



# A local cores-based hierarchical clustering algorithm for data sets with complex structures

Dongdong Cheng<sup>1</sup> · Qingsheng Zhu<sup>1</sup> · Jinlong Huang<sup>2</sup> · Quanwang Wu<sup>1</sup> · Lijun Yang<sup>3</sup>

Received: 25 February 2018 / Accepted: 13 July 2018 / Published online: 26 July 2018  
© The Natural Computing Applications Forum 2018

## Abstract

Hierarchical clustering is of great importance in data analysis. Although there are a number of hierarchical clustering algorithms including agglomerative methods, divisive methods and hybrid methods, most of them are sensitive to noise points, suffer from high computational cost and cannot effectively discover clusters with complex structures. When recognizing patterns from complex structures, humans intuitively tend to discover obvious clusters in dense regions firstly and then deal with objects on the border. Inspired by this idea, we propose a local cores-based hierarchical clustering algorithm called HCLORE. The proposed method first partitions the data set into several clusters by finding local cores, instead of optimizing an objective function through iteration like *K*-means; then temporarily removes points with lower local density, so that the boundary between clusters is clearer; after that merges clusters according to a newly defined similarities between clusters; and finally points with lower local density are assigned to the same clusters as their local cores belong to. The experimental results on synthetic data sets and real data sets show that our algorithm is more effective and efficient than existing methods when processing data sets with complex structures.

**Keywords** Hierarchical clustering · Local cores · Complex structures

**Electronic supplementary material** The online version of this article (<https://doi.org/10.1007/s00521-018-3641-8>) contains supplementary material, which is available to authorized users.

✉ Qingsheng Zhu  
qs Zhu@cqu.edu.cn

Dongdong Cheng  
cdd@cqu.edu.cn

Jinlong Huang  
h.jinlong@qq.com

Quanwang Wu  
wqw@cqu.edu.cn

Lijun Yang  
yljun@cqu.edu.cn

<sup>1</sup> College of Computer Science, Chongqing University, Chongqing, China

<sup>2</sup> College of Computer Engineering, Yangtze Normal University, Chongqing, China

<sup>3</sup> School of Computer Science and Technology, Southwest Minzu University, Chengdu, China

## 1 Introduction

The information age has brought rapid growth of data. How to mine useful knowledge from big data has attracted a significant amount of attention and research. Clustering, as an important subject of data mining, has been applied to bioinformatics, business management, community detection, pattern recognition, machine learning and recommender system [33, 34].

Clustering analysis means to arrange the objects in such a way that similar objects are in the same group, while dissimilar objects are in different groups. Many different clustering algorithms have been proposed in the literature [13].

Partitioning clustering algorithms, such as *K*-means [22] and *K*-centers [26], use an iterative strategy to optimize an objective function. These algorithms can effectively cluster data with Gaussian-like distribution. However, they are sensitive to the selection of initial cluster centers and the fact that a data point is always assigned to its nearest cluster center makes them fail to detect non-spherical clusters.

In order to avoid initializing cluster centers, some novel center-based clustering algorithms are proposed. In 2007, Frey and Dueck proposed a new clustering algorithm by

passing messages between data points, called Affinity Propagation (AP) [8]. In 2014, Rodriguez and Laio proposed a novel clustering algorithm by fast search and find of density peaks [25] (DP for short) which is on the assumption that cluster centers are surrounded by neighbors with lower local density and that they are at a relatively large distance from any points with a higher local density. Nevertheless, these algorithms still have difficulty in discovering complex structured clusters.

Density-based clustering [7, 21] assumes that clusters are dense regions separated by sparse regions, and therefore, it can discover clusters with arbitrary shapes. DBSCAN [7] is a typical density-based clustering algorithm. It globally defines the density of points using two thresholds (the radius  $\varepsilon$  and the minimum number of points *minpts*). It is insensitive to noise data. However, the main difficulties of DBSCAN are parameter selection and detection clusters with different densities.

Hierarchical clustering seeks to build a hierarchy of clusters. Single-link algorithm [23], complete-link algorithm [17] and Chameleon [16] are the representative algorithms. Chameleon is a hierarchical method which integrates bottom-up and top-down strategies. It can find more natural clusters of various shapes as it measures the similarity of two clusters dynamically by considering data distribution within a cluster. However, it is time-consuming and susceptible to noise points and cannot process data sets with complex structures.

Spectral clustering [20] utilizes spectral graph theory to map the data into a low-dimensional space and combines traditional clustering algorithms for clustering. Compared with the classic methods, spectral clustering algorithm is able to discover non-convex clusters. A density-adaptive affinity propagation clustering algorithm based on spectral dimension reduction (DAAP) [15] is proposed to improve the performance of AP when processing data sets with complex structures. However, these algorithms are computationally intensive.

Data sets with complex structures generally refer to those which contain clusters with diverse shapes (including spherical, non-spherical, manifold), sizes and densities. Classical clustering methods (e.g.,  $K$ -means, DBSCAN) do not perform well on them. In order to efficiently process data sets with complex structures, we propose a hierarchical clustering algorithm based on local cores called HCLORE. Our method is motivated by the reality that when recognizing clusters from complex structures, humans intuitively tend to discover obvious clusters in dense regions first and then deal with potentially ambiguous objects with lower local density. The steps of our method are as follows. First, it computes the local density and obtains the local cores which are points with local maximum density in the local neighbors, and each remaining point, including points with lower local density, is assigned to the cluster that its local core belongs to. As a

result, the original data set is divided into small clusters based on local cores, instead of optimizing an object function through iteration. After that, points with lower local density are temporarily removed from the data set to make the boundary between clusters clearer. Then, we define a new metric to measure the similarity between clusters and obtain the clusters by continuously merging the most similar clusters. Finally, points with lower local density are assigned to the cluster their local cores belong to. We compare our method HCLORE with the partitioning clustering algorithm  $K$ -means [22], hierarchical clustering algorithm Chameleon [16], the density-based clustering algorithm DBSCAN [7], the spectral-based clustering algorithm DAAP [15] and the center-based clustering algorithm DP [25]. The experimental results on synthetic and real data sets show that HCLORE is more effective than the existing algorithms when processing data sets with complex structures.

The rest of this paper is organized as follows. Section 2 reviews related work about center-based clustering and hierarchical clustering. Section 3 introduces local density and natural neighbor. The proposed algorithm HCLORE is presented in Sect. 4, and Sect. 5 shows the experimental results and analysis. Section 6 concludes this study.

## 2 Related work

In this section, we review related center-based clustering algorithms and hierarchical clustering algorithms.

### 2.1 Center-based clustering

For many classical algorithms, a key step is to find cluster centers. For example,  $K$ -means [22] and  $K$ -centers [26] obtain clustering results by optimizing an objective function, typically the sum of the distance to a set of putative cluster centers. However, the initialization of centers is a difficult task. QCC [12] assumes that the density of a cluster center is the highest in its  $k$  nearest neighbors or revers  $k$  nearest neighbors. The main purpose of AP algorithm [8] is to find the optimal cluster centers by exchanging real-valued messages between data points. Different from  $K$ -centers, AP regards all objects as potential cluster centers instead of specifying the initial cluster centers in advance. However, AP cannot obtain satisfactory results since the number of clusters is affected by user-defined preference parameter.  $K$ -AP [36], an improved algorithm of AP, exploits the immediate results of  $K$  clusters by introducing a constraint in the process of message passing. A density-adaptive affinity propagation clustering algorithm based on spectral dimension reduction (DAAP) [15] is an improved version of  $K$ -AP. DAAP firstly constructs Laplacian matrix according to spectral graph theory

and then uses the eigenvectors to map the original data points to a low-dimensional space and subsequently constructs density-adaptive similarity measurement and employs K-AP to cluster. It does better work than K-AP when detecting clusters with manifold structures, but has high computational complexity.

DP algorithm [25] uses decision graph to map the data into a two-dimensional graph w.r.t the local density  $\rho$  and the distance  $\delta$ . Cluster centers stand out with anomalously large value of  $\rho$  and  $\delta$ . DP does not require continuous iteration to get cluster centers, making it an efficient algorithm. However, DP does have some drawbacks. First, it has to set the cutoff distance which can influence the clustering results. Second, the decision graph might produce the redundant centers [18]. Third, the assignment strategy for the remaining points can create a Domino Effect, that is, once one point is assigned erroneously, there may be more points subsequently misassigned. Some improved algorithms [4, 6, 18, 29, 31] are proposed to solve these problems, yet these algorithms have problems of discovering clusters with complex structures.

## 2.2 Hierarchical clustering

Hierarchical clustering is to establish a hierarchical structure. Two strategies for hierarchical clustering are top-down (division) and bottom-up (agglomeration). Divisive methods initially assume that all objects are in one cluster, and splits are performed as one moves down the hierarchy. The time complexity of divisive methods is high,  $O(2^N)$  ( $N$  is the number of objects in a data set.) Agglomerative methods start with every individual object as a cluster, and two most similar clusters are merged as one moves up the hierarchy, such as the single-link algorithm [23] and the complete-link algorithm [17]. The single-link algorithm uses the distance between the two closest points of the two clusters as the distance between clusters, while the complete-link algorithm uses the distance between two farthest points.

Some clustering algorithms [1, 16, 19, 35] combine partitioning methods with hierarchical methods, which is called hybrid clustering algorithms. The main differences of hybrid clustering algorithms are the partition methods and the distance or similarity measure between clusters. BIRCH [35] first partitions the data set into many small clusters with CF-Tree and applies a global clustering algorithm on those clusters to achieve the final results. Chameleon [16] first uses a graph-based partition algorithm to divide the data set into many clusters and then measures the similarity between clusters based on relative interconnectivity (RI) and relative closeness (RC). The algorithms in [1, 19] both employ  $K$ -means to partition the data

set in the first phase. In [19], a new similarity measure, referred to as cohesion, is proposed to measure the inter-cluster distances. An efficient agglomerative hierarchical clustering algorithm [1] builds a hierarchy based on a group of centers rather than raw data points.

Some newly hierarchical methods are recently proposed. Synchronization-inspired partitioning and hierarchical clustering algorithm [28] uses the extensive Kuramoto model to cluster. It discovers interpretable cluster hierarchies through continually expanding the neighbor search range. DenPEHC [32] is an improved algorithm for DP algorithm, which directly generates hierarchical clusters on each possible clustering layer.

## 3 Preliminaries

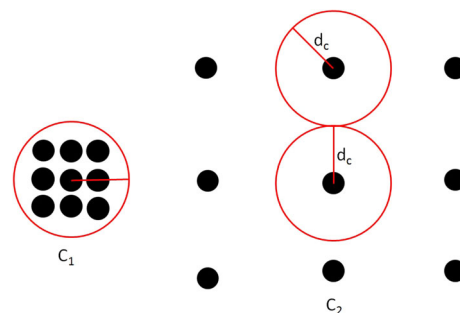
### 3.1 Local density

Density-based clustering algorithms, such as DBSCAN and DP, define the density of each point  $p$  as the number of neighbors whose distance to point  $p$  is less than the cutoff distance  $d_c$ , as shown below:

$$\rho(p) = \sum_q \chi(\text{dist}(p, q) - d_c) \quad (1)$$

where if  $x < 0$ ,  $\chi(x) = 1$ , otherwise,  $\chi(x) = 0$  and  $d_c$  is the cutoff distance. Thus,  $\rho(p)$  is equal to the number of points whose distance to point  $p$  is less than  $d_c$ .

However, when the inter-cluster density is significantly different, it will be hard to set an appropriate cutoff distance, which will influence clustering results, just as shown in Fig. 1. If the value of  $d_c$  is set inappropriately, then there are no neighbors for points in cluster  $C_2$  and they will be regarded as noises by DBSCAN. As for DP algorithm, points in cluster  $C_2$  are all with local maximum density, making it hard to select cluster



**Fig. 1** Illustration of great density variations, the value of cutoff distance is hard to set

### 3.2 Natural neighbor

Let  $dist(x, y)$  be the Euclidean distance between points  $x$  and  $y$ . In the data set  $D$ , point  $o$  is the  $k$ -th nearest neighbor of point  $p$ . The definitions of  $k$  nearest neighbors and reverse  $k$  nearest neighbors can be given as follows.

**Definition 1** (*K nearest neighbors*) The  $k$  nearest neighbors of point  $p$  are a set of points  $x$  with  $dist(p, x) \leq dist(p, o)$ , that is,  $NN_k(p) = \{x \in D | dist(p, x) \leq dist(p, o)\}$ .

**Definition 2** (*Reverse k nearest neighbors*) The reverse  $k$  nearest neighbors of point  $p$  are a set of points  $x$  which consider  $p$  as its  $k$  nearest neighbor, that is,  $RNN_k(p) = \{x \in D | p \in NN_k(x)\}$ .

Natural neighbor [37] is a new neighbor concept. Compared with  $k$  nearest neighbor, it does not need to set

**Definition 3** (*Natural neighbor*) Natural neighbor of  $p$  is defined as follows.

$$NaN(p) = \{x | x \in NN_\lambda(p) \wedge p \in NN_\lambda(x)\} \quad (2)$$

**Definition 4** (*Natural neighbor graph NNG*) The graph constructed by linking each point and its natural neighbors is natural neighbor graph.

**Definition 5** (*Saturated neighbor graph SNG*) The graph constructed by linking each point and its  $\lambda$  nearest neighbors is saturated neighbor graph.

Since KD-tree is introduced into NaN-Searching algorithm, the time complexity of searching local neighbors is  $O(N \log N)$ . ( $N$  is the number of objects in a data set.) The value of  $nb(x)$  is larger for points in dense region than that in sparse region, which is a good reflection of the point local characteristic. Thus, it is reasonable to use  $nb(x)$  as the number of point  $x$ 's neighbors.

---

#### Algorithm 1: NaN-Searching

---

**Input:**  $D$  (the data set)

**Output:**  $\lambda, nb$

Initializing:  $r=2$ ,  $nb(i)=0$ ,  $NN_0(i) = \phi$ ,  $count(1) = N$ ,  $RNN_0(i) = \phi$ ;

**while true do**

**for** each data point  $p$  in  $D$  **do**

        Use KD-tree to find the  $r$ -th neighbor  $q$  of  $p$ ;

$nb(q) = nb(q) + 1$ ;

$NN_r(p) = NN_{r-1}(p) \cup \{q\}$ ;

$RNN_r(q) = RNN_{r-1}(q) \cup \{p\}$ ;

**end**

$count(r) = \text{length}(\text{find}(nb == 0))$ ;

**if**  $count(r) == count(r - 1)$  **then**

        Break;

**end**

$r = r + 1$ ;

**end**

$\lambda = r$ ;

---

any parameters and is an adaptive neighbor method. In [37], natural neighbor is mainly used for classification and outlier detection. In this paper, the result of natural neighbor method is combined with a hierarchical strategy for clustering analysis. The formation of natural neighbor is shown in Algorithm 2, where  $\lambda$  is the natural characteristic value and  $nb(q)$  is the number of reverse neighbors of point  $q$ . The algorithm indicates that we continually expand neighbor searching region  $r$ , until the number of points without reverse neighbors does not change, and we call it reaches natural stable state, and the searching range at this moment is natural characteristic value  $\lambda$ .

### 4 The proposed algorithm

The proposed method is a hybrid clustering algorithm. In the partition phase, we find local cores which are with higher density than their neighbors. Each remaining point, including points with lower local density, is assigned to the same cluster as its local core belongs to. Thus, the data set is divided into small clusters. Since our method discovers obvious clusters in dense regions first and then deals with potentially ambiguous objects with lower local density, we determine a density threshold and points with lower local density than the threshold are temporarily removed to make the boundary between clusters clearer. In the merge phase,

we compute similarities between clusters and then continuously merge the most similar clusters until the desired cluster number is obtained. Finally, the points with lower local density are assigned to the same clusters as their local cores belong to. In Fig. 2, we give an example to illustrate the steps of HCLORE. In the following, the details of the proposed clustering algorithm HCLORE are given.

#### 4.1 Local density

The density of points in a dense region is usually larger than that in a sparse region. This means that the sum of the distances between a point and its neighbors in a dense region is usually smaller than that in a sparse region. As for two different points, we find the same number of neighbors. The larger the sum of distance between the point and its neighbors is, the sparser region the point lies, and vice versa. STclu algorithm [29] employs  $K$  nearest neighbors to compute the local density. In this paper, the local density is defined in a similar way and the local density of point  $p$  is computed as shown in Eq. (3).

$$\rho(p) = \frac{\mu}{\sum_{x \in NN_{\mu}(p)} \text{dist}(p, x)} \quad (3)$$

where  $\mu$  is the maximum value of  $nb$ . Different from definition in [29], the new defined local density does not need to set parameter  $k$ , because NaN-Searching algorithm is committed to find the appropriate parameter  $k$ .

#### 4.2 Local cores

Local representatives are proposed in [5]. Since local representatives take the same number of neighbors into consideration, they cannot well represent the local feature of the data set. In this section, local cores are introduced to represent the data set. The detailed definitions are as follows.

**Definition 6** (*Local Neighbors (LN)*) For each point  $p$ , the  $nb(p)$  nearest neighbors are its local neighbors, that is,  $LN(p) = NN_{nb(p)}(p)$ ;

Definition 6 means that different points have different number of neighbors. Points in dense region have more neighbors than that in sparse region.

**Definition 7** (*Representative*) If the density of point  $q$  is the maximum among the local neighbors of point  $p$ , then  $q$  is the representative of  $p$  and its local neighbors. We denote it as  $Rep(p) = q$ .

According to Definition 7, there exists a situation that a point has two or more representatives at the same time. The representatives will compete for being representative of the point, and the representative competition rule is defined as follows.

**Representative competition rule (RCR)** For point  $p$ , if  $Rep(p) = R_1$  and  $Rep(p) = R_2$  at the same time, then  $rep(p) = \arg \min_{x \in \{R_1, R_2\}} \{\text{dist}(p, x)\}$ , that is, the representative

closer to point  $p$  will be the final representative of  $p$ .

If the representative of point  $p$  is  $q$ , and the representative of  $q$  is  $r$ , then the representative of point  $p$  will be changed to  $r$ . We call it representative transfer rule which is defined as follows.

**Representative transfer rule (RTR)** If  $Rep(p) = q$ , and  $Rep(q) = r$ , then  $Rep(p) = r$ .

**Definition 8** (*Local Core*) A point  $p$  is a local core if  $Rep(p) = p$ .

Local cores are points with local maximum density. Each local core becomes the initial cluster center. The rest points are assigned to the cluster that their local cores belong to. Local cores searching algorithm (LORE) is shown in Algorithm 1, where  $Rep(p)$  is the representative of point  $p$ ,  $localCores(N_l)$  is the  $N_l$ -th local core, and  $cl(p)$  is the cluster label of point  $p$ . Since we search local cores in descending order of points density, the input order of points has no impact on the result of LORE algorithm. As shown in Fig. 3, for each data set, the red star points are local cores found by LORE. Then the data sets are divided into several clusters according to the local cores.

**Algorithm 2: LORE**


---

**Input:**  $LN$  (the local neighbors),  $\rho$  (the density)  
**Output:**  $Rep$  (the representatives),  $localCores$  (the local cores),  $cl$  (the cluster label)

Initializing:  $Rep(i)=\phi$ ,  $localCores=\phi$ ;

**for** each point  $i$  in descending order of density **do**  
  Find the point  $y$  with the maximum density in  $LN(i)$ ;  
  **for** each point  $p$  in  $LN(i)$  **do**  
    **if**  $Rep(p) == \phi$  **then**  
       $Rep(p) = y$ ;  
    **end**  
    **if**  $Rep(p) == x$  and  $x \neq y$  **then**  
      Determine  $Rep(p)$  according to RCR;  
    **end**  
    **for** each point  $z$  in the data set **do**  
      **if**  $Rep(z) == p$  **then**  
        Determine  $Rep(z)$  according to RTR;  
      **end**  
    **end**  
  **end**  
**end**  
 $N_l=0$ ;  
**for** each point  $p$  in the data set **do**  
  **if**  $Rep(p) == p$  **then**  
     $N_l=N_l+1$ ;  
     $localCores(N_l)=p$ ;  
     $cl(p)=N_l$ ;  
  **end**  
**end**  
**for** each point  $p$  in the data set **do**  
   $cl(p)=cl(Rep(p))$ ;  
**end**

---

**4.3 The density threshold**

Since our method discovers obvious clusters in dense regions first and then deals with potentially ambiguous objects with lower local density, we should set a density threshold and those points whose density are lower than the density threshold are removed temporarily. Here, we can use two schemes to determine the threshold.

**User-specified percentage** All the points are sorted in ascending order with respect to the density. The density of  $\alpha\%$ -th point is the density threshold, and thus, the first  $\alpha\%$  points are declared to be points with lower local density.

**Density curve** First, all the points are sorted in ascending order with respect to the density. Then we draw density increasing curve. We assume that the density of noise points is far less than that of ordinary points. Thus, there will be a mutation at the junction of noises and ordinary points. We can get the mutation by calculating the maximum value of the discrete second derivative of the density. Figure 4 shows the method. Figure 4b is the density increasing curve, and the red point is the mutation we determined. Thus, the threshold is the density value of the red point. Figure 4c is the corresponding result after

distinguishing noise points, and the black star points are objects with lower local density. As for some data sets, although the mutation between noise and ordinary points is not so obvious, this method still provides a good basis for determining the density threshold.

**4.4 Similarity between clusters**

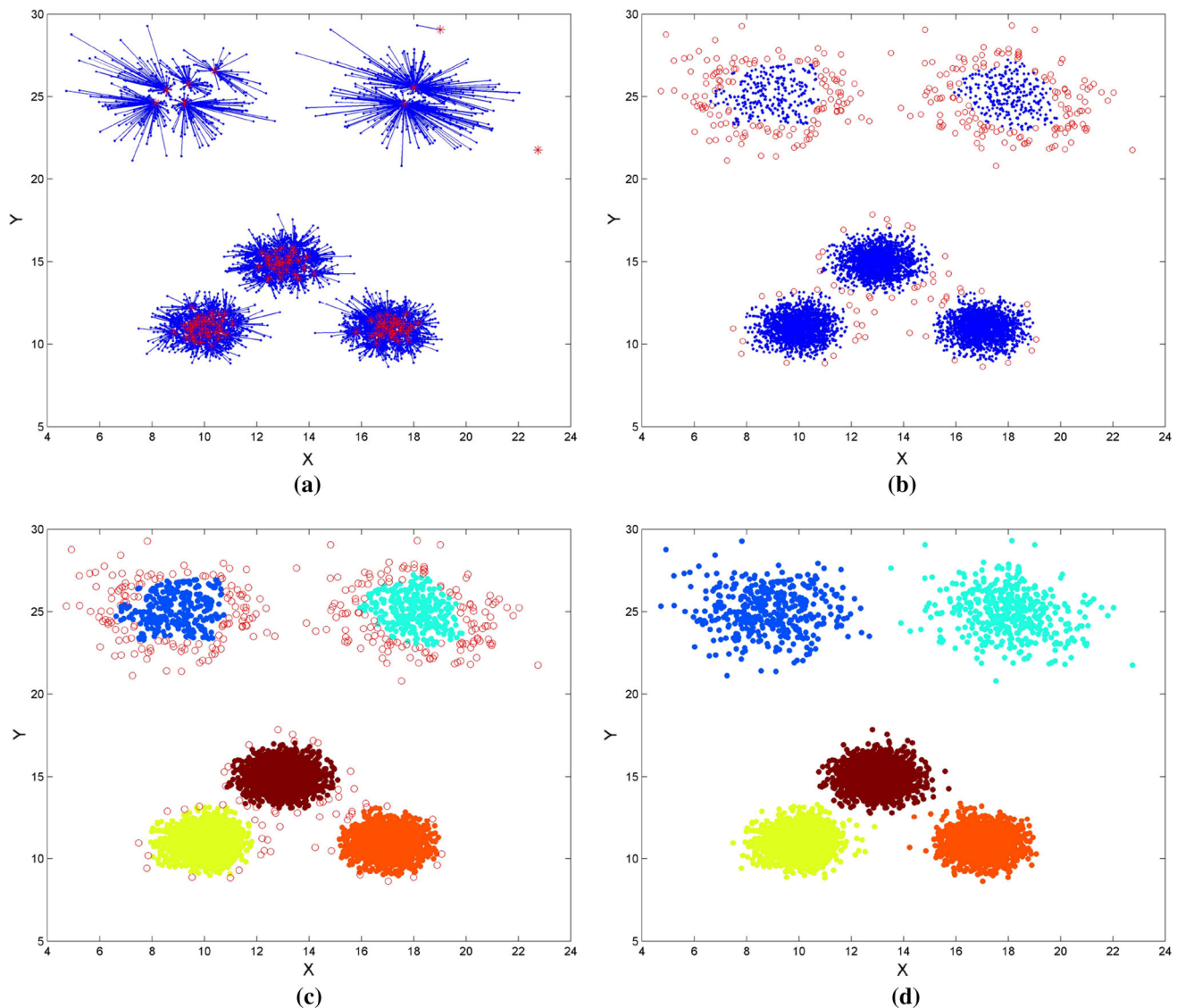
The similarity between clusters is computed based on the inter-connectivity and closeness between clusters. For clusters  $C_i$  and  $C_j$ , there exist  $v_i \in C_i$  and  $v_j \in C_j$ , then the similarity between clusters  $C_i$  and  $C_j$  is defined as follows.

The inter-connectivity between clusters  $C_i$  and  $C_j$  is the weight sum of cut edges straddling the two clusters and it is computed as follows:

$$Conn(C_i, C_j) = \sum_{e(v_i, v_j) \in CE(C_i, C_j)} w(v_i, v_j) \quad (4)$$

where  $CE(C_i, C_j)$  is the cut edges between clusters  $C_i$  and  $C_j$ ,  $w(v_i, v_j)$  is the weight of edge between points  $v_i$  and  $v_j$ , and  $w(v_i, v_j) = \frac{1}{1+dist(v_i, v_j)}$ .





**Fig. 2** Steps of HCLORE. **a** The data set is divided into small clusters by local cores (the red star points). **b** Points with lower local density are marked with red circles. **c** Small clusters are merged. **d** Points

The closeness between clusters  $C_i$  and  $C_j$  is the average weight of cut edges straddling the two clusters and it is computed as follows.

$$Close(C_i, C_j) = \frac{\sum_{e(v_i, v_j) \in CE(C_i, C_j)} w(v_i, v_j)}{|CE(C_i, C_j)|} \quad (5)$$

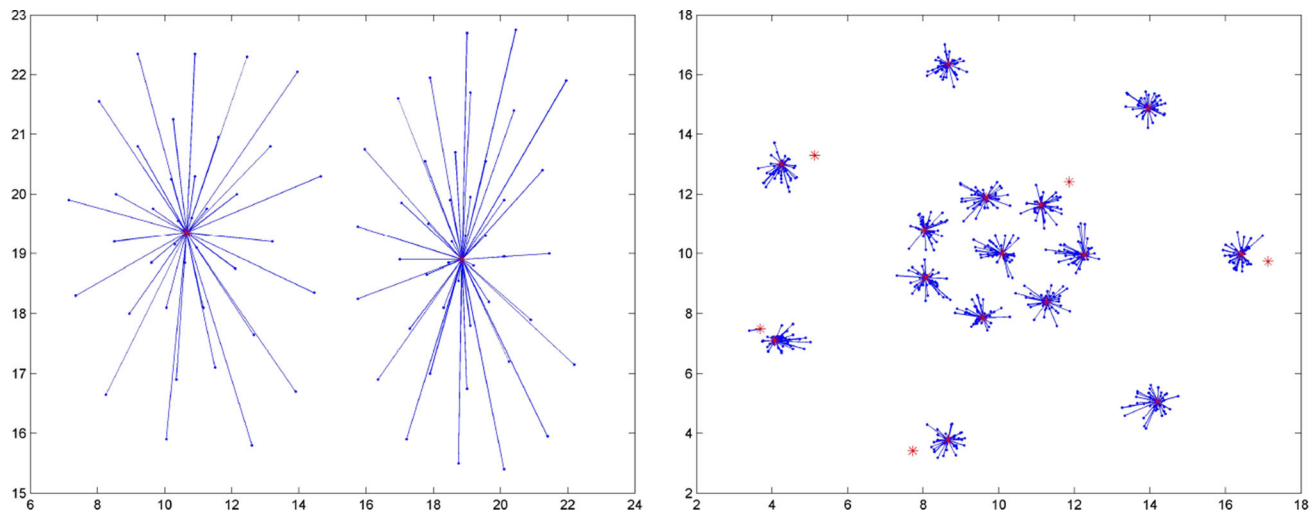
The similarity between clusters  $C_i$  and  $C_j$  is a function that combines the inter-connectivity and closeness and is computed as follows.

$$Sim(C_i, C_j) = Conn(C_i, C_j) \times Close(C_i, C_j)^2 \quad (6)$$

with lower local density are assigned to the cluster their local cores belong to (color figure online)

#### 4.5 Hierarchical clustering based on local cores (HCLORE)

Based on the above definitions, we propose a hierarchical clustering algorithm HCLORE. We first find local cores and the data set is divided into small clusters according to the local cores, instead of optimizing an objective function through iteration like K-means. According to the user-specified percentage or density curve to determine a density threshold, points with lower local density than the threshold are temporarily removed. Then we build a saturated neighbor graph on the rest points to compute the similarity between clusters. After that, we get the clustering result by repeatedly merging the most similar clusters.



**Fig. 3** Points are assigned to their representatives, and red star points are local cores found by LORE algorithm (color figure online)

Finally, the points with lower local density are assigned to the clusters that their local cores belong to. The procedure of HCLORE is detailed in Algorithm 3.

In this section, we describe the details of our method: local density, local cores, the density threshold, similarity between clusters and the main steps of HCLORE. Overall,

---

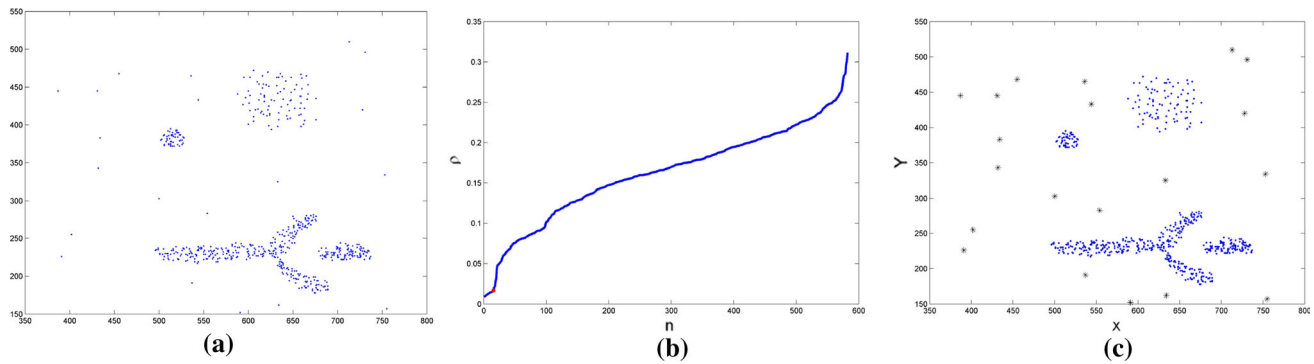
### Algorithm 3: HCLORE

---

**Input:**  $D$  (the data set),  $N_C$  (the desired cluster number)  
**Output:**  $cl$  (the cluster label of each point)  
 $(\lambda, nb) = \text{NaN-Searching}(D)$ ;  
**for** each point  $p$  in  $D$  **do**  
     $LN(p) = NN_{nb(p)}(p)$ ;  
**end**  
Compute the density  $\rho(p)$  according to Eq. (4);  
 $(Rep, localCores, cl) = \text{LORE}(LN, \rho)$ ;  
Determine the density threshold  $\rho_T$ ;  
**for** each point  $p$  in  $D$  **do**  
    **if**  $\rho(p) > \rho_T$  **then**  
         $D' = D' \cup p$ ;  
    **end**  
**end**  
Construct SNG on  $D'$ ;  
 $K = \text{length}(localCores)$ ;  
Compute the similarity matrix  $Sim$  between initial clusters according to Eq. (6);  
**while**  $K > N_C$  **do**  
     $(i, j) = \max(Sim)$ ;  
     $C_i = C_i \cup C_j$ ;  $C_j = \phi$ ;  
    Update the similarity between cluster  $C_i$  and other clusters;  
     $K = K - 1$ ;  
**end**  
**for** each point  $x$  in cluster  $C_i$  **do**  
     $cl(x) = i$ ;  
**end**  
**for** each local core  $p$  with  $\rho(p) \leq \rho_T$  **do**  
    Find the nearest neighbor  $x$  whose density is larger than  $\rho_T$ ;  
     $cl(p) = cl(x)$ ;  
**end**  
**for** each point  $p$  with  $\rho(p) \leq \rho_T$  **do**  
     $cl(p) = cl(Rep(p))$ ;  
**end**

---





**Fig. 4** **a** The original data set. **b** Density increasing curve. **c** The corresponding result. The black star points are identified by the density increasing curve

HCCLORE first finds local cores which are with higher density than their neighbors and each remaining point are assigned to the cluster its local core belongs to, and thus the data set is divided into clusters by local cores; then determines a density threshold and temporarily removes points with lower local density than the threshold, which makes the boundary between clusters clearer; after that computes similarities between clusters and continuously merge the most similar clusters until the desired cluster number is obtained; and finally assigns each point with lower local density to the same cluster as its local core belongs to. The temporary removal of lower density points makes the boundary between clusters clearer, and therefore, it is more robust against noise points. The hierarchical strategy and the idea of discovering obvious clusters in dense regions first and then dealing with potentially ambiguous points with lower local density both contribute to processing data sets with complex structures.

HCCLORE mainly contains three steps: divide the data set into clusters by search of local cores, determine the density threshold and merge clusters. The first step includes searching local neighbors for each object ( $O(N \log N)$ ), finding local cores ( $O(N)$ ) and assigning each point to the cluster its local core belongs to ( $O(N)$ ). Therefore, the time complexity of the first step is  $O(N \log N)$ . When determining the density threshold, it requires sorting the points; thus, its time complexity is also  $O(N \log N)$ . The number of initial clusters is equal to the number of local cores. Assuming the number of local cores is  $N_l$  ( $N_l < N$ ). Since the number of edges in SNG is less than  $N * \lambda/2$ , the number of cut edges is far less than  $N * \lambda/2$ . Consequently, the time complexity of computing the similarity matrix by counting cut edges is  $O(N_l * N)$ . When updating the similarity matrix, we just need to recalculate the similarity between the merged cluster and its associated clusters. The time complexity of merging process in each iteration is  $O(N)$ . Therefore, the overall time complexity of HCCLORE is  $O(N \log N)$ .

## 5 Experiments and performance evaluation

In order to demonstrate the effectiveness of HCCLORE, we compare the proposed method with  $K$ -means, DBSCAN, Chameleon, DAAP and DP algorithms on synthetic data sets and real data sets. We use two criteria to evaluate the clustering performance. The first one is the accuracy (ACC) [3, 30]. It is defined as

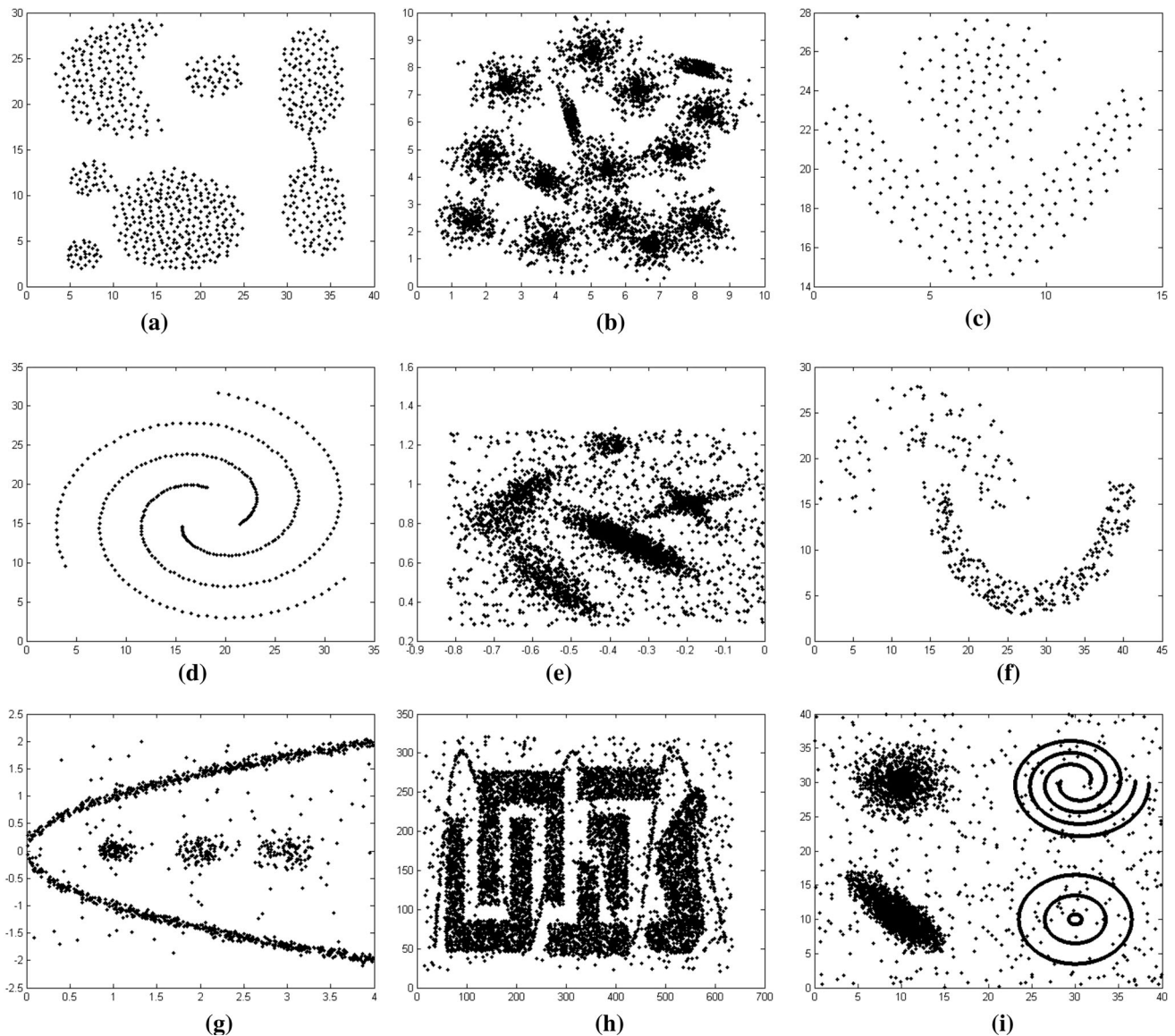
$$ACC = \frac{1}{n} \sum_{i=1}^n \delta(y_i, \text{map}(c_i)) \quad (7)$$

where  $y_i$  is the real cluster label,  $c_i$  is the serial number obtained by a clustering algorithm,  $\delta(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$  is a discrimination function, and  $\text{map}(\cdot)$  is a permutation function that maps each cluster label to a category label. Since the serial number of every cluster will be resorted after the clustering results are obtained, e.g., the first cluster of original data set may be specified as the second cluster by an algorithm, the mapping function is necessary. The optimal matching can be found by the Hungarian algorithm [24].  $ACC \in [0, 1]$ . A higher value of ACC means a better clustering result.

The second one is normalized mutual information (NMI) [3]. It is defined as

$$NMI(X, Y) = \frac{MI(X, Y)}{\sqrt{H(X)H(Y)}} \quad (8)$$

where  $MI(X, Y)$  is the mutual information between two random variables  $X$  and  $Y$  and  $H(\cdot)$  is the random variable entropy, which is used for normalizing the mutual information to be in the range of  $[0, 1]$ . Given cluster label vector and clustering results, the NMI measures how good the clustering results is.  $NMI = 1$  means the clustering result perfectly matches the cluster label vector, and  $NMI = 0$  means the clustering result is useless. The higher the NMI score, the better the clustering quality.



**Fig. 5** Synthetic data sets. **a** Aggregation, **b** S2, **c** Flame, **d** Spiral, **e** Figure2b, **f** Jain, **g** D6, **h** T4, **i** E6

The configuration of the computer used in our experiment is as follows: processor is Intel Core i5 2.80 GHz; memory size is 4 GB; programming environment is MATLAB R2013a.<sup>1</sup>

### 5.1 Clustering on synthetic data sets

The synthetic data sets are shown in Fig. 5. The first five data sets are also used in DP algorithm [25]. Data set 1 (Aggregation), a total of 788 points, taken from [10], contains 7 clusters with concave and convex shapes. Data set 2 (S2) containing 5000 points, taken from [30], has 15 clusters with high overlap in data distribution. Here, the

attribute values of S2 are scaled to [0, 10] by decimal scaling. Data set 3 (Flame) [9] is composed of two clusters with different sizes and shapes, a total of 240 points. There are three spiral clusters composed of 312 points in Data set 4 (Spiral) which is from [2]. Data set 5 (Figure2b), presented in [25], is generated from a probability distribution and contains 5 clusters and lots of noise points, a total of 4000 points. Data set 6 (Jain), taken from [14], involves two concave clusters with large variation in density, a total of 373 points. Data set 7 (D6), taken from [11], is composed of three spherical clusters, one manifold cluster and some noise points, a total of 1400 points. Data set 8 (T4) is from [16] and consists of 6 clusters with complex patterns, a total of 8000 points. Data set 9 (E6), taken from [13],

<sup>1</sup> The MATLAB code is available upon request.

contains 7 clusters with different shapes, sizes and density and amount of noise points, a total of 8537 points.

The clustering results of HCLORE algorithm are shown in Fig. 6. The best clustering results and corresponding parameters of  $K$ -means, Chameleon, DBSCAN, DAAP and DP algorithms on synthetic data sets are, respectively, presented in Fig. 1–Fig. 5 of Supplementary Materials. The ACC and NMI scores are shown in Table 1. The running time of each method on synthetic data sets is displayed in Table 2.

For  $K$ -means, the parameter  $K$  is set as the desired cluster number and the initial cluster centers are selected randomly. We run  $K$ -means for several times to get the best clustering results. The ACC and NMI scores of  $K$ -means listed in Table 1 are not high. From the results, we can see that  $K$ -means fails to discover the non-spherical clusters in the synthetic data sets.

Chameleon uses two different schemes to implement merge process. The first merges only those pairs of clusters exceeding user-specified thresholds for RI and RC. The second uses the product of RI and RC as the similarity between clusters and repeatedly merges the most similar clusters until the desired cluster number is reached. In order to avoid setting too many parameters, here we select the second scheme to implement the merge process. The clustering results in Fig. 2 of Supplementary Materials show that Chameleon is susceptible to noise points or outlier clusters. Consequently, it cannot correctly discover the clusters in S2, Flame, Figure2b, D6, T4 and E6.

For DBSCAN algorithm, it reruns many times with different parameters in an attempt to obtain the best clustering results. The results in Fig. 3 of Supplementary Materials show that DBSCAN correctly obtains the clusters in data sets Aggregation, Flame, Spiral and D6. As for data sets with large number of noise points, such as Figure2b, T4 and E6, DBSCAN takes some noise points as small clusters. As for S2 and Jain, DBSCAN fails to get the right clusters because of the variation of inter-cluster density. The clustering results of DBSCAN show that it does good work when discovering clusters with different shapes and different size, but it cannot discover clusters with different densities.

DAAP introduces density-adaptive similarity measurement, making it adaptive to the manifold structures, and uses K-AP to remedy the disadvantages of AP algorithm. However, the results in Fig. 4 of Supplementary Materials show that it still cannot handle the data sets with complex patterns, such as D6, T4 and E6. Additionally, the running time listed in Table 2 tells us that DAAP has high computational complexity and spends much more time than other algorithms.

For DP algorithm, the cutoff distance is set as the 2%-th shortest distance, which is suggested by [25], so that the

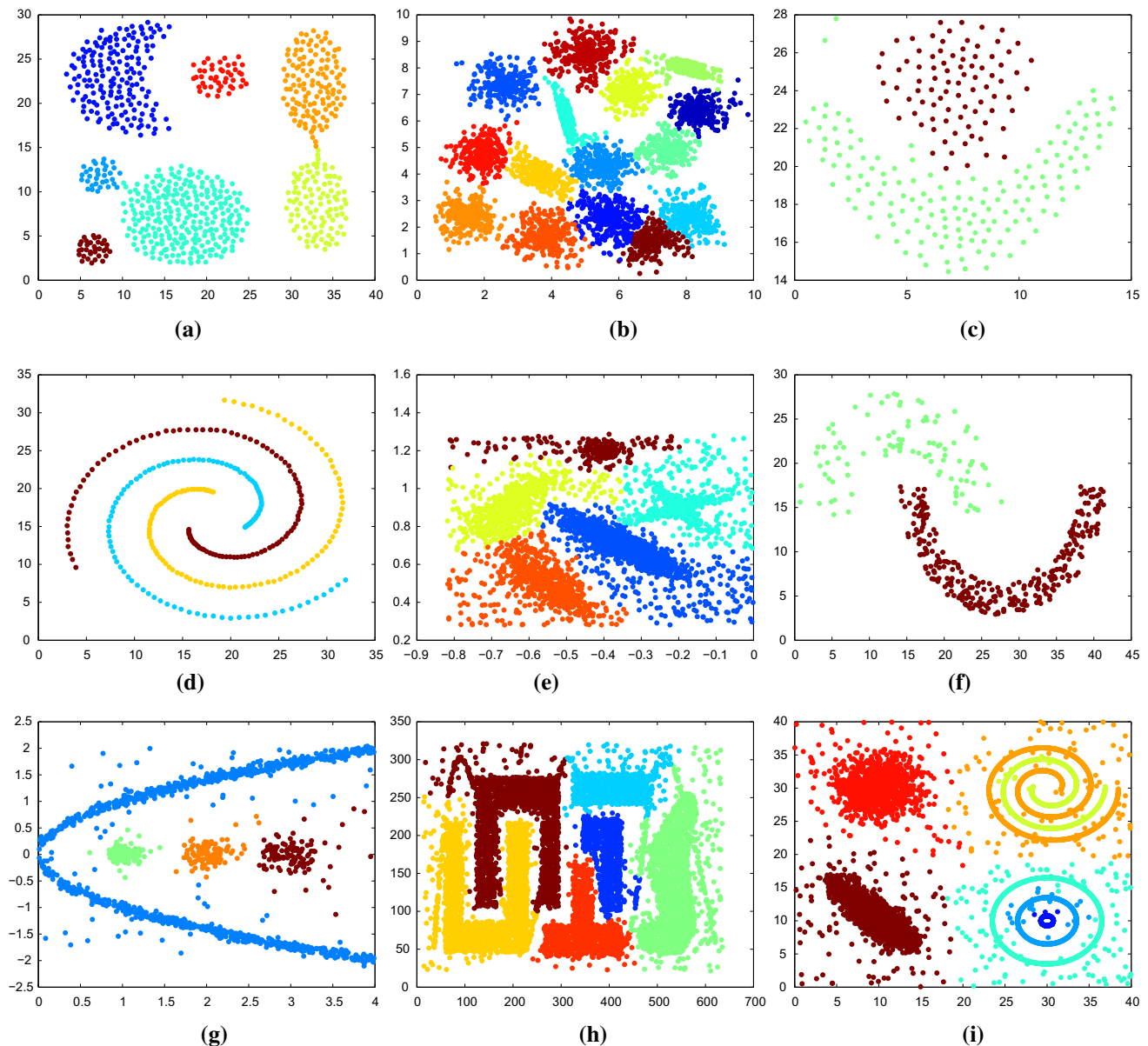
average number of neighbors is around 1–2%, and we adopt the exponential kernel to compute the density. The results in Fig. 5 of Supplementary Materials demonstrate that DP is effective when discovering spherical clusters and has a certain capacity to cluster non-spherical data sets. However, when processing data sets with complex structures, like data sets D6, T4 and E6, DP cannot select the right cluster centers through decision graph.

The results of HCLORE are presented in Fig. 6. HCLORE divides the data set according to the local cores, instead of optimizing an objective function through iteration. The temporary removal of points with lower local density makes HCLORE robust against noise points. The merging process helps HCLORE discover clusters with complex structures. The ACC and NMI scores in Table 1 of HCLORE for most data sets are higher than other methods. The bold values are the best results. As for S2 and Figure2b, the results of HCLORE are close to the best ones. The results prove that HCLORE algorithm is robust to noise points and more powerful when processing data sets with complex structures than other clustering algorithms. Besides, the running time of HCLORE in Table 2 is much less than that of Chameleon, DBSCAN and DAAP algorithm.

## 5.2 Clustering on real data sets from UCI

To further demonstrate the effectiveness of our algorithm, we compare our algorithm with  $K$ -means, Chameleon, DBSCAN, DAAP and DP on several benchmarking real data sets from UCI Machine Learning Repository, which include Iris, Wine, Control, Segment, Pageblocks, Pendigits and Letter. The details of these data sets are shown in Table 3.

The comparison of ACC and NMI scores is illustrated in Table 4. The best results are shown in bold. Iris and Wine are both composed of 3 clusters, and there are two clusters heavily overlapped for Iris. The results show that the ACC and NMI scores of HCLORE are both higher than  $K$ -means, Chameleon, DBSCAN, DAAP and DP algorithms. Control and Segment are both high-dimensional data sets. The ACC scores of HCLORE are much better than  $K$ -means, Chameleon, DBSCAN, DAAP and DP algorithms, and the NMI scores of HCLORE ranked in the second place only after Chameleon and are much better than  $K$ -means, DBSCAN, DAAP and DP algorithms. Pageblocks contain 5473 instances and 5 clusters. One of the clusters has 4913 instances, which makes the NMI scores lower than other data sets. HCLORE obtains the highest ACC and NMI scores for Pageblocks, and the clustering accuracy of HCLORE reaches 0.904. Pendigits and Letter both contain manifold clusters. The ACC and NMI scores of HCLORE are higher than other algorithms. The experimental results



**Fig. 6** Clustering results of HCLORE. With the analysis of the density increasing curve, the proportion points with lower local density are set as 0, 0.1, 0.1, 0, 0.15, 0, 0.05, 0.1, 0.05 for each data

set. The desired cluster number is set as the real cluster number. **a** Aggregation, **b** S2, **c** Flame, **d** Spiral, **e** Figure2b, **f** Jain, **g** D6, **h** T4, **i** E6

show that HCLORE outperforms *K*-means, Chameleon, DBSCAN, DAAP and DP algorithms.

### 5.3 Clustering on Olivetti Face Database

We also test the performance of HCLORE by comparing with other algorithms on Olivetti Face Database [27]. The Olivetti Face Database contains 400 face images from 40 persons, taken at different times and varying the lighting, facial expressions and facial details. The size of each image is  $92 \times 112$  pixels. We use the first 100 images, that is, 10 clusters of the database to do the experiment. The

similarity between two images, denoted as  $S(A, B)$ , is computed by the following equation.

$$S(A, B) = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{(\sum_m \sum_n (A_{mn} - \bar{A})^2)(\sum_m \sum_n (B_{mn} - \bar{B})^2)}} \quad (9)$$

where  $A$  and  $B$  are the subjects of Olivetti Face Database.  $A_{mn}$  and  $B_{mn}$  represent the pixels of the two images. The value of  $S$  is scaled to  $[0, 1]$ . The bigger the value of  $S$  is, the more similar the two images are. The distance between two images is denoted as  $d(A, B)$  and computed as follows.



**Table 1** Comparison of clustering algorithms on synthetic data sets

|             |     |  | <i>K</i> -means | Chameleon    | DBSCAN       | DAAP         | DP           | HCLORE       |
|-------------|-----|--|-----------------|--------------|--------------|--------------|--------------|--------------|
| Aggregation | ACC |  | 0.797           | 0.997        | 0.992        | 0.940        | <b>1.000</b> | <b>1.000</b> |
|             | NMI |  | 0.846           | 0.992        | 0.983        | 0.945        | <b>1.000</b> | <b>1.000</b> |
| S2          | ACC |  | 0.877           | 0.831        | 0.774        | <b>0.970</b> | 0.965        | 0.964        |
|             | NMI |  | 0.909           | 0.905        | 0.765        | <b>0.945</b> | 0.939        | 0.940        |
| Flame       | ACC |  | 0.829           | 0.646        | 0.913        | 0.867        | 0.788        | <b>0.983</b> |
|             | NMI |  | 0.394           | 0.048        | 0.765        | 0.544        | 0.413        | <b>0.872</b> |
| Spiral      | ACC |  | 0.363           | <b>1.000</b> | <b>1.000</b> | <b>1.000</b> | <b>1.000</b> | <b>1.000</b> |
|             | NMI |  | 0.001           | <b>1.000</b> | <b>1.000</b> | <b>1.000</b> | <b>1.000</b> | <b>1.000</b> |
| Figure2b    | ACC |  | 0.931           | 0.583        | 0.795        | 0.961        | <b>1.000</b> | 0.990        |
|             | NMI |  | 0.831           | 0.472        | 0.725        | 0.887        | <b>1.000</b> | 0.956        |
| Jain        | ACC |  | 0.788           | <b>1.000</b> | 0.807        | <b>1.000</b> | 0.861        | <b>1.000</b> |
|             | NMI |  | 0.374           | <b>1.000</b> | 0.261        | <b>1.000</b> | 0.507        | <b>1.000</b> |
| D6          | ACC |  | 0.371           | 0.702        | 0.960        | 0.411        | 0.384        | <b>1.000</b> |
|             | NMI |  | 0.183           | 0.035        | 0.888        | 0.479        | 0.277        | <b>1.000</b> |
| T4          | ACC |  | 0.639           | 0.675        | 0.898        | 0.803        | 0.782        | <b>1.000</b> |
|             | NMI |  | 0.614           | 0.779        | 0.852        | 0.748        | 0.843        | <b>1.000</b> |
| E6          | ACC |  | 0.571           | 0.562        | 0.792        | 0.619        | 0.734        | <b>1.000</b> |
|             | NMI |  | 0.704           | 0.657        | 0.859        | 0.642        | 0.797        | <b>0.999</b> |

**Table 2** Running time of the six methods on synthetic data sets (s)

|             |  | <i>K</i> -means | Chameleon | DBSCAN  | DAAP   | DP     | HCLORE |
|-------------|--|-----------------|-----------|---------|--------|--------|--------|
| Aggregation |  | 0.003           | 84.511    | 34.766  | 185.75 | 0.648  | 1.658  |
| S2          |  | 0.056           | 482.291   | 131.957 | 22135  | 18.182 | 22.341 |
| Flame       |  | 0.003           | 26.837    | 4.913   | 22.328 | 0.122  | 0.391  |
| Spiral      |  | 0.006           | 29.039    | 7.870   | 28.287 | 0.165  | 0.552  |
| Figure2b    |  | 0.066           | 456.641   | 262.200 | 17323  | 10.929 | 25.338 |
| Jain        |  | 0.004           | 36.501    | 27.768  | 62.548 | 0.184  | 0.644  |
| D6          |  | 0.027           | 209.780   | 165.741 | 1217.6 | 1.465  | 3.875  |
| T4          |  | 0.008           | 689.672   | 441.500 | 41997  | 45.312 | 73.482 |
| E6          |  | 0.094           | 777.708   | 497.098 | 51788  | 49.189 | 94.590 |

**Table 3** Real data sets from UCI

| Data sets  | Instances | Attributes | Clusters |
|------------|-----------|------------|----------|
| Iris       | 150       | 4          | 3        |
| Wine       | 178       | 13         | 3        |
| Control    | 600       | 60         | 6        |
| Segment    | 2310      | 19         | 7        |
| Pageblocks | 5473      | 10         | 5        |
| Pendigits  | 7494      | 16         | 10       |
| Letter     | 20000     | 16         | 26       |

$$d(A, B) = 1 - S(A, B) \quad (10)$$

For such a small data set, we only take the nearest neighbor of each point as the local neighbor in HCLORE algorithm. Figure 7 shows clustering results of HCLORE. Only two images enclosed with red box are assigned to the wrong

cluster. The results of *K*-means, Chameleon, DBSCAN, DAAP and DP are displayed in Fig. 6–Fig. 10 of Supplementary Materials, respectively. In DP algorithm, the density is estimated by a Gaussian kernel with variance and *dc* is set to 0.07 and we do not consider finding noises. In DBSCAN algorithm, the radius  $\epsilon$  is set to 0.2 and *minpts* is 2. DBSCAN detects 12 clusters. DAAP discovers 10 clusters. To analyze the performance of *K*-means, Chameleon, DBSCAN, DAAP, DP and HCLORE, we also consider ACC and NMI scores as the measurement standards. The results are shown in Table 5. The best results are shown in bold. The scores of ACC and NMI of HCLORE are higher than other algorithms.

HCLORE first partitions the data sets by finding local cores and then repeatedly merges most similar clusters until a given number of clusters number are obtained. The temporary removal points with lower local density make the boundary between clusters clearer and avoid the effect



**Table 4** Performance comparison of the algorithms on real data sets

| Data sets  |     | <i>K</i> -means | Chameleon    | DBSCAN | DAAP  | DP    | HCLORE       |
|------------|-----|-----------------|--------------|--------|-------|-------|--------------|
| Iris       | ACC | 0.887           | 0.693        | 0.667  | 0.867 | 0.907 | <b>0.960</b> |
|            | NMI | 0.742           | 0.723        | 0.761  | 0.700 | 0.806 | <b>0.871</b> |
| Wine       | ACC | 0.944           | 0.382        | 0.691  | 0.933 | 0.882 | <b>0.955</b> |
|            | NMI | 0.816           | 0.040        | 0.527  | 0.812 | 0.710 | <b>0.848</b> |
| Control    | ACC | 0.582           | 0.608        | 0.285  | 0.302 | 0.557 | <b>0.820</b> |
|            | NMI | 0.709           | <b>0.819</b> | 0.094  | 0.080 | 0.746 | 0.803        |
| Segment    | ACC | 0.481           | 0.532        | 0.530  | 0.361 | 0.520 | <b>0.609</b> |
|            | NMI | 0.461           | <b>0.696</b> | 0.591  | 0.349 | 0.511 | 0.629        |
| Pageblocks | ACC | 0.711           | 0.672        | 0.790  | 0.302 | 0.899 | <b>0.904</b> |
|            | NMI | 0.049           | 0.164        | 0.093  | 0.111 | 0.110 | <b>0.318</b> |
| Pendigits  | ACC | 0.689           | 0.699        | 0.513  | 0.704 | 0.766 | <b>0.782</b> |
|            | NMI | 0.682           | 0.775        | 0.664  | 0.659 | 0.765 | <b>0.776</b> |
| Letter     | ACC | 0.336           | 0.212        | 0.140  | 0.355 | 0.325 | <b>0.575</b> |
|            | NMI | 0.302           | 0.137        | 0.191  | 0.362 | 0.282 | <b>0.480</b> |

**Fig. 7** Clustering results of HCLORE on the Olivetti Face Database. Faces with the same color belong to the same cluster. The faces enclosed by red box are assigned to the wrong cluster (color figure online)

of noise points. The hybrid process helps HCLORE discover clusters with complex structures. Through above experiments on synthetic data sets and real data sets, it is obvious that the results of HCLORE outperform *K*-means, Chameleon, DBSCAN, DAAP and DP algorithms, especially when processing data sets with complex structures.

#### 5.4 The selection of density threshold

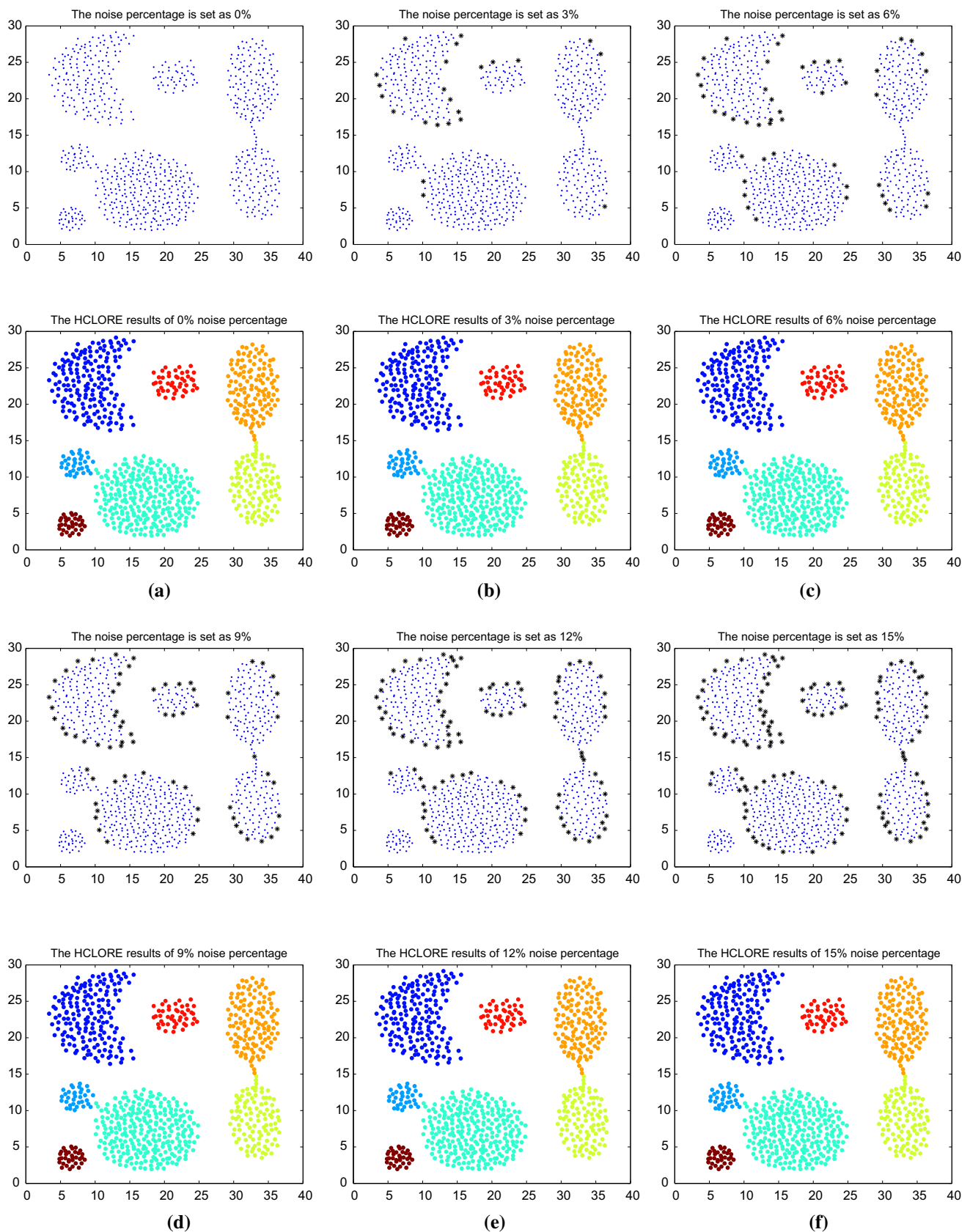
In Sect. 4.3, we describe two schemes to determine the density threshold. We do the following experiment to demonstrate that HCLORE is insensitive to the density threshold.

As for the first scheme, we do experiments on two synthetic data sets Aggregation and E6. (Aggregation does not contain noise points, and E6 contains a number of noise points.) We set the value of  $\alpha$  changing from 0 to 15. The experimental results of data sets Aggregation and E6 are, respectively, shown in Figs. 8 and 9. From the results, we can see that HCLORE can obtain the correct clustering results with different values of  $\alpha$ .

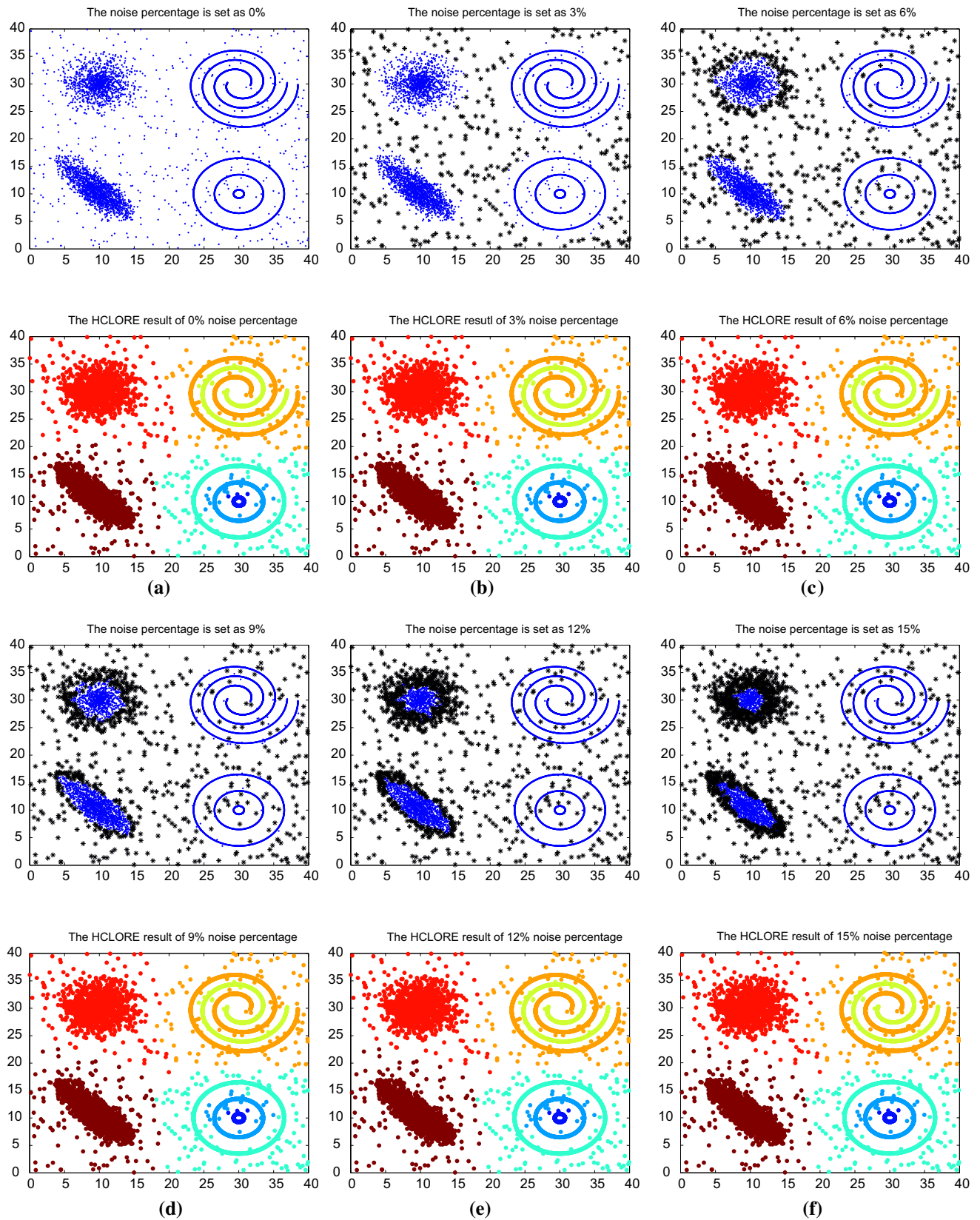
Figure 10 shows the second scheme to determine the noise density threshold of data sets Aggregation and E6. From the density increasing curves, we can see that as for the first 15% points, there is a mutation on the curve of E6, but not on that of Aggregation, which indicates that the

**Table 5** Comparison of ACC and NMI on the Olivetti Face Database

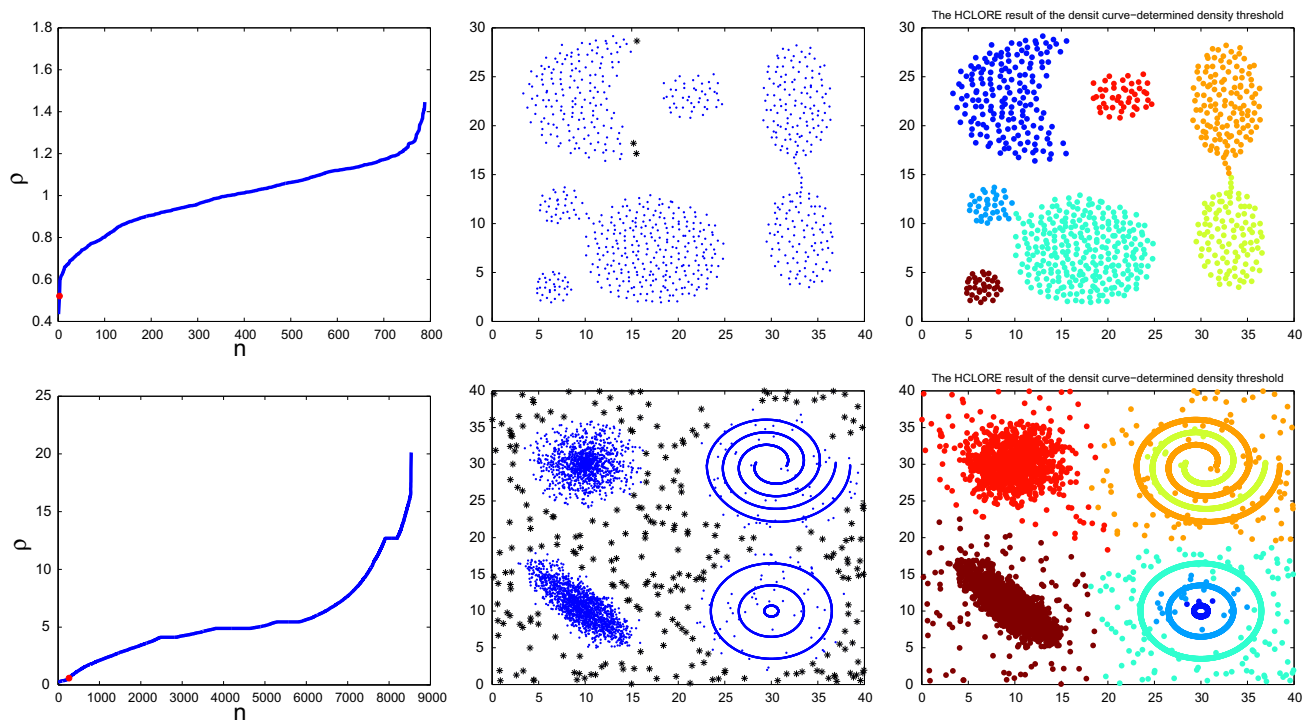
|     | <i>K</i> -means | Chameleon | DBSCAN | DAAP | DP   | HCLORE      |
|-----|-----------------|-----------|--------|------|------|-------------|
| ACC | 0.66            | 0.43      | 0.76   | 0.61 | 0.78 | <b>0.98</b> |
| NMI | 0.71            | 0.68      | 0.82   | 0.68 | 0.85 | <b>0.97</b> |



**Fig. 8** Robustness experiment with respect to  $\alpha$  on Aggregation. The black star points are noise points. **a**  $\alpha = 0\%$ , **b**  $\alpha = 3\%$ , **c**  $\alpha = 6\%$ , **d**  $\alpha = 9\%$ , **e**  $\alpha = 12\%$ , **f**  $\alpha = 15\%$



**Fig. 9** Robustness experiment with respect to  $\alpha$  on E6. The black star points are noise points. **a**  $\alpha = 0\%$ , **b**  $\alpha = 3\%$ , **c**  $\alpha = 6\%$ , **d**  $\alpha = 9\%$ , **e**  $\alpha = 12\%$ , **f**  $\alpha = 15\%$



**Fig. 10** HCLORE results of density curve-determined threshold. The red point in the density increasing curve is the threshold we determined. Since there is no mutation on the density increasing

density increasing curve can help us analyze the noise points in a data set. As for some data sets, although the mutation between noises and ordinary points is not so obvious, it still provides a good basis for determining the density threshold.

## 6 Conclusions

Inspired by the reality that when recognizing clusters from complex structures, humans tend to discover obvious clusters in dense regions first and then deal with potentially ambiguous objects with lower local density, we propose a new hierarchical clustering algorithm HCLORE in this paper. The data set is first well divided into clusters by finding local cores, instead of continuous iteration to optimize the objective function like *K*-means. Local cores are points with local maximum density in the local neighbors. Then, the points with lower local density are removed temporarily, so that the boundary between clusters is clearer, making HCLORE insensitive to noise points. After that, clusters are merged according to the similarity between clusters, which helps HCLORE discover clusters with complex structures. Finally, the lower local density points are assigned to the clusters their local cores belong to. The experimental results on both synthetic data sets and real data sets demonstrate that compared with other methods,

curve of Aggregation, the threshold we determined is smaller than any points (color figure online)

HCLORE algorithm is significantly more effective and efficient when discovering clusters with complex structures.

**Acknowledgements** This work is supported by National Nature Science Foundation of China (61502060, 61702060), Project of Chongqing Education Commission (KJZH17104) and Science and Technology Project of Chongqing Municipal Education Commission (KJ15012014).

## Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

1. Bouguettaya A, Yu Q, Liu XM, Zhou XM, Song A (2015) Efficient agglomerative hierarchical clustering. *Expert Syst Appl* 42(5):2785–2797
2. Chang H, Yeung DY (2008) Robust path-based spectral clustering. *Pattern Recognit* 41(1):191–203
3. Chen WY, Song YQ, Bai HJ, Lin CJ, Chang EY (2011) Parallel spectral clustering in distributed systems. *IEEE Trans Pattern Anal Mach Intell* 33(3):568–586
4. Chen Y, Tang S, Zhou L, Wang C, Du J, Wang T, Pei S (2016) Decentralized clustering by finding loose and distributed density cores. *Inf Sci* (in press)
5. Cheng D, Zhu Q, Huang J, Yang L, Wu Q (2017) Natural neighbor-based clustering algorithm with local representatives. *Knowl-Based Syst* 123(C):238–253



6. Du MJ, Ding SF, Jia HJ (2016) Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowl-Based Syst* 99:135–145
7. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD* 96:226–231
8. Frey BJ, Dueck D (2007) Clustering by passing messages between data points. *Science* 315(5814):972–976
9. Fu LM, Medico E (2007) Flame, a novel fuzzy clustering method for the analysis of dna microarray data. *BMC Bioinform* 8:3
10. Gionis A, Mannila H (2007) Clustering aggregation. *ACM Trans Knowl Discov Data* 1(1):1–30
11. Ha J, Seok S, Lee JS (2014) Robust outlier detection using the instability factor. *Knowl-Based Syst* 63:15–23
12. Huang J, Zhu Q, Yang L, Cheng D, Wu Q (2017) Qcc: a novel clustering algorithm based on quasi-cluster centers. *Mach Learn* 106(3):337–357
13. Jain AK (2010) Data clustering: 50 years beyond k-means. *Pattern Recognit Lett* 31(8):651–666
14. Jain AK, Law MHC (2005) Data clustering: a user's dilemma. *Pattern Recognit Mach Intell Proc* 3776:1–10
15. Jia HJ, Ding SF, Meng LH, Fan SY (2014) A density-adaptive affinity propagation clustering algorithm based on spectral dimension reduction. *Neural Comput Appl* 25(7–8):1557–1567
16. Karypis G, Han EH, Kumar V (1999) Chameleon: hierarchical clustering using dynamic modeling. *Computer* 32(8):68–75
17. King B (1967) Step-wise clustering procedures. *J Am Stat Assoc* 62(317):86–101
18. Liang Z, Chen P (2016) Delta-density based clustering with a divide-and-conquer strategy: 3dc clustering. *Pattern Recognit Lett* 73:52–59
19. Lin CR, Chen MS (2005) Combining partitional and hierarchical algorithms for robust and efficient data clustering with cohesion self-merging. *IEEE Trans Knowl Data Eng* 17(2):145–159
20. von Luxburg U (2007) A tutorial on spectral clustering. *Stat Comput* 17(4):395–416
21. Lv Y, Ma T, Tang M, Cao J, Tian Y, Al-Dhelaan A, Al-Rodhaan M (2016) An efficient and scalable density-based clustering algorithm for datasets with complex structures. *Neurocomputing* 171(C):9–22
22. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol 1. Oakland, CA, USA, pp 281–297
23. Moss WW, Hendrick JA (1973) Numerical taxonomy. *Ann Rev Entomol* 18:227–258
24. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: algorithms and complexity. Courier Corporation, North Chelmsford
25. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. *Science* 344(6191):1492–1496
26. Rousseeuw PJ (2009) Finding groups in data: an introduction to cluster analysis. Wiley, London
27. Samaria FS, Hater AC (2014) Parameterisation of a stochastic model for human face identification. In: *Proceedings of the second IEEE workshop on applications of computer vision*. IEEE, pp 138–142
28. Shao JM, He X, Bohm C, Yang QL, Plant C (2013) Synchronization-inspired partitioning and hierarchical clustering. *IEEE Trans Knowl Data Eng* 25(4):893–905
29. Wang G, Son Q (2016) Automatic clustering via outward statistical testing on density metrics. *IEEE Trans Knowl Data Eng* 28(8):1971–1985
30. Schölkopf B, Platt J, Hofmann T (2007) A local learning approach for clustering. In: *Advances in neural information processing systems 19: Proceedings of the 2006 conference*. MIT Press, pp 1529–1536. ISBN:9780262256919. <https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6287388>
31. Xie JY, Gao HC, Xie WX, Liu XH, Grant PW (2016) Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors. *Inf Sci* 354:19–40
32. Xu J, Wang G, Deng W (2016) Denpehc: density peak based efficient hierarchical clustering. *Inf Sci* 373:200–218
33. Zhang H, Chow TWS, Wu QMJ (2016) Organizing books and authors by multilayer som. *IEEE Trans Neural Netw Learn Syst* 27(12):2537–2550. <https://doi.org/10.1109/TNNLS.2015.2496281>
34. Zhang H, Wang S, Xu X, Chow TWS, Wu QMJ (2018) Tree2vector: learning a vectorial representation for tree-structured data. *IEEE Trans Neural Netw Learn Syst*. <https://doi.org/10.1109/TNNLS.2018.2797060>
35. Zhang T, Ramakrishnan R, Livny M (1996) Birch: an efficient data clustering method for very large databases. In: *ACM SIGMOD record*, vol 25. ACM, pp 103–114
36. Zhang X, Wang W, Norvag K, Sebag M (2010) K-ap: generating specified k clusters by efficient affinity propagation. In: *2010 IEEE 10th international conference on data mining (ICDM)*. IEEE, pp 1187–1192
37. Zhu QS, Feng J, Huang JL (2016) Natural neighbor: a self-adaptive neighborhood method without parameter k. *Pattern Recognit Lett* 80:30–36