International Journal of Computer Applications (0975 – 8887)
Volume 185 – No.17, June 2023

Developments in KD Tree and KNN Searches

Vijay R. Tiwari
Jai Hind College
Churchgate
Mumbai, India

ABSTRACT

KNN (K-nearest neighbor) is an important tool in machine learning and it is used in classification and prediction problems. In recent years several modified versions of KNN search algorithm have been developed and employed to improve the efficiency of search. KNN has enormous real life applications and is widely used in data mining. Data structures like KD tree (or K dimensional tree) are used for implementing KNN effectively. A KD tree is a multidimensional binary search tree that can be balanced or unbalanced. With the increase in dimension of space the computational time of KNN-KD search goes high. Certain modifications that can help in improvising the search time has been developed in recent years.

General Terms

Data mining and Data Structures.

Keywords

KNN search, KD tree, Supervised Machine Learning, KNN-KD tree, Point Cloud Data.

1. INTRODUCTION

KNN (K – Nearest Neighbor) is a non-parametric lazy learning algorithm and is used to find K (n positive integer) nearest neighbor of a given query point by searching every point in the dataset. In lazy learning, machine stores the data at training time and delays learning until classification time. The nearest neighbor search was first introduced in 1951, by Fix and Hodges in their technical report which was never published as an official paper. Later Silverman and Jones [22] wrote commentary on this technical report thereby re-introducing the concept briefly. Cover and Hart [6] first proposed that KNN can be used to solve classification problems in machine learning. KNN performs best when points are in lower dimension but increase in dimension creates an over-fitting problem and exponential growth in computation time [30]. Grouped techniques like weighted, reduction, additive, reverse, continuous, principal axis, etc. are used for implementing nearest neighbor search [18]. KNN search is used in pattern recognition, data compression, computational statistics, information retrieval, databases and data mining, etc. It labels the given object by finding the most similar labelled objects and copying their labels. There are numerous distance measuring formulas used in metric and non-metric spaces to calculate the distance of a query point from points in the dataset. Performance of KNN is affected by the type of distance formula used. Andoni [1] proved that the classical approach of nearest neighbor search fails when used with certain distance formulas like string edit distances. This problem is solved by using the Ulam distance formula.

A data structure is a format of data organization, management and storage. It can be thought of collection of data values and their relations. Different data structures are used for implementing nearest neighbor search. Ball tree, KD tree, LSH (Locally sensitive hashing) are few of them. These techniques use structure for indexing points and searching points.

Binary search tree is a key in building a multidimensional binary search tree. KD trees are binary trees which stores K-dimensional data. It is used to partition space into smaller number of cells in a hierarchical manner. A major challenge in optimizing search arises when KD is unbalanced. In an unbalanced tree nodes appear to cluster heavily on one side. There is no choice to reach the leaf node on heavy side as cutting branch is impossible. Traversing each and every node is required thereby, increasing the run time and making the search linear. This issue is resolved by building a balanced tree in which runtime is logarithmic. A balanced KD tree is built by dividing the data points using median. Bentley [3] showed that if the runtime of finding median, of n data points is of order n, then runtime of building balanced KD tree is of order $n \log n$. It is complicated to find an algorithm for computing median with runtime of order n. Quicksort algorithm [10] finds median with run time of order n in the best case. Merge sort algorithm [8] computes median whose run time is of order $n \log n$. This helps in building balanced KD tree whose runtime is of order n. Heapsort algorithm [23] does the same process. Improved version of KD tree search algorithm [20] performs well in case when data set points belong to higher dimension. This algorithm finds an c approximate nearest neighbor where $c > 1$ is a parameter denoting how closer is the searched point from its neighbors. An c approximate nearest neighbor is a point at most c times the distance of nearest neighbor. Brown [4] proposed a method of building balanced KD tree for n points in k - dimension whose runtime is of order $kn \log n$ in the worst case. Zhai [25] proposed an improved KNN algorithm that increases the efficiency of classification and it improves KNN search, by combining Principal component analysis (PCA) with KD tree data structures.

2. BINARY SEARCH TREE

A binary search tree is an enhanced binary tree where a node will have a child to the left if the key value of the child node is lesser than or equal to the key value of parent node value. The child node will be attached to the right if the key value of the child node is greater than the key value of the parent node. A binary search tree can be skewed or balanced. If the left sub-tree and right sub-tree have the same number of nodes then the tree is balanced otherwise it is called skewed. A diagram of balanced binary search tree shown in Fig 1:

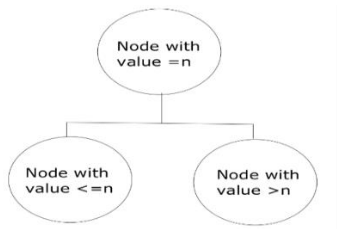

Fig 1: Balanced Binary Tree of level 1

3. KD TREE

KD tree is a K-dimensional non-linear data structure, which is used for space partitioning based on certain conditions. It can be balanced or unbalanced. It is used for storing the data in an efficient manner so that certain search algorithms can be implemented in optimal way thus reducing the computation time. Balanced binary search tree leads to faster construction of space partitions and new query search is also optimal in these cases.

3.1 Construction of balanced

For given points in K-dimension, a balanced binary search tree is constructed using median. If $m_1, m_2, \cdots, m_n$ are n discrete data points the median is $m_{\frac{n+1}{2}}$, if n is odd. If n is even the median is $\frac{1}{2}\left(m_{\frac{n}{2}} + m_{\frac{n}{2}+1}\right)$.

Example: Data points (1, 3), (5, 7), (2, 6), (9, 4), (4, 8), (7, 10), (3, 9).

Step 1: Sorting in ascending value of first coordinate (i.e. x coordinate), the new data set is (1, 3), (2, 6), (3, 9), (4, 8), (5, 7), (7, 10), (9, 4). Median of $x\ coordinate$ of these points is 4. Thus select the root node as (4, 8). Draw the line $x = 4$ as shown in fig 2:

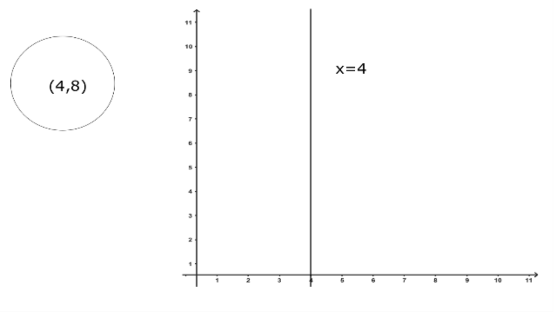

Fig 2: Step 1 Balanced Search Tree

Step 2: Left Sub-Tree

For the left sub-tree take the points with $x\ coordinate$ less than or equal to 4. The points are (1, 3), (2, 6), (3, 9). Now sorting these points in ascending value of the second coordinate (i.e. $y\ coordinate$) the new list is (1, 3), (2, 6), (3, 9). Median of $y\ coordinate$ of these points is 6.Thus, (2, 6) is selected as root of the left sub-tree. Compare $y\ coordinate$ of these points with $y\ coordinate$ of (2, 6). Thus (1, 3) will form the left child and (3, 9) will form the right child. Draw the line $y = 6, x = 1$, and then $x = 3$ as shown in fig 3:

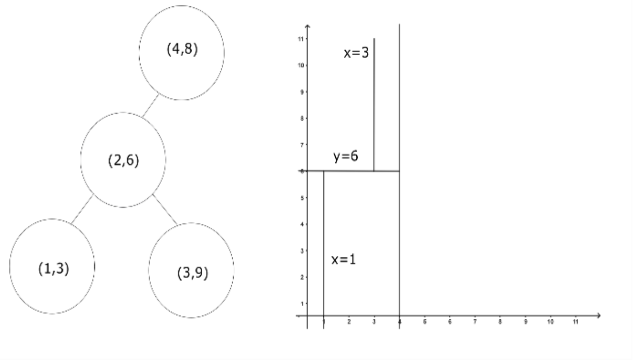

Fig 3: Step 2 Balanced Search Tree

Step 3: Right Sub-Tree

For the right sub-tree, points with $x\ coordinate$ greater than 4 are selected and they are (5, 7), (7, 10), (9, 4). Arrange these points in ascending value of $y\ coordinate$ the new list as (9, 4), (5, 7), (7, 10). Median of $y\ coordinate$ of these points is 7. Thus the root of the right sub-tree is (5, 7). Now (9, 4) will form the left child and (7, 10) will form the right child as shown in fig 4:

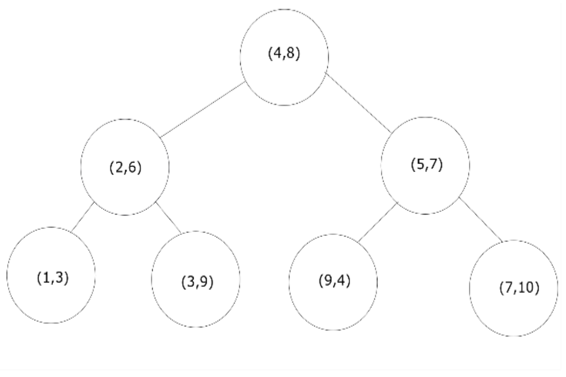

Fig 4: Step 3 Balanced Search Tree

Draw the lines $y = 7, x = 9$ and $x = 7$ as shown in Fig5:

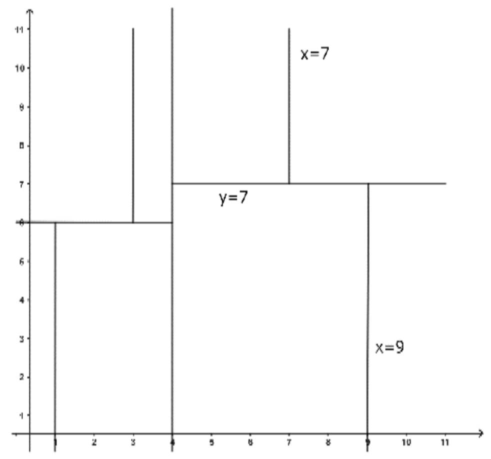

Fig 5: Step 3 Space Partition

4. K NEAREST NEIGHBOR (KNN)

KNN is a branch of supervised machine learning used to solve classification and prediction problems that occur in daily life. K is an integer denoting the number of neighbors taken for that particular problem. Methods like cross validation is used to find the optimal values of K. Trial and error method on given data with different values of K is also used to find the optimal number of neighbors. Important steps in KNN algorithm are:

1. Find the optimal value of K as mentioned above.
2. Calculate the distance between the query observation and given observations.
3. Rank the observations based on the distances measured in increasing order with rank 1 given to observations with minimum distance.
4. Arrange the observations based on rank and determine the K nearest neighbors.

5. Gather the category of nearest neighbors.
6. The majority nature of the category of these K neighbors is the prediction of the query observation.

4.1 Example on Classification

Consider the data in Table 1 of a soap manufacturing company:

Table 1: Soap Data

Potassium Content in mg (x)	PH (y)	Category
12.6	7	Good
9.5	8	Bad
9.8	9	Bad
12.1	10	Good
12.8	7	Good
9.1	9	Good
12.2	8	Bad

Company manufactures a new soap with Potassium content equal to 12 grams and PH value 9. Classify new soap a good or bad by doing three neighbor test and using square of Euclidean distance which is given by

$$d\big((x_1, y_1), (x_2, y_2)\big) = (x_1 - x_2)^2 + (y_1 - y_2)^2.$$

Data given in Table 2 is obtained after finding the distance and arranging the data based on rank.

Table 2: Arranged Data

Potassium Content in mg (x)	PH (y)	Category	$d((x, y), (12,9))$	Rank
12.1	10	Good	1.01	1
12.2	8	Bad	1.04	2
12.6	7	Good	4.36	3
12.8	7	Good	4.64	4
9.8	9	Bad	4.84	5
9.5	8	Bad	7.25	6
9.1	9	Good	8.41	7

The first three ranks are the three nearest neighbors of the query observation (12,9). From these three nearest neighbors two are in good category. Since the major category falls under the good category, thus the soap is declared good.

4.2 Example on Prediction

Given set of observations: $(1, 4), (2, 9), (4, 8), (7, 2), (9, 9), (3, 5), (6, 3), (8, 0)$ and if the query observation is (5, -) with missing y coordinate then use distance formula: $d(x_1, x_2) = |x_1 - x_2|$.

Table 3Distance from Query

Observations$(x_1, y_1))$	$d(x_1, 5)$	Rank
(1, 4)	4	4
(2, 9)	3	3
(4, 8)	1	1
(7, 2)	2	2
(9, 9)	4	4
(3, 5)	2	2
(6, 3)	1	1
(8, 0)	2	2

If $k = 5$ then there is a tie in observations with rank 1 and rank 2 hence choose $k = 2$ in order to avoid such clashes. The two nearest neighbors are $(6, 3)$ and $(4, 8)$. The predicted $y\,coordinate$ of the query (5, -) is the average of $y\,coordinates$ of the nearest neighbors and it is $\frac{3+8}{2} = 5.5$.

4.3 Improving KNN

The computation cost of KNN search is high and the accuracy of classification is affected by the selection of distance formula. Survey report [12] gives three main issues in implementing KNN along with some proposed solutions.

4.3.1 KNN is affected by the Choice of Distance Formula Selected.

A data point x having n attributes say $A_1, \dots, A_n$ is denoted by the vector $\big(\big(a_1(x), \dots, a_n(x)\big)\big)$, where $a_i(x)$ denotes value of attribute A_i, for $1 \le i \le n$. In most cases Euclidean distance is employed that uses each coordinate for measuring distance. Thus for large values of n computation cost is very high. A new approach [17] eliminates least relevant attributes from the space thereby reducing the computation time. Another method calculates the distance by attaching weight to each attribute depending on their importance. If $w_1, \dots, w_n$ are the weights attached to attributes $A_1, \dots, A_n$ then the distance formula used for measuring the distance between any two instances say x and y is

$$d(x, y) = \sqrt{\left(\sum_{i=1}^{n} w_i^2 \big(a_i(x) - a_i(y)\big)^2\right)}$$

4.3.2 KNN is affected by Method Used for Finding K

Finding the optimal number of neighbors is important for accuracy of classification. SNSB (Selective neighborhood naïve Bayes) model [24] finds the best value of K. The computation time of this model is high hence it is less efficient. Another efficient method to find the value of K is cross validation. It is done by crossValidate () function in the file ofweka.classifier.lazy.IBK.java. DNNAW (Dynamic K-nearest neighbor naïve Bayes with attribute weighted) [13] determines the best value of K by combining SNSB with weighted attribute technique. In this method, attributes are weighted first and then machine learns a local naïve Bayesian classifier for the test data.

4.3.3 KNN is affected by method used for voting

A simple method to classify or label a query attributes is to label it with maximum vote obtained from labels of its K neighbors. KNNDW (K-nearest neighbor with distance weighted) [17] is another approach to label the query. In this method votes of different neighbors are weighted based on their distance. Combining KNN with naïve Bayes [15] is another technique. LWNB (locally weighted naïve Bayes) [7] weighs K nearest neighbors first and then built a local naïve Bayes. ICLNB (Cloning local naïve Bayes) generates a number of clones of each neighborhood. These clones are added to training data and then naïve Bayes is trained on this expanded data.

5. APPLYING KNN IN KD TREE

Finding K nearest neighbors of a query observation using KD trees is done by traversing a sub-tree based on minimum distance. This process helps in pruning a sub-tree, thus saving the time in finding the nearest neighbor. The method for two dimensional data is discussed and it can be generalized to any data set in finite dimension. The smallest box containing all the data points in a region of a space partition is called a tight box. If the query observation belongs to any tight box then the minimum distance is set to be infinite.

5.1. Steps in Algorithm

1. Compute distance of query and root and call this as minimum distance.

2. At level one if $x\ coordinate$ of query is greater than $x\ coordinate$ of root then traverse the right sub-tree first and then left sub-tree otherwise traverse left sub-tree first and then right sub-tree.

Sub-tree traverse (right or left):

- Traverse each data point in a sub-tree using $x\ coordinate$ and $y\ coordinate$ alternate.
- While traversing a sub-tree check if the query belongs to tight box corresponding to left sub sub-tree or right sub sub-tree. If it does not, then find the distance between the query point and the tight box. If it is less than minimum distance then traverse that sub sub-tree or else prune that sub sub-tree.
- Find the distance between query observation and data point at each level of sub sub-tree. If this distance is less than current minimum then update this value as minimum distance.

3. From step 2, minimum distance and one nearest neighbor is found. Delete the nearest neighbor and repeat all processes again to get the second nearest neighbor. Continuing to get all K nearest neighbors.

Example: Data points (1, 3), (5, 7), (2, 6), (9, 4), (4, 8), (7, 10), (3, 9). Query observation (4.5, 10).

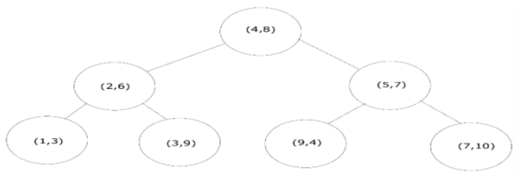

Figure 6: Nearest Neighbor

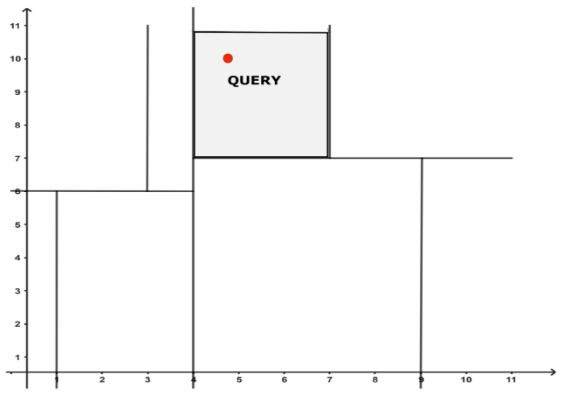

Figure 7: Nearest Neighbor

Step 1: At level 1, root is (4, 8) and distance of query from root is $d\big((4,8),(4.5,10)\big) = (4-4.5)^2 + (8-10)^2 = 4.25$. Therefore set the minimum distance to 4.25. Note that $query\ x\ \ coordinate(4.5) > root\ x\ coordinate(4)$ traverse right sub-tree first and then left sub-tree. Since the query lies in the space partition corresponding to the right sub-tree, do not measure the distance of the query from the tight box corresponding to this sub-tree.

Step 2 (Right sub-tree traversal):

At level 2, root is (5, 7) (see fig 5.1.1), and the distance of query from root is $d\big((5,7),(4.5,10)\big) = (5-4.5)^2 + (7-10)^2 = 9.25$. Since this distance is greater than the set minimum distance (which is 4.25), do not update minimum distance. Since $query\ y\ coordinate\ (10) > root\ y\ coordinate\ (7)$ traverse right child first and then left child.

Right child traversal
Right Child is (7, 10) and $d\big((7{,}10),(4.5,10)\big) = (7-4.5)^2 + (10-10)^2 = \ 6.25$. Since this distance is greater than the set minimum distance (which is 4.25), do not update minimum distance.

Left child traversal
Left Child is (9, 4) and $d\big((9,4),(4.5,10)\big) = (9-4.5)^2 + (4-10)^2 = \ 56.25$. Since this distance is greater than the set minimum distance (which is 4.25) do not update minimum distance. Now the query lies in right part of partition thus find the distance of query from tight box corresponding to left sub-tree as shown in fig 8:

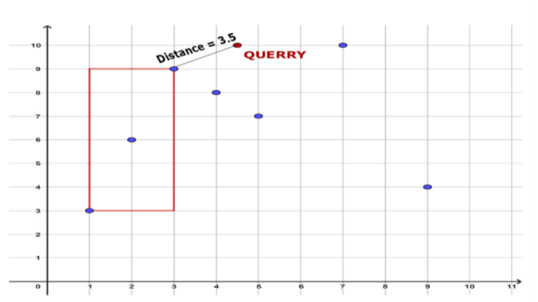

Figure 8: Query Distance from Tight Box

Step 3 (Left Sub-tree traversal)
At level 2, root is (2, 6) (fig 6) and the distance of the query from root is $d\big((2,6),(4.5,10)\big) = (2-4.5)^2 + (6-10)^2 = 22.25$. Since this distance is greater than the set minimum distance (which is 4.25), do not update minimum distance. Since $query\ y\ coordinare\ (10) > root\ \ coordinate(6)$ thus traverse right child first and then left child.

Right child traversal
Right Child is (3, 9) (fig 6), thus $d\big((3,9),(4.5,10)\big) = (3-4.5)^2 + (9-10)^2 = \ 3.25$. Since this distance is less than the set minimum distance (which is 4.25), update minimum distance = 3.25.

Left child traversal
Left Child is (1, 3) (fig 6), thus $d\big((1,3),(4.5,10)\big) = (1-4.5)^2 + (3-10)^2 = \ 61.25$. Since this distance is greater than minimum distance (which is 3.25), do not update minimum distance. Thus the minimum distance is 3.25 and nearest neighbor is (3, 9). This neighbor does not belong to the space partition of the query.

5.2 Pruning in KD Trees

KNN search is enhanced when used with KD tree as a branch of a tree can be pruned thereby reducing the cost of measuring distance from each point in the dataset.

Example: Data points (1, 3), (5, 7), (2, 6), (9, 4), (4, 8), (7, 10), (3, 9) and query observation (7, 2). Following the steps of algorithm described in section 5.1 the first nearest neighbor obtained is (9, 4). This process is completed by pruning the left sub-tree (light shaded part), as shown in the fig 9:

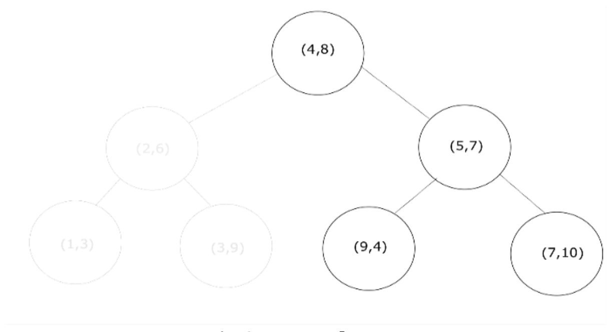

Fig 9: Pruned Tree

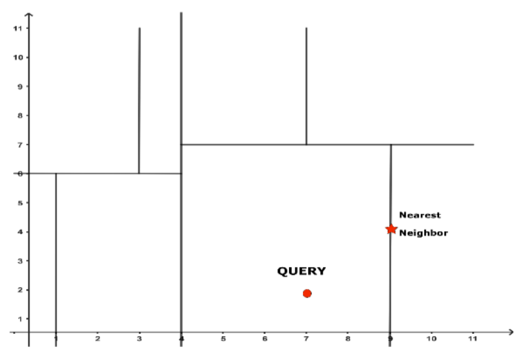

Fig 10: First Nearest Neighbor

6. TIME COMPLEXITY

The time complexity known as $\boldsymbol{Big\ O}$ is a mathematical notation denoting order of a function. It measures the time taken to run the algorithm as the input size grows.

Definition ([27], Big O notation)
For a positive integer n, we say $f(n)$ is big oh of $g(n)$ if there are positive constants c and k depending on f only such that $0 \leq f(n) \leq c\ g(n)$ for every $n \geq k$.

Example: A function f is given as $f(n) = n^2 + 2n$. Since $2n \leq n^2$ if $n \geq 2$. Thus $n^2 + 2n \leq 2\,n^2$ for every $n \geq 2$. Choose $g(n) = n^2$, $c = 2$ and $k = 2$ $0 \leq f(n) \leq c\ g(n)$. Hence $f(n)$ has time complexity of order n^2.

The bubble sort algorithm has the time complexity of order n^2. Thus to sort 1000 numbers bubble sort will require $1000^2 = 1000000$ steps. The same sorting can be done by a quick sort algorithm in fewer steps. The time complexity of quick sort is of order $n \log_2 n$. Thus to sort 1000 numbers quicksort will require $1000\ \log_2 1000 = 10000$ steps. The logarithmic runtime is faster than quadratic. $Big\ O$ gives the upper bound of the time required to run an algorithm. This is called the worst case of time complexity of an algorithm.

Time Complexity of Building KD Tree Building KD tree that employs $\boldsymbol{O(n)}$ median finding search algorithm has time complexity $\boldsymbol{O(n \log n)}$. Whereas KD tree that employs $\boldsymbol{O(\log n)}$ median finding search algorithm has time complexity $\boldsymbol{O((n \log n)^2)}$ as time taken $\boldsymbol{\log n * n * \log n}$. Finding nearest neighbor in balanced KD tree has time complexity $\boldsymbol{O(\log n)}$. Deleting root node has time complexity $\boldsymbol{O(n^{(k-1)/k})}$ and deleting a random node has time complexity $\boldsymbol{O(n\ \log n)}$.

6.1 Run-Time and Accuracy Comparison between KNN and KNN – KD Search

Experiment [11] show that KNN-KD search was much better when dataset consisted of n points of k dimension with $n \gg 2^k$. Experiment was performed with eleven UCI dataset on a computer with win10 system, Intel i5-4200U, 4GB RAM, 64-bit operating system and Pycharm 2017. The run-time of KNN-KD is lesser than the run-time of KNN search whereas accuracy remains almost the same. The runtime and accuracy comparison is shown in table 4.

Table 4: Run Time and Accuracy Comparison

Data set	KNN search	KNN-KD search	% Rate of increase	Accuracy % of KNN	Accuracy % of KNN-KD
Iris	0.02923	0.01696	72.347	96.33	96.333
Haberman	0.14135	0.05747	145.954	74.146	74.193
Blood	0.56106	0.13267	322.899	71.855	72.126
Seeds	0.03755	0.02749	36.595	89.512	89.551
Ecoli	0.08834	0.12173	-27.512	92.864	92.858
Wine	0.03203	0.05951	-46.177	71.235	71.276
Hayes	0.01958	0.0152	28.816	67.012	67.487
Liver	0.11220	0.08919	25.799	41.41	41.824
CMC	2.2142	1.5891	39.337	53.773	53.572
Heart	0.06884	0.10556	-34.786	65.778	65.444
Cleveland	0.08177	0.17135	-52.279	52.704	52.638

7. APPLICATIONS OF KD TREE

7.1 Information Retrieval and Text Classification

Information retrieval is a process of searching information from databases. KD-tree helps in organizing data and accessing them efficiently. This feature is used a lot in search engines where information retrieval works on keyword search. A benefit of using information retrieval programs is that it can be used to get the location of documents containing information if this document exists. The drawback of an information retrieval program is that it cannot find the exact required information explicitly.

A record of a file is an ordered tuple of the form $(v_0, v_1, \ldots, v_{k-1})$ where each coordinate of this tuple are the keys of the record. When a query of a file is called, it specifies certain conditions that are specified by the keys. An associative search is initiated by the information system on arrival of query. Methods [14] like method of compounding attributes, superimposed coding systems and combinatorial hashing are used for information retrieval but none of these are suitable for associative searches. These methods have issues like large requirements of space, large run time etc. The KD tree search has proven better in such situations. The run time of the nearest neighbor search algorithm gets improved when combined with the KD trees. Another advantage of using the KD tree is that it allows deletion of roots by replacing it with its descendants.

Text categorization is a classification problem in which it is determined whether a document belongs to a set of pre-specified documents or not. It is useful in the situations where documents are indexed manually. Features are words occurring in document sets and hence it is very large. Decision trees based on C4.5, RIPPER, naïve Bayes Rainbow are few methods used for text classification. PEBLS (Parallel Exemplar-Based Learning System) algorithm [5] is used for text classification and is based on determining features of attributes. An improvement of PEBLS [16] is VSM (Variable-Kernel Similarity Metric) and works by improving weight in each iteration according to the optimization function. VSM has certain optimization limitations. WAKNN (Weight Adjusted K nearest neighbor) [9] outperforms PEBLS and VSM. In this algorithm KNN is used for improving objective function. For document d, if D is the training document and W is the weight vector objective function is

$$obj_{max}(O,W,P) = \{d \mid d \in D \text{ and } d \in Correct_{maj}(d,D,W,P)\},$$

Where P is the majority percentage and $Correct_{maj}(d,D,W,P)$ is a predicate.

CeKNN (Centroid based K nearest neighbor) [21] based on two step method is another approach. In the first step dimension of document is reduced by projecting feature space to lower dimensional class centroid space using centroid classifier. In the second step KD tree search is used to find K nearest neighbors.

7.2 Digital Elevation Model Production

A set of data points in space is called point cloud data. Point cloud data are mostly used to create CAD models for manufactured parts in 3D, Quality check and animation. One important application is production of digital elevation models that involve KD Tree. While acquiring the point cloud data in an image there are chances of getting elevation anomalies. These anomalies have two components: Low gross error that occurs because of obstructions due to moving vehicles and trees while scanning and high Gross error that occurs because of reflected signals of birds and other low flying objects [31]. These errors can be eliminated if the point cloud data is organized and managed properly. It is achieved using KD- tree. The organization of point cloud data is also crucial since point cloud data has its own features like massiveness, missing local data, uneven density, etc. An image ([31], p.721) before and after gross error elimination is shown in fig 11 and fig 12 respectively.

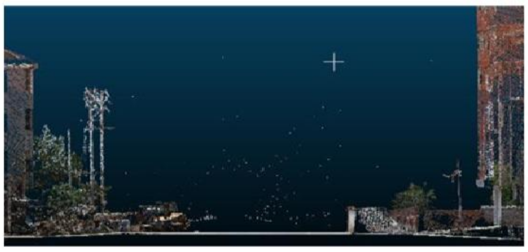

Fig 11: Before Gross Elimination

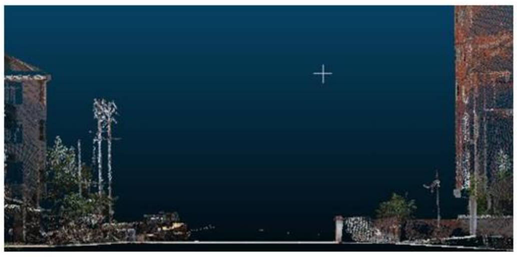

Fig 12: After Gross Elimination

8. FUTURE SCOPE OF STUDY

Setting efficient memory indexing algorithm is requirement of building strong text classifier. Performance of newly developed algorithms for nearest neighbor search in different metric and non-metric space is another direction of study.

9. REFERENCES

[1] Andoni, A. 2009. Nearest Neighbor Search: the Old, the New, and the Impossible. Doctoral dissertation, Massachusetts Institute of Technology. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4b4e7e0419644e9130666ed6e3ba8beee77ffa4e

[2] Atallah, M. J., and Blanton, M. 2009. Algorithms and Theory of Computation Handbook, Second Edition, volume 1: General Concepts and Techniques. Chapman & Hall / CRC Applied Algorithms and Data Structures series Edition: 2. ISBN: 1584888229; 9781584888222.

[3] Bently, J. L. 1975. Multidimensional Binary Search Trees used for Associative Searching. Communications of the ACM, 18(9). pp. 509-517. 517. doi:10.1145/361002.361007

[4] Brown, R.A. 2015. Building a balanced k - d Tree in O(kn log n) Time. Journal of Computer Graphics Techniques (JCGT), vol. 4, no. 1. pp. 50-68.

[5] Cost, S., and Salzberg, S. 1993. A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. Machine learning 10. pp. 57-78. doi: 10.1023/A:1022664626993

[6] Cover, T.M. and Hart, P.E. 1967. Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory, 13(1), pp. 21-27.

[7] Eibe, F., Hall, M., and Pfahringer, B. 2003. Locally Weighted Naïve Bayes. In Proceedings of the Conference on Uncertainty in Artificial Intelligence. pp. 249–256. Morgan Kaufmann.

[8] Goldstine, H., and Von Neumann, J. 1963. Coding of some combinatorial (sorting) problems. In John von Neumann Collected works Design of computers, Theory 66 of Automata and Numerical Analysis, A. Taub, Ed., vol. 5. Pergamon Press Ltd. and the Macmillan company, New York NY. pp. 196-214.

[9] Han, E., Karypis, G., and Kumar, V. 2001. Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. Pacific-Asia Conference on Knowledge Discovery and Data Mining.

[10] Hoare, C. A. R. 1962. Quicksort. The Computer Journal 5. pp. 10-15. doi: 10.1093/comjnl/5.1.10. 51

[11] Hou, W., Li, D., Xu, C., Zhang, H., and Li, T. 2018. An Advanced k Nearest Neighbor Classification Algorithm

Based on KD-tree. IEEE International Conference of Safety Produce Informatization (IICSPI), Chongqing, China, pp. 902-905. doi: 10.1109/IICSPI.2018.8690508.

[12] Jiang, L., Cai, Z., Wang, D., and Jiang, S. 2007. Survey of Improving K-Nearest-Neighbor for Classification. Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), Haikou, China, pp. 679-683. doi: 10.1109/FSKD.2007.552.

[13] Jiang, L., Zhang, H., and Cai, Z. 2006. Dynamic K-Nearest-Neighbor Naive Bayes with Attribute Weighted. International Conference on Fuzzy Systems and Knowledge Discovery. pp. 365–368. Springer, 2006.

[14] Knuth, D.E. 1973. The art of computer programming Volume III: Sorting and Searching. Addison - Wesley, Reading, Mass.

[15] Kohavi, R. 1996. Scaling up the accuracy of Naïve-Bayes Classifier: A decision tree hybrid. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. pp. 202–207. AAAI Press.

[16] Lowe, D.G. 1995. Similarity metric learning for a variable-kernel classifier. Neural computation, vol. 7, no. 1. pp. 72-85. doi: 10.1162/neco.1995.7.1.72.

[17] Mitchell, T. M. 1997. Machine Learning. McGraw-Hill Science/Engineering/Math. ISBN: 0070428077

[18] Mohammad Reza, A., Bijan, G., and Hassan, N. 2014. A Survey on Nearest Neighbor Search Methods. International Journal of Computer Applications. 95. pp. 39-52. doi: 10.5120/16754-7073.

[19] Necaise, R.D. n. d. Data Structures and Algorithms using python. Wiley Student edition.

[20] Panigrahy, R. 2008. An Improved Algorithm Finding Nearest Neighbor Using Kd-trees. In: laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds) LATIN 2008; Theoretical Informatics. LATIN 2008. Lecture Notes in Computer Science, vol 4957. Springer, Berlin, Heidelberg.doi: 10.1007/978-3-540-78773-0_34

[21] Priyanka, T., and Swamy, N. N. 2015. KNN Based Document Classifier Using Kd Tree: An Efficient Implementation. International Journal of Computer Science & Communication Networks, 5(5). pp. 270-274.

[22] Silverman, B. W., and Jones, M. C. 1989. E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges. International Statistical Review / Revue Internationale de Statistique, vol. 57, no. 3, 1989, pp. 233–238. doi: 10.2307/1403796.

[23] Williams, J. 1964. Heapsort (algorithm 232). Communications of the ACM 7. pp. 347-348.

[24] Xie, Z., Hsu, W., Liu, Z., and Lee, M. 2002. SNNB: A Selective Neighborhood Based Naïve Bayes for Lazy Learning. In Proceedings of the Sixth Pacific-Asia Conference on KDD. pp. 104–114. Springer. doi: 10.1007/3-540-47887-6_10.

[25] Zhai, H. 2022. *Improving KNN Algorithm Efficiency Based on PCA and KD-tree.* International Conference on Machine Learning and Knowledge Engineering (MLKE), Guilin, China, pp. 83-87. doi: 10.1109/MLKE55170.2022.00021.

[26] Barnwal, A. 2022. K Dimensional Tree | Set 1 (Search and Insert).https://www.geeksforgeeks.org/k-dimensional-tree/

[27] Black, P. E. 2019. Big - O notation. Dictionary of Algorithms and DataStructures. https://www.nist.gov/dads/HTML/bigOnotation.html

[28] Couto, D. D, and Napoli, J. 1998. kD Trees.http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/kd.html

[29] Dey, S. 2017. Implementing kd-tree for fast range-search, nearest-neighbor search and k-nearest-neighbor search algorithms in 2D (with applications in simulating the flocking boids: modeling the motion of a flock of birds and in learning a kNN classifier: a supervised ML model for binary classification) in Java and python.https://sandipanweb.wordpress.com/2017/09/10/implementing-kd-trees-along-with-the-fast-range-search-nearest-neighbor-search-and-k-nearest-neighbor-search-algorithms-in-2d-with-an-application-in-simulating-the-motion-of-a-flock-of-boids/

[30] Hachcham, A. 2023. The KNN Algorithm – Explanation, Opportunities, Limitations.https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations

[31] Kang, Q., Huang, G. M., and Yang, S.C. 2018. A Gross Error Elimination Method for Point Cloud data based on KD-Tree. ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2018, 719-722. https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-3/719/2018/isprs-archives-XLII-3-719-2018.pd

[32] MATWORKS from MATLAB. 2022. Classification Using Nearest Neighbors. https://in.mathworks.com/help/stats/classification-using-nearest-neighbors.html

[33] Mount, D. 2021.CMSC 420: Lecture 13 Answering Queries with kd-trees.https://www.cs.umd.edu/class/spring2021/cmsc420-0101/Lects/lect13-kd-query.pdf

[34] Padmaja, B. 2018. Lecture Notes on Data Structures.https://www.iare.ac.in/sites/default/files/lecture_notes/IARE_DS_LECTURE_NOTES_2.pdf