

Assignment No. 1

Minimum Distance to Class Mean classifier

MD. Saimom Islam

16.02.04.011

January 17, 2021

1 Abstract

”Minimum Distance to Class Mean Classifier” is used to classify unclassified sample vectors clustered in more than one classes are given. Basically The minimum distance classifier is used to classify unknown data to classes which minimize the distance between the image data and the class in multi-feature space. The distance is defined as an index of similarity so that the minimum distance is identical to the maximum similarity. For example, in a dataset containing n sample vectors of dimension d some given sample vectors are already clustered into classes and some are not. we can classify the unclassified sample vectors with Class Mean Classifier.

2 Given Task

Given the following two class set of prototypes. Let in train.txt which level is

$1 = w_1$ and which level is $2 = w_2$

$w_1 = \{(2 \ 2), (3 \ 1), (3 \ 3), (-1 \ -3), (4 \ 2), (-2 \ -2)\}$

$w_2 = \{(-4 \ 3), (2 \ 6), (0 \ 0), (-4 \ 2), (-1 \ -1), (-4 \ 2)\}$

1.

plot all sample points 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 following points by plotting them with the designated class-color but different marker

$$X_1 = (-1 \ -5)$$

$$X_2 = (3 \ 2)$$

$$X_3 = (-2 \ 1)$$

$$X_4 = (8 \ 2)$$

$$X_5 = (6 \ -1)$$

$$X_6 = (0 \ 2)$$

$$X_7 = (-3 \ 0)$$

The Linear Discriminant Function is :

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

3.

Draw the decision boundary between two classes.

The Formula is generated from below:

Let mean1 = y1 and mean2 = y2

$$y_1 = [1.5 \ .5]$$

$$y_2 = [-1.5 \ 2]$$

$$g_1(X) = [1.5 \ .5] \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$\Rightarrow 1.5X_1 + .5X_2 - 1.25$$

$$g_2(X) = [-1.5 \ 2] \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

$$\Rightarrow 3X_1 - 1.5X_2 - 1.25$$

Now, For decision boundary, $g_1 = g_2$

So,

$$1.5X_1 + .5X_2 - 1.25 = 3X_1 - 1.5X_2 - 1.25$$

$$X_2 = (3X_1 + 1.878)/1.5$$

This is the decision boundary line equation

3 Implementation

1.First of all i have to plot all the sample points of both classes from train dataset.For that train data was classified in two classes based on their label.All data which are labelled 1 in one class and lebelled 2 are in another class.then the classified data are stored in two different numpy arrays,and finally plotted with same color and marker of same class data.

2.At second step Mean of the two classes were calculated and plotted with different marker.test data were also classified based on their label and plotted with same marker and different color

3.After that the equation of decision boudary was calculated which is shown avobe and plotted every point and a decision boundary is plotted.

4.Finally the accuracy of test data was calculated compared with real class data.Using $\frac{N}{T} \times 100\%$

Here N = How many samples are correctly classified and T = Total Number of samples


4 Main Code

The Code is given below:

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

```
from google.colab import files

uploaded_train = files.upload()
```

 Choose Files No file chosen

```
[99] training_data = pd.read_csv(io.BytesIO(uploaded_train['train.txt']), sep=' ', header=None)
```

```
[101] training_data_arr = training_data.to_numpy() ;
```

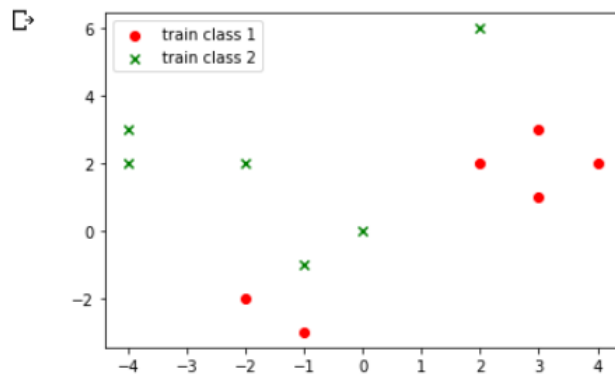
```
[102] class1=[[x[0],x[1]] for x in training_data_arr if x[2]==1]
      class2=[[x[0],x[1]] for x in training_data_arr if x[2]==2]

      train_class1 = np.array(class1)
      train_class2 = np.array(class2)
```

```

fig = plt.figure()
plt.scatter(train_class1[:,0], train_class1[:,1], c = 'r', marker = 'o', label = 'train class 1')
plt.scatter(train_class2[:,0], train_class2[:,1], c = 'g', marker = 'x', label = 'train class 2')
plt.legend(loc = 'best')
plt.show()

```



```

# calculating the mean
y1 = np.mat([train_class1[:,0].mean(), train_class1[:,1].mean()])
y2 = np.mat([train_class2[:,0].mean(), train_class2[:,1].mean()])

```

uploaded_test = files.upload()



Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser s

Saving test.txt to test (1).txt

```
[116] # reading the test samples
testing_data = pd.read_csv(io.BytesIO(uploaded_test['test.txt']), sep=' ', header=None)
testing_data_arr = testing_data.to_numpy() ;
```

```
[117] test_class1=[[x[0],x[1]] for x in training_data_arr if x[2]==1]
test_class2=[[x[0],x[1]] for x in training_data_arr if x[2]==2]
test_class1 = np.array(class1)
test_class2 = np.array(class2)
```



```
# converting features of test data into matrix
testing_x = [[x[0],x[1]] for x in testing_data_arr ]
testing_x_mat_arr = np.asmatrix(testing_x)
print(testing_x_mat_arr)
```

|

```

▶ # array for storing the class 1 and class 2 testing data according to minimum distance classifier
testing_x_class1 = np.empty(( 0, 2))
testing_x_class2 = np.empty(( 0, 2))
# list for storing the predictions
accuracy = []

[120] # minimum distance classifier with respect to class 'mean'
for i in range(len(testing_x_mat_arr)):
    g1 = (y1 * testing_x_mat_arr[i].transpose() ) - (0.5 * (y1 * y1.transpose()))
    g2 = (y2 * testing_x_mat_arr[i].transpose() ) - (0.5 * (y2 * y2.transpose()))
    if g1 > g2 :
        testing_x_class1 = np.append(testing_x_class1, [[testing_x_mat_arr[i].item(0), testing_x_mat_arr[i].item(1)]], axis=0)
        accuracy.append(1)
    else:
        testing_x_class2 = np.append(testing_x_class2, [[testing_x_mat_arr[i].item(0), testing_x_mat_arr[i].item(1)]], axis=0)
        accuracy.append(2)

[121]
# plotting the training samples
plt.scatter(train_class1[:,0], train_class1[:,1], c = 'r', marker = 'o', label = 'train class 1')
plt.scatter(train_class2[:,0], train_class2[:,1], c = 'g', marker = 'x', label = 'train class 2')

```



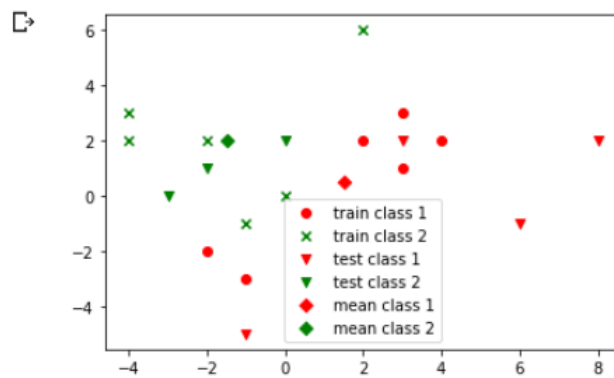
```

# plotting the testing samples
plt.scatter(testing_x_class1[:, 0], testing_x_class1[:, 1], c = 'r', marker = "v", label = 'test class 1')
plt.scatter(testing_x_class2[:, 0], testing_x_class2[:, 1], c = 'g', marker = "v", label = 'test class 2')
# plotting the means
plt.scatter(y1.item(0), y1.item(1), c = 'r', marker = "D", label = 'mean class 1')
plt.scatter(y2.item(0), y2.item(1), c = 'g', marker = "D", label = 'mean class 2')

plt.legend(loc = "best")

plt.show()

```



```

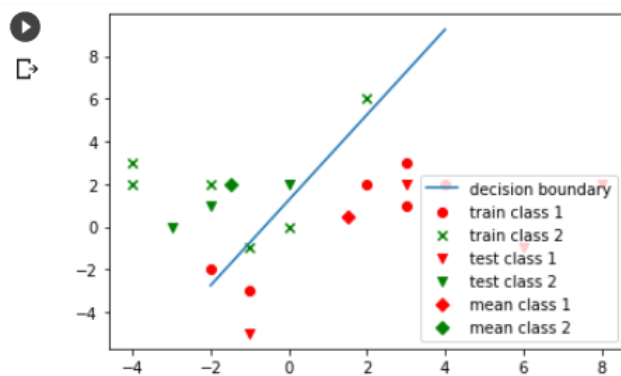
[122] x_db = []
      y_db = []
      for i in range(train_class1[:,0].min(), train_class1[:,0].max()+1):
          x_db.append(i)
          j = ((3 * i) + 1.875) / 1.5
          y_db.append(j)

[123] # plotting the training samples
      plt.scatter(train_class1[:,0], train_class1[:,1], c = 'r', marker = 'o', label = 'train class 1')
      plt.scatter(train_class2[:,0], train_class2[:,1], c = 'g', marker = 'x', label = 'train class 2')
      # plotting the testing samples
      plt.scatter(testing_x_class1[:, 0], testing_x_class1[:, 1], c = 'r', marker = "v", label = 'test class 1')
      plt.scatter(testing_x_class2[:, 0], testing_x_class2[:, 1], c = 'g', marker = "v", label = 'test class 2')
      # plotting the means
      plt.scatter(y1.item(0), y1.item(1), c = 'r', marker = "D", label = 'mean class 1')
      plt.scatter(y2.item(0), y2.item(1), c = 'g', marker = "D", label = 'mean class 2')
      # plotting decision boundary
      plt.plot( x_db, y_db, label = 'decision boundary')

      plt.legend(loc = 'best')

      plt.show()

```



```
[124] # measuring accuracy
      counter = 0
      for i in range(len(testing_data[2])):
          if measure_accuracy[i] == testing_data[2][i]:
              counter += 1

      accuracy = (counter / len(testing_data[2])) * 100
      print("accuracy = " , accuracy)
```

```
accuracy = 85.71428571428571
```

5 Result Analysis

The Results of Given tasks are given below:

1. Plotting all sample points from training data

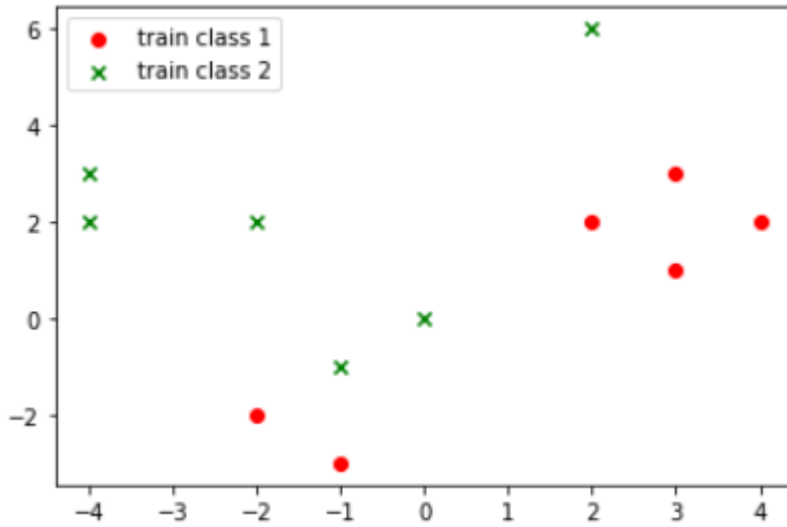


Figure 1: the two classes of train dataset are plotted using different marker and different color

2. Plotting the mean class data and test class data

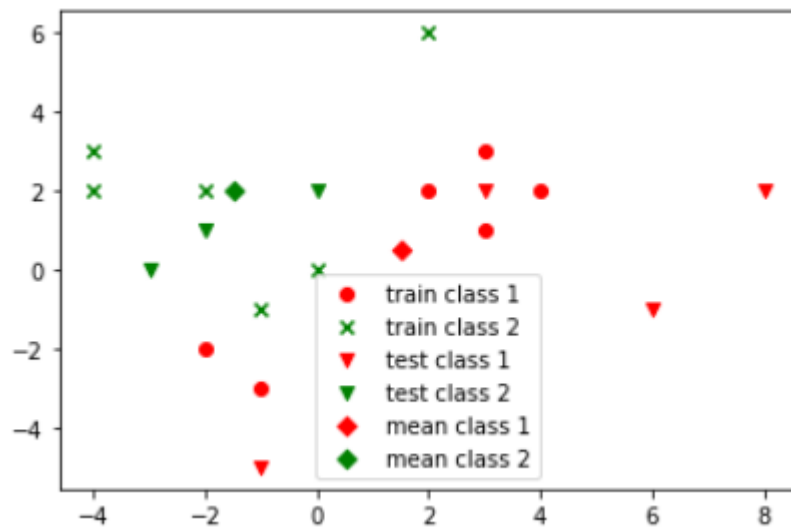


Figure 2: Minimum distance classifier with respect to class mean and classify some sample

3. Plotting the decision boundary

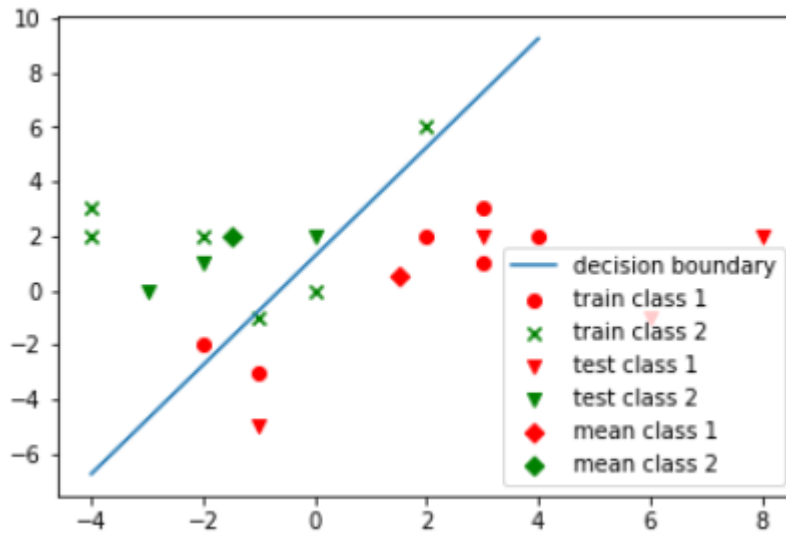


Figure 3: Decision Boundary to classify two class

4. Measure Accuracy

accuracy = 85.71428571428571

Figure 4: Accuracy

6 Conclusion

Minimum distance to class mean classifier is fast but the problem is it's performance is poor for big dataset. The rate of misclassification is high. Though the accuracy for this given dataset is pretty good. the accuracy is 85.712%. So, this classifier performed well for this given dataset