Implementing K-Nearest Neighbors (KNN)

MD Saimom Islam
Department of CSE
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204011@aust.edu

***Abstract*—The experiment is about to implement the KNN Classifier to classify some sample points using the KNN algorithm. For doing so I used Eucledian distance to classify the nearest data points of the classes.It minimizes the error rate of classification**

I. Introduction

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

A. Motivation

My main goal is to implement the KNN on given training sets and performing a distribution on given data and labeled them. Also, plot them in a 3D surface for clear visualization.

II. Experimental Design / Methodology

A. Task 1

1. Determine parameter K = number of nearest neighbors. Suppose use K = 3

B. Task 2

Then I drawn classified samples in 2D plane using different colored markers according to the assigned class label of training dataset

C. Task3

Then I drawn classified samples in 2D plane using different colored markers according to the assigned class label of test dataset

D. Task4

Then using eucledian distance from all the datapoint i measured the distance of the sample point.and then sorted them and take first 3 points and stored it in a file name prediction.txt.The majority probability will be the decided class for that sample point.

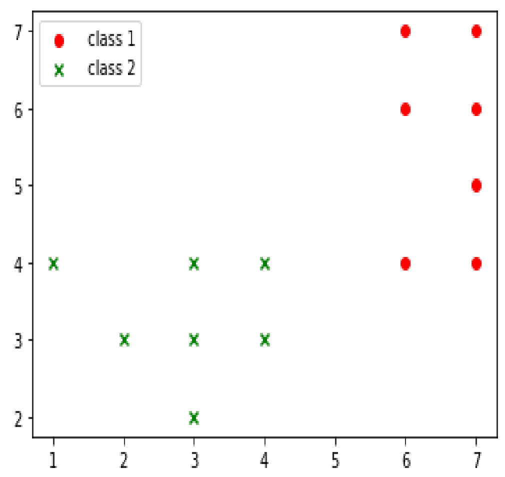

Fig. 1. Distribution of data points as Class 1 and Class 2

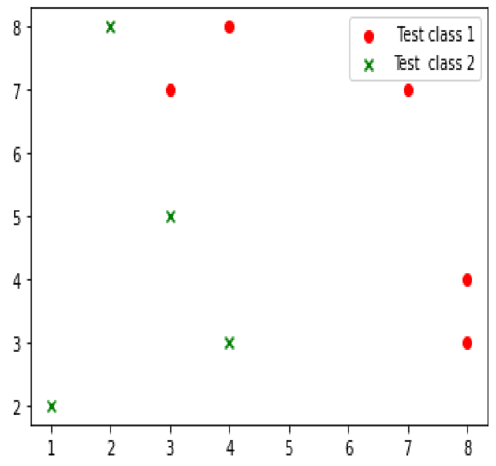

Fig. 2. Distribution of test data points as Class 1 and Class 2

III. Result Analysis

From the following figures, it can be seen that, the two classes are plotted in the figures with different colour and markers.then using knn algorithm sample points are classified.and the result is quite good.

IV. Conclusion

From following the KNN algorithm, test data are classified in their respective classes.

V. Algorithm Implementation / Code

```
#import necessary libaries
import math
import pandas as pd
import numpy as np
import io
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

from google.colab import files

traindataset = files.upload()
testdataset = files.upload()

traindataset=pd.read_csv(io.BytesIO(traindataset['train_knn.txt']), sep=",", header=None)
train=traindataset.to_numpy();
#print(train)
testdataset=pd.read_csv(io.BytesIO(testdataset['test_knn.txt']), sep=",", header=None)
#print(testdataset)
test=testdataset.to_numpy();
print(train[0])
print(test)

Trainclass1 =[([i[0],i[1]]) for i in train if i[2] == 1]
Trainclass2 =[([i[0],i[1]]) for i in train if i[2] == 2]
Trainclass1 = np.array(Trainclass1)
Trainclass2 = np.array(Trainclass2)
print(Trainclass1)
print(Trainclass2)

x = np.concatenate((Trainclass1, Trainclass2), axis=0)

fig = plt.figure()
plt.scatter(Trainclass1[:,0], Trainclass1[:,1], c = 'r', marker = 'o', label = ' class 1')
plt.scatter(Trainclass2[:,0], Trainclass2[:,1], c = 'g', marker = 'x', label = 'class 2')
plt.legend(loc = 'best')
plt.show()




def euclidian_dist(p1, p2):
    sum = 0
    sum = math.pow(p1[0] - p2[0], 2) + math.pow(p1[1] - p2[1], 2)
    return round(math.sqrt(sum),2)

new_sample = [3,7]
dists = {}
dx = []
dx1=[]
for i in range(len(x)):
    d = euclidian_dist(x[i], new_sample)
    dx.append(d)
    print(dx)
    dists[i] = d
    #print(dists[i])
dx.sort()
print(dx)
"""for i in range(3):
    print(dx[i])"""

k_neighbors = sorted(dists,key = dists.get)[:3]

#print(len(k_neighbours))
print(k_neighbors[0])
#print(dists[5])
#print(dists[7])
#print(dists[6])
"""for i in range(3):
    #dx1.append(dx[i])
print()"""
""""for i in k_neighbours:
    #print(i)"""
print(train[k_neighbors[0]][-1])


#dx1
def knn(x,train_set, new_sample, K):
    dists = {}
    for i in range(len(x)):
      d = euclidian_dist(x[i], new_sample)
      dists[i] = d

    k_neighbors = sorted(dists, key=dists.get)[:K]

    qty_label1 = 0
    qty_label2 = 0
    for index in k_neighbors:
        if train_set[index][-1] == 1:
            qty_label1 += 1
        else:
            qty_label2 += 1

    if qty_label1 > qty_label2:
        return 1
    else:
        return 2

K = int(input("Enter your value of k: ") )
l = []
for j in range(len(test)):
    label = knn(x,train, test[j], K)
    l.append(label)
test_label = np.array(l)
```

```
print(test_label)

y=np.zeros([9,3])
y[:,0]=test[:,0]
y[:,1]=test[:,1]
y[:,2]=test_label

test1 =[([i[0],i[1]]) for i in y if i[2] == 1]
test2 =[([i[0],i[1]]) for i in y if i[2] == 2]
test1 = np.array(test1)
test2 = np.array(test2)

fig = plt.figure()
plt.scatter(test1[:,0], test1[:,1], c = 'r',marker = 'o', label = 'Test class 1')
plt.scatter(test2[:,0], test2[:,1], c = 'g',marker = 'x', label = 'Test
class 2')
plt.legend(loc = 'best')
plt.show()

file = open("predict.txt", "w")

def knnx(x,train_set, new_sample, K):
    dists = {}
    dx = []
    dx1 = []

    qty_label1 = 0
    qty_label2 = 0

    for i in range(len(x)):
      d = euclidian_dist(x[i], new_sample)
      dists[i] = d
      dx.append(d)
      #print(dx)
    dx.sort()
    k_neighbors = sorted(dists, key=dists.get)[:K]



    for i in range(K):
        dx1.append(dx[i])



    for index in range(len(k_neighbors)):
        #print(index,':',dx1[index])

        if train_set[k_neighbors[index]][-1] == 1:
            print('distance ',index+1,':',dx1[index],'class',1)
            file.write("%s %s %s %s %s %s \n" %("distance",index+1,":",dx1[index],"class","1"))
            qty_label1 += 1
        else:
            print('distance ',index+1,':',dx1[index],'class',2)
            file.write("%s %s %s %s %s %s \n" %("distance",index+1,":",dx1[index],"class","2"))
            qty_label2 += 1
```

```
    if qty_label1 > qty_label2:
        #return 1
          #print(index,':',dx1[index],1)
        print('predicted class 1')
        file.write("%s\n" %("predicted
class 1"))
    else:
        print('predicted class 2')
        file.write("%s\n" %("predicted
class 2"))
        #return 2
          #print(index,':',dx1[index],2)

K= int(input("Enter your value of k: "))
for j in range(len(test)):
    file.write("%s %s\n" %("test point",test[j]))
    print('test point',test[j])
    knnx(x,train, test[j], K)
    file.write('\n')
    print('\n')
file.close()
```