A Dispersive Degree Based Clustering Algorithm Combined with Classification

Xianchao Zhang Shimin Shan Zhihang Yu He Jiang

School of Software, Dalian University of Technology, Dalian 116620, P. R. China

Abstract

The various-density problem has become one of the focuses in density based clustering research. A novel dispersive degree based algorithm combined with classification, called CDDC, is presented in this paper to remove the hurdle. In CDDC, a sequence is established for depicting the data distribution, discriminating cores and classifying edges. Clusters are discovered by utilizing the revealed information. Several experiments are performed and the results suggest that CDDC is effective in handling the various-density problem and is more efficient than the well-known algorithms such as DBSCAN, OPTICS and KNNCLUST.

Keywords: Clustering analysis, Various-density, Dispersive degree, Data mining

1. Introduction

One of the primary data mining tasks is clustering analysis. The goal of a clustering algorithm is to group the objects of a database into a set of meaningful subclasses (clusters). A clustering algorithm can be used either as a stand-alone tool to get insight into the distribution of a data set, or as a preprocessing step for other algorithms which operate on the detected clusters. Applications of clustering are, for instance, computational analysis, pattern recognition, medical diagnosis, web retrieval. For each of these applications specialties are requested [1], such as scalability, ability to deal with different types of attributes, ability to handle dynamic data, discovery of clusters with arbitrary shape, minimal requirements for domain knowledge, able to deal with noise and outliers, insensitive to order of input records, high dimensionality, incorporation of user-specified constraints, interpretability and usability, etc, which make the research of clustering algorithm to be challenging and magnetic.

Many clustering algorithms were raised heretofore, among which are partition method, hierarchical method, density-based method, grid-based method, model-based method, etc. Density-based clustering algorithm is an important embranchment of cluster analysis with the advantages of capability of discovering clusters with arbitrary shape and insensitivity to noise data. The basic idea is using a local cluster criterion, in which clusters are defined as regions in the data space where the objects are dense, and clusters are separated from one another by low-density regions. There are several concernful density-based algorithms: DBSCAN, OPTICS, and KNNCLUST [2]-[4].

The key idea of DBSCAN is that for each object of a cluster, its neighborhood at a given radius must contain at least a minimum number of objects (MinPts), then cluster the density-connected objects. It is significantly effective in discovering clusters of arbitrary shape and can deal with noise. But the two fixed input parameters, ε and MinPts, weaken its ability of dealing with data sets with various densities. Instead of producing clusters explicitly, OPTICS creates an augmented ordering to represent the density-based clustering structure of a data set. It is equivalent to cluster a broad range of parameter settings, which overcomes the drawbacks of DBSCAN in a way. But the result of OPTICS is a tree of clustering structure, on which every node is a cluster. The sons of a node are subclusters of their parent node. In other words, OPTICS can't attach an object exactly to a cluster. KNNCLUST method combined nonparametric k-nearest-neighbor and kernel density estimation, relies Bayes' decision rule to cluster objects. Although this technique makes it possible to model clusters of different densities in data sets and identify the number of clusters automatically, it's a "hard" algorithm which assigns object to one and only one cluster. It means that KNNCLUST can not identify noise. Furthermore, it is less suited for finding clusters with strange shapes.

Almost all of the well-known clustering algorithms require input parameters which are hard to determine but have a significant influence on the clustering results. Furthermore, for many real data sets there is not a global parameter setting which describes the intrinsic clustering structure accurately. For density-based clustering, this problem embodies as

that the intrinsic cluster structure of many real-data sets with various densities can not be characterized by global density parameters. Most of the widespread algorithms are not effective on handling practical datasets because of incapability of tackling the various densities. The various density problem has become one of the focuses for the density-based clustering research. In this paper, a new dispersive degree based algorithm combined with classification, called CDDC (Clustering using Dispersive Degree and Classification), is developed to tackle these situations. Enlightening by OPTICS and KNNCLUST, CDDC computes dispersive degree with KNN distance first, and produces an order(sequence of scanning) to depict the clustering information, then partitions the order into core and edge points, assigns the edge points to clusters applying KNN-kernel density estimation in the end.

The rest of the paper is organized as follows. The new dispersive degree based algorithm combined with classification, CDDC, is given in Section 2. In Section 3, its performance is evaluated by simulate experiments and compared to the results from DBSCAN, OPTICS and KNNCLUST. Finally, the work is summarized in Section 5.

In order to facilitate the description of CDDC, simple 2-dimension points are used to represent objects in data space in this paper.

2. CDDC clustering algorithm

2.1. Density-based clustering

Density-based clustering algorithms are built on definition of density and cluster which are given as follows.

Definition 1. (Neighborhood of a point) Let D be a set of points. The neighborhood of a point p in D wrt. a given radius ε, denoted by Neighbors(p, ε), is defined by Neighbors(p , ε) = {q ∈ D | dist(p,q) ⩽ε}, where dist(p,q) is the distance between p and q.

Definition 2. (Density of a point) The density of a point p wrt. ε in D , denoted by Density(p, ε), is defined by Density(p, ε) = |Neighbors(p, ε)|.

Definition 3. (Density-based cluster) A cluster C is a non-empty subset of D, in which the points have higher density and uniformity of density distribution.

Definition 4. (Noise) Let $C_1,C_2,\ldots,C_n$ be n clusters in D, i=1,2,…,n. noise is defined by Noise={p ∈ D | ∀ i: p ∉ C_i}.

It can be deduced that points in clusters should satisfy the following lemmas:

Lemma 1. Points in clusters have higher density than noise points. Density(p , ε)> Density(q, ε) ,where p ∈ C_i, q ∈ {noises}.

Lemma 2. The density difference between two points in same cluster is more distinct than that between one in cluster and the other in noise. |Density(o, ε)-Density(p, ε)| < |Density(o, ε)-Density(q, ε)|, where o, p ∈ C_i, q ∈ noise.

Definition 5. (outlier degree) The outlier degree of point p wrt. ε ,denoted by ODegree(p, ε), is defined by

$$ODegree(p,\varepsilon)=\frac{1}{\left|Neighbors(p,\varepsilon)\right|}\sum_{q_i}\frac{Density(p,\varepsilon)}{Density(q_i,\varepsilon)}$$

where ∀ i :q_i ∈ Neighbor(p, ε).

Depending on lemma 2, it is obvious that points in clusters have ODegree trending to 1.If the ODegree of a point is farther from 1, its probability to be a noise is higher.

2.2. Frame of CDDC algorithm

CDDC algorithm has four main steps:

- Compute dispersive degree
- Scan data
- Divide scan order
- Classify edge points

These steps are introduced in the following 4 subsections, respectively.

2.3. Computing dispersive degree

Definition 6. (KNN distance) Let p be a point in data set D, p's KNN distance is the distance from p to its *k*th nearest neighbor, denoted by KNN-dist(p,k) .

For a given k, the density of p is larger , its KNN-dist is smaller. So KNN-dist represents density of p from another aspect. This idea of KNN-density is first introduced by Loftsgaarden and Quesenberry [5], which is redefined in this paper as follows:

Definition 7. (KNN density) KNN density of p in D ,denoted by KNN-density , is the reciprocal of p's KNN-dist.

Outlier degree can be redefined with KNN-density as:

Definition 8. (K outlier degree) The K outlier degree of p, denoted by ODegree(p,k), is defined by

$$ODegree(p,k)=\frac{1}{\left|Neighbors(p,\varepsilon')\right|-1}\sum_{q_i}\frac{Density(p,k)}{Density(q_i,k)}$$

where ε' = KNN-dist(p,k), ∀ i : q_i ∈ Neighbor(p, ε'), $q_i \neq p$.

Definition 9. (Dispersive degree) The dispersive degree of p in data set D wrt. k, denoted by DDegree(p , k) , is defined by DDegree(p , k) = max(ODegree(p , k) , (ODegree(p , k))$^{-1}$)-1.

It is known from the definition above that the smaller the DDegree the more likely it is in a cluster; oppositely it is noise.

In this step, CDDC finds k nearest neighbors of each point in data set, records the distances to every neighbor, and computes DDegree with KNN-dist of the points. This procedure is illustrated by Figure 1.

```
ComputeDDegree(DataSet, k)
   FORALL Point FROM DataSet DO
      Point.FindNeighbors(DataSet,k);
      Point.knn_dist = Point.Neighbors[k].dist;
   FORALL Point FROM DataSet DO
      FORALL Neighbor FROM Point.Neighbors[]
      DO
          Point.ODegree = Point.ODegree
                          +Neighbor.knn_dist;
      Point.ODegree = Point.ODegree
                /(Point.knn_dist*Point.Neighbors[].size)
      Point.DDegree = Max(Point.ODegree,
                      1/ Point.ODegree)-1
END
```

Fig. 1: Procedure ComputeDDegree.

The array Point.Neighbors[] preserves all the points except Point itself in the neighborhood of Point wrt. Point.knn_dist and their distances to Point. Because there may be more than one point have the same distance equal to Point.knn_dist from Point, the size of array may be larger than k-1.

2.4. Scanning data

CDDC creates an order of a data set, additionally sorting the dispersive degree for each point. It can be seen that this order is sufficient to extract the density-based clusters. Figure 2 depicts the pseudo-code for the procedure ScanData.

Function GetNextStart() gets the point with the smallest DDegree which has not been scanned as the start of a new scan process, and inserts it into SeedsQueue, in which points are ascending sorted according to DDegree. Function GetFirst() fetches the first element of SeedsQueue, and adds it to ScanOrder. Then all neighbors of the first point were checked. If the neighbor hasn't been scanned and is not in SeedsQueue, it is added to ScanOrder. This repeats until ScanOrder is empty, when function GetNextStart() is called again to start a new loop. The procedure stops when all the points were add in ScanOrder.

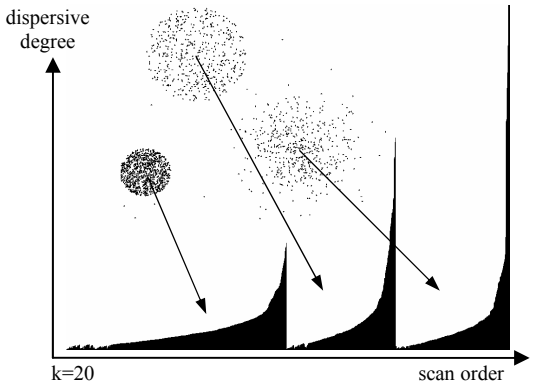
Fig. 2: Procedure ScanData.

```
ScanData(DataSet, ScanOrder)
   WHILE DataSet.GetNextStart() <> NULL DO
      SeedsQueue.Insert(DataSet.GetNextStart);
      WHILE NOT SeedsQueue.Empty() DO
         CurrentPoint = SeedsQueue.GetFirst();
         ScanOrder.Add(CurrentPoint);
         CurrentPoint.scanned = TRUE;
         FORALL Neighbor FROM
                  CurrentPoint.Neighbors[] DO
            IF Neighbor.scanned = FALSE
              AND NOT
                    SeedsQueue.Contain(Neighbor)
               SeedsQueue.Insert(Neighbor);
END
```

GetNextStart() insures that a scan process starts at a point in a cluster, which can be inferred from lemma 1; and it is ensured by GetFirst() that a point, which is most likely to be in the same cluster with the last point in ScanOrder, will be added next in conformity to lemma 2.

The scan order of a data set can be represented and understood graphically. The clustering structure can be seen if the dispersive degrees are plotted for each point in ScanOrder. Figure 3 depicts the DDegree-plot for a very simple data set (i.e. there are two uniform and one Gaussian distribution clusters with different densities in the data set).

Fig. 3: Illustration of the ScanOrder.



2.5. Dividing scanning order

After carrying out accurate observations, we found that points in the regions with lower DDegree are the core points of clusters, which distribute in the vicinity of the centers. The points in high DDegree region mainly are edge of clusters or noise. According to this fact, CDDC divides the scan order of a data set into sets of core points and corresponding edge points.

Definition 10. (downgrade point) Let p_i be a point in a ScanOrder S, i=1,2,…,n, n is the size of D. If the DDegree of p_i is larger than p_{i+1}, it's a downgrade point, denoted by DP. DPs={$p_i \in S \mid \forall i$: DDegree(p_i) > DDegree(p_{i+1})}

Definition 11. (contour point) Let p_i be a point in the ScanOrder S, i=1,2,…,n, n is the size of D. The contour point of p_i, denoted by CP(p_i), is the first point after p_i with the DDegree larger than it. CP(p_i)=q_j, where j=min{l>i | $\forall$ l: DDegree(o_l) > DDegree(p_i)}

Definition 12. (peak point) Let DPs be the set of downgrade points of a ScanOrder. Peak points denoted by PPs, are the subset of DPs, in which the DDegrees of the points are monotonically increasing and satisfy the following conditions: The DDegree of point must larger than a given threshold dt; the number of points between a downgrade point p and its contour point CP(p) must more than a given threshold mt.

Definition 13. (dividing point) Let p be a peak point in the set of peak points of a ScanOrder S. If p has the largest Degree in the set, it is a dividing point of S.

For instance, Figure 4 gives a fragment from a scan order of a data set. Point A, B, C, D and E are downgrade points. Point C' and D' are the contour points of C and D. D is a peak point but not C and E, because C' is not so "far" apart from C(i.e. points between them is less than mt) and E is too "low"(i.e. DDegree of E is smaller than dt).Point D is the dividing point of the scan order.

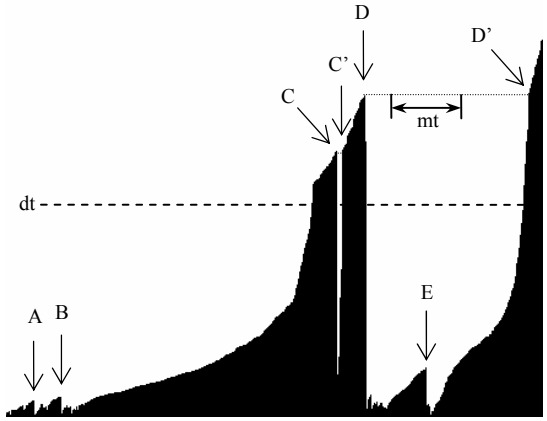


Fig. 4: Downgrade, contour, peak and dividing point.

Figure 5 shows the procedure of DivideScanOrder. In this step, CDDC iterative splits current order into two half by function BinSplit() in the position of its dividing point. The first half is pushed into a stack of orders, and the last will be dealt with next as the current order until it can not be divided again, i.e. function FindPeaks() can not find any peaks of current order. The function Filter() will distinguishes between core points and edge points, with the cutoff point which is the contour point of current order's last dividing point. When the stack is empty, the scan order is divided into sets of core points and edge points.

```
DivideScanOrder (ScanOrder, CoreSet, EdgeSet, dt,
mt)
  Stack.push(ScanOrder);
  WHILE NOT Stack.Empty() DO
    IF CurrentOrder=NULL;
      CurrentOrder=Stack.pop();
    CurrentOrder.FindPeaks(dt, mt);
      IF NOT CurrentOrder.Peaks.Empty
        CurrentOrder.BinSplit();
      ELSE
        CurrentOrder.Filter(CoreSet,EdgeSet);
        CurrentOrder=NULL;
END
```
Fig. 5: Procedure DivideScanOrder.

2.6. Classifying edge points

Each set of core points is considered as a cluster. The Points at the end of scan order are always having the highest dispersive degree in the data set. Therefore the last set of edge points in the tail of the scan order is considered to be not in any cluster but noise. Step 4 of CDDC is to classify edges to these clusters or noise. There are lots of classification algorithms, but we choose the KNN-kernel density-based method [6]. Because it is suited for various density data and can fully make use of the KNN information extracted by step 1.

The KNN-kernel method was developed by Terrell and Scott [7]:

Definition 13. (KNN-kernel density estimate) The KNN-kernel density estimate obtained at a object x in a N×d dimensional data set with kernel K is defined as

$$\hat{f}(x) = \frac{1}{NV_x}\sum_{i=1}^{n} K((x - x_i)./H_x)$$

where V_x is the data volume adjusted to include the KNN objects, H_x is a scale vector[h^1_x … h^d_x] of V_x.

The most common classification rules are based on Bayes' decision rule:

$$p(x \mid C_i)\,p(C_i) > p(x \mid C_j)\,p(C_j), \quad \forall i \neq j$$

where p(x|C_i) is the class-conditional density function at x of each class C_i, and p(C_i) is the prior probability function. The class-conditional density function can be estimated by KNN-kernel defined above:

$$\hat{p}(x \mid C_i) = \frac{1}{n_i V_x} \sum_{x_j \in C_i} K((x - x_j)./H_x)$$

where n_i is the size of cluster C_i. Bayes' KNN-kernel class-condition can be rewritten as:

$$\frac{1}{n_i V_x}\left(\sum_{x_l \in C_i} K((x-x_l)./H_x)\right) p(C_i) > \frac{1}{n_j V_x}\left(\sum_{x_l \in C_j} K((x-x_l)./H_x)\right) p(C_j)$$

$p(C_i)$ and $p(C_j)$ are normally estimated by n_i/N and n_j/N. Then the expression above can be simplified:

$$\sum_{x_l \in C_i} K((x-x_l)./H_x) > \sum_{x_l \in C_j} K((x-x_l)./H_x)$$

For CDDC clustering algorithm, an edge point p is assigned to a cluster C_i (noise is denoted by C_0 here) in order to maximize the priority function defined by

$$\mathrm{Prob}(C_i) = \sum_{q_l \in Ci} K\left(\frac{dis(p,q_l)}{dis(p,o)}\right)$$

where the dissimilarity, dis(p,q) is defined by:

$$dis(p,q) = dist(p,q) \times \left|DDegree(p) - DDegree(q)\right|$$

o is the point which is the most dissimilar k-nearest-neighbor of p, K is triangular kernel, defined by K(z)=1-|z|, if |z|<1; K(z)=0,otherwise.

2.7. Determining the parameters

CDDC clustering algorithm needs only one parameter, k, the number of neighbors to depict the clustering of a data set. Some researches indicate that if k is too small, there will be too many modes; otherwise, too large, modes have disappeared [8]. But CDDC is roughly insensitive to the input parameter. Figure 6 shows the effects of different k on the DDegree-plot for the same data set used in figure 3.In the first plot a smaller k is used and larger for the second. For smaller k the DDegree-plot looks more jagged and larger k smoothen the curve. However, with this difference the clustering structure could be recognized in all these plots in the same way. It is a further advantage of CDDC compared to other methods that it's not difficult to find a range of k for which clustering results are stable. CDDC can always get good results using values between 10 and 20.

The other two parameters, mt and dt are easy to be chosen according to the DDegree-plot. The parameter mt is used to filter the clusters containing points less than it, which users do not care about. And the order with a diving point lower than dt will not be divide anymore because the DDegree of the points is small enough to be core points of a cluster. So both the parameter can be fixed visually and interactively.

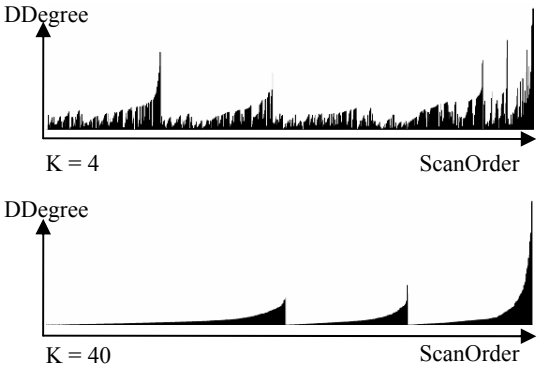


Fig. 6: Effects of k setting on the scan order.

2.8. Computational complexity

Let n be the size of a data set, including m clusters and l edge points, k be the number of neighbors. Then the time complexity would be $O(n^2)$ for computing DDegree, O(n) for producing ScanOrder, about O((m-1)n) for dividing ScanOrder and O(lkm) for classification. Hence the total time complexity is $O(n^2+mn+lkm)$.Because m ,l and k are far less than n in general, the CDDC's runtime would be $O(n^2)$. If a tree-based index such as the R*-tree [9] or the X-tree [10] can be used, the run-time is reduced to O(n*log*n).

3. Performance evaluation

In this section, we evaluate the performance of CDDC. We compare it with the performance of DBSCAN, OPTICS and KNNCLUST, because the first one is an acknowledged classical density-based clustering algorithm, and the other two are designed to clustering data set with various densities. We have implemented CDDC, DBSCAN and OPTICS in java with out any index. All experiments have been run on the workstation with Pentium D 2.8G CPU, 512MB memory and Windows XP pro sp2.

To compare these algorithms in terms of effectivity, we use two synthetic sample data sets which are depicted in figure 7. They have clusters of different shape, size, density, and orientation, as well as random noise. Note that DS2 was obtained from Chameleon [11], whereas we synthetically generated the remaining data set.

DS1:7118 points

DS2:8000 points

Fig. 7: The two data sets used in experiments.

Precision, recall and F-measure are common measurements used in information retrieval for evaluation [12]. Their comparison of DBSCAN, OPTICS and CDDC is shown in figure 8. Because the KNNCLUST can not find clusters of strange shape and gives poor results in our experiment, we omit its items. The ideal clusters are Clockwise direction marked by C1-C9. Parameters of DBSCAN is fine tuned to find all the ideal clusters (Eps = 11, MinPts = 4), but in this case, C8 and C9 (the triangles) are recognized as a single cluster and lots of noise point are clustered mistakenly. OPTICS can not tell when to stop dividing the a cluster, so we artificial select the "best" result (i.e. most similar to the ideal clusters) with the parameters Eps = 22 and MinPts = 20, ξ = 0.01. Parameters of CDDC , k, mt and dt are set to 20, 250 and 1.097. The two thresholds are determined according to the DDegree-plot.

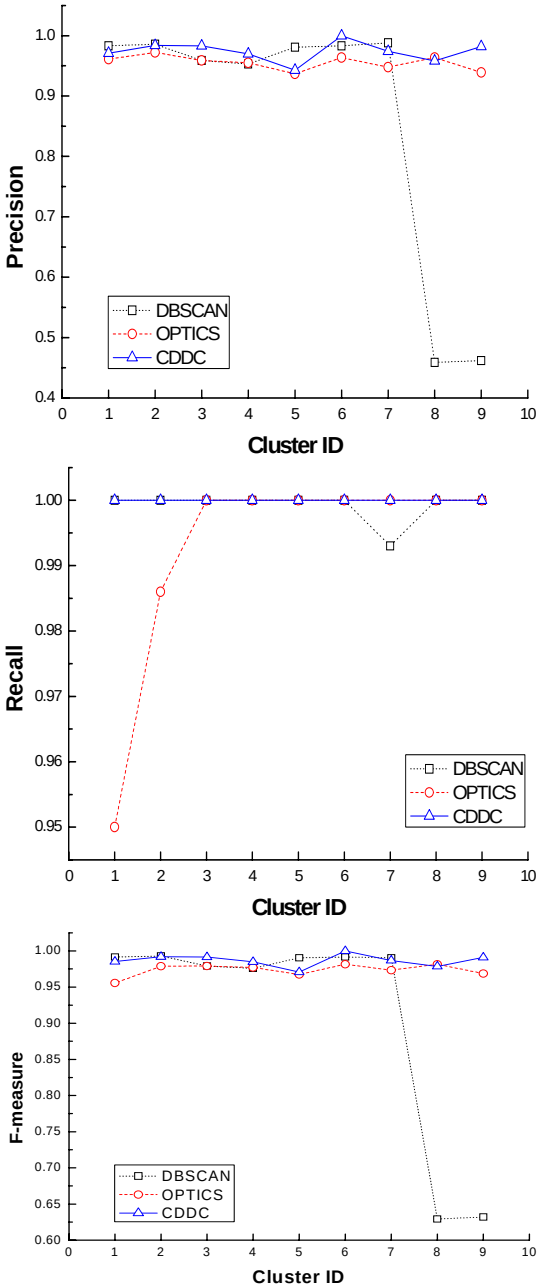


Fig. 8: Comparison of DBSCAN, OPTICS and CDDC.

Since there is no common internal quality evaluations for density-based clustering algorithms, we evaluate the results on DS2 by visual inspection. The comparison is depicted in figure 9.

Obviously, CDDC gets the best results of all.

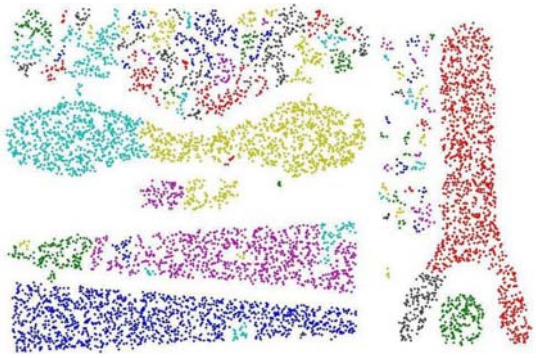


Fig. 9: Clusters found by DBSCAN.

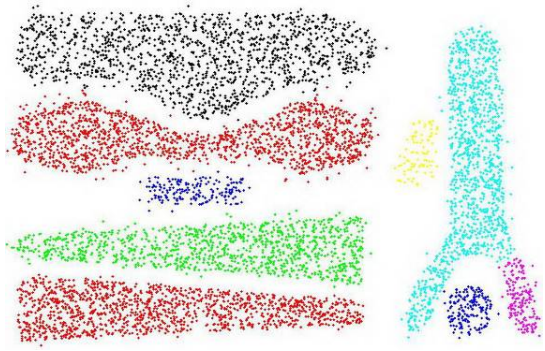


Fig. 10: Clusters found by OPTICS.

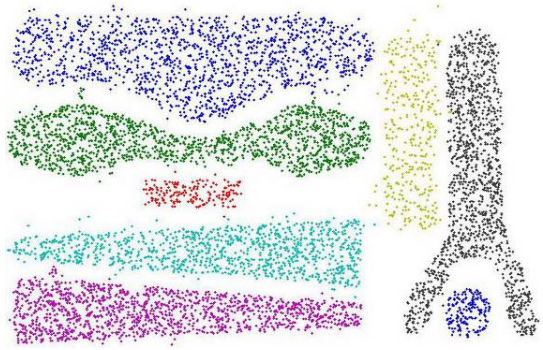


Fig. 11: Clusters found by CDDC.

4. Conclusions

Many density-based clustering algorithms such as DBSCAN, suffer from the problem of clusters with different densities. This is not the only purpose for our new proposed algorithm, CDDC, making use of dispersive degree and kernel density estimation based on KNN. CDDC need only one parameter, k, the number of neighbors, to discover the essential clustering structure, according to which mt and dt can be determined easily to produce useful clusters. It is suited for finding arbitrary shape and density clusters and noise. Furthermore it has the same time complexity as DBSCAN. In conclusion, CDDC is a good method to cluster data set where the clusters are very different in densities.

Acknowledgement

This work is partially supported by National Nature Science Foundation of China (Grant No. 60673066).

References

[1] J.W. Han, M. Kambr, *Data Mining Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.

[2] M. Ester, HP. Kriegel, J. Sander, A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Knowledge Discovery and Data Mining*, pp. 226-231, 1996.

[3] M. Ankerst, M. Breunig, H.P. Kriegel and J. Sander, OPTICS: Ordering Points to Identify the Clustering Structure. Proc. *ACM SIGMOD'99 Int. Conf. on Management of Data*, pp. 49-60, 1996.

[4] T.N. Tran, R. Wehrens, Lutgarde M.C.Buydens, KNN-kernel density-based clustering for high-dimensional. *Computational Statistics & Data Analysis,* 51: 513-525, 2006.

[5] D.O. Loftsgaarden, C.P. Quesenberry, A nonparametric estimation of a multivariate density function. *Ann.Math.Statist* 36: 1049-1051, 1965.

[6] A. Webb, *Statistical Pattern Recognition*, Wiley, Malvern, 2002.

[7] G.R. Terrell, D.W. Scott, Variable kernel density estimation. *Ann. Statist.* 20: 1236-1265, 1992.

[8] http://www.ficcs.org/meetings/ficcs2/ficcs2.html. http://web.maths.unsw.edu.au/~tduong/research/index.html.

[9] N. Beckmann, H.P. Kriegel, R. Schneider and B. Seeger, The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 322-331, 1990.

[10] S. Berchthold, D. Keim, H.P. Kriegel, The X-Tree: An Index Structure for High-Dimensional Data. *22nd Conf. on Very Large Data Bases*, pp. 28-39, 1996.

[11] G. Karypis, E.H. Han, Vipin Kumar, *IEEE Computer: Special Issue on Data Analysis and Mining*, 32: 68-75, 1999.

[12] D. Crabtree, P.Andreae, X. Gao. QC4-A Clustering Evaluation Method. *Proc. of PAKDD*, pp. 59-70, 2007.