CLUSTER ANALYSIS ON HIGH-DIMENSIONAL NON-LINEAR DATA: DENSITY-BASED CLUSTERING ALGORITHMS

R.NANDHAKUMAR[1] and **Dr.ANTONY SELVADOSS THANAMANI** [2]

[1.] *Assistant Professor, Department of Computer Science,*

Nallamuthu Gounder Mahalingam College, Pollachi-642001, India

[2.] *Associate Professor & Head, Department of Computer Science,*

Nallamuthu Gounder Mahalingam College, Pollachi-642001, India

E-Mail: nkumarram@gmail.com, Mobile:9965551124

Abstract: The effectiveness and efficiency of the existing cluster analysis methods are limited, especially when the referred data has high dimensions or when the clusters within the data are not well-separated and having different densities, sizes and shapes. Density-based clustering algorithms have been proven able to discovered clusters with those characteristics. Previous researchers which explored density-based clustering algorithms focused on the analyzing the parameters essential for creating meaningful spatial clusters. The aim of this paper is to provide a comparative study of three well know density-based clustering algorithms including DBSCAN, DENCLUE and LTKC. The merits of them were evaluated of their ability to cluster several high-dimensional artificial data. We concluded that each density-based data clustering algorithm has their individual merits for high-dimensional data. However, further research is needed in the application of the techniques to analyze other high-dimensional non linear data, to permit a comprehensive evaluation of their respective strengths and limitations as powerful cluster analysis methods.

Key words: Cluster analysis; high-dimensional non linear data; density-based clustering; DBSCAN; DENCLUE; LTKC.

INTRODUCTION

Clustering is a powerful exploratory technique for extracting knowledge of given data (Witten, Frank, & Hall, 2011). Clustering is an unsupervised process, because in the clustering process, there are no predefined classes and no examples that would show what kind of desirable relations should be valid among the data (Halkidi, Batistakis, & Vazirgiannis, 2001). Thus, clustering process uses the similarity and dissimilarity pair of data within the dataset for grouping unlabelled datasets.

Recently, several clustering techniques have been introduced and proposed, such as *k*-means (Anderson, 2009; Golob & Recker, 2004; Shekhar, Lu, Chawla, & Zhang, 2001), hierarchical clustering (Skyving, Berg, & Laflamme, 2009) and Support Vector Machine-based approach (SVM) (W. Chang, Zeng, & Chen, 2005). *K*-means technique is probably the most popular and is a simple solution for clustering. However, the weakness with this technique is in determining the proper number of clusters and potential to being trapped in local optimal (Tan, Steinbach, & Kumar, 2006). Determining the proper number of clusters is an essential issue in clustering task, since it is difficult to choose an appropriate number of clusters, especially when the data has high dimensions, or when clusters within the data are not well-separated and having different densities, sizes and shapes.

In addition, the data with high dimensional data normally have a problem with curse of dimensionality. The curse of dimensionality refers to the phenomenon in which many types of data analysis become significantly harder as the dimensionality of the data increases. For clustering, the definition of density and the distance between points, which are critical for clustering would often become meaningless (Tan, et al., 2006). This problem indicates that the complexity of clustering the data grows exponentially with the number of dimensions (i.e., variables) that it comprises. For example, in similarity searching (finding nearest neighbours), it indicates that the number of data points in the dataset that need to be observed in getting the estimate, grow exponentially with the underlying dimension. As a result, many clustering algorithms have trouble with high-dimensional non linear data since it can be reduce the accuracy rate and produce poor quality clusters (Tan, et al., 2006). Therefore, developing clustering algorithms that can handle the complexity of high-dimensional non linear data and the heterogenic of the clusters is still a challenging issue in cluster analysis domain.

Among several clustering algorithm types, density-based clustering algorithm is so far the most efficient in detecting the cluster with different densities, since it determines the number of clusters automatically by analyzing the density of data points in a region (Martin Ester, Hans-Peter Krigel, Jorg Sander, & Xiaowei Xu, 1996a). A cluster in these algorithms is a dense region of data point that is surrounded by a region of low density. Density-based clustering algorithm uses local cluster criterion in which the clusters are defined as region in data space whose data points are dense, and the clusters are separated from one another with low-density region (Martin Ester, Hans-Peter Krigel, Jorg Sander, & Xiawei Xu, 1996b).

Previous researchers, such as (Parimala, Lopez, & Senthilkumar, 2011), which explored clustering algorithms focused on the analyzing density-based clustering algorithm in term of the parameters essential for creating meaningful spatial clusters. In particular, they tended to concentrate on the variants of DBSCAN algorithm in the clustering spatial data. The aim of this paper is to provide a comparative study of several well know density-based clustering algorithms. The merits of three density-based clustering algorithms; DBSCAN, DENCLUE and LTKC, were evaluated of their ability to cluster high-dimensional non linear data. To evaluate the performance of the density-based clustering algorithms, we utilized various benchmarked artificial data. The following sections describe each of the density-based clustering algorithms and the experimental results of various benchmarked artificial data.

Density based Spatial Clustering of Application with Noise (DBSCAN):

Introduction:

DBSCAN proposed by (Ester, et al., 1996a) and used by (Le-Khac, Bue, Whelan, & Kechadi, 2010), (Whelan, Le-Khac, & Kechadi, 2011), (Kechadi & Bertolotto, 2007) and (Birant & Kut, 2007), defines density of point by counting the number of point in a region of specified radius *Eps* around the point. Points with density above the specified threshold are classified as core points, while noise points are defined as non-core points that do not have a core points within specified radius. Noise points are discarded, while clusters are formed around the core points. If two core points are alongside each other, then their clusters are joined. Non- noise and non-border points, which are called boundary points, are assigned to the clusters associated with any core point within their radius. Thus, the core points form the skeleton of the clusters, while the border points flesh out this skeleton. Figure 1 shows the definition of core, border and noise points in DBSCAN algorithm.

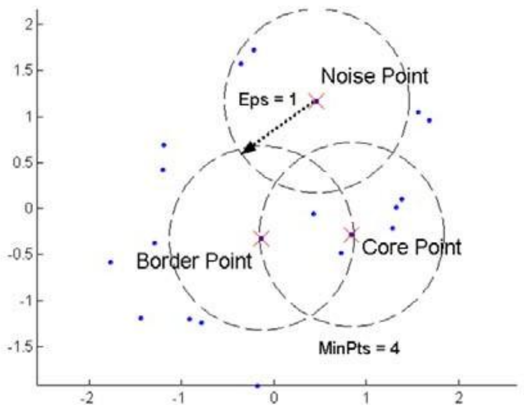


Fig. 1: Definition of points in DBSCAN.

Each point in a cluster in the neighborhood of a given radius has to contain at least a minimum number of points *minpts*. It implies that the density in the neighborhood has to exceed the threshold, *Eps*. The shape of a neighborhood is defined by the choice of a distance function for two points *p* and *q*, denoted by *dist*(*p*,*q*). For example, in two dimensional data, the shape of the neighborhood is circle when Euclidean distance is used or is rectangular when Manhattan distance is used.

Description of the Algorithm:

In this sub section, the DBSCAN algorithm is explained. The steps involved in DBSCAN algorithm are as follow:

1. Labelling all point as core, border, or noise points.
2. Putting an edge between all core points that are within *Eps* of each other
3. Making each group of connected core points into separate cluster
4. Assigning each border point to one of the clusters of its associated core points

Impact of the Algorithm:

DBSCAN requires two parameters: minimum number of neighbours *minpts* and minimum radius *Eps* (Ester, et al., 1996a). This is done by using the concept of determining the density that is based on *Eps* which makes DBSCAN has the capability to find clusters of arbitrary shapes. In addition, DBSCAN holds good for data with big size (Parimala, et al., 2011).

Future Works:

Otherwise, since its density is based on the definition of core points, DBSCAN cannot identify the core points of varying density. As a result, DBSCAN has difficulties to find clusters of differing densities. In addition, two parameters that are required for clustering high-dimensional non linear data are also difficult to be determined even when using heuristic methods (Ester, et al., 1996a).

DENsity-based CLUstEring (DENCLUE):

Introduction:

DENCLUE (DENsity CLUstEring) proposed by (Hinneburg & Keim, 1998) and modified by (Hinneburg & Gabriel, 2007) and (Hinneburg & Keim, 2003), improves DBSCAN through the use of kernel function as the influence function to express the contribution of each data point to the overall density function, instead of only using *Eps* value. The influence function describes the impact of a data point within its neighborhood. Examples for influence functions are parabolic functions, square wave function, or the Gaussian functions. The influence function is applied to each data point.

The overall density of the data space can be calculated as the sum of the influence function of all data points. Clusters can then be determined mathematically by identifying the density-attractors. Density attractors are the local maxima of the overall density function. If the overall density function is continuous and differentiable at any point, determining the density attractors can be done efficiently by using a hill-climbing procedure which is guided by the gradient of the overall density function. In addition, the mathematical form of the overall density function allows the clusters of arbitrary shape to be described in a very compact mathematical form, namely by using a simple equation of the overall density function.

Summary of Algorithm:

The DENCLUE algorithm procedure is summarized as follow:

1. Calculating the density function of each point
2. Identifying the points that are local maxima, usually called density attractor.
3. Defining clusters which consist of points associated with a particular density attractor.
4. Discarding clusters whose density attractor has a density less than a *mindens*.
5. Combining clusters that are connected by path of points that all have density of *mindens* or higher

Impact of Algorithm:

Similar to DBSCAN, DENCLUE also requires two parameters: minimum density *mindens* and wide of bandwidth *h*. By using kernel function as the influence function to express the contribution of each data point to the overall density function, DENCLUE is relatively stable with respect to outliers and still capable to find arbitrarily shaped clusters.

Future Works:

However, the DENCLUE algorithm is still improvable for more efficiency or for larger datasets. Firstly, DENCLUE needs to hold all of the original data to fulfill its clustering procedure. Thus, its behavior depends on the memory availability. For a large dataset that cannot fit into the main memory, swap in and swap out of the data objects can degrade its behavior sharply. Secondly, for each data object of the dataset, the algorithm has to compute its local density value by summing up all the influences of nearby objects with a point wise manner, whether those points are crowded together or distributed sparsely. It pays no attention to the statistical information of the grids around a data object. This negligence also complicates the algorithm markedly. Third, DENCLUE is relatively resistant to noise (Tan, et al., 2006). In addition, DENCLUE is computationally expensive than DBSCAN.

Local Triangular Kernel Clustering (LTKC):

Introduction:

LTKC algorithm proposed by (Musdholifah & Mohd. Hashim, 2010) is density-based clustering that determines the density of data points using combination of two nonparametric density estimation procedures. LTKC combines *k*-nearest neighbour (KNN) and kernel density estimation. KNN density estimation is extended and combined with triangular kernel function. LTKC uses Bayesian decision rule in order to assign objects to respective clusters. If the kernel nearest-neighbour density function value of object *x* in cluster ω_i is maximized, and then the target data will belong to the cluster ω_i .

Therefore, using triangular kernel function and KNN density estimation approach, the density cluster is determined with respect to the object *x*; next called triangular kernel density cluster ω of object *x* , Kt_ω, which can be defined as:

$$Kt_\omega = \frac{1}{n_\omega}\sum_{j=1}^{n_\omega}\left(1 - dist(x, x_j)\right) \qquad (1)$$

where x_j is the *k*-nearest neighbour of object *x* and member of cluster *ω, n_ω* is the number of objects in cluster *ω* and *dist*(*x*,x_j) is the distance of object *x* and x_j. In this research, the distance was considered as Euclidean distance and defined as

$$dist(x, x_j) = \sqrt[2]{\sum_{d=1}^{L}\left(x^d - x_j^d\right)} \qquad (2)$$

where *L* is the dimension of the data.

Summary of Algorithm:

The clustering algorithms with *k* number of nearest neighbours, was carried out in the following manner:

1. For each data point x_i in *d*-dimension, their Euclidean distance to all data points were calculated as

$$dist(x_i, x_j) = \sqrt[2]{\sum_{k=1}^{d}\left(x_i^k - x_j^k\right)} \text{ where } (i, j = 1, \dots, n).$$

2. Creating similarity matrix (size *n* x *n*), then sorting them in descending order.
3. Finding the *k*-nearest neighbour data points of each data point x_i ; which refers to first data point until k^{th} data point in the list of the sorted similarity matrix, and then creating *k-nn* table.
4. Calculating the triangular kernel function of each *k*-nearest data point of each data point x_i using distance value stored in *similarity matrix* and *k-nn* table.
5. Repeat

Each data point x_i was assigned to cluster ω that maximized the triangular kernel density of data point x_i, $Kt_\omega(x_i)$, locally,

$$Clust_\omega = \max_{\omega \in \omega_{nci}}\left(Kt_\omega(x_i)\right) \qquad (3)$$

where ω_{nci} is all cluster indices of the *k*-nearest neighbours of the data point x_i and $Kt_\omega(x_i)$ is triangular kernel function of data point x_i defined as Eq. 3.

Re-indexing the cluster label

6. Until the cluster structure was unable to change.

Impact of Algorithm:

LTKC considers the *k* nearest data point that the nearest data points refers to the data points within the *k*-top list of data point with smallest kernel density function. Thus, LTKC only requires a parameter input, which is the number of nearest neighbours, *k*. From the LTKC algorithm, it can be stated the number of clusters is not given to the algorithm as a parameter. Depending on the nature of the data, the algorithm finds “natural” clusters. In addition, all the objects in the data are clustered using out algorithm.

Future Works:

LTKC algorithm requires an input parameter *k* number of nearest neighbours. In order to determine the “optimal” *k*, LTKC clustering is executed using different *k* number of nearest neighbours; by running single LTKC once for each *k* from 2 to *n-1*. For each of the clustering results, the chosen validation measure is calculated; for instance accuracy rate, and finally, the clustering with best validation value (for instance, if the silhouette coefficient is used then the best validation value is the maximum accuracy rate). Unfortunately, the run time of this approach is prohibited for large *n*, because it implies *O*(*n*) calls of LTKC. Thus, one of the issues in LTKC clustering is how to determine the optimal number of nearest neighbours that produces good clusters.

Experimental Results and Discussions:

Experimental Data sets:

The ideal input for clustering algorithm is a dataset which is without noise and has a known number of equal size, equal density, and globular. When the data deviates from these properties, it poses different problems for different types of algorithms. Nine artificial datasets were utilized to evaluate the performance of the density-based clustering algorithms for discovering clusters on high-dimensional non linear data with different shapes, sizes and densities. Some datasets contained true clusters with soft boundary, arbitrary shapes and densities.

Table 1 represents the summary of the characteristic of the true clusters within the datasets. The distribution of points within the nine artificial datasets is shown in

Fig. 2. Different colours and symbols were used to demonstrate the different classes. From

Fig. 2 shows that all datasets used had different shapes and number of classes or true clusters, in range from 2 to 31.

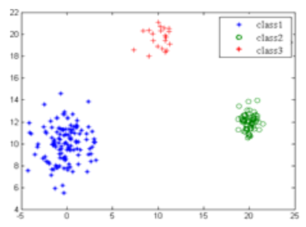
The artificial data 1 utilized in this experiment was an artificial spherical or globular data. In addition, all classes within this data were convex objects. An object is convex in Euclidean shape if every pair of points is within the object and the considered object on the straight line segment that joins them is also within the object. However, most benchmark and real data sets are not spherical data. Therefore, the objective of experiment on the artificial data 1 was to evaluate the performance of the proposed algorithm on spherical data, which was convex, centred, well separated and compact.

The artificial data 1 was generated through the use of a random number generator. This data had 200 instances with two attributes. The data distribution of each class was generated using various means $\mu_1 = [0,10]$, $\mu_2 = [20,10]$, and $\mu_3 = [10,20]$; variance $\sigma_1 = [1.7,1.7]$, $\sigma_1 = [0.7,0.7]$, and $\sigma_1 = [1.0,1.0]$ and number of data $n_1 = 120$, $n_2 = 60$, $n_3 = 20$. The data distribution is shown in

Fig. 2.a.

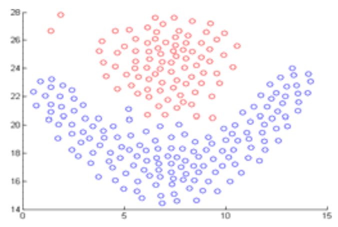
Table 1: Characteristics of the artificial dataset.

Artificial Data	*Number of true clusters*	*Number of points*	*Boundary*		*Shape type*	
			Soft	Hard	Convex	Non-convex
1	3	200	-	√	√	-
2	2	240	√	-	√	√
3	2	373	-	√	-	√
4	3	312	-	√	-	√
5	3	300	√	-	√	√
6	7	788	√	√	√	√
7	6	399	√	√	√	√
8	15	600	√	√	√	-
9	31	3100	√	√	√	-

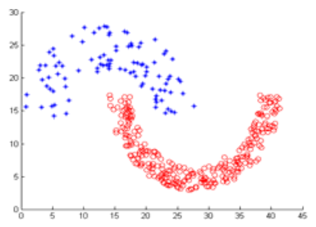


Artificial data 1

(a)

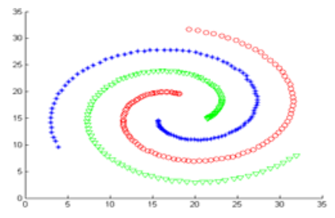


(b) Artificial data 2

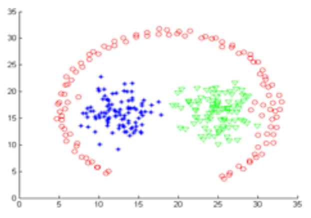


Artificial data 3

(c)

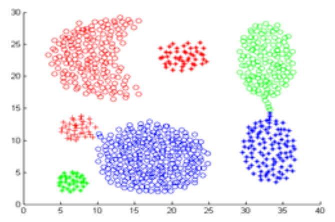


(d) Artificial data 4

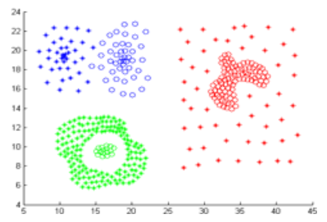


Artificial data 5

(e)

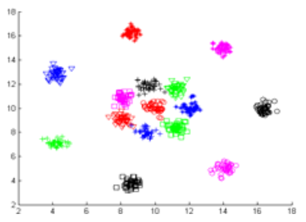


(f) Artificial data 6

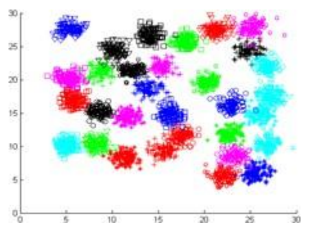


Artificial data 7

(g)



(h) Artificial data 8



(i) Artificial data 9

Fig. 2: Distribution of data points within the artificial data sets

The eight remaining artificial data sets were provided by Speech and Image Processing Unit ("Clustering datasets," 2012). In the artificial data 2 that was originally used by (Fu & Medico, 2007), there were two classes which one of them was non-convex shape and visualized by blue circle symbol. The motivation of utilizing the artificial data 2 that is shown in

Fig. 2.b was to analyze the capability LTKC algorithm on the data that contained different shapes and soft boundary.

The artificial data 3, initially used by (Jain & Law, 2005), formed two half-rings and contained 373 two-dimensional patterns; upper class had 97 patterns and lower class had 276 patterns. In addition, the classes within the artificial data 3 that is shown in

Fig. 2.c were non-convex shapes. Therefore, it was interesting to evaluate the performance of LTKC on the data sets that all the classes within the data set were non-convex.

The artificial data 4that is shown in

Fig. 2.d composed three classes with spiral shapes (Chang and Yeung, 2008). A spiral can be defined as a curve which derives from a central point which gets farther away as it resolves around the data point. However, all points in the artificial data 4 were not classified based on distance between the centre data points and the considered data points.

The artificial data 5, originally used by(H. Chang & D.-Y., 2008), had two circle classes inside a circular class. Each class shown in

Fig. 2.e contained 100 data points. In addition, the points in both circles were generated by using Gaussian distribution. This sample could be observed by utilizing two Gaussian classes with addition of some Gaussian noises which tended to connect the classes together. However, it was interesting to analyze whether LTKC discovered three clusters in a way similar to natural classes or LTKC would firstly consider the circular class as noise data.

The rest of the artificial data sets were more complex than previous artificial data sets because of the number of classes and shapes of classes contained in the data sets. For instance, artificial data 6 was a two-dimensional synthetic data that consisted of seven perceptually distinct classes.

Fig. 2.f shows a link between two globular classes that belonged to one of the two globular classes.

In the artificial data 7, introduced by (Zahn, 1971), and shown in

Fig. 2.g, there were five classes of different shapes and sizes with additional noise data points. The artificial data 8, shown in

Fig. 2.h was a ring data and contained 3100 random patterns generated by 15 similar 2D Gaussian distributions. Meanwhile, the artificial data 9 was the largest data among all data sets used in this research; it consisted of 31 randomly placed 2D Gaussian classes of 100 patterns each.

Comparison of Clustering Accuracy:

The proposed LTKC algorithm was conducted and applied on nine artificial data. The proposed LTKC algorithm only required one parameter; *k*, number of neighbours. LTKC was executed once for each *k* from 2 to *n*-1 where *n* is the number of data points. For each of the clustering results, the F-measure (Gullo, Ponti, & Tagarelli, 2008) was calculated and the clustering with maximum F-measure value was chosen as the true clusters. Meanwhile, applying DBSCAN algorithm required two parameters, minimum number of neighbour points, *MinPts* and maximum distance value or radius, *Pts*. However, according to (Ester, et al., 1996a), we should eliminated the parameter *MinPts* to 4 for all data sets. DENCLUE algorithm also required two parameters: the threshold of attractor merge, *mindens* and the width, *h*.

Table 2 describes the value of parameter used for all algorithms to discover the true clusters hidden in the artificial data sets.

From the experimental results of the artificial data 1, LTKC produced clusters which were equal to true clusters (classes) and were 100% accurate. The clustering result is shown in Figure 3(a). It was observed that each cluster was well-separated while the distance between points within a cluster to points within another cluster was large. Therefore, all clusters were discovered as having compact shapes since the distance of each pair of points within the same cluster was very close. Based on the experimental results, it has been proven that LTKC has ability to discover true clusters from spherical data sets.

For the artificial data 2, LTKC achieved the highest accuracy with 99.17%. In addition, from Figure 3(b), only two points were assigned to incorrect cluster due to their Euclidean distance values. Both points had relatively lower similarity values to other points within the upper cluster than those points closer to the lower cluster. The result of experiment on artificial data 2 demonstrated the ability of the LTKC approach to solve the

two clusters touching at a neck problem (Jain & Law, 2005) with a medium-sized neighbourhood (*k* equals to 10 as shown in
Table 2). There was no distinct point pair distance inconsistency at the neck, so that triangular kernel density function would be unable to break the data at the neck point.

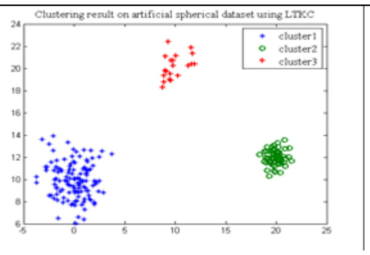
Again, the performance of LTKC was evaluated for non-convex shape class. In the artificial dataset 3, LTKC was proven capable to achieve two non-convex clusters as well as true class with 100% accuracy. However, the visual and quantitative analysis of experimental results on the artificial data 2 and 3 indicated that LTKC outperformed DBSCAN and DENCLUE for clustering non-convex shape classes.

Next, the artificial data 4 with three spiral classes was considered. The privileged clustering result was achieved by using LTKC, and DBSCAN. Two density-based clustering algorithms discovered correctly three spiral clusters, as shown in Figure 3(d) by using the neighbouring concepts and inter-point Euclidean distance between points within a cluster. The experiment results on the artificial data 4 showed the effects of variation of neighbourhood size quite clearly with respect to the continuity concept.

This research also included experiments on the artificial data 5 that contained three classes; a circular class with an opening near the bottom and two Gaussian distributed classes inside. All algorithms could not find the three clusters correctly as shown in Figure 3(e).

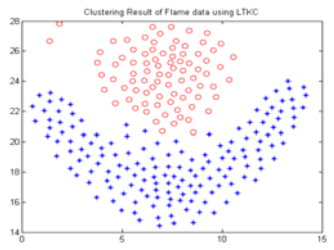
Table 2: Parameters of all algorithms used to discover true clusters of the nine artificial data sets.

Artificial Dataset	*LTKC*	*DBSCAN*		*DENCLUE*	
	k	*MinPts*	*Pts*	*Mindens*	*h*
1	10	4	0.985	12	1
2	10	4	0.985	12	1
3	22	7	2.446	23	2.5
4	7	4	1.843	20	1.85
5	20	4	1.839	20	1.85
6	16	4	1.208	12	1.2
7	15	4	1.393	10	1.4
8	38	4	0.635	1	0.6
9	71	4	0.535	4	0.9



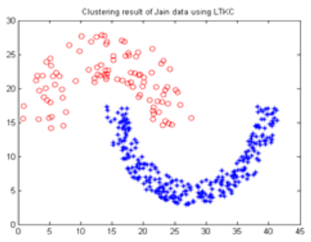
Artificial data 1

(a)



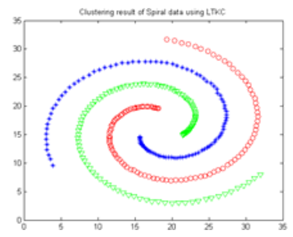
Artificial data 2

(b)

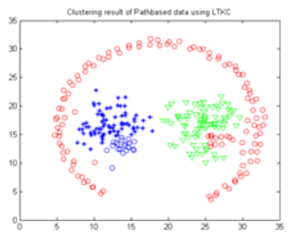


Artificial data 3

(c)

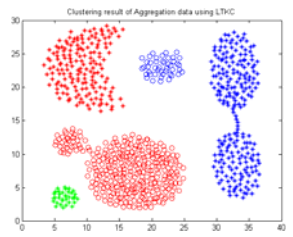


(d) **Artificial data 4**

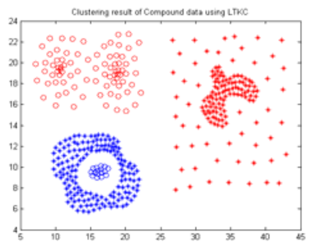


Artificial data 5

(e)

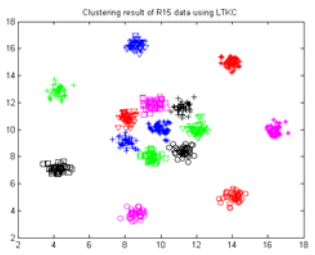


(f) **Artificial data 6**



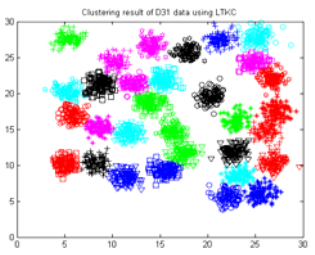
Artificial data 7

(g)



Artificial data 8

(h)



(i) **Artificial data 9**

Fig. 3: Clusters of the artificial data sets produced by LTKC.

Table 3: Percentage of F-measure for the experiment results of all algorithms on the artificial data sets.

Artificial Data	LTKC	DBSCAN	DENCLUE
1	100	100	100
2	99.17	64.58	80.83
3	100	73.99	81.77
4	100	100	34.29
5	87.00	80.67	53.67
6	78.43	82.23	58.5
7	87.22	96.99	66.42
8	99.67	53.33	99.50
9	97.55	62.54	76.03

Again, the proposed LTKC algorithm gave more satisfactory result, as shown in Table 3. However, some inter-cluster points were misclustered. Our proposed method was successful in assigning the points that were closer to these points and hence essentially detecting them as outliers. As a result, they had relatively lower similarity values to other points within a cluster than those points closer to the Gaussian centres or on the incomplete circle.

The clustering result for the artificial data 6 produced by LTKC algorithm was imperfect as shown in Figure 3(f). In fact, this data contained seven classes but LTKC discovered five clusters only. Some points formed narrow bridge between of two classes as these two classes were assigned into one cluster.

In the artificial data 8, the proposed LTKC algorithm produced 15 circle clusters with 99.67% accuracy. It indicates only two points were not clustered correctly because the points were closer to the clusters than to the true class. Remarkably, the DBSCAN algorithm was not able to discover the originating cluster structure. Moreover, the accuracy of clustering result produced by DBSCAN was less than 55%.

Again, LTKC algorithm outperformed than DBSCAN and DENCLUE in the largest data. 31 clusters were found by LTKC with 97.55 % accuracy. 76 points were misclustered since the Euclidean distance values between the points intra-cluster were lower those of inter-cluster.

From the experimental results on the artificial data sets, it can be stated that the density based clustering algorithm such as LTKC and DBSCAN are suitable for discovering true clusters as well as natural clusters with arbitrary shapes on datasets that contain points that are close within one class or well separated. In addition, LTKC algorithms that utilized Euclidean distance measure had an accuracy close to 100% on data which contained classes with circle shapes, because Euclidean distance is suitable for objects with circle shape.

In conclusion, the promising performance of LTKC algorithms had been demonstrated well in dealing some difficult data with arbitrary shapes but there existed some other situations when these algorithms did not perform well. One such condition was when there were many noise points within the dataset, such as in artificial data 6 and 7, where LTKC still assigned them as a group or a cluster.

Comparison of Time Complexity:

In density-based clustering algorithms, the problem of finding neighbours is somewhat similar to the point location query. In particular, in many algorithms, the first step is to locate the query object q. The second step consists of the actual search. Most of the classical methods employed a depth-first strategy. An alternative, they employed a best-first search strategy. The difference between these two strategies is that, in the best-first strategy, the elements of the structure where the objects are stored are explored in an ascending order of their distance from q. In contrast, the dept-first strategy, the order in which the elements of the structure where the objects are stored are explored is a result of performing a depth-first traversal of the structure using the distance to the currently nearest objects to prune the search. However, in this research, a best-first search strategy was chosen for searching the nearest neighbours of object.

Another issue that can influence the performance of the clustering algorithm is calculating of the distance a pair of objects. The commonly distance function between two objects used is Euclidean distance. Although the Euclidean metric was mentioned in first step of LTKC algorithm, the method was with no mean restricted to this measure and any suitable measure could be used. In addition, all k-nearest neighbours could be found with relatively little computational expense over that finding of the nearest neighbour by using the previously described scheme. For each neighbourhood list row generation, only n distance measures are needed to be calculated; if d is large, this represents the largest computational expense algorithm. Therefore, the computational complexity of calculating the neighbour table is of the order $(n^2)d + C(k)$ operations where C is relatively a small factor to allow for the extra overhead of testing for all k-near neighbours for each data point. Only $n(n+1)/2$ distance measure are necessary, but in the algorithm implemented redundant distance calculations were tolerated to enhance programming simplicity and converse storage. The computing complexity for one pass of the neighbourhood table of explore clusters for neighbourhood size k is of the order of (at most) $(n(n-1)/2)(k+1)^2$ integer comparisons, plus the data order and threshold dependent cost of the link listing procedure that is evoked only when the matches of sufficient vote are detected. The distance calculation process requires $O(n^2)$ memory.

However, for huge dataset (over 10,000 objects), it is suggested that there is a need to separate the distance calculation process and store the distance matrix in another file or variables. Thus, the computational complexity of LTKC will depend mainly on the calculation of the k-nearest neighbour list through the kernel density estimation stage that requires the search for the data points failing in the neighbourhood. In the results, their complexity will be relative to very expensive. The proposed LTKC has the simplest way to solve this problem. LTKC orders all the distances from the considered point to other objects, which leads to a complexity of $O(n\log(n))$.

Next, the computational complexity of two other clustering algorithms was described. DBSCAN and DENCLUE visited each point within the dataset, possibly multiple times. For practical considerations, however, the time complexity was mostly governed by the number of region Query invocations. DBSCAN executed exactly one such query for each point, and if an indexing structure was used to execute such a neighbourhood in $O\log(n)$, an overall runtime complexity of $O(n\log(n))$ would be obtained.

Conclusions:

In this paper, we reviewed and compared three well-known density-based clustering algorithms which are DBSCAN, DENCLUE and LTKC based on the essential requirement needed for any clustering algorithms in high-dimensional non linear data. Each algorithm is unique with its own characteristics. A comparative study in term of input parameters, shapes of the cluster, density and the type of the data is given in **Error! Reference source not found.**. Furthermore, the result of investigation can be used to other researchers to study, apply and enhance the density-based clustering algorithm.

From **Error! Reference source not found.**, it can be stated that LTKC algorithm is able to tackle the difficulties of high dimensional data and cluster with different densities, shapes and sizes. The experiments showed that the LTKC algorithm could achieve more accurate clustering result and use lesser processing time compared to DBSCAN and DENCLUE. Furthermore, the result of investigating can be used by other researchers to study, apply and enhance the density-based clustering algorithm.

Table 4: Comparison of three density-based clustering algorithms.

Algorithm	*Different shapes*	*Different sizes*	*Different densities*	*High-dimensional*
DBSCAN	√	√	-	√
DENCLUE	√	√	-	√
LTKC	√	√	√	√

REFERENCES

1. Anderson, T.K., 2009. Kernel Density Estimation and K-means Clustering to Profile Road Accident Hotspots. *Accident Analysis and Prevention,* 41(3): 359-364.
2. Birant, D., A. Kut, 2007. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data and Knowledge Engineering,* 60(1): 208-221.
3. Clustering datasets, 2012. from Speech and Image Processing Unit, School of Computing, University of Eastern Finland:
4. Halkidi, M., Y. Batistakis, M. Vazirgiannis, 2001. On clustering validation techniques. *Journal of Intelligent Information Systems,* 17(2-3): 107-145.
5. Hinneburg, A., H.H. Gabriel, 2007. DENCLUE 2.0: Fast Clustering Based on Kernel Density Estimation.
6. *LNCS,* 4723: 70-80.
7. Hinneburg, A., D.A. Keim, 1998. *An effecient approach to clustering in large multimedia databases with noise.* Paper presented at the The fourth international conference on knowledge discovery and data

mining (KDD'98), Menlo Park, CA.

8. Jain, A.K., M.H.C. Law, 2005. Data clustering: A user's dilemma. *Lecture Notes in Computer Science,*

9. 3776: 1-10.

10. Kechadi, M.T., M. Bertolotto, 2007. Scalable 2-Pass Data Mining Technique for Large Scale Spatio-Temporak Datasets. *LNAI,* 4693: 785-792.

11. Shekhar, S., C.T. Lu, S. Chawla, P. Zhang, 2001. *Data Mining and Visualization of Twin-cities Traffic Data*: University of Minnesota.

12. Witten, I.H., E. Frank, M.A. Hall, 2011. *Data mining: Practical machine learning tools and techniques*.

13. Burlington, USA: Morgan Kauffmann.

14. Zahn, C.T., 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transaction on*, 100(1): 68-86.

AUTHORS PROFILE

Name: R.NANDHAKUMAR

Assistant Professor, Department of Computer Science

Nallamuthu Gounder Mahalingam College, Pollachi-642001, India.

Name: Dr. ANTONY SELVADOSS THANAMANI

Associate Professor and Head

Nallamuthu Gounder Mahalingam College,Pollachi-642001,India