World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:9, 2008

DCBOR: A Density Clustering Based on Outlier Removal

A. M. Fahim, G. Saake, A. M. Salem, F. A. Torkey and M. A. Ramadan

Abstract—Data clustering is an important data exploration technique with many applications in data mining. We present an enhanced version of the well known single link clustering algorithm. We will refer to this algorithm as DCBOR. The proposed algorithm alleviates the chain effect by removing the outliers from the given dataset. So this algorithm provides outlier detection and data clustering simultaneously. This algorithm does not need to update the distance matrix, since the algorithm depends on merging the most k-nearest objects in one step and the cluster continues grow as long as possible under specified condition. So the algorithm consists of two phases; at the first phase, it removes the outliers from the input dataset. At the second phase, it performs the clustering process. This algorithm discovers clusters of different shapes, sizes, densities and requires only one input parameter; this parameter represents a threshold for outlier points. The value of the input parameter is ranging from 0 to 1. The algorithm supports the user in determining an appropriate value for it. We have tested this algorithm on different datasets contain outlier and connecting clusters by chain of density points, and the algorithm discovers the correct clusters. The results of our experiments demonstrate the effectiveness and the efficiency of DCBOR.

Keywords—Data Clustering, Clustering Algorithms, Handling Noise, Arbitrary Shape of Clusters.

I. Introduction

NUMEROUS applications require the management of spatial data, i.e. data related to space. Spatial Database Systems(SDBS) are database systems for the management of spatial data. Increasingly large amounts of data are obtained from satellite images, X-ray crystallography or other automatic equipment. Therefore, automated knowledge discovery becomes more and more important in spatial databases. Clustering is an important task of knowledge discovery in databases. Clustering is the organization of a database D into homogeneous and separated groups with respect to a distance or a similarity measure. Its objective is to assign the similar objects to the same cluster, and dissimilar objects to different clusters. Clustering methods basically classified into two types; partitional and hierarchical methods [9]. Partitioning algorithms construct a partition of a database D of n objects into a set of k clusters; k is an input parameter for these algorithms. A partitioning algorithm typically starts with an initial partition of D and then uses an iterative control strategy to optimize an objective function. The square error criterion, defined below in (1), is the most commonly used (m_i is the mean of cluster C_i).

$$\sum_{i=1}^{k} \sum_{p \in C_i} \parallel p - m_i \parallel^2 \qquad (1)$$

The square-error is a good measure of the within cluster variation across all the partitions. The objective is to find k partitions that minimize the square error. Thus, square error clustering tries to make the k clusters as compact and separated as possible, and works well when clusters are compact clouds that are rather well separated from one another. Each cluster is represented by the gravity center of the cluster ($k - means\ algorithm$) or by one of the objects of the cluster located near its center ($k-medoid\ algorithms$). Consequently, partitioning algorithms use a two-step procedure. First, determine k representatives minimizing the objective function. Second, assign each object to the cluster with its representative "closest" to the considered object.

Hierarchical algorithms create a hierarchical decomposition of D. The hierarchical decomposition is represented by a dendrogram; a tree that iteratively splits D into smaller subsets until each subset consists of only one object. In such a hierarchy, each node of the tree represents a cluster of D. The dendrogram can either be created from the leaves up to the root (agglomerative approach) or from the root down to the leaves (divisive approach) by merging or dividing clusters at each step. Agglomerative hierarchical clustering (AHC) is more stable but its computation and computer memory used are very expensive, and thus, it is not feasible for a large data set. Moreover, for examples, the single link approaches are very susceptible to noise and differences in density. While group average and complete link are not as susceptible to noise, they have trouble with varying densities and cannot handle clusters of different shapes and sizes [4].

Besides the partitional and hierarchical approaches, density based clustering methods such as DenClue [7] and DBSCAN [3] form a third clustering type. These are often used in data mining for knowledge discovery. Density-based clustering uses a local cluster criterion, in which clusters are defined as regions in the data space where the objects are dense, and remain, separated from one another by low-density regions. Density-based clustering has advantages over k-clustering and AHC in discovering clusters of arbitrary shapes and sizes. However, it was shown that current density based clustering works well only on a simple data set where cluster densities

A.M. Fahim is with the faculty of information, Otto-von-Guericke University, Magdeburg,Germany, email: ahmed.fahim@iti.cs.uni-magdeburg.de.
G. Saake is with the faculty of information, Otto-von-Guericke University, Magdeburg,Germany, email: saake@iti.cs.uni-magdeburg.de.
A. M. Salem is with the faculty of Computers and Information, Ain Shams University, Cairo, Egypt, email:absalem@asunet.shams.edu.eg.
F. A. Torkey is with the Kafer el Shiekh University, Kafer el Sheikh, Egypt,email:fatorkey@Yahoo.com.
M. A. Ramadan is with the Faculty of Science, Minufiya University, Shbien el koum, Egypt,email: mramadan@mailer.eun.eg.

are similar [1]. Density based clustering is important for knowledge discovery in databases. Its practical application areas include biomedical image segmentation [2], molecular biology and geospatial data clustering [13], and earth science tasks [3].

In this paper we propose an algorithm that merges between hierarchical, partional and density based methods. This algorithm builds the k nearest neighbors graph, estimates the density for each point from this graph, and removes the low density points according to the input parameter ranging from 0 to 1. This parameter represents a threshold for outliers in data and is the only parameter required by our algorithm. After outlier removal the algorithm start the clustering process, starting from the most density point, applying the idea of the single link [14] with some modification to overcome the computational complexity, and depending on the density estimation, the algorithm eliminates the chain effect problem of the single link algorithm. Also this algorithm does not require O(n^2) since we use the idea of canopy to construct the k nearest neighbors graph. We do not use a distance matrix. The running time of this algorithm is O(n). Also we can cluster all points in the data set if we want by assigning each outlier point to the nearest clustered point starting with the most density outlier point. Our experimental results demonstrate the effectiveness and efficiency of the proposed algorithm. This paper is organized as follows, in section 2 we review some of related works. In section 3, some details of the proposed algorithm are presented. We present some experimental evaluation of the proposed algorithm in section 4 and conclude with section 5.

II. Related Work

Existing clustering algorithms can be broadly classified into hierarchical and partitioning clustering algorithms [8]. The Single-Link method is a commonly used hierarchical clustering method [14]. Starting with the clusters obtained by placing every object in a unique cluster, in every step the two closest clusters in the current clustering are merged until all objects are in one cluster. This algorithm is time consuming, and is very sensitive to outliers. BIRCH method [15] can be classified as a hierarchical method. BIRCH constructs a CF-tree which is a hierarchical data structure designed for a multiphase clustering method. First, the data set is scanned to build an initial in memory CF-tree which can be seen as a multi-level compression of the data that tries to preserve the inherent clustering structure of the data. Second, an arbitrary clustering algorithm can be used to cluster the leaf nodes of the CF-tree. This algorithm uses the idea of diameter and the cluster center to cluster the leaf nodes of the CF-tree. And this leads the algorithm to be more similar to partitional algorithms. Another hierarchical algorithm CURE has been proposed in [5]. This algorithm stops the creation of a cluster hierarchy if a level consists of k clusters where k is one of several input parameters. It utilizes multiple representative points to evaluate the distance between clusters, thereby adjusting well to arbitrary shaped clusters and avoiding the single-link effect. This results in a very good clustering quality. But this algorithm has several parameters. The parameter setting does have a profound influence on the result.

Partitioning algorithms typically represent clusters by a prototype. Points are assigned to the cluster represented by the most similar prototype. These clustering algorithms are effective in determining a good clustering if the clusters are of convex shape, similar size and density, and if their number k can be reasonably estimated. Depending on the kind of prototypes, one can distinguish k-means, k-modes and k-medoid algorithms. The algorithm CLARANS introduced in [12] is an improved k-medoid type algorithm restricting the huge search space by using two additional user-supplied parameters.

Density-based approaches apply a local cluster criterion and are very popular for the purpose of database mining. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise). In [3] a density-based clustering method is presented. The basic idea for the algorithm DBSCAN is that for each point of a cluster the neighborhood of a given radius (e) has to contain at least a minimum number of points (MinPts) where e and MinPts are input parameters. These two parameters are globular for the data set. And it is not easy to determine the best value for e. In [7] the density-based algorithm DenClue is proposed. This algorithm uses a grid but is very efficient because it only keeps information about grid cells that do actually contain data points and manages these cells in a tree-based access structure. This algorithm generalizes some other clustering approaches which, however, results in a large number of input parameters. We propose the DCBOR that uses the density notion and a level of dissimilarity used in single link.

III. The algorithm

In this section we present our proposed algorithm that operates on a sparse graph in which nodes represent data items, and weighted edges represent the distances between the data points. This algorithm finds the clusters in the data set by using a two phase algorithm. During the first phase, it builds the k-nearest neighbors graph and removes the outlier from the data set. During the second phase, it uses the single link with simple modification to discover the genuine clusters. We describe the details in the following subsections.

A. *Outlier removal*

To remove the outlier from the dataset we use the idea of graph, since the graph shows the relationship between the data points. So we build the k-nearest neighbors graph. In this graph every nod is connected to its k nearest neighbors. The edge between two nods represents the distance between them. We compute the influence of each point within its k nearest neighbors using influence function; the influence function describes the impact of a point within its neighbors, we use the Euclidean distance to present the influence function. As the distance between the two points increase the influence of a point within its neighbor decrease. Also we calculate the density for each point in the dataset, note that not all data points contribute the density of a point, so we use this observation to calculate the density for a point x, only k points of the dataset which are close to x actually contribute to the density. All other points may be neglected without making a substantial error. Here, we do not use fixed radius for neighborhood, but we use the k nearest neighbors. Since the distance to the k nearest neighbors is more impressive than EPS neighborhood in the DBSCAN algorithm. And we calculate the density of point as the summation of the influence in its k nearest neighbors which is better than counting the points in EPS neighborhood. This method is more accurate than that of DBSCAN algorithm to determine the core point based on neighborhood radius and the minimum number of points that must be within this radius, we remove the most low density points from the dataset as outliers. To get the k nearest neighbors for each point, we use a data structure called canopy [11], that partition the data space into overlapping subsets based on the average radius of the data space. A data point may be covered by more than one canopy, and every point must be at least in one canopy. For each point, we keep track to its nearest canopy, and use this canopy to answer the query region for this point. After computing the density for all points in the dataset and removing the lowest density points, we are ready to perform clustering based on merging ideas from the slink and DBSCAN algorithm.

1) The influence and density function: The main idea of our algorithm is to remove the lowest density points from the data. We compute the density of point according to the following functions: Influence function represent the impact of point x on point y as the Euclidean distance between them, where $x, y \in R^d$.

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

As the distance between the two points decrease the impact of x on y increase, and vice versa. The density function for a point x is defined as the summation of (influence functions within the k nearest neighbors) distances between the point x and the k nearest neighbors. The density function is defined as

$$DEN(x, y_1, y_2, ..., y_k) = \sum_{i=1}^{k} INF(x, y_i) \tag{3}$$

The definition of density based on the summation of distances of the k nearest neighbors is better than counting the points within the Eps-neighborhood radius. Consider the following example as in Figure1. As we see in Figure 1.a and 1.b both

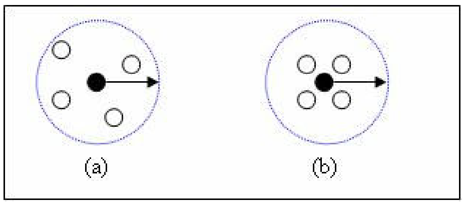

Fig. 1. Density based on distance to k nearest neighbors.

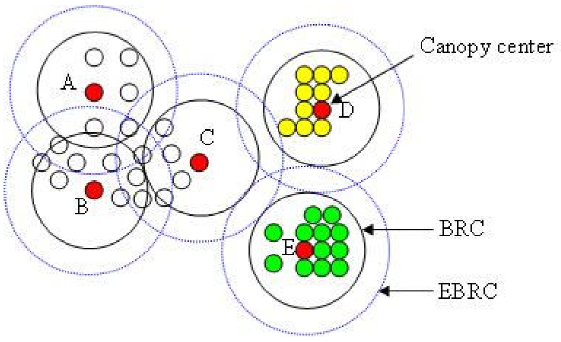

Fig. 2. An example of three data clusters and canopies cover them.

black points has four points in neighbor (the Eps is the same represented by black arrow) but Figure 1.b is more density than Figure 1.a. based on the summation of distances reflects this fact accurately. But based on the Eps-neighborhood radius as in DBSCAN algorithm there is no difference, because each point has four points within its Eps. And we can note that the value of Eps vary according to the density using the k nearest 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 neighbor surrounding it from all directions. So density based on distances summation is better than points numeration within Eps-neighborhood radius.

2) Creating Canopies: Our algorithm based on outlier removal and density calculation, also based on a simple and efficient data structure to find the k nearest neighbors, this data structure is called canopy which is simply a number of overlapped hyper spheres of the same radius cover the data space. This data structure is used to partition the data space like the grid but here there is overlap between cells and the cells are not (hyper) rectangular regions but (hyper) spherical regions. To get the k nearest neighbors for a point x, only we consider the points lie in the same canopy of x. Also a point x may be covered by more than one canopy, so we select the best (nearest) canopy to get them, (i.e. we select the canopy where x is in its core). To create canopies, we find the means of the dataset, then find the average radius of the dataset and divide this value by six. This value referred to it as BRC (Basic Radius of Canopy) and we extend this radius to EBRC which equals 1.5 of BRC. We start from the first point in the dataset as the center of the first canopy, all points that are within the distance BRC are the core of the canopy, and are excluded from being a center for other canopy. But the points that are within the distance between BRC and EBRC are added to the canopy but one or more of them may form center(s) for other canopy (canopies). To answer the query region for a point only one canopy is tested. And we reach this canopy directly, since each point labeled by its canopy number. The circles with dashed (blue) outlines in Figure 2 show an example of overlapping canopies that cover a data set. The red points from A to E represent the centers for canopies.

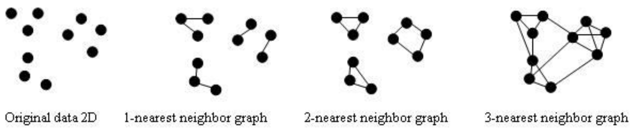

Fig. 3. *k*-nearest graphs from an original data in 2D.

3) The k-nearest neighbor graph: After creating canopies, for each point x we compute its distance to the other points at the canopy, and keep the distances to the k-nearest neighbors. These distances are arranged in ascending order, and we start to build the k-nearest neighbors graph. Each vertex of the k-nearest neighbors graph represents a point, and there exists an edge between two vertices, if a point corresponding to either of the nodes is among the k-most nearest points of the data point corresponding to the other node. Figure 3 illustrates the 1-, 2-, and 3-nearest neighbor graphs of a simple data set. Note that since our algorithm operates on a sparse graph, each cluster is nothing more than a sub-graph of the original sparse graph representation of the data set.

There are several advantages of representing data using a k-nearest neighbors graph G_k. Firstly, data points that are far apart are completely disconnected in the G_k. Secondly, G_k captures 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. Compared to the model defined by DBSCAN [3] in which a global neighborhood density is specified, G_k captures more natural neighborhood. Finally, the density of the region is recorded as the weights of the edges. The edge weights of dense regions in G_k (with edge weights representing distances) tend to be small and the edge weights of sparse regions tend to be large. After creating the k-nearest neighbors graph and computing the density of points using equation (3), the algorithm removes the lowest density points as outlier, the number of discarded points depends on the input parameter ranging from 0 to 1. When we want to cluster all data points we can assign each outlier point to the nearest clustered point, or cluster the outlier themselves. After removing the outliers the algorithm moves to the second phase to cluster the data depending on the idea of single link algorithm [14] and DBSCAN [3].

4) The Threshold of Outlier: The outlier threshold is the only required input parameter for DCBOR. The algorithm supports the user in determining an appropriate value for it. To detect the outlier we assign outlying factor for each data point, this value is based on the local density. The outlying factor for a point x_i is given by the following Equation (4).

$$OF(x_i) = \frac{\sum_{j=1}^{k} d(x_i, y_j)}{max(\sum_{j=1}^{k} d(z, y_j))} \tag{4}$$

$d(x, y)$ is the Euclidean distance between the two points, the numerator represents the local density at the point x_i which was computed by Equation (3), the denominator represents the local density of the lowest density point z in the data set. So the outlying factor for a point x_i is ranging from 0 to 1, as the outlying factor become closer to one as the higher probability for the point to be outlier. The algorithm divides the interval [0,1] into 20 sub-intervals, and determine the count of points in each sub-interval. By examining this information the user can determine appropriate value for the outlier threshold. All points that have outlying factor larger than the input value are discarded as outliers. Outlier detection is an important problem. You can see for example [6].

B. Clustering Stage

As we know that the single link is susceptible to outliers. We overcome this problem by removing the outliers at the first phase. And slink require $O(n^2)$, where n is the number of points in dataset. This high computational complexity makes this algorithm not suitable to large data set. But based on canopy to get the most similar point for each other, and the density notion we solve the problem of complexity and the chain effect. And our proposed algorithm is very efficient for clustering very large datasets. As we know the single link is an agglomerative hierarchical clustering algorithm, and this family of algorithms create a hierarchical dendrogram from the leaf node up to the root, and the input parameter for this algorithm may be the required number of clusters or a threshold parameter used to cut the dendrogram at specified level. And it is not easy to determine the number of cluster, so in this paper we will depends on the threshold. We can deduce a suitable value for the threshold from the graph after removing the outliers. We search for the maximum distance between a point and its first nearest neighbor. This distance is the ideal distance for threshold (level of dissimilarity between clusters) according to the main idea of the single link algorithm. After that we perform the clustering process as in the following steps:

1) Search for the most density point to be the starting point for the current cluster
2) Expand the current cluster according to the threshold until no point can be added to it.
3) Start new cluster and repeat steps 1 and 2 until all points are clustered.
4) (Optional) cluster outliers by assigning each point to the nearest cluster.

Step 2 of clustering process is a middle ground between the single link and DBSCAN, since in single link two points are merged in each step, but here all points at distance from the current point satisfy the threshold are assigned to the current cluster. On the other hand we do not consider how many points satisfy the threshold as in DBSCAN. So our proposed algorithm can discover clusters having different shapes, sizes and densities. Our algorithm is presented in Figure 4.

1) Time Complexity: As we have seen before, the algorithm build canopies that cover all data space, say m canopies are required, suppose the data points are uniformly distributed, so each canopy will contain $\frac{n}{m}$ points; where n is the size of the input dataset. The canopies creation process require $O(mn)$, m is very small compared by n. To get the k-nearest neighbors

```
1. M = the mean of the data
2. Avr = the average radius of the data
3. Create canopies with the following radius
4.     BRC= Avr/6
5.     EBRC= BRC*1.5
6. For each point return the $k$-nearest neighbor arranged in ascending order
7. Compute the density of each point
8. Remove outliers from the data according to the input parameter value.
9. Threshold = max distance to the first nearest neighbor
10. Get the most density unclassified point P
11. ClusterId=1
12. Expand_Cluster( Dataset,ClusterId, P, Threshold)
13. Get the most density unclassified point P
14. If P exist Then
15. ClusterId= ClusterId +1
16. Goto step 12
17. endif

Expand_Cluster( Dataset,ClusterId, P, Threshold)
{
   Seeds:=Dataset.RegionQuery(P,Threshold)
  If Seeds.Size>0 then
    Dataset.ChangeClusterId(Seeds, ClusterId)
    Seeds.Delete(P)
    While Seeds <> Empty DO
       CurrentP = Seeds.First()
        Result = Dataset.RegionQuery(CurrentP, Threshold)
        If Result.size >0 then
            For i= 1 to Result.Size DO
              {
                ResultP = Result.Get(i)
                If ResultP unclassified then
                   Seeds.Append(ResultP)
                    Dataset.changeClusterId(ResultP,ClusterId)
                 Endif
              }
            Endif
           Seeds.Delete(CurrentP)
     End While
  Endif
}
```

Fig. 4. DCBOR algorithm.

for a point, this requires distance calculation between this point and the other points shared the same canopy which present small portion of the data set, this process requires $O(h)$; where $h = \frac{n}{m}$, h is very small compared with the dataset size. For each point we keep the distances to the 10^{th} nearest neighbors and only we use the 5^{th} nearest neighbors to compute the density of it. To remove the outliers point this requires $O(n)$. To get the highest density point we check the density for each point in $O(n)$. To perform clustering process this requires $O(n)$. And the overall complexity of the algorithm is $O(nm + nh + n + n)$.

IV. Experimental Results

In this section we evaluate the performance of DCBOR. We implemented this algorithm in C++. We have used the synthetic datasets that used in [10] to test our proposed algorithm. We experimented with four different data sets containing points in two dimensions whose geometric shape are shown in Figure 5. The first data set, DS1, has six clusters of different size, shape, and orientation, as well as random noise points and special artifacts such as streaks running across clusters. The second data set, DS2, has nine clusters of different shape, size, and orientation, some of which are inside the space enclosed by other clusters. Moreover, DS2 also contains random noise and special artifacts, such as a collection of points forming vertical streaks. The third data set, DS3, has eight clusters of different shape, size, density, and orientation, as well as random noise. A particularly challenging feature of this data set is that clusters are very close to each other and they have different densities. Finally, the fourth data set, DS4, has six clusters of different shape connected by density chain of points. The size of these data sets ranges from 8000 to 10000 points, and their exact size is indicated in Figure 5

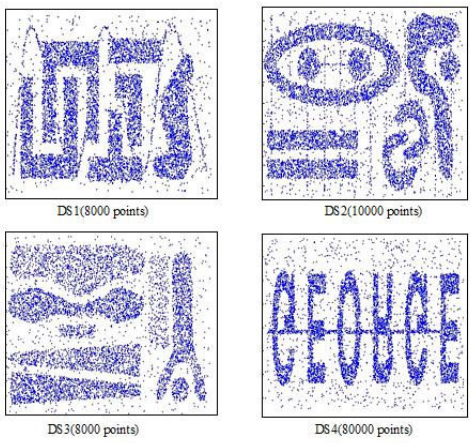

Fig. 5. The data sets used in our experiments.

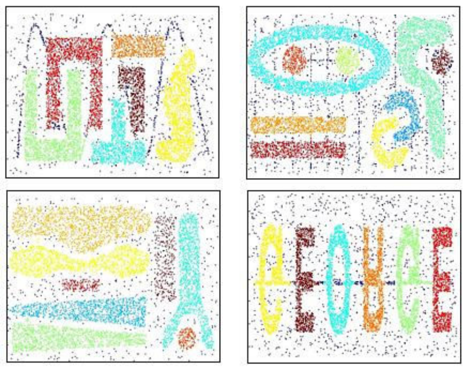

Fig. 6. The data sets after determining the outliers.

At the first stage of DCBOR, it determines the outliers from the dataset. Figure 6 shows the datasets from Figure 5 after determining the outliers (the gray points). Figure 7 shows the clusters discovered in each data set.

The following table 1 presents the execution time of DCBOR with and without the canopy data structure.

TABLE I
Execution time of DCBOR algorithm

Dataset	Canopy count	Time in sec	Time in sec without canopy
DS1	148	1	18
DS2	151	2	28
DS3	158	1	18
DS4	79	2	18

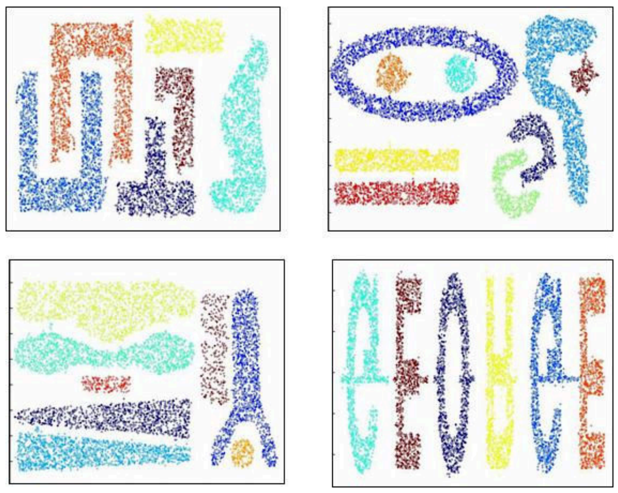

Fig. 7. The final results of DCBOR algorithm.

V. Conclusion

In this paper we have presented a clustering algorithm, that introduces a middle ground between the hierarchical clustering and density clustering, this algorithm called DCBOR. It removes the outliers from the data set based on the density notion, and applies the clustering process based on the idea of single link and growing cluster in all possible direction as in density clustering algorithm like DBSCAN. This algorithm discovers clusters with different shapes, sizes and densities. Our experimental results demonstrated the efficiency of this algorithm from two direction; the first from the quality of clusters, the second from the high speed of producing (discovering) the clusters.

Acknowledgment

I would like to thank Dr. B. Borah who provided me with the datasets to test my algorithm, also great thanks for all my supervisors.

References

[1] M. Ankerst , M. M. Breunig and H-P. Kriegel , "OPTICS: Ordering Points to Identify the Clustering Structure". in Proc. ACM SIGMOD, 1999, pp. 49-60.
[2] M. Emre Celebi , Y. Alp Aslandogan , and P. R. Bergstresser "Mining biomedical images with density-based clustering." In ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing, volume I, pages 163-168, Washington, DC, USA, 2005. IEEE Computer Society.
[3] M. Ester , H.-P. Kriegel , J. Sander and X. Xu , "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". in Proc. Knowledge Discovery and Data Mining, 1996, pp. 226-231.
[4] L. Ertoz , M. Steinbach and V. Kumar , "A new shared nearest neighbor clustering algorithm and its applications", AHPCRC, Tech. Rep. 134, 2002.
[5] S. Guha, R. Rastogi , and K. Shim, "CURE: An Efficient Clustering Algorithms for Large Databases." Proc. ACM SIGMOD Int. Conf. on Management of Data. Seattle, WA, 1998, pp. 73-84.
[6] V. Hautamaeki , S. Cherednichenko , I. Kaerkkaeinen , T. Kinnunen , and P. Fraenti "Improving K-Means by Outlier Removal", LNCS Springer Berlin / Heidelberg, may 2005, pp. 978-987.
[7] A. Hinneburg and D. A. Keim , "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," in Proc. Knowledge Discovery and Data Mining, 1998, pp. 58-65.
[8] A. K. Jain and, R. C.Dubes "Algorithms for Clustering Data." Prentice Hall, 1988.
[9] A. K. Jain , M. N. Murty , and P. J. Flynn , "Data Clustering: A Review", ACM Computing Surveys, vol. 31, no 3, pp. 264-323, Sep. 1999.
[10] G. Karypis , E. H. Han and V. Kumar, "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling." Computer,32, pp. 68-75, 1999.
[11] A. McCallum , K. Nigam and L. H. Ungar "Efficient Clustering of HighDimensional Data Sets with Application to Reference Matching." In Proceedings of KDD'2000. pp.169-178.
[12] R. T. Ng and J. Han "Efficient and Effective Clustering Methods for Spatial Data Mining". Proc. 20^{th} Int. Conf. on Very Large Data Bases. Morgan Kaufmann Publishers, San Francisco, CA, 1994, p. 144-155.
[13] J. Sander , M. Ester , H-P. Kriegel , and X. Xu "Density-based clustering in spatial databases: The algorithm gdbscan and its applications." Data Mining and Knowledge Discovery, 2(2):169-194, 1998.
[14] R. Sibson, SLINK: an optimally efficient algorithm for the single-link cluster method. The Comp. Journal, 1973, 16(1), p. 30-34.
[15] T. Zhang , R. Ramakrishnan and M. Linvy BIRCH: An Efficient Data Clustering Method for Very Large Databases. Proc. ACM SIGMOD Int. Conf. on Management of Data. ACM Press, New York, 1996, p. 103-114.