out of the 7 given data points for testing being misclassified.

Conclusion

The model created by the minimum distance to mean classifier performed well on the dataset provided, with an accuracy of 85.71%. The drawback of this classifier is that it is a linear classifier, meaning the feature space is only linearly separable by the decision boundary. This limits the classifier when some data points need to be mapped to a higher plane in order to be properly separated. Fig 1 displays that 5 coordinates, some from training class 1 and some from 2, fall on the same line and cannot be separated by a straight line. The minimum distance to mean classifier has a fast computation speed compared to other classifiers.