ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/220704900

An Enhanced Density Based Spatial clustering of Applications with Noise.

Conference Paper · January 2009

Source: DBLP

CITATIONS
18

READS
1,983

5 authors, including:

Ahmed M. Fahim

21 PUBLICATIONS **427** CITATIONS

SEE PROFILE

Gunter Saake

Otto-von-Guericke-Universität Magdeburg

677 PUBLICATIONS **8,640** CITATIONS

SEE PROFILE

Abdel-Badeeh M.Salem

Ain Shams University

329 PUBLICATIONS **3,452** CITATIONS

SEE PROFILE

Mohamed A. Ramadan

Menoufia University

150 PUBLICATIONS **1,630** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Polarity Opinion Mining View project

Data warehouse Technologies View project

All content following this page was uploaded by Ahmed M. Fahim on 02 June 2014.

The user has requested enhancement of the downloaded file.

An Enhanced Density Based Spatial clustering of Applications with Noise

A. Fahim[1], **G. Saake**[1], **A. Salem**[2], **F. Torkey**[3] **and M. Ramadan**[4]
[1]Faculty of Information, Magdeburg University, Magdeburg, Germany
[2]Faculty of Information, Ain Shams University, Cairo, Egypt
[3]Kafrelshiekh University President, Kafrelshiekh, Egypt
[4]Faculty of Science, Menufiya University, Shebin El-Kom, Egypt

Abstract – *Cluster analysis is a primary method for data mining. Finding clusters with varying sizes, shapes and densities is a challenging job. DBSCAN can find clusters with varying shapes and sizes. But it has a trouble in finding clusters with varying densities, because it depends on a global value for its parameter Eps. This paper presents enhanced DBSCAN which clusters databases containing clusters with varying densities effectively. The idea is to use varied values for Eps according to the local density of the starting point in each cluster. The clustering process starts from the highest local density point towards the lowest local density one. For each value of Eps, DBSCAN is adopted to make sure that all density reachable points with respect to current Eps are clustered. At the next process, the clustered points are ignored, to avoid merging among denser clusters with sparser ones.*

Keywords: DBSCAN, Data Clustering, Density Clustering.

1 Introduction

Spatial data clustering is one of the promising techniques of data mining. The main goal of clustering is to group data objects into clusters such that objects belonging to the same cluster are similar, while those belonging to different ones are dissimilar. Data clustering algorithms can be classified into four categories; (1) partitioning, (2) hierarchical, (3) density-based and (4) grid-based. However, some algorithms may fall into more than one category. By clustering one can identify dense and sparse regions and, therefore, discover overall distribution patterns. Finding clusters in data is challenging when the clusters are of widely differing sizes, shapes and densities and when the data contains noise and outliers. Although many algorithms exist for finding clusters with different sizes and shapes, there are a few algorithms that can detect clusters with different densities.

Basic density based clustering techniques such as DBSCAN [5] and DENCLUE [6] treat clusters as regions of high densities separated by regions of no or low densities. So they are able to suitably handle clusters of different sizes and shapes besides effectively separating noise and outliers. But they fail to identify clusters with varying densities unless the clusters are separated by sparse regions [3]. There are some algorithms which can handle clusters of different densities, like OPTICS [1] but it does not produce explicit clusters. Traditional DBSCAN may have trouble with clusters of varying densities. For example, in the dataset shown in Fig. 1.a, DBSCAN fails to find the four clusters, because this data set has four levels of density, the clusters are not totally separated by sparse regions and the value of *Eps* is global for the dataset. So we need an approach able to stop the expanding of cluster at different levels of density. In Fig. 1.b there are two problems; they are in the smallest sparse cluster and the largest cluster which contains three dense clusters within it. In Fig. 1.c, DBSCAN discovers only the three small clusters and considers the other two large clusters as outliers or merges the three small clusters in one cluster to be able to find the other two large clusters.

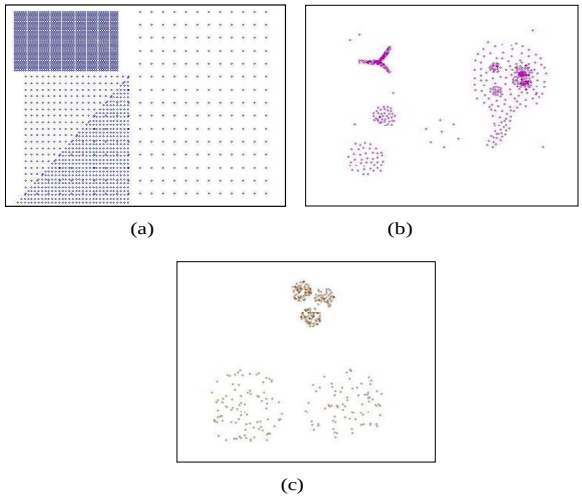

Fig. 1. clusters with varying densities

This paper introduces an enhanced version of the DBSCAN algorithm which is able to discover clusters with varying densities. It selects suitable values for its parameter *Eps* for each cluster. It is based up on the local density of the

starting point in each cluster, and adopts the traditional DBSCAN for each value of *Eps*. The idea of the proposed algorithm depends on discovering the highest density clusters first, and then the *Eps* is adapted to discover the next low density clusters with ignoring the previously clustered points. The proposed algorithm requires two input parameters; they are *Minpts* and *Maxpts*. The *Maxpts* allows the value of *Eps* to be different from one cluster to another according to the local density of the initial point in each one. The *Minpts* determines the lowest level of density allowed inside the cluster, while the *Maxpts* limits the highest level of density allowed inside the cluster in addition to controlling the values of *Eps* neighboring radius of the traditional DBSCAN algorithm. We can note that the DBSCAN starts to create a cluster from any core point, while the enhanced DBSCAN starts the clustering process from the highest density core point. We refer to this core point as the starting point or the initial point.

Rest of the paper is organized as follows. Section 2 surveys the main definitions of DBSCAN. Section 3 briefly surveys some of recent clustering methods. The proposed algorithm is presented in section 4. Section 5 presents some experimental results to evaluate the algorithm. Finally, section 6 presents a conclusion.

2 Density based clustering (DBSCAN)

Since the proposed algorithm is an enhanced version of DBSCAN algorithm, it is important to review the main definitions of DBSCAN algorithm, for details see [5]. Let D is a database of N points in d-dimensional space R^d. The distance between two data points p and q is given by the Euclidean distance and denoted by $d(p,q)$. The DBSCAN depends on the following definitions:

- The *Eps-neighborhood* of a point p, denoted by $NEps(p)$, is defined by $NEps(p) = \{q \in D \mid d(p,q) \leq Eps\}$.
- A point p is *directly density-reachable* from a point q wrt. *Eps* and *Minpts* if $p \in NEps(q)$ and $|NEps(q)| \geq Minpts$ (core point condition).
- A point p is *density-reachable* from a point q wrt. *Eps* and *Minpts* if there is a chain of points $p_1, ..., p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i.
- A point p is *density-connected* to a point q wrt. *Eps* and *Minpts* if there is a point o such that both, p and q are density-reachable from o wrt. *Eps* and *Minpts*.
- A *cluster* C wrt. *Eps* and *Minpts* is a non-empty subset of D satisfying the following conditions:
  - $\forall p, q$: if $p \in C$ *and* q *is* density-reachable from p wrt. *Eps and Minpts*, then $q \in C$. (Maximality).
  - $\forall p, q \in C$: p *is* density-connected *to* q wrt. *Eps* and *Minpts*. (Connectivity).
- If $C_1, ..., C_k$ be the clusters of the database D wrt. parameters Eps_i and $Minpts_i$, $i = 1, ..., k$. Then the noise is defined as the set of points in the database D not belonging to any cluster C_i, i.e. noise = $\{p \in D \mid \forall i: p \notin C_i\}$.
- If $d(p,q) \leq Eps$, q is core point and $|NEps(p)| < Minpts$ then p is a border point.

DBSCAN searches for clusters by checking the *Eps neighborhood* of each point in the database. If the *Eps neighborhood* of a point P contains *Minpts* points or more, a new cluster with P as *core point* is created. Then DBSCAN iteratively collects directly *density-reachable* points from these core points, which may involve the merge of a few *density-reachable* clusters. The process terminates when no new points can be added to any cluster. In DBSCAN the *Minpts* is fixed to 4 points, and the number of discovered clusters depends on the *Eps* value which is fixed during the execution time, and this value is not suitable for discovering clusters with different densities except when the clusters are totally separated. But when the clusters are not totally separated the DBSCAN algorithm faces the problem of varying densities and produces inaccurate clusters.

3 Related works

The DBSCAN (Density Based Spatial Clustering of Applications with Noise) [5] is a basic density based clustering algorithm. The density associated with a point p is obtained by counting the number of points in a region of specified radius *Eps* around p.

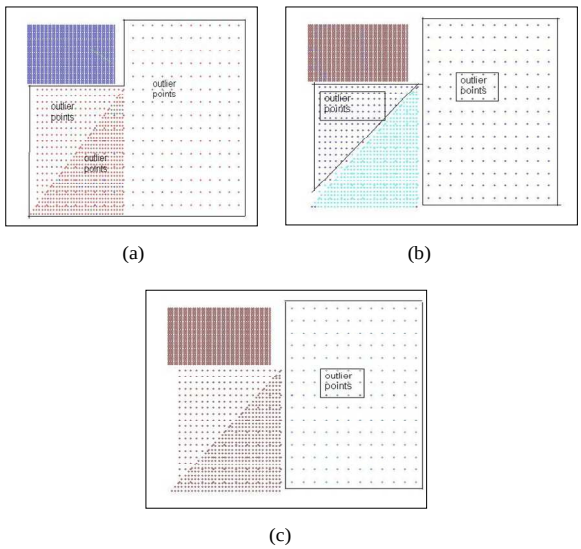

Fig.2. (a) DBSCAN discovers only the densest cluster (*Eps*=0.25). In (b) it discovers two clusters satisfying the lowest possible density (*Eps*=0.4). In (c) it merges the two clusters in (b) with the intermediate points to produce only one cluster (*Eps*=0.5) and discards the other points as noise.

A point with density greater than or equal to a specified threshold *Minpts* is treated as core point (dense), otherwise non-core (sparse). Non-core points that do not have a core point within the specified radius are discarded as noise. Clusters are formed around core points by finding sets of density connected points that are maximal with respect to density-reachability. DBSCAN can find clusters having varying sizes and shapes, but there may be wide variation in local densities within a cluster since it uses global density parameters *Minpts* and *Eps*, which specifies only the lowest possible density of any cluster without restricting the highest possible density. Fig. 2 shows the results of applying the DBSCAN on the dataset in Fig. 1.a. From Fig. 2, we note that for some datasets we need many different values for the *Eps* parameter of DBSCAN algorithm to discover clusters with different densities, and we can not depend on a single value for this sensitive parameter. This problem motivates us to propose an enhanced version of DBSCAN which is able to discover clusters from datasets having varying densities.

To find clusters that are naturally presented in dataset very different local densities needed to be identified and separated into clusters. The proposed algorithm performs this task efficiently. DBSCAN is very interesting algorithm; it has received a lot of attention from many researchers. But most of them concentrated their efforts in improving the scalability of it. We briefly review some of these works; for example in IDBSCAN [4], the researchers improved the scalability of the algorithm by reducing the number of query region. This has been done by sampling the points in the *Eps* neighborhood of core point *p* and marking these points as Marked Boundary Objects (*MBO*), for each of these *MBOs*, the closest point in the *Eps*-neighborhood -as in Fig. 3- is identified and selected as a seed. If the same point is identified as the nearest point for more than one *MBO* then this point must be regarded only once as a seed. Therefore total number of seeds selected may be less than or equal to eight in 2-dimensional case. In general, if the points are of dimension *d*, then there will be $(3^d\text{-}1)$ *MBOs*, 2^d quadrants and number of seeds selected is at most $3^d\text{-}1$. This number is lower than the corresponding number in DBSCAN. IDBSCAN [4] is more efficient than DBSCAN, but it occasionally may treat some boundary objects as noise producing a few more noise points than DBSCAN.

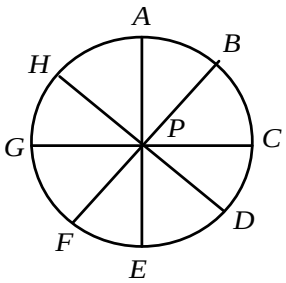

Fig. 3. Circle with eight *MBO*

Another algorithm was introduced to improve the efficiency of DBSCAN by merging between the k-means and IDBSCAN, this algorithm is known as KIDBSCAN [9]. K-means yields the core points of clusters. Then, clusters are expanded from these core points by executing IDBSCAN. The purpose of the k-means is to determine the *k* highest density center points and the points that are closest to these center points. These points are moved to the front of data. And the IDBSCAN starts the expansion of clusters from high density center points. This process reduces the number of redundant steps and increases efficiency of IDBSCAN.

In [8] the authors introduced VDBSCAN (Varied Density Based Spatial Clustering of Applications with Noise). First, VDBSCAN calculates and stores k-dist for each point. K-dist plot is drawn for selection of parameters Eps_i and analysis of density levels of the dataset. Second, the number of densities is given intuitively by k-dist plot. Third, the user chooses parameters Eps_i for each density. Fourth, VDBSCAN scans the dataset and clusters different densities using corresponding Eps_i. And finally, it displays the valid clusters corresponding with varied densities. VDBSCAN has two steps: choosing parameters Eps_i and clustering in varied densities. This method depends on seeing several smooth curves connected by greatly variational ones, and in many cases we see only one smooth curve, referring to our experimental results we see only one curve in dataset 8 and two curves in dataset 7 and more than 12 curves in 4-dist plot in dataset 3. And this is only true for dataset 4 where the points of each region are uniformly distributed.

Recently DDSC (Density Differentiated Spatial Clustering) [3] was proposed. It is an extension of the DBSCAN algorithm to detect clusters with differing densities. Adjacent regions are separated into different clusters if there is significant change in densities. DDSC starts a cluster with a homogeneous core object and goes on expanding it by including other directly density-reachable homogeneous core objects until non homogeneous core objects are detected. A homogeneous core object *p* is a core object whose density (the count of points in its *Eps*-neighborhood) is neither more nor less than α time the density of any of its neighbors, where $\alpha > 1$ is a constant. DDSC requires three input parameters, they are *Eps*, *Minpts* and α. As we know *Eps* and *Minpts* depend on each other. However, finding the correct parameters for standard density based clustering [5] is more of an art than science. In addition to the third parameter α, which represents the allowed variance in densities within each cluster, and proper tuning of the parameter values is very important for getting good quality results.

4 The Proposed Algorithm (Enhanced DBSCAN)

In this section, we describe the details of the proposed algorithm. As DBSCAN is sensitive to *Eps*, the proposed algorithm will adjust different values for this parameter in each cluster. The algorithm finds the k-nearest neighbors for each point in given dataset as DBSCAN does, but here the

enhanced algorithm does not build the *sorted k-dist graph*. Based on the k-nearest neighbors, a local density function is used to find the local density at each point which is an approximation of the overall density function [6]. The overall density of the data space can be calculated as the sum of the influence functions of all data points. The influence function can be seen as a function which describes the impact of a data point within its neighborhood, and it is applied to each data point. Local density at a data point is computed according to the following functions:

Influence function represents the impact of point *x* on point *y* as the Euclidean distance between them.

$$INF(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \qquad (1)$$

Where *d* is the dimensionality of points. As the distance between the two points decrease the impact of *x* on *y* increase, and vice versa.

The ***local density function*** at point *x* is defined as the sum of the (influence functions within the *k*-nearest neighbors) distances among the point *x* and its k-nearest neighbors, this is shown in (2).

$$DEN(x, y_1, ..., y_k) = \sum_{i=1}^{k} INF(x, y_i) \qquad (2)$$

The definition of local density based on the sum of distances of the k-nearest neighbors is better than counting the points in the neighborhood radius (*Eps*). Consider the following example as in Fig. 4.

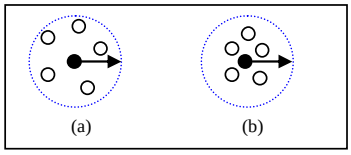

Fig. 4. Density based on k distances.

As we see in Fig. 4.a and 4.b both black points have five points in *Eps*-neighborhood (the *Eps* is the same represented by black arrow) but Fig. 4.b is denser than Fig. 4.a. The sum of distances reflects this fact accurately. In this example *Maxpts* =5, and *Minpts* = 4. So the proposed algorithm detects appropriate value for *Eps* in each cluster that will be *Maxpts*-distance from the starting point in the cluster. But based on fixed *Eps*-neighborhood radius as in DBSCAN algorithm there is no difference, because each point has five points in its *Eps*. And we can note that the value of *Eps* varies according to local density by accumulating the distances to the *Maxpts*-neighbors. Also the point inside the cluster has high density, on the other hand the point at the edge (border) of cluster has low density, since the neighbors for this point lie on one side of it, but the point at the core of cluster has its neighbors surrounding it from all sides. So density based on summing of distances is better than counting points in neighborhood radius.

We use the advantage of k-nearest neighbors to determine a suitable value for *Eps* in each cluster separately. The k-nearest neighbors capture the concept of neighborhood dynamically. The neighborhood radius of a data point is determined by the density of the region in which this data point resides. In a dense region, the neighborhood is defined narrowly and in a sparse region, the neighborhood is defined more widely [7]. By examining Figure 4, if we determine the value of *Eps* to be the distance to the 5^{th} neighbor for the densest point in each cluster, we will find two different values for the *Eps* parameter, the first value will be for the black point in cluster b and the other will be for the black point in cluster a. The exact two input parameters for the proposed algorithm are *Minpts* and *Maxpts*; *Minpts* < *Maxpts* < 20. These two parameters determine the minimum and maximum density for core points respectively. *Maxpts* also determines the *Eps* for each cluster according to the highest local density of its starting point. The *Maxpts*-distance is the same as *k*-distance or the distance to the *k*-nearest neighbor.

The algorithm finds the *k*-nearest neighbors for each point *p*, and keeps them in ascending order according to their distances to *p*. (i.e. $N_k(p)=\{q \in D,\ d(p,q_{i-1}) \leq d(p,q_i),\ i=1,\ldots,k\}$), for each neighbor q_i of *p* its distance to *p* and its input order in the input dataset are kept in one vector. The algorithm computes the density on each point *p* according to (2), and arranges the points in dataset in descending order according to the local density of each point using the quick sort. So the first point will be the densest point and the last will be the sparsest one. The algorithm initializes all points as unclassified (ClusId = -1), and sets value of *Minpts* to 4. The user inputs *Maxpts* that will be used to determine *Eps* parameter. Starting from the unclassified point *p* with the highest local density, the algorithm determines the *Eps* as the distance to *Maxpts*-neighbor for this point *p*.

The algorithm expands the cluster around the point *p* starting from the nearest directly density-reachable point *q*, if *q* is a core point, then its unclassified neighbors at *Eps* will be appended to the seed list, otherwise *q* is border point and no points are directly density-reachable from it. This process is continue until the seed list is empty, and the algorithm starts new cluster with new value for *Eps*, and ignores the clustered points. Because the points are arranged in descending order according to their local density, the denser clusters will be created first and the sparser clusters will be created later. The distance to the k-nearest neighbor in dense region tends to be small while in sparse region it tends to be large. This distance is a very good indicator for a suitable value of *Eps* in each cluster (region). Based on this idea, the proposed algorithm is able to change *Eps* to be suitable for the current cluster. The

experimental results confirm this idea. And this algorithm is able to discover clusters having different densities even if there is no separation among them. The following are the main steps of the proposed algorithm:-

1. Find the *k*-nearest neighbors for each point *p*. (i.e. $N_k(p)$) and keep them in ascending order from *p*, in this step *k*=20.
2. Set local density value for each point *p* as $DEN(p,y_1,\ldots,y_k)$, in this step *k*=10.
3. Rearrange the data points in descending order according to their local densities.
4. *ClusId*=1.
5. Starting from the first unclassified point *p* in the sorted data do the following:-
   a. *Eps* = distance to *maxpts*-nieghbor for the point *p*.
   b. Assign the point *p* to the current cluster (*ClusId*).
   c. Append its unclassified neighbor q_i, wrt. *Eps* and *Minpts* to the seed list SL_p in ascending order of their distance to *p*, Continue expanding current cluster until no point can be assigned to it.
6. *ClusId* = *ClusId* +1.
7. Assign the next unclassified point to the current cluster and go to step 5 until all points are classified.

Step 1 of the proposed algorithm imposes an ordering on the *k*-nearest neighbors, and these neighbors will be appended to the seed list SL_p in the same order as in step 1. Step 3 makes the creation of clusters start from the densest cluster and ended with the sparsest one, at the same time allows variance in density within a cluster according the deference between *Maxpts* and *Minpts*. And *Maxpts* is the only input parameter the user set to the algorithm. Step 5.c appends the points in ascending order to the seed list to impose growing of cluster in contiguous regions. Since the density of every point in the cluster compared with the density of the starting point of it, the algorithm may discover small high density clusters. The algorithm discards the small clusters as outlier.

4.1 Time complexity

The most time consuming part of the algorithm is to find the k-nearest neighbors. The neighborhood size (*k*=20) is very small compared to the size of the dataset. So, the different tests performed on the neighborhood to arrange them will not consume much time. While expanding a cluster the list of newly contributed seeds by each object of the cluster are already sorted before. For all objects only a small fraction of the neighbors become new seeds, whereas some points contribute no new seeds at all. The time required for a neighborhood query is O(*log n*) by using a spatial access method such as R*-tree [2]. Neighborhood query is performed for each of the *n* points in the dataset. Also we arrange the points according to their local densities using quick sort, which requires O(*n log n*). If we don't arrange the points, we must search for the densest point in each new cluster creation, this process requires O(*nm*); where *m* is the number of obtained clusters and *n* is the size of the input dataset. So the run time complexity is O(*n log n*) which is the same as that of DBSCAN.

5 Experimental Results

In this section we evaluate the performance of the proposed algorithm. We implemented this algorithm in C++. We have used many synthetic datasets to test the proposed algorithm. We experimented with eight different data sets containing points in two dimensions whose geometric shapes are shown in Fig. 5. The first dataset has six clusters of different sizes, shapes, and orientation, as well as random noise and special artifacts such as streaks running across clusters.

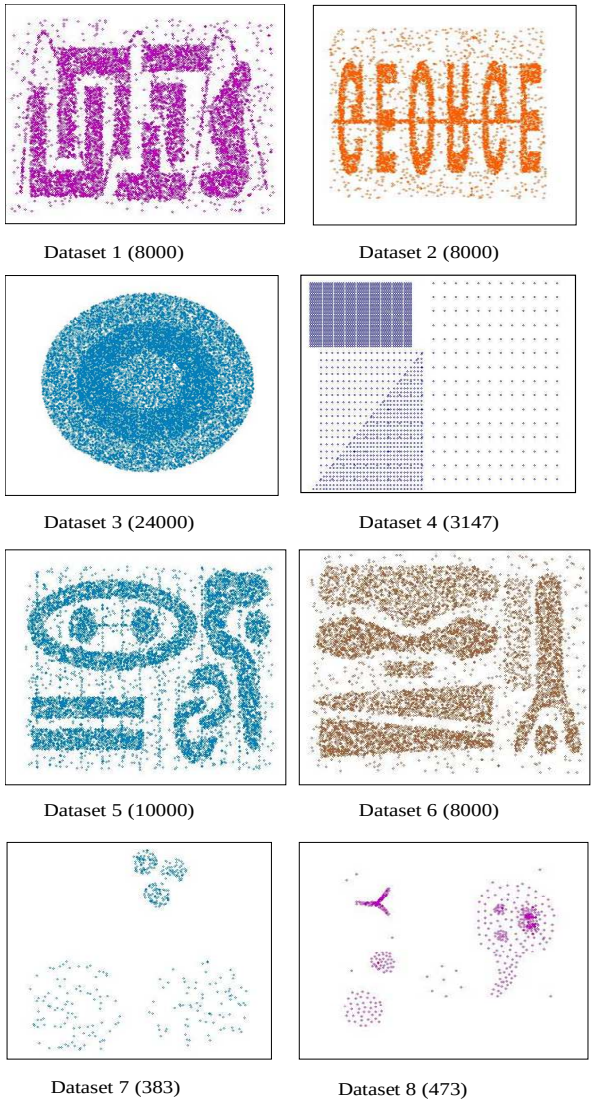

Fig. 5. Datasets used to evaluate the algorithm and their sizes

The second dataset has six clusters of different shapes. Moreover, it also contains random noise and special artifacts, such as a collection of points forming horizontal streak. The

third dataset has three nested circular clusters of different sizes, and densities. A particularly challenging feature of this dataset is that clusters are very close to each other and they have different densities and there is no separation between them. The fourth dataset has four clusters of different shapes, sizes, and densities. There are no sparse regions among clusters. The fifth and sixth data set contain clusters of different shapes and densities which are very close to each other. The seventh data set contains five clusters, three of them have high density and are very close, which motivate DBSCAN to merge them in single cluster to detect the other two, or to detect these three clusters and discard the two large as noise. Finally, in eighth data set that contains clusters with varying densities in addition to three dense clusters involved in large intermediate dense cluster. The size of these data sets ranges from 383 to 24000 points, and their exact sizes are indicated in Fig.5.

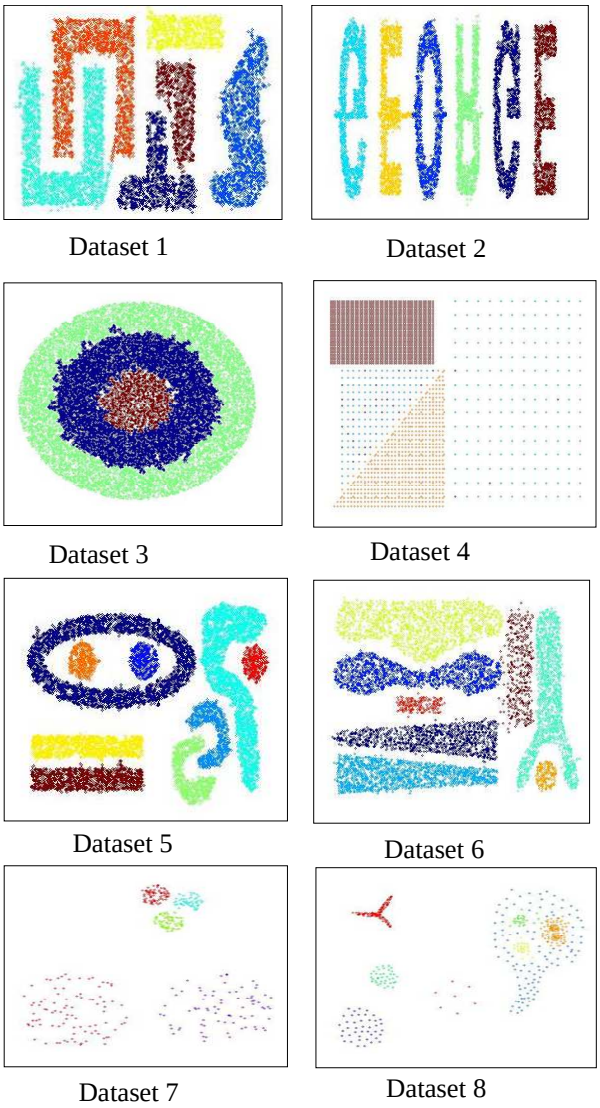

Fig. 6. The results from the proposed algorithm

The clusters resulting from applying the proposed algorithm on these eight datasets in Fig.5 are shown in Fig.6. Different colors are used to indicate the clusters. The very small clusters are discarded as noise. It can be seen from Fig.6 that the triangular and rectangular clusters are discovered and extracted based on differences in densities although they are not separated by sparse regions. This is will be clear when we compare the result of dataset 4 in Fig.6 with that in Fig.2 that result from the DBSCAN algorithm.

Table 1 the different values for Eps in each dataset according to the Maxpts-distance of the starting point in each cluster

	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5	Dataset6	Dataset7	Dataset8
Eps1	5.233	3.045	4.243	0.224	7.006	9.191	0.672	1.374
Eps2	6.666	4.111	7.280	0.424	6.248	7.189	0.783	1.772
Eps3	7.169	3.488	7.071	0.539	6.805	7.580	0.763	3.100
Eps4	5.233	4.250		1.170	6.765	8.977	2.785	3.236
Eps5	5.735	4.226			7.689	8.474	2.911	4.955
Eps6	6.052	3.851			5.918	8.696		6.576
Eps7					6.334	9.610		6.519
Eps8					8.501	16.812		18.355
Eps9					7.976			

From Fig. 6, the proposed algorithm discovers the correct clusters in datasets 3 and 4, where there is no separation between the clusters which have varying densities. Also it discovers the right clusters in datasets 7 and 8. We concentrate on these four datasets because they contain

clusters with varying densities. Table 1 shows the exact value for the *Eps* in each cluster according to the *Maxpts*-distance of the starting point in each one. We can't determine these accurate values from *k*-dist plot as stated in VDBSCAN [8], because this method depends on seeing several smooth curves connected by greatly variational ones. The following Fig. 7 presents the 4-dist plot for some datasets that have clusters with varying densities. Examining this Figure, we should select values that cluster dataset 3 into more than three clusters. For dataset 7 it is very difficult to determine from the *k*-dist plot those values that cluster it correctly, and this is the same situation for dataset 8.

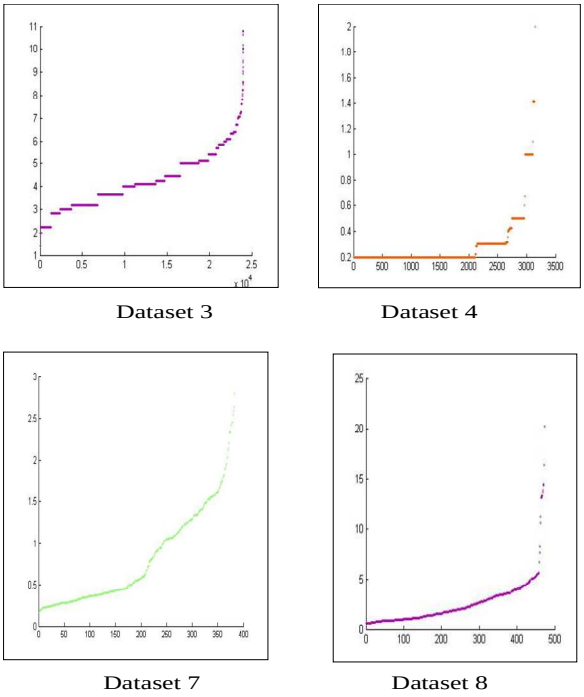

Fig. 7. The 4-dist plot

6 Conclusion

In this paper, we have introduced a competent idea to improve the results of DBSCAN algorithm by detecting clusters with varying densities without requiring any separation between clusters. The *Minpts* is fixed to 4 as in traditional DBSCAN. The proposed algorithm requires only one input parameter (*Maxpts*) which specifies the maximum number of points around a core point; in other word, it specifies maximum level of density for a core point within any cluster. As the value of *Maxpts* decrease the number of discovered cluster increase and the points within a cluster are more similar to each other, and the vice versa. The difference between *Maxpts* and *Minpts* represents the allowed difference in density inside the cluster. The time complexity of the algorithm remains O(*n* log *n*). The experimental results are the evidence on the efficiency of the proposed algorithm.

7 References

[1] M. Ankerst, M. Breunig, H.P. Kriegel, and J. Sandler. "OPTICS: Ordering Points to to Identify the Clustering Structure"; proceedings of the Int. Conf. on Management of Data (SIGMOD'99), pp. 49-60, 1999.

[2] N. Beckmann, H.P. Kriegel, and R. Schneider. "The R*-tree an Efficient and Robust Access Method for Points and Rectangles"; proc. ACM SIGMOD, Utlantic City, USA, pp. 322-331, May 1990.

[3] B. Borah, and D.K. Bhattacharyya. "DDSC: A Density Differentiated Spatial Clustering Technique"; Journal of Computers, Vol. 3, No. 2, pp. 72-79, February 2008.

[4] B. Borah, and D.K. Bhattacharyya. "An Improved Sampling-Based DBSCAN for Large Spatial Databases"; proceedings of International Conference on Intelligent Sensing and Information, pp. 92-96, 2004.

[5] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. "A density based algorithm for discovering clusters in large spatial data sets with noise"; in 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231, 1996.

[6] A. Hinneburg and D. Keim. "An efficient approach to clustering in large multimedia data sets with noise"; in 4th International Conference on Knowledge Discovery and Data Mining, pp. 58–65, 1998.

[7] G. Karypis, E. H. Han, and V. Kumar. "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling"; Computer, 32, pp. 68-75, 1999.

[8] P. Liu, D. Zhou, and N. Wu. "VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise"; International Conference on Service Systems and Service Management (ICSSSM), pp.1-4, 2007.

[9] C-F. Tsai, and C-W. Liu. "KIDBSCAN: A New Efficient Data Clustering Algorithm"; ICAISC, PP. 702-711, 2006.

View publication stats