K-NN

Thursday, July 2, 2020 08:19

For the glory of God

What is the K-NN (K-Nearest Neighbor) algorithm ?

· It is a type of instance-based learning where the function is only approximated locally.

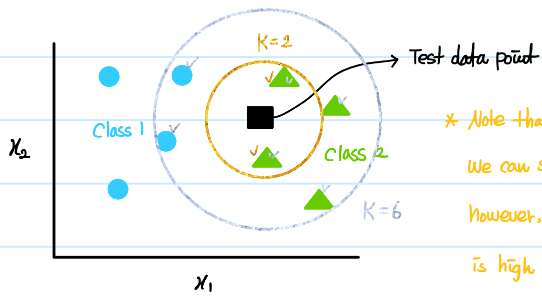
↳ It's a family of learning algorithms that does not perform model generalization but construct hypotheses directly from the training instances themselves.

· It can be used for both classification and regression.

- Classification : the output is a class membership that is classified by a plurality vote of its neighbors
- Regression : the output is a value that is the average of k nearest neighbors

How does the K-NN algorithm work ?

· Let us suppose that we are expected to solve the following classification problem ;



* Note that :

We can specify a range instead of specifying the k value ;

however, it may experience curse of dimensionality if the domain is high dimension.

· At a glance, it's easy to say that the test data point is highly likely classified as the class 2.

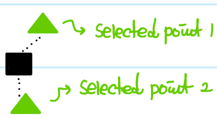
· Note that the decision made by eyeball judgement !

· The question is 'how does a computer figure out the decision by itself ?'

- The answer would be as follows : (i.e. by using the K-NN algorithm)

a.) Specify the hyperparameter (k) ⇒ It's always challenging to isolate the best k value.

e.g. If k = 2, the algorithm will be looking for two points that are closest to the test data point.



; Here, note that the algorithm would take one of distance metrics (e.g. Manhattan distance) to calculate distance between two points in a way that the algorithm could figure out which pairs are closest to the test data point.

e.g. If Euclidean distance is chosen as the metric, the algorithm would use the following equation ;

$$\text{Distance}(\text{Point 1}, \text{Point 2}) = \sqrt{\sum_{i=1}^{n} (\text{Point 1}_i - \text{Point 2}_i)^2}$$

b) Determine a class membership based on the selected points

e.g. For this example, the algorithm would generate the output as a green class.

· When applying the K-NN algorithm, keep in mind that normalization is necessary to reduce an error related to calculating distance between two points.

- Min Max normalization is typically recommended ;

$$x_{normalized} = \frac{x_{original} - \min(x)}{\max(x) - \min(x)}$$

Pros and Cons of the K-NN algorithm

Advantages

· It's easy to implement.

· It's a robust algorithm especially if it deals with large volume of training data.

· It does not require to generalize a model. (i.e. instance-based learning)

↳ Thus, it is not affected by the quality of data.

Disadvantages

· It is challenging to choose the best k parameter properly.

Think about the Brute-Force algorithm

· It is computationally expensive especially if it must account for calculating distance with many data points.

↳ Thus, there have been many attempts to reduce the computational complexity.

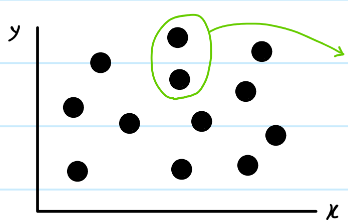
e.g. Ball Tree

Algorithms for improving the efficiency of K-NN algorithm

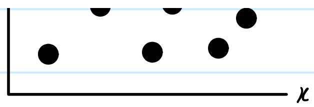
Introduction

· As we discussed, even if the k-NN classifier is a simple algorithm that can be applied to a variety of engineering problems, one concern is that it's computationally expensive.

- To be more specific, let's see the following example ;



Is this a pair of points with the smallest distance?



; Here, the question is "Given N points in the xy plane, find a pair of points with the smallest Euclidean distance"

· In fact, we can employ the Brute-Force algorithm ; thus, the time complexity is $O(N^2)$.

↳ But can we do better in case N increases dramatically ?

- Actually, there have already been many attempts to answer the question.

- We will be talking about a few representative algorithms (e.g. Ball tree) in this hand-written note.

KD (K-Dimensional) Tree

· In computer science, KD Tree is a space-partitioning data structure for organizing points in a K dimensional space.

· The KD Tree is a binary tree in which every non-leaf node can be thought of as implicitly generating a splitting hyperplane that divides the space into two parts, known as half-spaces.

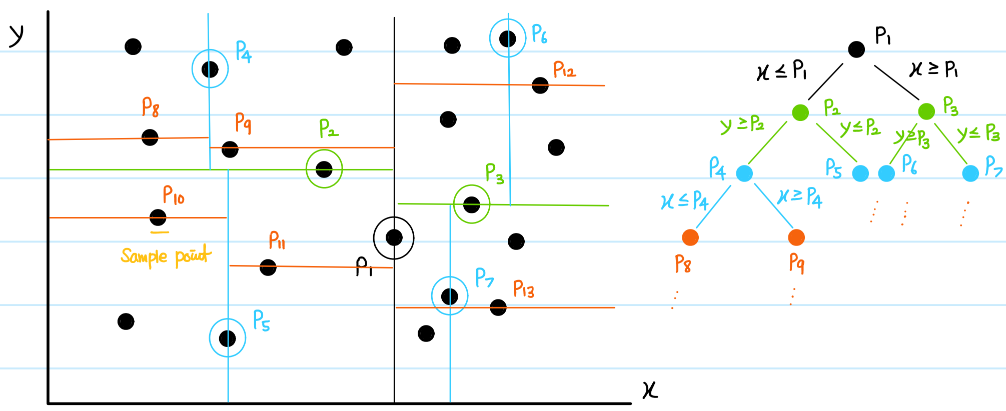
It is a tree data structure in which each node has at most two children.

· Why does the KD Tree do better than the naive (O^2) approach ?

- The basic idea is that it neglects to calculate some points that are far from the sample point.

- How does it work ? Let's suppose that we want to find a pair of points with the smallest distance between them.

↳ Note that we first need to build a KD Tree with the data points in order to find out the pair of points ;



Step 1) We begin by choosing a root node. Typically, the median value in X axis is selected.

↳ Then we split the data points into two groups.

Step 2) We recursively build the KD Tree in both left and right half-spaces by picking other points to split.

↳ In this case, we are going to select the median value in y axis from each half-space.

Step 3) We continue this construction to completion.

↳ Note that we need to cut X axis → Y axis → X axis → …

e.g. $O(N \log N)$ vs. $O(N^2)$

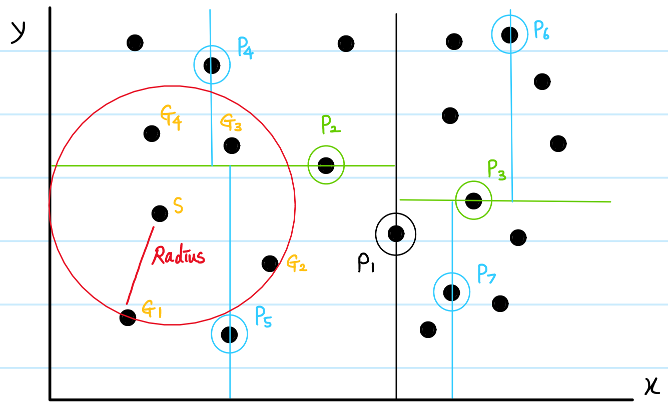
Step 4) Our resulting KD Tree will look like this. Note that a tree structure is better than other structures.

↳ At this point, we have generated the KD Tree for Nearest Neighboring search.

Step 5) Now, the question is "Which point in the KD Tree is closest to the sample point?"

e.g. What is the closest waypoint nearby the position of aircraft?

↳ Let's see how to answer the question.



· Suppose that the point "S" is the sample point.

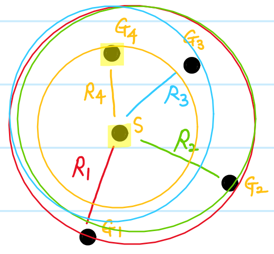
· Given the KD Tree, we know that the root node P_5 has two leaves that include S and G_1.

· Then we can guess the pair ($S-G_1$) is the closest point. (i.e. current guess)

· Based on the current guess, we need to draw a circle with the Radius that is connected by the line of two points.

↳ { If there is a point in the circle, we need to update the current guess.
If not, the current guess is so correct that they are the closest point in the xy space.

· For example,

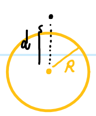


⇒ Note that this is just a notional sketch.

↳ This is true : $R_1 > R_2 > R_3 > R_4$

; Here, within R_1, we have three other points. Thus, it's not nearest neighbor.

within R_2, we have two other points. ⇒ How can we say this mathematically?



if $d \leq R$

within R_3, we have another point.

within R_4, there is no point ; therefore, they are a pair of points closest between them.

· In this example, we used a circle to define a region ; however, it's worth mentioning that it would be a sphere in three dimensions.

· This is critical because we can start eliminating parts of trees (i.e. data points) that are not necessary to be considered as a candidate.

Ball Tree (i.e. advanced KD Tree)

· Although the KD Tree is an amazing algorithm to reduce the complexity of K-NN, the obvious downside of the algorithm is related to curse of dimensionality.

↳ In other words, the KD Tree only works well given that K is reasonably small.

- In general, the KD Tree is effective for $N > 2^K$

· The Ball Tree has been introduced to deal with the issue related to curse of dimensionality.

↳ It's also a space partitioning data structure in which it gets its name from the fact that it partitions data points into a nested set of hyperspheres known as "Balls".

· The key difference between two algorithms is as follows ;

- The KD Tree partitions data points with Cartesian axes such as $X/Y/Z$.

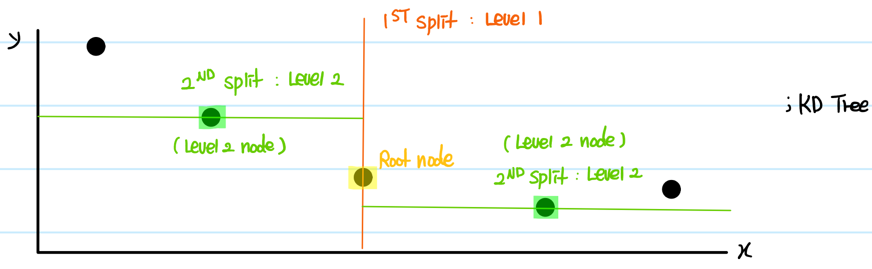
- The Ball Tree partitions data points with a nested set of hyperspheres.

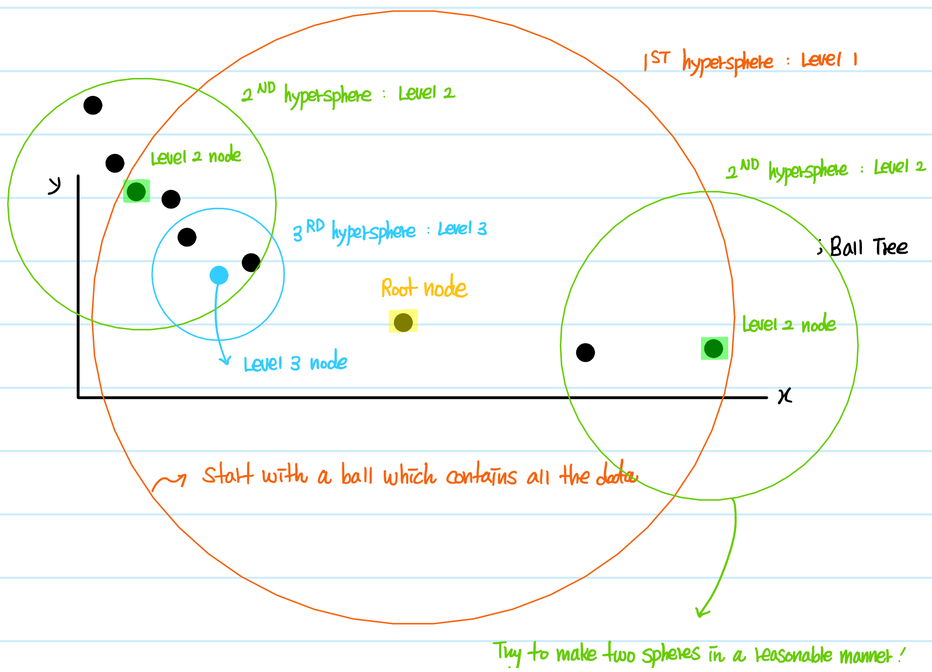
↳ Because of this hypersphere structure, the Ball Tree is more complex than the KD Tree especially for generating a tree.

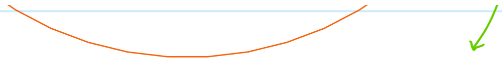
e.g. $\begin{cases} \text{Ball Tree} = O(N \log N) \\ \text{KD Tree} = O(N) \end{cases}$

↳ However, the Ball Tree works well with high K values (i.e. high dimensionality).

- For example,







Try to make two spheres in a reasonable manner!

- The specific criteria (i.e. reasonable manner) will depend on the type of question being answered and the distribution of data.
- Specifying the criteria is a difficult task but there are several heuristics that partition the data well in practice.

↳ In general, it aims to minimize the total volume of its internal nodes.