Machine Learning
Project-3 K means Clustering

Saachi Shivhare - 1001830083

Abstract

This report summarizes the analysis done for KNN. It is a supervised machine learning algorithm which learns from a labeled input and produces an apt output when given new unlabeled data. Accuracy is also computed to check the efficiency of the classifier.

❖ Introduction

The k-nearest neighbors (KNN) algorithm is supervised machine learning algorithm that can be used to solve both classification and regression problem. It relies on the labeled input data to learn a function and provides output for the unlabeled unseen data.
Supervised machine learning algorithms are used to solve classification and regression problems.

A classification problem has a discrete value as its output whereas a regression problem does not have a discrete value.

The KNN algorithm assumes that similar things exist in proximity. KNN algorithm works on the idea of similarity with some mathematics for calculating the distance between points. There are multiple ways of calculating distance like Euclidean distance, Manhattan distance etc.

- The KNN Algorithm:

1. Load we load the data on which we must work.
2. Split the data into two sets – training data and testing data.
3. Initialize value of K (k implies the number of neighbors)
4. Calculate distance for each point in the testing data set, add it to a set.
5. Sort the set-in ascending order by the distances.
6. Pick the first k entries for classifying the points.
7. Get the labels of the k selected entries.
8. Return the mode of the k labels.

❖ Dataset

The data being used for this project is an Iris dataset. There are total 150 data points. 50 data points are of Iris-setosa, 50 data points are of Iris-Versicolor and 50 data points are of Iris-virginica. The first attribute is sepal length, second attribute is sepal width, third attribute is petal length, and fourth attribute is petal width. All the attributes are in centimeter. I have replaced the target variable into integer values 0 for Iris-setosa, 1 for Tris-versicolor and 2 for Iris-virginica.

5.1,3.5,1.4,0.2, Iris-setosa
4.9,3.0,1.4,0.2, Iris-setosa
4.7,3.2,1.3,0.2, Iris-setosa
4.6,3.1,1.5,0.2, Iris-setosa
5.0,3.6,1.4,0.2, Iris-setosa
5.4,3.9,1.7,0.4, Iris-setosa
…
7.0,3.2,4.7,1.4, Iris-versicolor
6.4,3.2,4.5,1.5, Iris-versicolor
6.9,3.1,4.9,1.5, Iris-versicolor
5.5,2.3,4.0,1.3, Iris-versicolor
6.5,2.8,4.6,1.5, Iris-versicolor
…..
7.7,3.0,6.1,2.3, Iris-virginica
6.3,3.4,5.6,2.4, Iris-virginica
6.4,3.1,5.5,1.8, Iris-virginica
6.0,3.0,4.8,1.8, Iris-virginica
6.9,3.1,5.4,2.1, Iris-virginica
6.7,3.1,5.6,2.4, Iris-virginica

❖ K-nearest neighbor

A classifier is a machine learning model that is used to discriminate different objects based on certain features.

In this project, I first shuffled the data using random library and then divided the data into 80% - 20% i.e. 80 % training data and 20% testing data. After shuffling the data, for each data point in testing set, I computed the Euclidean distance between the testing data point and the training data points. After computing the Euclidean Distance, sorted the values and choose first k values to find the most frequent label as the target label. Value of K must be odd.

Euclidean distance between two points (x1, y1) and (x2, y2) is given by,

$$\text{Distance} = ((x2-x1)^2 + (y2-y1)^2)^{1/2}$$

After computing the label, I also calculated the accuracy of the model, which is given by,

$$\text{Accuracy} = a/b * 100$$

a: correctly identified datapoints
b: total number of datapoints

❖ Libraries used

- NumPy
- Random
- Math
- Operator
- Counter

❖ Preprocessing

To develop a fair classifier, I have used random function from random library which divides the data into two sets i.e. training set and testing set.

I have also separated the label from the data as we will not be using the labels for training the model. As I divided the data into two sets, in similar way I have divided the labels in two sets. I have also updated the string labels by integer values. Iris-versicolor is replaced by integer 1.0, Iris-virginica is replaced by 2.0 and Iris-setosa is replaced by 0.0.

❖ Training the Classifier

After preprocessing, the number of samples used for training the model are 105 and number of samples used for testing the model are 45.

In KNN, while training the model I computed the Euclidean Distance for each testing sample and training sample. The computed distance is then sorted in ascending order and first k values are considered for finding the most frequent label. I have used lambda function to sort the values by the Euclidean Distance.

❖ Testing the Classifier

After computing the labels for all the testing samples, I compared the predicted label with the actual label.

In the output I have shown the testing sample, which is being tested and its k Euclidean Distances, from a particular training sample. I have also displayed its actual and predicted label. The most frequent label with its frequency has also been displayed. I have used counter from collections to compute the most common label.

In this way, we get the clear picture of every sample if it has been classified correctly.

Below is one such example,

```
Calculating Euclidean Distance for Test data: [6.1, 2.9, 4.7, 1.4]
--TEST DATA               TRAIN DATA          EUC. DISTANCE         LABEL  PREDICTED LABEL--
[6.1, 2.9, 4.7, 1.4]    [6.1, 3.0, 4.6, 1.4]    0.14142135623730995     1.0     1.0
[6.1, 2.9, 4.7, 1.4]    [6.1, 2.8, 4.7, 1.2]    0.22360679774997896     1.0     1.0
[6.1, 2.9, 4.7, 1.4]    [6.0, 3.0, 4.8, 1.8]    0.43588989435406733     1.0     2.0
Considering K nearest neighbour 3
Most Common LABEL & its frequency: [(1.0, 2)]
```

In this example we can see that currently the model is working on [6.1, 2.9, 4.7, 1.4]. Now, compute the distance from each training sample. I have only displayed the first k distances after sorting the distances. The actual label and the predicted label are also

displayed. In the above example, one label is incorrectly predicted whereas remaining two labels were correctly predicted.

I followed same steps for the remaining test samples and kept incrementing the counter for the correctly classified samples.

After predicting all the labels, I have displayed all the labels as mentioned below. In this we can easily see which samples of the training set were correctly classified and which were not.

```
++++++++++++++++++++++++++++PREDICTED LABELS AND ACTUAL LABELS ++++++++++++++++++++++++++++
Most common labels with their frequency and actual labels for the training samples are displayed below:


PREDICTED      ACTUAL
[(2.0, 3)]      2.0
[(0.0, 3)]      0.0
[(1.0, 3)]      1.0
[(0.0, 3)]      0.0
[(0.0, 3)]      0.0
[(1.0, 3)]      1.0
[(1.0, 3)]      1.0
```

Accuracy of the model is also computed.

❖ Results

I have used two values of k i.e. K = 3 and K = 5 but, in both the case out of 45 samples only one sample was incorrectly classified. Accuracy of the model in both the cases is 97.77 %.

❖ Summary

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification problems. It is easy to implement and understand but has a major drawback of becoming significantly slows as the size of that data in use grows.

In the case of classification, we saw that choosing the right K for our data is done by trying several Ks and picking the one that works best. I tried two K values and got same accuracy both the times.

❖ References

1. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
2. https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26
3. https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
4. https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/
5. https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7
6. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761