Designing a Minimum Distance to Class Mean Classifier

Rashedun Nobi Chowdhury
ID: 160204039
Group: A2
Email: 160204039@aust.edu

***Abstract*—Machine Learning algorithms serve different purposes and utilities. Of the many applications of Machine Learning, one is predicting the target class of an unknown data. Classification techniques are usually trained using a labeled dataset. They are then able to predict the class of new data based on the training dataset. In this assignment, I work with a simple classification technique named Minimum Distance to Class Mean classifier. I train the model using a provided dataset and perform some basic experiments.**

I. Introduction

Minimum Distance to Class Mean classifier is a simple classification technique that is trained using a labeled dataset. It is therefore regarded as a supervised learning technique. In the training phase, the model creates a mean vector μ for all class C. When a new data point is presented to the trained model, the algorithm first finds the euclidean distance between the data and mean vector for all the classes. The data is predicted to be the class it shares minimum distance with. In this experiment, I'm working with two classes, thus making this model a binary classifier. The rest of the report is organized as follows, section 2 contains the experimental design. In section 3 I briefly describe the results. In section 4 I conclude the report with some discussions on the advantages and disadvantages of the algorithm. Finally in section 5, I attach a snapshot of my implemented code.

II. Experimental Design / Methodology

To design the classifier, I use the python programming language. I have used the **pandas** library to read the dataset. **Numpy** was used to do basic mathematical operations. I also use **MatPlotLib** to plot the data points and decision boundary.

Algorithm 1 Minimum Distance to Class Mean Classifier

```
1) Read the train and test dataset
2) Split the dependent and independent variables
3) Compute mean vector $\overline{Y_i}$ for each class i
4) For all test data X, calculate $i$
```

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

```
5) if $g_1(X) > g_2(X)$ class = 1 $else$ class = 2
6) Plot the points and draw decision boundary
7) Calculate the accuracy of the classifier
```

III. Result Analysis

The classifier I designed achieved an accuracy of 85.7142% when tested using the test dataset. In *fig 1* I have plotted all the data points in the train and test dataset. I have also drawn a decision boundary. Decision boundary however is not flawless as some data points are placed in the wrong side of the boundary.

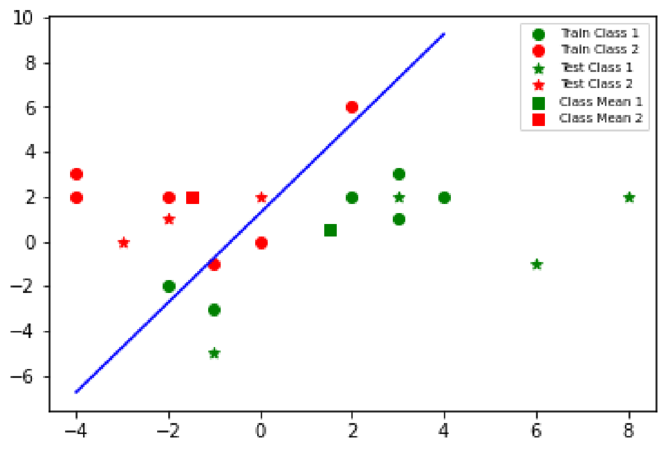


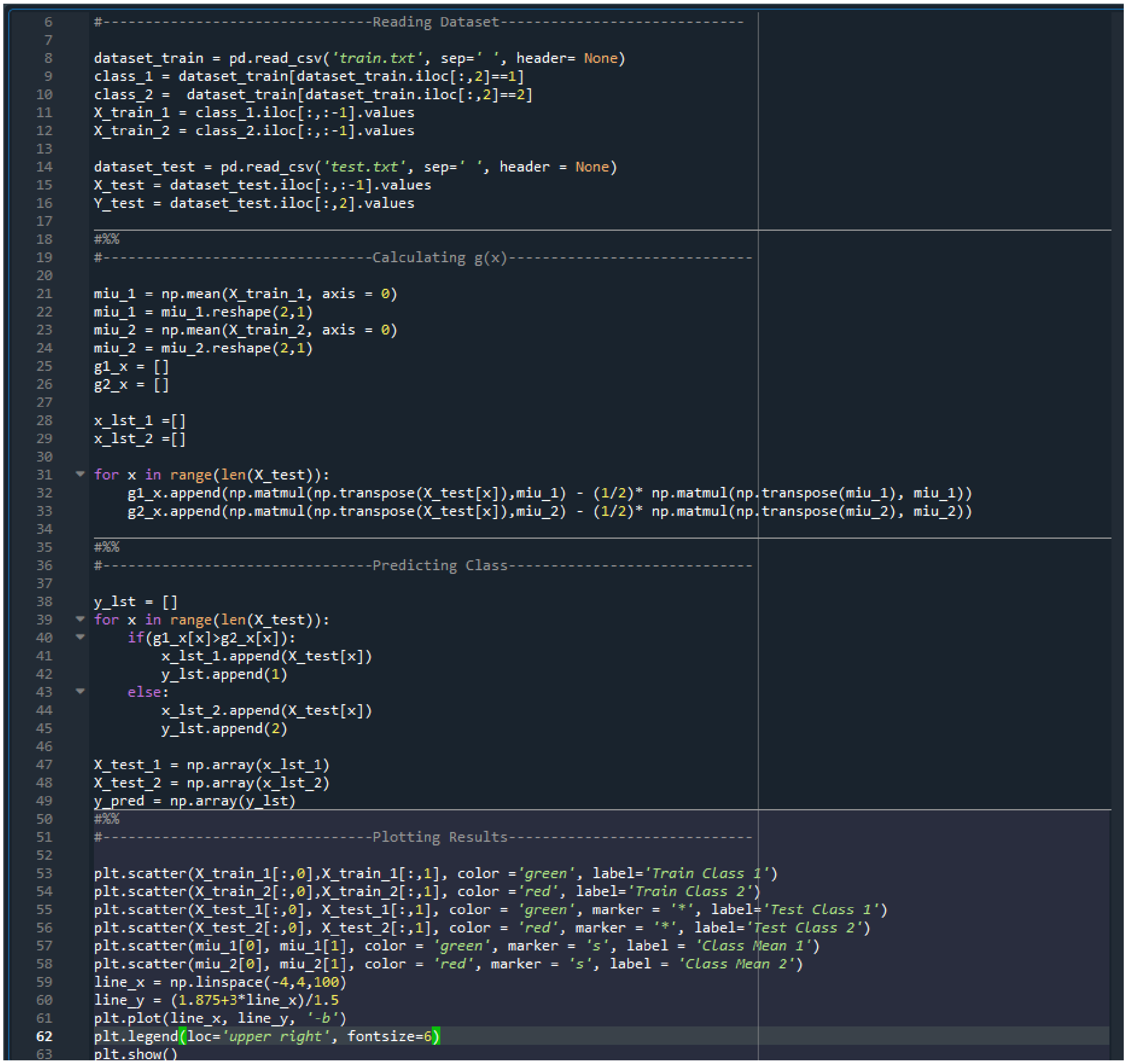
Fig. 1. Distribution of Data Points

IV. Conclusion

Minimum Distance to Class Mean classifier is a basic classifier that is easy to implement and lightweight. For this assignment, I have designed the model as a binary classifier using python. The model was able to achieve a good accuracy score. It should be noted that the dataset was linearly separable. For data that are not linearly separable, the model may perform poorly. Another thing to note is that outlier cases may negatively affect the model by shifting the class mean vector.

V. Algorithm Implementation / Code

A snapshot of my implementation of the algorithm in python in included in the next page.



```python
#--------------------------------Reading Dataset----------------------------

dataset_train = pd.read_csv('train.txt', sep=' ', header= None)
class_1 = dataset_train[dataset_train.iloc[:,2]==1]
class_2 =  dataset_train[dataset_train.iloc[:,2]==2]
X_train_1 = class_1.iloc[:,:-1].values
X_train_2 = class_2.iloc[:,:-1].values

dataset_test = pd.read_csv('test.txt', sep=' ', header = None)
X_test = dataset_test.iloc[:,:-1].values
Y_test = dataset_test.iloc[:,2].values

#%%
#--------------------------------Calculating g(x)----------------------------

miu_1 = np.mean(X_train_1, axis = 0)
miu_1 = miu_1.reshape(2,1)
miu_2 = np.mean(X_train_2, axis = 0)
miu_2 = miu_2.reshape(2,1)
g1_x = []
g2_x = []

x_lst_1 =[]
x_lst_2 =[]

for x in range(len(X_test)):
    g1_x.append(np.matmul(np.transpose(X_test[x]),miu_1) - (1/2)* np.matmul(np.transpose(miu_1), miu_1))
    g2_x.append(np.matmul(np.transpose(X_test[x]),miu_2) - (1/2)* np.matmul(np.transpose(miu_2), miu_2))

#%%
#--------------------------------Predicting Class----------------------------

y_lst = []
for x in range(len(X_test)):
    if(g1_x[x]>g2_x[x]):
        x_lst_1.append(X_test[x])
        y_lst.append(1)
    else:
        x_lst_2.append(X_test[x])
        y_lst.append(2)

X_test_1 = np.array(x_lst_1)
X_test_2 = np.array(x_lst_2)
y_pred = np.array(y_lst)
#%%
#--------------------------------Plotting Results----------------------------

plt.scatter(X_train_1[:,0],X_train_1[:,1], color ='green', label='Train Class 1')
plt.scatter(X_train_2[:,0],X_train_2[:,1], color ='red', label='Train Class 2')
plt.scatter(X_test_1[:,0], X_test_1[:,1], color = 'green', marker = '*', label='Test Class 1')
plt.scatter(X_test_2[:,0], X_test_2[:,1], color = 'red', marker = '*', label='Test Class 2')
plt.scatter(miu_1[0], miu_1[1], color = 'green', marker = 's', label = 'Class Mean 1')
plt.scatter(miu_2[0], miu_2[1], color = 'red', marker = 's', label = 'Class Mean 2')
line_x = np.linspace(-4,4,100)
line_y = (1.875+3*line_x)/1.5
plt.plot(line_x, line_y, '-b')
plt.legend(loc='upper right', fontsize=6)
plt.show()
```