ESKISEHIR OSMANGAZI UNIVERSITY
ELECTRICAL AND ELECTRONICS
ENGINEERING

2022-2023 OBJECT ORIENTED PROGRAMMING I

K-Nearest Neighbours

Serdar Söylemez -151220182033

Contents

Abstract

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered “plurality voting”, the term, “majority vote” is more commonly used in literature. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.A commonly used distance metric for continuous variables is Euclidean distance.

Introduction

The k-nearest neighbors algorithm (k-NN) was first developed by Evelyn Fix and Joseph Hodges in 1951[1] and later by Thomas Cover[2]. It is used for classification and regression. In both cases, the input consists of the k closest training examples in a data set. The output depends on whether the k-NN is used for classification or regression.In k-NN classification, the output is a class membership. An object is classified by the plural of its neighbors, and the object is assigned the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, the object is simply expected to be assigned to that nearest neighbor's class.[3]

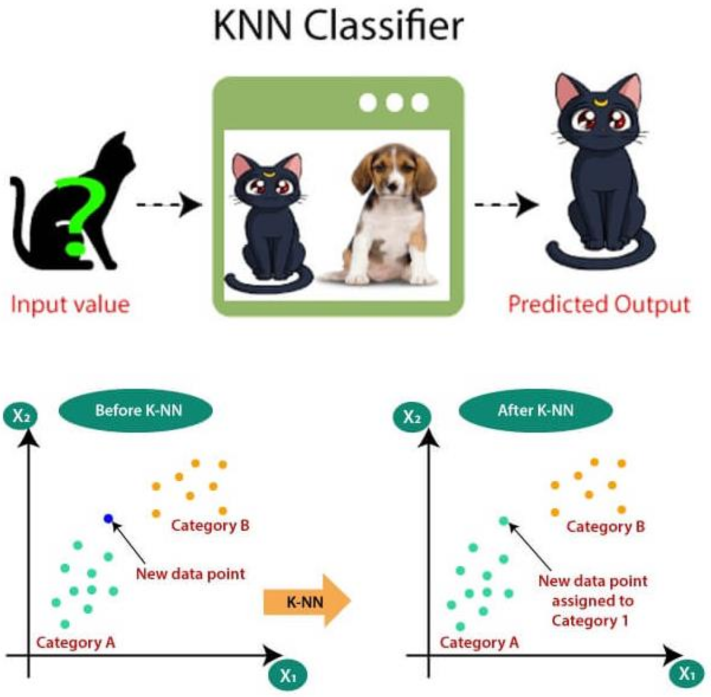

Figure 1 Example of KNN Algorithm

Project

In this project, primarily training data and test data were created. The size of the training data is 30 and the size of the test data is 10. Each training data has 2 features and 3 categories. These features are randomly selected between -5 and +5. The category value of traning data with ID value between 0 and 9 is assigned as 0. The category value of traning data between 10 and 19 is assigned as 1. The category value of traning data between 20 and 29 is assigned as 2. It is not clear. Which category was predicted as a result of the KNN algorithm. The ID values of the test data are between 30 and 39. Their features are between -5 and +5 and were determined randomly. After these features are determined, the categories of the test data are estimated by determining the Euclidean distance and plurality vote of neighbor's criterion.

Coding Hierarchy

First, I created a DataSet class in which the ID, category ID, Feature X, Feature Y which are properties of the training data used as vector. I stored each of the features I created in vector.

Then I created the KNN class, where I calculate the distance, nearest neighborhood and estimated category IDs between the test data and training data.

I created the Graph class, which visually outputs the training data and predicted category values. I also used the graphics library to visualize the data.[4]

In the driver code, I determined the properties of the training data. I created an object for each class and called them in order.

- **DataSet.h**
- **DataSet.cpp**
- **KNN.h**
- **KNN.cpp**
- **Graph.h**
- **Graph.cpp**
- **main.cpp**

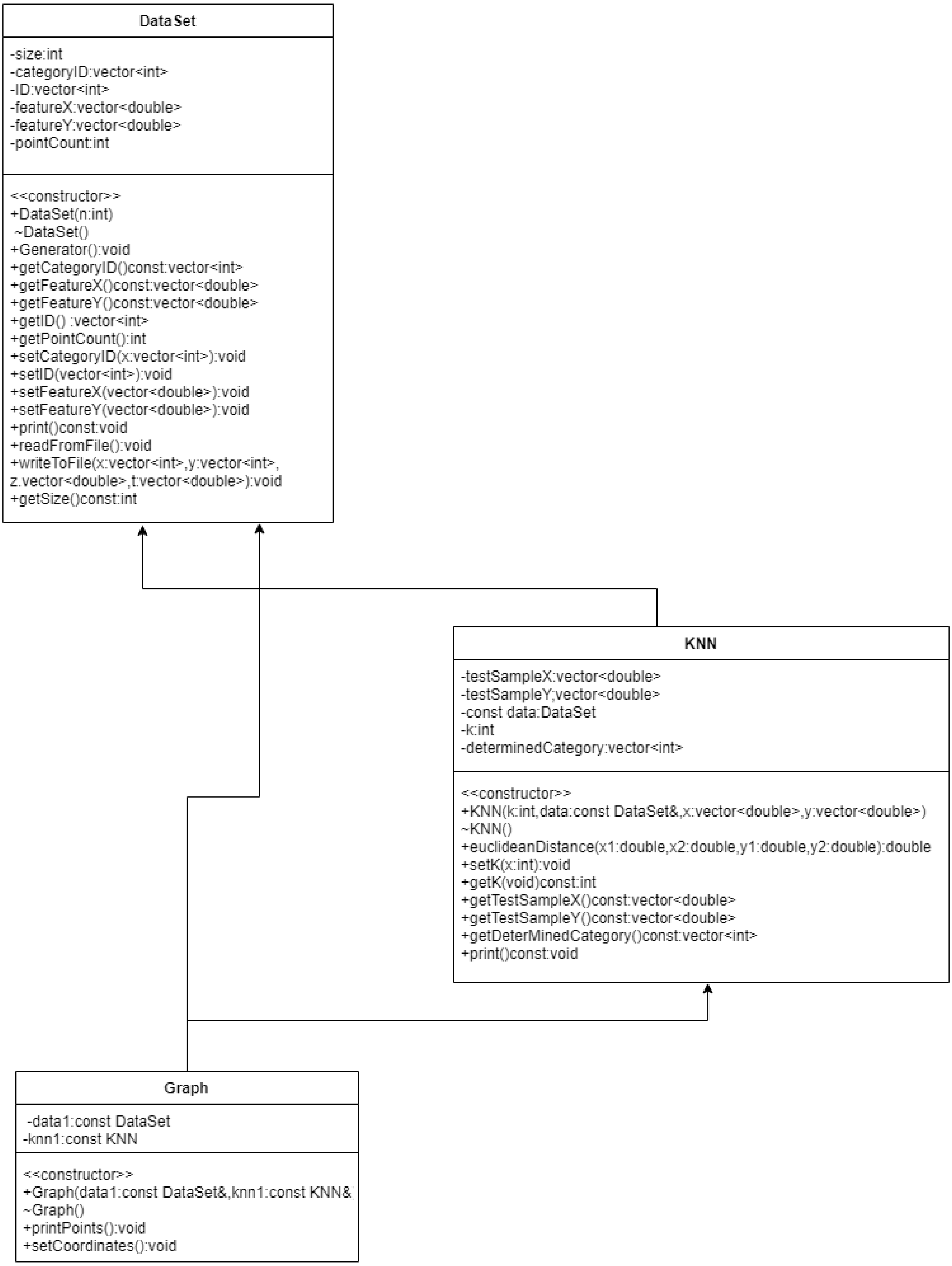
Data Set
-size:int
-categoryID:vector<int>
-ID:vector<int>
-featureX:vector<double>
-featureY:vector<double>
-pointCount:int
<<constructor>>
+DataSet(n:int)
~DataSet()
+Generator():void
+getCategoryID()const:vector<int>
+getFeatureX()const:vector<double>
+getFeatureY()const:vector<double>
+getID() :vector<int>
+getPointCount():int
+setCategoryID(x:vector<int>):void
+setID(vector<int>):void
+setFeatureX(vector<double>):void
+setFeatureY(vector<double>):void
+print()const:void
+readFromFile():void
+writeToFile(x:vector<int>,y:vector<int>,
z.vector<double>,t:vector<double>):void
+getSize()const:int
KNN
-testSampleX:vector<double>
-testSampleY;vector<double>
-const data:DataSet
-k:int
-determinedCategory:vector<int>
<<constructor>>
+KNN(k:int,data:const DataSet&,x:vector<double>,y:vector<double>)
~KNN()
+euclideanDistance(x1:double,x2:double,y1:double,y2:double):double
+setK(x:int):void
+getK(void)const:int
+getTestSampleX()const:vector<double>
+getTestSampleY()const:vector<double>
+getDeterMinedCategory()const:vector<int>
+print()const:void
Graph
-data1:const DataSet
-knn1:const KNN
<<constructor>>
+Graph(data1:const DataSet&,knn1:const KNN&
~Graph()
+printPoints():void
+setCoordinates():void

Figure 2 UML diagram of project

Results

At the end of this project, information was gained about how the KNN algorithm is used, euclidean distance, and neighborhood criteria. The data members of the classes were used as private. The constructor and destructor of each class were created. Exception handling was used if an incorrect value was entered. Set get functions were used. Functions for file writing and reading operations were written. Operator overloading was used to suppress the object. STL container and STL algorithms were used. Also composition was used. Finally, the grafics library was used to visualize the outputs.

References

[1] Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, *57*(3), 238-247.

[2] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175-185.

[3] *What is the k-nearest neighbors algorithm?* (2020). IBM: https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.

[4] https://www.youtube.com/watch?v=CHFyEnlMnxg&t=2s