Implementing K-Nearest Neighbors (KNN)

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

***Abstract*—Machine Learning is an important sector of computer science with many use cases and purposes. Classifying a real data based on previous knowledge is one of its major applications. K Nearest Neighbors is a very popular and easy to implement machine learning model. For any new data point the data is predicted to be of the majority class of 'k' closest data points. In this assignment, I implement the K Nearest Neighbor model for binary classification. I also conduct some basic experiments on the model.**

***Index Terms*—KNN, Machine Learning, Classification Techniques, Binary Classification**

I. Introduction

K Nearest Neighbor(KNN) is a simple machine learning technique used for data classification. It is a supervised learning technique. The KNN model is first trained using a training set. When a test data appears, the model computes the distance between the test data and all training data points. The data is classified based on the majority among the 'k' closest neighbors. For example, if k=3, then the model will first calculate the distance of all the points from the test data. The distances are then sorted in ascending order. The 3 closest data points are then chosen. The test is predicted based on which class it shares most nearest neighbors with. For this experiment, we are computing the distance using euclidean distance,

$$distance = \sqrt{(x_{train}[0] - x_{test}[0])^2 + (x_{train}[1] - x_{test}[1])^2} \quad (1)$$

It should be mentioned that our given dataset contains data in two dimension. 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

For this experiment, I have implemented the algorithm in python. A brief description of the algorithm is as follows,

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 Error Rate Classifier

```
1) Read the train dataset
2) Read the test dataset
3) For each data, calculate distance using 1
4) Sort the train data based on the distance in ascending
   order
5) Take k closest neighbors
6) Classify the test according to the majority class of
   nearest neighbors
7) Plot the data using appropriate markers
```

III. Result Analysis

In figure 1, I've plotted the train data points in a 2D graph. I've also plotted the test data after prediction.

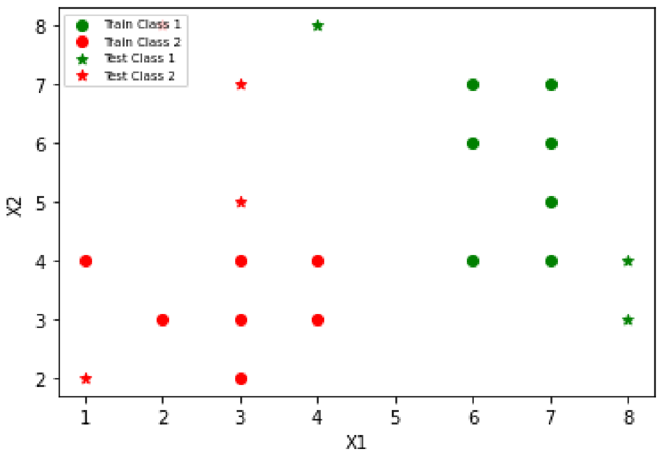

Fig. 1. Distribution of Data Points

Of the 8 data points in test class, 4 are predicted to be from Class 1 whereas 4 other have been predicted to be from class 2. I would like to mention that 1 is drawn when taking into consideration that **k = 3**

IV. Conclusion

In this experiment, I have implemented the KNN algorithm for Binary Classification. The model takes euclidean distance as the metric to compute distances between two data points and find nearest neighbors. The model is easy to implement. However as the number of test data points increase, the model may take longer time to execute.

V. Algorithm Implementation / Code

A snapshot of my implementation of the algorithm in python is given below,

```
x_lst_1 =[]
x_lst_2 =[]
num_neighbors = int(input('Enter␣number␣of␣Neighbors:␣'))
file1 = open("prediction.txt","w+")

for x in range(len(X_test)):
    dist_list = []
    class_1_count = 0
    class_2_count = 0
    for y in range(len(X_train_1)):
        dist_list.append(DataPoint(X_train_1[y], 1, math.sqrt((X_test[x][0]-X_train_1[y][0])**
    for y in range(len(X_train_2)):
        dist_list.append(DataPoint(X_train_2[y], 2, math.sqrt((X_test[x][0]-X_train_2[y][0])**

    dist_list.sort(key=sorter)

    print('Test␣Point:␣', X_test[x],file = file1)
    for i in range(num_neighbors):
        if(dist_list[i]._class_ == 1):
            class_1_count = class_1_count + 1
            print('Distance', i+1,':', dist_list[i]._distance_, '\tClass:', 1, file = file1)
        else:
            class_2_count = class_2_count + 1
            print('Distance', i+1,':', dist_list[i]._distance_, '\tClass:', 2,file = file1)

    if(class_1_count>class_2_count):
        x_lst_1.append(X_test[x])
        print('Predicted␣Class:',1, file = file1)
    else:
        x_lst_2.append(X_test[x])
        print('Predicted␣Class:',2, file = file1)
file1.close()
```