A Review of various k-Nearest Neighbor Query Processing Techniques

S. Dhanabal
Asst. Professor, Dept. of CSE,
Jansons Institute of Technology,
Coimbatore, Tamilnadu, INDIA

Dr. S. Chandramathi
Professor & Head, Dept. of ECE,
Sri Krishna College of Engineering & Technology,
Coimbatore, Tamilnadu, INDIA

ABSTRACT
Identifying the queried object, from a large volume of given uncertain dataset, is a tedious task which involves time complexity and computational complexity. To solve these complexities, various research techniques were proposed. Among these, the simple, highly efficient and effective technique is, finding the K-Nearest Neighbor (kNN) algorithm. It is a technique which has applications in various fields such as pattern recognition, text categorization, moving object recognition etc. Different kNN techniques are proposed by various researchers under various situations. In this paper, we classified these techniques into two ways: (1) structure based (2) non-structure based kNN techniques. The aim of this paper is to analyze the key idea, merits, demerits and target data behind each kNN techniques. The structure based kNN techniques such as Ball Tree, k-d Tree, Principal Axis Tree (PAT), Orthogonal Structure Tree (OST), Nearest Feature Line (NFL), Center Line (CL) and Non-structured kNN techniques such as Weighted kNN, Condensed NN, Model based k-NN, Ranked NN (RNN), Pseudo/Generalized NN, Clustered k-NN(CkNN), Mutual kNN (MkNN), Constrained RkNN etc., are analyzed in this paper. It is observed that the structure based kNN techniques suffer due to memory limit whereas the Non-structure based kNN techniques suffer due to computation complexity. Hence, structure based kNN techniques can be applied to small volume of data whereas Non-structure kNN techniques can be applied to large volume of data.

KEYWORDS
Query processing – Nearest Neighbor - kNN

1. INTRODUCTION
Query processing technique is usually applied to the smallest datasets which follows any of the in-memory algorithms like B+tree, R-trees etc., to get the result. But, when the datasets are large, high dimensional or uncertain, it is highly impossible for traditional query processing technique to retrieve the required data within the stipulated time. The nearest neighbor techniques play a vital role in these situations. The nearest neighbor (NN) technique is very simple, highly efficient and effective in the field of pattern recognition, text categorization, object recognition etc. It has so many advantages like simplicity, robust to noisy training data, improved query time and memory requirements etc., and also have disadvantages like Computation Complexity, Memory limitation and high cost in execution of algorithm. The nearest neighbor (NN) rule identifies the category of unknown data point on the basis of its nearest neighbor whose class is already known. This rule is widely used in pattern recognition [1,2], text categorization [3-5], ranking models [6], object recognition [7] and event recognition [8] applications. A number of methods have been proposed for efficient processing of nearest neighbor queries for stationary points. The k-nearest neighbor lies in first category in which whole data is classified into training data and sample data point. Distance is evaluated from all training points to sample point and the point with lowest distance is called nearest neighbor. This technique is very easy to implement but value of k affects the result in some cases. The NN training data set can be structured using various techniques to improve over memory limitation of kNN. The kNN technique can be implemented using ball tree [19, 20], k-d tree [21], nearest feature line (NFL) [22], tunable metric [24], principal orthogonal search tree [26], axis search tree [27] and Continuous RkNN[47]. In tree structure, training data is divided into nodes, whereas in techniques like NFL and tunable metric, the training data set is divided according to planes. These algorithms increase the speed of basic kNN algorithm.

Non-structured k-NN technique has been improved to meet the increase in dimensionality of the data space. T. M. Cover et al. proposed that the nearest neighbor can be calculated based on the value of k which specifies the number of nearest neighbors to define a class of sample data point[9]. Later, it was improved based on weights [33]. The training points are assigned weights according to their distances from sample data point. The computational complexity and memory requirements are the two main issues related to the above techniques. To overcome memory limitation, size of data set is reduced by the repeated patterns, which do not add extra information and the data points which do not affect the result are eliminated from training data set. Apart from the time and memory limitation, the value of k is mainly considered to find the category of the unknown sample. To improve speed of classical kNN, many techniques such as ranking, false neighbor information, clustering etc., are used. Non-structured k-NN techniques are further amplified in the areas of moving object which uses query indexing technique instead of object indexing. Object indexing technique is not suited for the database where the objects are continuously moving and so uses Query indexing. Most of the indexing techniques such as R-trees, B+ trees etc., are based on the disk-based indexing which is not well-suited for moving objects for the following reasons:- (i) updating the index when the object moves; (ii) Frequent revaluation of queries when any object moves; and (iii) achieving minimum execution times for large number of moving objects and queries. Also, the cost of executing these algorithms from main memory is high. In both the cases, kNN techniques are well suited for finding the solution. Reverse kNN technique is the complementary problem to that of finding the *k*-nearest neighbors (*k*-NN) of a query object whose goal is to find the influence object of the whole dataset [41-46]. Continuous RkNN[47], Constrained RkNN [48], Mutual k-NN(MkNN)[50], are some of the k-NN techniques used in the

continuous moving object datasets whereas Aggregate kNN[49] is used to find the nearest neighbor using aggregate functions.

2. K-NEAREST NEIGHBOR TECHNIQUES

k-NN techniques are classified in two types i) Structure based kNN and ii) Non- structure based kNN.

2.1 Structure *based k-NN Technique*

Structure based k-NN technique uses tree structures to represent the training datasets. Berchtold proposed a method on Voronoi cells which is built specifically for nearest neighbor queries [10]. Range queries uses index structures that works based on Branch-and-bound methods. Roussopoulos proposed an influential algorithm [11], for finding the *k* nearest neighbours in which an R-tree [12] indexes the points, and *depth-first* traversal of the tree is used. During the traversal, entries in the nodes of the tree are ordered and pruned based on a number of heuristics. Cheung and Fu [13] simplified this algorithm without reducing its efficiency. To better suit the nearest neighbor problems, branch-and- bound algorithms are modified by various methods especially when applied for high-dimensional data [14]. Next, a number of incremental algorithms for similarity ranking have been proposed that can efficiently compute the *(k* +1*)*-st nearest neighbor, after the *k* nearest neighbors are returned [15, 16]. They use a global priority queue of the objects to be visited in an R-tree. More specifically, Hjaltason et al. [16] proposed an incremental nearest neighbor algorithm, which uses a priority queue of the objects to be visited in an R+-tree [17]. They show that such a *best-first* traversal is optimal for a given R-tree. A very similar algorithm was proposed by Henrich [15], which employs two priority queues. For high-dimensional data, multi-step nearest neighbor query processing techniques are usually used [18].

Ball Tree concept was proposed by Ting Liu. It is a binary tree in which, leaves contain information and the internal nodes are used to efficiently search through leaves. This can be shown in the Figure.1. It follows top down approach and has a better speed over kNN[19,20].

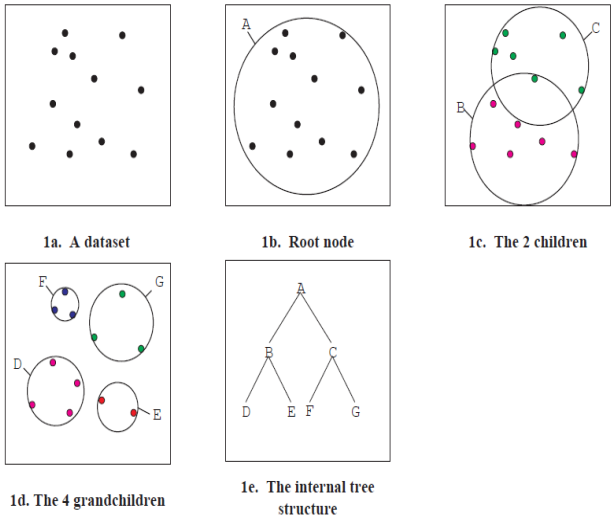


Fig1. An Example of a Ball-tree

In the k-dimensional trees, the training data are divided into right node and left node. According to query records. left or right side of tree is searched. Once the terminal node is reached, records in that node are examined to find the closest data node to query record [21]. Stan Z.Li et al. proposed the concept of NFL [22] which divides the training data into plane. It is used to enhance the representational capacity of a sample set of limited size by using the feature lines which passes through each pair of the samples belonging to the same class. An example to illustrate the NFL classification method is given in Figure.2 The evaluated distances are sorted into ascending order and the NFL distance is assigned as rank 1. An improvement made over NFL is Local Nearest Neighbor, proposed by W. Zheng et al., evaluates the feature line and feature point in each class, for points only, whose corresponding prototypes are neighbors of query point[23]. Yongli Zhou et al. introduce [24] new metric, called "Tunable Metric", which is used for evaluating distances for NFL rather than feature line. At first stage it uses tunable metric to calculate distance and then implement steps of NFL. Center Based Nearest Neighbor [25] is improvement over NFL and Tunable Nearest Neighbor in which Center base Line [CL] connects sample point with known labeled points. CL is calculated based on the straight line passing through training sample and center of class and then distance is evaluated from query point to CL and nearest neighbor is evaluated. Principal Axis Tree [PAT] [26] divide the training data in an efficient manner in terms of speed for nearest neighbor evaluation. It consists of two phases 1) PAT Construction 2) PAT Search. PAT uses principal component analysis (PCA) and divides the data set into regions containing the same number of points. Once tree is formed kNN is used to search nearest neighbor in PAT. The regions can be determined for given point using binary search. The Orthogonal Search Tree [OST][27] uses orthogonal vector. It is an improvement over PAT to speed up the process. It uses concept of "length (norm)", which is evaluated at first stage. Then orthogonal search tree is formed by creating a root node and assigning all data points to this node. Then the left and right nodes are formed using pop operation.

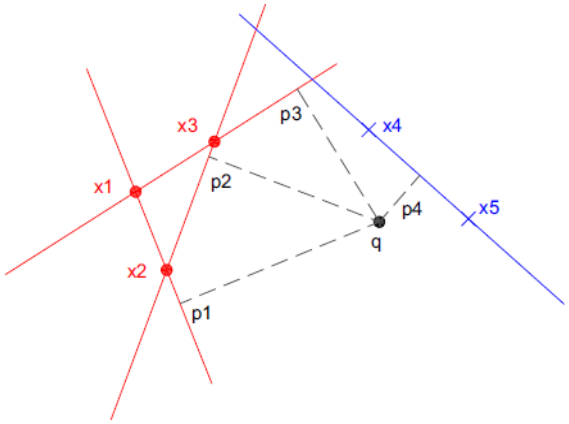


Figure.2 An example to illustrate the NFL classification method. The feature points *p*1, *p*2, *p*3 and *p*4 are the projections of query *q* on the feature lines *x*1*x*2, *x*2*x*3, *x*1*x*3 and *x*4*x*5, respectively.

2.2 Non-structure based KNN Techniques

Kollios et al. [28] proposed an elegant solution for answering nearest neighbor queries for moving objects in one dimensional space. Their algorithm uses a duality transformation, where the future trajectory of a moving point $x(t) = x0 + vx\ t$ is transformed into a point $(x0, vx)$ in a so-called *dual space*. The solution is generalized to the "1.5- dimensional" case where the objects are moving in the plane, but with their movements being restricted to a number of line segments (e.g.,

corresponding to a road network). However, Tian Xia and Donghui Zhang [47] investigated the processing of Continuous RNN (CRNN) queries when k=1. Their method is based on the 60-degree-pruning technique. The monitoring region of a CRNN query is defined as six pie-regions (determined by the query point and the six candidates) and six cir-regions (determined by the six candidates and their nearest neighbors). Then for a CRNN query: in each sub-space a *continuous constrained nearest neighbour* query is used to monitor the candidate in that sub-space, called continuous filter. For each candidate, a *continuous nearest neighbor* query is used to do continuous refinement. Frequent Updated R-Tree (FUR) and the hash tables are used to store the cir-region whereas optimization techniques like *lazy-update* and *partial-insert* are also proposed to avoid unnecessary NN searches and reduce the updates on the FUR-tree. The work of Albers et al. [29] investigated Voronoi diagrams of continuously moving points, relates to the problem of nearest neighbor queries. Even though such diagrams change continuously as points move, their topological structures change only when certain discrete events occur. The authors show a non-trivial upper bound of the number of such events. They also provide an algorithm to maintain such continuously changing Voronoi diagrams.

Song et al. [30] proposed a solution for finding the *k* nearest neighbors for a moving query point. However, the data points are assumed to be static. In addition, the time is not assumed to be continuous instead a periodical sampling technique is used. The time period is divided into *n* equal-length intervals. When computing the result set for some sample, the algorithm tries to reuse the information contained in the result sets of the previous samples. Whereas Raptopoulou et al. [31] and Tao et al. [32] considered the nearest neighbor problem for a query point moving on a line segment for static and for moving data points.

Bailey [33] uses weights with classical kNN and proposed weighted kNN (WkNN) algorithm. WkNN evaluates the distances as per value of k and a weight is assigned to each calculated value, and then nearest neighbor is decided and class is assigned to sample data point. The Condensed Nearest Neighbor (CNN) algorithm stores the patterns one by one and eliminates the duplicate ones. Hence, CNN removes the data points which do not add more information and show similarity with other training data set [34]. The Reduced Nearest Neighbor (RNN) is an improvement over CNN; it includes one more step to eliminate the patterns which are not affecting the training data set result[35]. The another technique called Model Based kNN selects similarity measures and create a 'similarity matrix' from the given training set. Then, in the same category, largest local neighbor is found that covers large number of neighbors and a data tuple is located with largest global neighborhood. These steps are repeated until all data tuples are grouped. Once data is formed using model, kNN is executed to specify category of unknown sample[36].

Subash C et al. [37] improved the kNN by introducing the concept of ranks. The method pools all the observations belonging to different categories and assigns ranks to each category of data in ascending order. Then observations are counted on the basis of rank and class is assigned to unknown sample. It is very much useful in case of multi-variants data. In Modified kNN, which is a modification of WkNN, validity of all data samples in the training data set is computed, accordingly weights are assigned and then validity and weight both together set basis for classifying the class of the sample data point[38].

Yong zeng et al. [39] defines a new concept to classify sample data point. The method introduces the pseudo neighbor, which is not the actual nearest neighbor; but a new nearest neighbor is selected on the basis of value of weighted sum of distances of kNN of unclassified patterns in each class. Then Euclidean distance is evaluated and pseudo neighbor with greater weight is found and classified for unknown sample. In this technique proposed by Zhou Yong [40], Clustering is used to calculate nearest neighbor. In the first step, samples which are nearer to the border of the training set are removed. Then, each training dataset are clustered based on the 'k' value and all cluster centers form a new training set. Then, weights are assigned to each cluster according to number of training samples in clusters.

3. REVERSE KNN TECHNIQUES

A reverse *k*-nearest neighbor (R*k*NN) query returns the data objects that have the query object in the set of their *k*-nearest neighbors. It is the complementary problem to that of finding the *k*-nearest neighbors (*k*-NN) of a query object. The goal of a reverse *k*-nearest neighbor query is to identify the "influence" of a query object on the whole data set. Although the reverse *k*-nearest neighbor problem is the complement of the *k*-nearest neighbor problem, the relationship between *k*-NN and R*k*NN is not symmetric and the number of the reverse *k*-nearest neighbors of a query object is not known in advance. A naive solution of the R*k*NN problem requires the running time of $O(n^2)$ whereas kNN requires the running time of O(n).Also RkNN is more expensive than k-NN queries.

Several different solutions have been proposed for computing RNN queries for non-moving points in two and higher dimensional spaces. Stanoi et al. [41] present a solution for answering RNN queries in two-dimensional space.

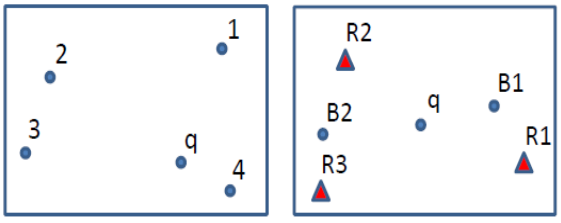
Definition 1:
Let p be an NN point of q among the points in Si. Then, either q is an NN point of p (and then p is an RNN point of q), or q has no RNN point in Si .

Stanoi et al. has proved this property [41]. These observations enable a reduction of the RNN problem to the NN problem. For each region *Si*, an NN point of *q* in that region is found. In another solution for answering RNN queries, Korn and Muthukrishnan [42] use two R-trees for the querying, insertion, and deletion of points. In the first, the RNN-tree, the minimum bounding rectangles of circles having a point as their centre and the distance to the nearest neighbor of that point as their radius are stored. The second, the NN-tree, is simply an R*-tree that stores the data points.

Yang and Lin [43] improve the solution of Korn and Muthukrishnan by introducing an Rdnn-tree, which makes it possible to answer both RNN queries and NN queries using a single tree. Structurally, the Rdnn-tree is an R+-tree, where each leaf entry is augmented with the distance to its nearest neighbor (dnn), and where a non-leaf entry stores the maximum of its children's dnn's. Maheshwari et al. [44] proposed main memory data structures for answering RNN queries in two dimensions. For each point their structures maintain the distance to its nearest neighbor. In contrast to the approach of Stanoi et al., updates of the database are problematic in the last three approaches mentioned. On the other hand, the approach of Stanoi et al. does not easily scale up to more than two dimensions because the number of regions where RNN candidates are found increases exponentially with the dimensionality. To alleviate this problem, Singh et al. [45] proposed an algorithm where RkNN candidates are found by performing a regular kNN query. The disadvantage of such an

approach is that it does not always find all RkNN points. The recent approach by Tao et al. [46] fixes this problem. Their so-called TPL algorithm, similar to Stanoi et al., works in two phases—a filtering phase and a refinement phase—but no subdivision of the underlying space into regions is necessary in the refinement phase. Thus, the algorithm gracefully scales to more than two dimensions.

Tobias Emrich et al. proposed a new RkNN technique based on some constrains. In this, they formalize the novel concept of Constrained Reverse k-Nearest Neighbor (CRkNN) search on mobile objects (clients) performed at a central server. The CRkNN query computes the set RkNN(q) of objects, for a given query object q, having q as one of their k-nearest neighbors, if and only if the result set exceeds a specific threshold, say 'm', else the query reports an empty result. Their approach minimizes the amount of communication between clients and central server by using the approximation of the positions to identify true hits and true drops. This approach provides only approximate results for bichromatic cases.[47]. An example of Constrained Reverse RkNN for monochromatic and bichromatic cases is shown in [Figure.3]. It illustrates the concept of CRkNN queries in both cases. The mono-chromatic CR1NN query [Figure.3a] for point q returns point 1 and 4 if, for some threshold value m ϵ {1, 2} or nothing if m $\leq$ 3. Points 2 and 3 are not returned for any choice of m, because they do not find q as their 1-nearest neighbor. In the bi-chromatic case, two object sets Dred (red objects) and Dblue (blue objects) are considered. The bi-chromatic CRkNN query returns all elements of Dred that have the query point as on of their k-nearest neighbors if all other red objects are ignored.Figure.3b shows the bi-chromatic CR1NN query for a set of lions (red objects R1, R2, and R3) and a set of potential prey (blue objects B1, B2, and q). The query object q (from the blue object set) could be a young elephant that is not yet able to defend itself. In this example, the bi-chromatic CR1NN query yields no results for any value of m, because each lion observes another animal as nearest neighbour.



(a) Mono-chromatic case. (b) Bi-chromatic case.

Figure.3 An example of Constrained Reverse RkNN

4. OTHER NEAREST NEIGHBOR TECHNIQUES

Dimitris Papadias et al. [48] deals with the aggregate nearest neighbor technique which states that for the given two spatial datasets *P* (e.g., facilities) and *Q* (queries), an *aggregate nearest neighbor* (ANN) query retrieves the point(s) of *P* with the smallest aggregate distance(s) to points in *Q*. They provide algorithms for memory-resident queries and cost models that accurately predict their performance in terms of node accesses and also developed methods for disk-resident query sets and approximate retrieval. But the cost for implementing disk-resident query model is very high because of the multiple reads of *Q* required by this processing technique.

Yunjun Gao et al.[49] proposed a technique on finding the mutual nearest neighbor which deals with the given set 'D' of trajectories, a query object q, and a query time extent C, a mutual (i.e., symmetric) nearest neighbor (MNN) query over trajectories from D, the set of trajectories that are among the k1 nearest neighbors (NNs) of q within C, and will have q as one of their k2 NNs. They proposed two types of MNN queries, i.e., MNNP and MNNT queries, which are defined with respect to stationary query points and moving query trajectories, respectively. They utilize the batch processing and reusing technology to reduce the I/O cost (i.e., number of node/page accesses) and CPU time significantly and also proposed techniques to tackle historical continuous MNN (HCMNN) search for moving object trajectories, which returns the mutual nearest neighbors of q (for a specified k1 and k2) at any time instance, say 'c'.But this algorithm doesn't deal with the bichromatic datasets and can be further pruned to increase the performance.Comparisons of various kNN techniques are given in table 1.

TABLE I. COMPARISON OF VARIOUS K-NEAREST NEIGHBOR TECHNIQUES

S. No	Technique	Concept	Merits	Demerits	Applications
1	Ball Tree k nearest neighbor (BTKNN) [19,20]	To improve the speed	1.Compatibile with high dimensional Objects. 2. Represented data are tuned well to structure 3.Simple to implement 4. Especially used for geometric learning	1. Implementation cost is high. 2. When distance is increased, performance is decreased.	Robotic, vision, speech, graphics
2	k-d tree nearest neighbor (kdNN) [21]	To divide the training data sets into two halves	1.Perfect balanced trees are formed 2.It is fast and simple	1.Computational complexity 2.Exhaustive search is required 3. Chance of misleading the points as it blindly splits the points into two halves.	Multidimensional data points.
3	Nearest feature Line Neighbor (NFL) [22]	To have multiple template per class for classificatio n	1.Accurate classification 2.Effective algorithm for small datasets. 3.Ignored information in nearest neighbor are used	1.Chance of failure if the model in NFL is far away from query point 2.Computational Complexity 3. Hard to illustrate the feature point in straight line.	Face Recognition problems
4	Local Nearest Neighbor [23]	To focus on nearest neighbor prototype of query point	1.Overcomes the limitations of NFL	1.Increase in number of computations	Face Recognition
5	Tunable Nearest Neighbor (TNN) [24]	Calculates the distance first and then implements the steps of NFL	1.Effective for small data sets	1.Large number of computations	Bias problems
6	Center based Nearest Neighbor (CNN) [25]	To connect sample point with known labeled points on a center line	1.Highly efficient for small data sets	1. Large number of computations	Pattern Recognition
7	Orthogonal Search Tree Nearest Neighbor [26]	To speed up the process , Orthogonal search trees are used	1.Less Computation time 2.Effective for large data sets	1.Query time is more	Pattern Recognition

8	Principal Axis Tree Nearest Neighbor (PAT) [27]	Uses PAT construction and PAT search	1.Good perform ance 2.Fast Search	1.Computational Time is more	Pattern Recognition
9.	k Nearest Neighbor (kNN) [9]	To find the nearest neighbor based on 'k' value	1.Training is very fast 2.Simple and easy to learn 3.Robust to noisy training data 4 .Effective if training data is large 5.It is symmetric.	1. Biased by value of k 2.Computation Complexity 3.Memory limitation 4.Being a supervised learning lazy algorithm i.e. runs slowly 5.Easily fooled by irrelevant	Large sample data
10.	Weighted k nearest neighbor (WkNN) [33]	To assign weights to neighbors based on distance calculated	1. Overcomes limitations of kNN by assigning equal weight to k neighbors implicitly. 2. Uses all training samples not just k. 3. Makes the algorithm global one	1.Computational complexity increases in calculating weights 2.Slow in execution	Large sample data
11.	Condensed nearest neighbor (CNN) [34]	To eliminate data sets which show similarity without adding extra information	1. Reduce size of training data 2. Improve query time and memory requirements 3. Reduce the recognition rate	1.CNN is order dependent; it is unlikely to pick up points on boundary. 2. Computational Complexity	Data set where memory requirement is a main concern
12.	Reduced Nearest Neighbor (RNN) [35]	To remove patterns which do not affect the training data set results	1. Reduced size of training data and eliminate templates 2. Improved query time and memory requirements 3. Reduced recognition rate	1.Computational Complexity 2.Cost is high 3.Time Consuming	Large data set
13.	Model based k Nearest Neighbor (MkNN) [36]	To construct a model from data and classify new data using these model	1. More classification accuracy 2.Value of k is selected automatically 3.Highly efficient due to reduced number of data points	1.Do not consider marginal data outside the region	Dynamic web mining for large repository
14.	Rank nearest neighbor (kRNN) [37]	To assign ranks to training data for each category	1.Performs better when there are too much variations between features 2.Robust as based on rank	1.Multivariate kRNN depends on distribution of the data	Class distribution of Gaussian nature
15.	Modified k nearest neighbor (MkNN) [38]	To classify nearest neighbor based on weights and validity of data point	1.Partially overcomes low accuracy of WkNN 2.Stable and robust	1.Computational Complexity	Methods facing outlets

16.	Pseudo/Generalized Nearest Neighbor (GNN) [39]	To utilize information of (n-1) neighbors also	1.Uses(n-1) classes which consider the whole training data set	1.Does not hold good for small data 2.Computational complexity	Large data set
17.	Clustered k nearest neighbor [40]	To select the nearest neighbor from the clusters	1.Overcome defects of uneven distributions of training samples 2.Robust in nature	1.Selection of threshold parameter is difficult before running algorithm 2.Biased by value of k for clustering	Text Classification
18.	Reverse k nearest neighbor [41-46]	Objects that have the query object as their nearest Neighbour, have to be found.	1. Approximate results can be obtained very fast. 2. Well suited for 2-Dimensional sets 3. Well suited for finite, stored data sets 4. Provides decision support	1. requires $O(n^2)$ time 2. do not support arbitrary values of k 3. cannot deal efficiently with database updates, 4. are applicable only to 2D	Spatial data set
19	Continuous RkNN [47]	To monitor the regions upon updates using FUR tree	1. Overcomes the difficulties of using the kNN and RkNN queries on moving objects. 2. Best suited for monochromatic cases	1. Not suited for bichromatic cases 2. Not suited for large population of continuously moving objects. 3. Memory Limitation	Moving object data set
20	Constrained RkNN [48]	To find the RkNN on moving objects based on constrains	1. Communication load is minimized. 2. CRkNN can be applied to both monochromatic and bichromatic cases.	1. Approximate result can be obtained for bichromatic cases.	Moving object data set especially in GPS
21	Aggregate kNN [49]	To use aggregate function for finding the nearest neighbor	1.Provides memory-resident queries and cost models that accurately predict their performance in terms of node accesses 2.Approximate result can be obtained for	1. Cost for evaluating the disk-resident query model is high. 2. Lazy algorithm	Spatial data set
22	Mutual Nearest Neighbor [50]	To find the Mutual Nearest Neighbor using TB-tree.	1 .Uses batch processing and reuse technology for reducing I/O cost and CPU time. 2. HCMNN is used to reduce the searching time of all the data again and again.	1. Applied only to the monochromatic datasets. 2. Computational complexity	Moving object data set

5. CONCLUSION

In this paper, the various types of structure based k-Nearest Neighbor techniques and Non-structure based k-Nearest Neighbor techniques are compared based on key ideas. Their merits and demerits are analyzed and the results are given in table 1. Limited memory allocation and more execution time are the two main issues of structure based algorithms such as Ball Tree, k-d Tree, Principal Axis Tree (PAT), and Orthogonal Structure Tree (OST), Nearest Feature Line (NFL), and Center Line (CL) techniques. But they are easy to construct and cost effective. Non structure based kNN techniques like simple k-NN technique has the drawback of

memory limitation and biased by 'k' value whereas Weighted kNN, Condensed NN, Model based k-NN, Ranked NN (RNN), Pseudo/Generalized NN, Clustered k-NN(CkNN), Continuous RkNN, Mutual kNN (MkNN), Constrained RkNN etc., are having more computational complexity. Moreover, kNN techniques like Constrained RkNN, Continuous RkNN and Mutual Nearest Neighbor are widely used in the moving object datasets whereas Aggregate kNN and Reverse kNN techniques are used in spatial dataset. Also, Constrained RkNN is well suited for both monochromatic and bichromatic datasets, Reverse kNN is especially used in 2D sets whereas Mutual NN, Aggregate NN, Continuous RkNN are designed for monochromatic datasets.In general, to reduce the time complexity and computational complexity, various kNN algorithms are proposed. Each algorithm is found to be suitable for a particular situation. kNN techniques are not suited for multidimensional environment because of large volume of data involved in it. Only few algorithms are there to reduce the dimensionality and is yet to be analyzed by the researchers.

6. REFERENCES

[1] V.Vaidehi, S.Vasuhi, " Person Authentication using Face Recognition", Proceedings of the world Congress on Engineering and Computer Science, 2008.

[2] Shizen, Y. Wu, "An Algorithm for Remote Sensing Image Classification based on Artificial Immune b-cell Network", Springer Berlin, Vol 40.

[3] G. Toker, O. Kirmemis, "Text Categorization using k Nearest Neighbor Classification", Survey Paper, Middle East Technical University.

[4] Y. Liao, V. R. Vemuri, "Using Text Categorization Technique for Intrusion detection", Survey Paper, University of California.

[5] E. M. Elnahrawy, "Log Based Chat Room Monitoring Using Text Categorization: A Comparative Study", University of Maryland.

[6] X. Geng et. al, "Query Dependent Ranking Using k Nearest Neighbor", SIGIR, 2008.

[7] F. Bajramovie et. al "A Comparison of Nearest Neighbor Search Algorithms for Generic Object Recognition", ACIVS 2006, LNCS 4179, pp 1186-1197.

[8] Y. Yang and T. Ault, "Improving Text Categorization Methods for event tracking", Carnegie Mellon University.

[9] T.M.Cover and P.E. Hart, "Nearest Neighbor Pattern Classification", IEEE Trans. Inform. Theory, Vol. IT-13, pp 21-27, Jan 1967.

[10] Berchtold, S., Ertl, B., Keim, D.A., Kriegel, H.P., Seidl, T."Fast nearest neighbor search in high-dimensional space" in Proceedings of the International Conference on Data Engineering, pp. 209–218 (1998)

[11] Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 71–79 (1995)

[12] Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 47–57 (1984)

[13] Cheung, K.L., Fu, A.W.-C.: Enhanced nearest neighbor search on the R-tree. ACM SIGMOD Record 27(3), 16–21 (1998)

[14] Katayama, N., Satoh, S.: The SR-tree: An index structure for high-dimensional nearest neighbor queries. In: Proceedings of the ACM SIGMOD International Conference on Management of Data,pp.369–380(1997)

[15] Henrich, A.: A distance scan algorithm for spatial access structures. In: Proceedings of the Second ACM Workshop on Geographic Information Systems, pp. 136–143 (1994)

[16]. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. ACM Trans. Database Sys. 24(2), 265–318 (1999)

[17] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*- tree: An efficient and robust access method for points and rectangles. In: Proceedings of the ACM SIGMOD International Conference On Management of Data, pp. 322–331 (1990)

[18] Seidl, T., Kriegel, H.P.: Optimal multi-step k-nearest neighbor search. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 154–165 (1998)

[19] T. Liu, A. W. Moore, A. Gray, "New Algorithms for Efficient High Dimensional Non-Parametric Classification", Journal of Machine Learning Research, 2006, pp 1135-1158.

[20] S. N. Omohundro, "Five Ball Tree Construction Algorithms", 1989, Technical Report.

[21] R. F Sproull, "Refinements to Nearest Neighbor Searching", Technical Report, International Computer Science, ACM (18) 9, pp 507-517. I*nternational Journal of Computer Science and Information Security,Vol. 8, No. 2, 2010*

[22] S. Z Li, K. L. Chan, "Performance Evaluation of The NFL Method in Image Classification and Retrieval", IEEE Trans On Pattern Analysis and Machine Intelligence, Vol 22, 2000.

[23] W. Zheng, L. Zhao, C. Zou, "Locally Nearest Neighbor Classifier for Pattern Classification", Pattern Recognition, 2004, pp 1307-1309.

[24] Y. Zhou, C. Zhang, "Tunable Nearest Neighbor Classifier", DAGM 2004, LNCS 3175, pp 204-211.

[25] Q. B. Gao, Z. Z. Wang, "Center Based Nearest Neighbor Class", Pattern Recognition, 2007, pp 346-349.

[26] Y. C. Liaw, M. L. Leou, "Fast Exact k Nearest Neighbor Search using Orthogonal Search Tree", Pattern Recognition 43 No. 6, pp 2351-2358.

[27] J.Mcname, "Fast Nearest Neighbor Algorithm based on Principal Axis Search Tree", IEEE Trans on Pattern Analysis and Machine Intelligence, Vol 23, pp 964-976.

[28] Kollios, G., Gunopulos, D., Tsotras, V.J.: Nearest neighbor queries in a mobile environment. In: Proceedings of the International Workshop on Spatio-Temporal Database Management, pp. 119–134 (1999)

[29] Albers, G., Guibas, L.J., Mitchell J.S.B, Roos, T.: Voronoi Diagrams of moving points. Int. J. Comput. Geom. Appl. **8**(3), 365– 380 (1998)

[30] Song, Z., Roussopoulos, N.: K-nearest neighbor search for moving query point. In: Proceedings of the International Symposium on Spatial and Temporal Databases, pp. 79–96 (2001)

[31] Raptopoulou, K., Papadopoulos, A., Manolopoulos, Y.: Fast nearest-neighbor query processing in moving-object databases. GeoInformatica **7**(2), 113–137(2003)

[32] Tao, Y., Papadias, D.: Spatial queries in dynamic environments. ACM TODS 28(2), 101–139 (2003)

[33] T. Bailey and A. K. Jain, "A note on Distance weighted k-nearest neighbor rules", IEEE Trans. Systems, Man Cybernatics, Vol.8, pp 311-313, 1978.

[34] E Alpaydin, "Voting Over Multiple Condensed Nearest Neighbors", Artificial Intelligent Review 11:115-132, 1997.

[35] Geoffrey W. Gates, "Reduced Nearest Neighbor Rule", IEEE Trans Information Theory, Vol. 18 No. 3, pp 431-433.

[36] G. Guo, H. Wang, D. Bell, "KNN Model based Approach in Classification", Springer Berlin Vol 2888.

[37] S. C. Bagui, S. Bagui, K. Pal, "Breast Cancer Detection using Nearest Neighbor Classification Rules", Pattern Recognition 36, pp 25-34, 2003.

[38] H. Parvin, H. Alizadeh and B. Minaei, "Modified k Nearest Neighbor", Proceedings of the world congress on Engg. and computer science 2008.

[39] Y. Zeng, Y. Yang, L. Zhou, "Pseudo Nearest Neighbor Rule for Pattern Recognition", Expert Systems with Applications (36) pp 3587-3595, 2009.

[40] Z. Yong, "An Improved kNN Text Classification Algorithm based on Clustering", Journal of Computers, Vol. 4, No. 3, March 2009.

[41] Stanoi, I., Agrawal, D., El Abbadi, A.: Reverse nearest neighbor queries for dynamic databases. In: Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 44–53 (2000)

[42] Korn, F., Muthukrishnan, S.: Influence sets based on reverse nearest neighbor queries. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 201–212 (2000)

[43] Yang, C., Lin, K.-Ip.: An index structure for efficient reverse nearest neighbor queries. In: Proceedings of the International Conference on Data Engineering, pp. 485–492 (2001)

[44] Maheshwari, A., Vahrenhold, J., Zeh, N.: On reverse nearest neighbor queries. In: Proceedings of the Canadian Conference on Computational Geometry, pp. 128–132 (2002)

[45] Singh, A., Ferhatosmanoglu, H., Tosun, A.: High dimensional reverse nearest neighbor queries. In: Proceedings of ACM CIKM International Conference on Information and Knowledge Management, pp. 91–98 (2003)

[46] Tao, Y., Papadias, D., Lian, X.: Reverse kNN search in arbitrary dimensionality In: Proceedings of the VLDB Conference, pp. 744–755 (2004)

[47] T. Xia and D. Zhang. Continuous reverse nearest neighbor monitoring. In :Processings of the IEEE *International Conference on Data Engineering*. 2006.

[48] Tobias Emrich, Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Andreas Züfle," Constrained Reverse Nearest Neighbor Search on Mobile Objects ",ACM GIS '09 , November 4-6, 2009

[49] Dimitris Papadias,Yufei Tao, Kyriakos Mouratidis and Chun Kit Hui ," Aggregate Nearest Neighbor Queries in Spatial Databases", In: Proceedings of ACM Transactions on Database Systems, Vol. 30, No. 2, June 2005, Pages 529–576.

[50] Yunjun Gao, Baihua Zheng, Gencai Chen, Qing Li, Chun Chen and Gang Chen "On efficient mutual nearest neighbor query processing in spatial databases" ,Elsievier Data & Knowledge Engineering, Vol. 68, May 2009, Pages 70.

7. AUTHORS PROFILE

S. Dhanabal received the Master degree in Computer Science and Engineering from Anna University, Chennai, India in 2008, and currently pursuing Ph.D. degree in Computer Science and Engineering in Anna University of Technology, Coimbatore, India. He is having more than 12 years of experience in teaching. He is currently working as an Assistant Professor of CSE department in Jansons Institute of Technology, Anna University of Technology, Coimbatore, India. His research interests include spatial databases, spatiotemporal databases, mobile/pervasive computing, and geographic information systems. He has presented 5 papers in national conferences. He is a member of Computer Society of India.

Dr. S. Chandramathi received her Ph.D. from College of Engineering, Anna University - Chennai. She is having more than 26 years of teaching experience. She is currently working as Professor and Head of Electronics and Communication department in Sri Krishna College of Engineering & Technology, Anna University of Technology, Coimbatore, India. Her research interests include Computer Networks, Wireless Communication, Sensor Networks and Digital Communication. She has published 8 papers in International Journals and 40 papers in National / International Conferences.