



# A hierarchical clustering algorithm based on noise removal

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: 20 September 2017 / Accepted: 31 May 2018 / Published online: 29 June 2018  
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

## Abstract

Noise is irrelevant or meaningless data and hinders most types of data analysis. The existing clustering algorithms seldom take the noise points into consideration and cannot detect arbitrary-shaped clusters. This paper presents a Hierarchical Clustering algorithm Based on Noise Removal (HCBNR). It is robust against noise points and good at discovering clusters with arbitrary shapes. In this work, natural neighbor-based density is applied to remove noise points in a data set firstly. Then we construct a saturated neighbor graph on the rest points, and a novel modularity-based graph partitioning algorithm is used to divide the graph into small clusters. Finally, the small clusters are repeatedly merged according to a novel similarity metric between clusters until the desired cluster number is obtained. The experimental results on synthetic data sets and real data sets show that our method can accurately identify noise points and obtain better clustering results than existing clustering algorithms when discovering arbitrary-shaped clusters.

**Keywords** Hierarchical clustering · Natural neighbor · Noise removal

## 1 Introduction

Clustering [10] is one of the most important methods of data mining. Intuitively, it groups a set of data in a way that maximizes the similarity within a cluster and minimizes the similarity between different clusters. The discovered clusters help to explain the hidden features of a data set. Clustering has been widely used in statistics, computer science, biology, social science and psychology.

Generally, existing clustering algorithms can be roughly divided into several types according to the clustering strategies, such as partitioning clustering, density-based clustering and hierarchical clustering.

Partitioning methods use an iterative control strategy to optimize an objective function, such as K-means [16] and K-medoids, among which partitioning around medoids (PAM) [13] is known to be the most powerful. However, the choice of initial cluster centers tends to affect clustering results. Since a point is always assigned to the nearest center, these approaches are not able to detect non-spherical clusters. Therefore, researchers have proposed many improved algorithms. In [5], the authors devised a method called affinity propagation, which takes as input measures of similarity between pairs of data points. Real-valued messages are exchanged between data points until a high-quality set of exemplars and corresponding clusters gradually emerge. QCC [9] thinks that the density of a cluster center should be the highest in its  $k$  nearest neighbors or reverse  $k$  nearest neighbors. a novel clustering algorithm [20] (denoted as DP for convenience in this paper) assumes 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. However, these algorithms still can not process data sets containing clusters with arbitrary shapes.

---

✉ 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

Density-based clustering assumes that clusters are dense regions separated by sparse regions. DBSCAN [4] is a representative algorithm of density-based clustering. The high density data set can be effectively divided into clusters, and arbitrary-shaped clusters can be found in the spatial data set with noises. However, DBSCAN cannot be applied to high-dimensional data with large variations in density and choosing appropriate parameters can be nontrivial. As for problems in DBSCAN algorithm, an efficient and scalable density-based clustering algorithm [15] was proposed.

Hierarchical clustering seeks to build a hierarchy of clusters. CURE [7] and Chameleon [12] are important hierarchical clustering algorithms. CURE selects well scattered points from the cluster as representatives, so that it can discover non-spherical clusters and the shrinking helps to dampen the effects of noises. Chameleon can find more natural clusters of various shapes since it measures the similarity of two clusters dynamically considering data distribution within a cluster. However, the results of CURE and Chameleon are very sensitive to parameters. Besides, Chameleon is susceptible to noise points.

Clustering is an important data analysis method. However, when a data set contains amount of noise points, most clustering algorithms tend to become less effective, because noise points may distort the analysis. A practical solution is to eliminate irrelevant or weakly relevant noise points before clustering. In this paper, we propose a Hierarchical Clustering algorithm Based on Noise Removal (HCBNR), which is robust against noise points and good at discovering clusters with arbitrary shapes. First it removes noise points according to the natural neighbor-based density. Then a hierarchical clustering algorithm is used to cluster the remaining points and it consists of three steps: (1) natural neighbor is used to construct a Saturated Neighbor Graph (SNG); (2) the SNG is divided into small clusters with a novel modularity-based graph partition method; (3) the small clusters are repeatedly merged according to the new defined similarity between clusters, until the desired cluster number is obtained.

The rest of this paper is organized as follows. Section 2 is the related work. Section 3 introduces natural neighbor. Section 4 presents the proposed method. Experiments and performance evaluation are shown in Section 5. Finally, we make conclusions on our work and point out the future work in Section 6.

## 2 Related work

DBSCAN [4] can discover clusters with arbitrary shapes. It chooses two parameters (the neighborhood radius  $Eps$  and the minimum number of points  $MinPts$ ) to compute the density threshold, discards points in regions with densities lower than the threshold as noise points, and assigns disconnected

regions of high density to different clusters. To find a cluster, DBSCAN starts with an arbitrary points  $p$  and retrieves all points density-reachable from  $p$  with respect to  $Eps$  and  $MinPts$ . If  $p$  is a core point, all the density-reachable points from  $p$  are classified into the same cluster. If  $p$  is a border point, no points are density-reachable from  $p$  and DBSCAN visits the next point of the data set.

Strategies for hierarchical clustering are generally divided into bottom-up and top-down methods. The former starts with every individual data object as a cluster, and two closest clusters are merged as one moves up the hierarchy, while the latter first assumes that all data objects are in one cluster, and splits are performed recursively as one moves down the hierarchy. BIRCH [27], CURE [7], ROCK [6], and Chameleon [12] are the representative methods. ROCK employs links and not distances when merging clusters, so it can be applied to data sets with categorical attributes. BIRCH first constructs CF-tree, and then deletes sparse clusters and merges dense clusters in leaf nodes. It works well for convex or spherical clusters of uniform size, but it is unsuitable when clusters are non-spherical and have different sizes. CURE and Chameleon identify clusters with non-spherical shapes and wide variances in size.

CURE represents each cluster by a certain fixed number  $n_r$  of points that are generated by selecting well scattered points from cluster and then shrinks them toward the center of the cluster by a specified fraction  $\alpha$ . CURE defines the distance between two clusters as the distance between the closest pair of representatives belonging to different clusters and identifies noises as clusters that are growing very slowly. This is achieved by proceeding with the clustering for several times until the number of clusters decreases below a certain fraction of the initial number of clusters. At this time, clusters with very few points are detected as noises. In [7], the fraction is suggested to be  $1/3$  of the number of points in a data set.

Chameleon produces clusters through three steps. First, it represents data objects as a graph structure. Then, it partitions this graph into initial sub-clusters by using hMetis which is a high-quality graph-partitioning algorithm. Consequently, it obtains the final clustering results by repeatedly merging these initial clusters showing highest similarities by comparing similarities between sub-clusters. However, the quality of clusters produced by Chameleon is significantly affected by its parameters. Besides, the computation of RI (relative interconnectivity) and RC (relative closeness) is sophisticated, making its time complexity high.

Recently, DP algorithm [20] was proposed. The algorithm first computes the local density  $\rho$  and a new defined distance  $\delta$  for each point  $i$ ,  $\delta$  is measured by computing the minimum distance between the point  $i$  and any other points with higher density. For the point with the highest density, the value of  $\delta$  is the maximum distance between it and any other points. According to the definitions, cluster centers

are recognized as points for which the values of  $\delta$  and  $\rho$  are anomalously large. After the cluster centers have been found, each remaining point is assigned to the same cluster as its nearest neighbor of higher density. The algorithm can find clustering results fast, but it fails to detect arbitrary-shaped clusters. As for the problems in DP, some novel algorithms [3, 23, 25] were proposed.

### 3 Preliminaries

Natural neighbor [28] is a new defined neighbor, which is inspired by the friendship of human society that when the relationship between people is stable, the number of one's real friends should be the number of persons who take him or her as friend and he or she takes the person as friend at the same time. In other words, given two points  $x$  and  $y$ , if  $x$  and  $y$  take each other as neighbor at the same time, then  $x$  and  $y$  are natural neighbor. It is conventional that objects lying in the sparse region should possess few neighbors, whereas objects lying in the dense region should possess a great number of neighbors.

The method to obtain natural neighbors is that we continuously expand neighbor searching range, and every time compute the number of each point's reverse neighbors, until the number of points without reverse neighbor does not change.

The whole computation procedure of natural neighbor can be automatically fulfilled without any parameters. The detailed algorithm is elaborated in Algorithm 1, where  $r$  is the searching range,  $nb(y)$  is the number of  $y$ 's neighbors and  $sup_k$  is natural characteristic value. Since KD-tree is introduced into NaN-Searching, the time complexity of Algorithm 1 is  $O(n \log n)$  ( $n$  is the number of objects in a data set). The following is the related definitions.

**Definition 1** (Natural Neighbor) Based on natural neighbor searching algorithm, if point  $x$  is one of the neighbors of point  $y$  and  $y$  is one of the neighbors of point  $x$ , then  $x$  is the natural neighbor of  $y$  and  $y$  is the natural neighbor of  $x$ .

Compared with  $k$  nearest neighbor which has been widely used, the main difference is that natural neighbor does not need to set parameters and that the number of neighbors for each object is not identical. According to natural neighbor searching algorithm, the number of each point  $x$ 's neighbors is  $nb(x)$  and the natural characteristic value is defined as follows.

**Definition 2** (Natural characteristic value  $sup_k$ ) The formula of computing  $sup_k$  is as follows.

$$sup_k = \min\{r | \forall x \exists y (y \neq x \wedge x \in NN_r(y)) \text{ or } \forall x (|RNN_r(x)| = 0) = |RNN_{r-1}(x)| = 0\} \quad (1)$$

where  $NN_r(y)$  is the  $r$ -nearest neighbors of  $y$ .  $RNN_r(x)$  is the reverse nearest neighbors of  $x$ . This formula means that with the increase of  $r$ , the minimum value of  $r$  satisfying one of the following conditions is natural characteristic value  $sup_k$ : (1) all points are considered as neighbors of other points; (2) the number of points without neighbors (i.e.,  $nb(x) = 0$ ) is the same in two successive iterations. Given a data set  $D$  containing  $n$  points, according to the graph theory, we know that  $n \times sup_k = \sum_{x \in D} nb(x)$ , thus,  $sup_k$  is the average number of neighbors.

**Definition 3** (Saturated Neighbor Graph SNG) The graph constructed by linking  $sup_k$  nearest neighbors of each point is called saturated neighbor graph.

---

#### Algorithm 1: NaN-Searching

---

**Input:**  $D$ : the data set  
**Output:**  $sup_k, nb, NN_r$   
 Initializing:  $r=1, nb(i)=0, NN_0(i) = \phi, RNN_0(i) = \phi$ ;  
**while true do**  
   **for each data point  $x$  in  $D$  do**  
     Use KD-tree to find the  $r$ -th neighbor  $y$  of  $x$ ;  
      $nb(y) = nb(y) + 1$ ;  
      $NN_r(x) = NN_{r-1}(x) \cup \{y\}$ ;  
      $RNN_r(y) = RNN_{r-1}(y) \cup \{x\}$ ;  
   **end**  
    $Numb = \text{length}(\text{find}(nb(x)=0))$ ;  
   **if the number  $Numb$  does not change then**  
     Break;  
   **end**  
    $r=r+1$ ;  
**end**  
 $sup_k = r$ ;  
 Output the  $sup_k, nb, NN_r(i)$ ;

---

## 4 The proposed method

In this paper, a new density definition based on natural neighbor is proposed. We employ the new local density to remove noise points in a data set. In order to avoid setting the threshold of the size of sub-clusters, we propose a new modularity-based partitioning method. The basic idea of HCBNR is: (1) obtain the density threshold according to a density increasing curve and the points with lower density than the threshold are marked as noises and removed; (2) construct the SNG on the rest points and the modularity-based graph partitioning method is used to divide SNG into small clusters; (3) merge the most similar clusters until the expected number of clusters is obtained.

### 4.1 Density-based noise removal

The density of a point in dense region is larger than that in sparse region, which 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. The density of a point is inversely proportional to the distance. Thus, the density based on natural neighbor is defined as follows.

**Definition 4** (Natural neighbor-based Density ) The natural neighbor-based density of point  $i$  is computed as the following formula.

$$\rho_i = \frac{sup_k}{\sum_{j \in NN_{sup_k}(i)} dist(i, j)} \quad (2)$$

where  $sup_k$  is the natural characteristic value.  $NN_{sup_k}(i)$  is the  $sup_k$  nearest neighbors of point  $i$ , and  $dist(i, j)$  is the distance between point  $i$  and  $j$ .

In [23], a new concept of density based on  $k$  nearest neighbor is proposed. Density based on natural neighbor employs the similar definition. The difference is that natural neighbor-based density does not need to set parameter  $k$ , and NaN-Searching algorithm is committed to find the appropriate parameter.

Noise data refers to the unexplained variation or randomness that is found within a given data set. If there are a great many noise points in a data set so that they affect the data analysis, then it is necessary to remove the noise points. Traditionally, outlier detection techniques can be used to remove noises. LOF [1] is a density-based outlier detection method by assigning to each object a local outlier factor, for measuring the outlier degree of every object in the data set. In [24], LOF is employed to remove noises before clustering. Several other noise removal techniques have been proposed in [26].

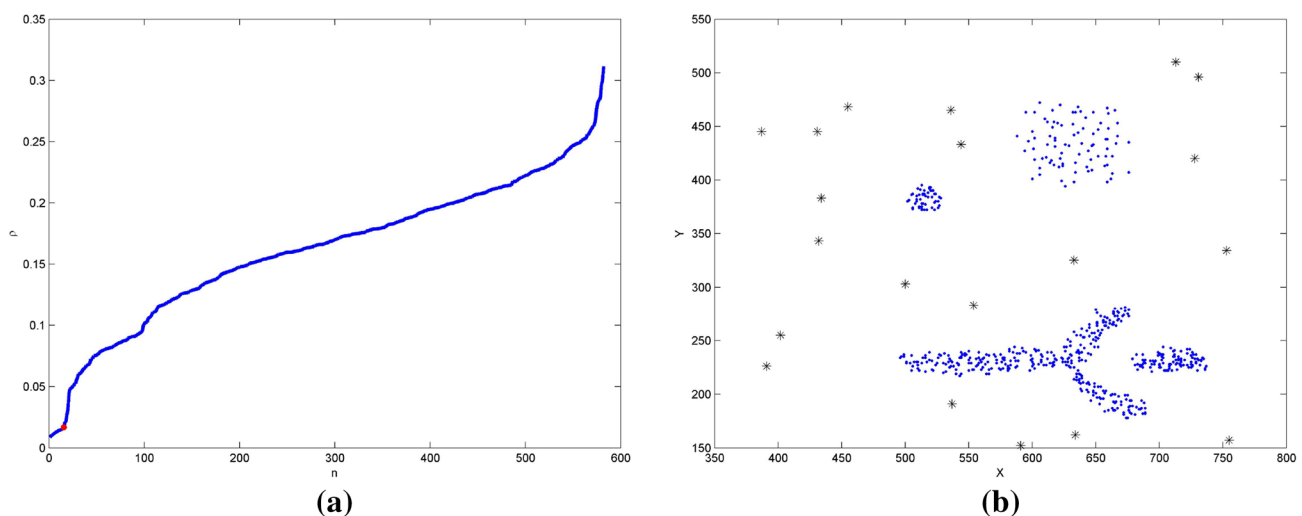
The density of noise points is often smaller than others. Through the analysis of data, we can set a density threshold,

and the points whose density are less than the threshold will be regarded as noises and removed from the data set. In this paper, we use density increasing curve to determine the threshold.

We first sort all the objects in ascending order according to the density, and then draw density increasing curve. According to our hypothesis, the densities of noise points are less than that of the normal points, so that the density curve will mutate at the junction of the noise points and normal points. The user can determine the density threshold according to the density curve. Fig. 1 has shown the method. Fig. 1a is the density increasing curve, here the vertical axis is the density, and the red point is the demarcation between noise points and normal points. Fig. 1b is the corresponding result after distinguishing noise points, and the black star points are noise points. We can also use the density increasing curve to determine the proportion of noise points, which is the ratio of the horizontal axis of the demarcation and the number of points in the data set. Although for some data sets, the demarcation between noise points and normal points is not so obvious, this method still provides a good basis for determining the density threshold.

## 4.2 Modularity-based graph partition method

The graph partitioning method in Chameleon has to set the minimum size of sub-clusters as the end condition of bisection. If the size is set large, then sub-clusters that need to be separated may be not divided, which will lead to incorrect clustering results. If the size is set small, the number of sub-clusters increases, which will affect the efficiency of algorithm. Here, we introduce modularity to obtain the



**Fig. 1** **a** Density increasing curve. **b** The corresponding result. The black star points are noises identified by the density increasing curve

compact sub-clusters and avoid setting the minimum size of sub-clusters.

Modularity [17–19] is inspired by the idea that if the number of edges between groups is significantly less than that we expect by chance and the number within groups is significantly more, it is reasonable to conclude that something interesting is going on. So Modularity is the number of edges falling within groups minus the expected number in an equivalent network with edges placed at random. Modularity is for non-weighted graph, but it has been proved in [18], a weighted graph can be considered as the superposition of several non-weighted graphs. So the weighted graph modularity is defined as follows.

**Definition 5** (Modularity) Suppose a weighted graph  $Graph = (V, E, W)$ . For a particular division of the graph into two groups *subcluster1* and *subcluster2*, for  $\forall v_i, v_j \in V$ , we have

$$S_i = \begin{cases} 1 & \text{if } v_i \in \text{subcluster1} \\ -1 & \text{if } v_i \in \text{subcluster2} \end{cases} \quad (3)$$

If  $v_i, v_j$  are in the same sub-cluster, then  $\frac{1}{2}(S_i S_j + 1) = 1$ , else  $\frac{1}{2}(S_i S_j + 1) = 0$ . The number of edges between  $v_i$  and  $v_j$  is  $w_{ij}$  (for weighted graph, it is the weight of edges between  $v_i$  and  $v_j$ ). At the same time, the expected number of edges between  $v_i$  and  $v_j$  is  $\frac{k_i k_j}{2m}$ , where  $k_i$  and  $k_j$  are the degrees of the points and  $m = \frac{1}{2} \sum_{v_i, v_j \in V} w_{ij}$  is the total weight of edges. We can then express the modularity as

$$Q = \frac{1}{4m} \sum_{v_i, v_j \in V} \left( w_{ij} - \frac{k_i k_j}{2m} \right) (S_i S_j + 1) \quad (4)$$

In Definition 4, modularity is for a particular division that a cluster is divided into two sub-clusters. In fact, the result of clustering includes several sub-clusters, in order to get the maximum modularity, a good idea is to recursively divide cluster into two sub-clusters and compute the additional contribution for modularity  $\Delta Q$  of every partition, if  $\Delta Q \leq 0$ , we stop dividing.  $\Delta Q$  is computed as follows:

$$\Delta Q = \frac{1}{2m} \left( \frac{1}{2} \sum_{v_i, v_j \in V} B_{ij} (S_i S_j + 1) - \sum_{v_i, v_j \in V} B_{ij} \right) \quad (5)$$

where  $B_{ij} = w_{ij} - \frac{k_i k_j}{2m}$ .

The modularity-based graph partitioning method recursively divides one graph into two sub-graphs by minimizing the weight of cut edges until  $\Delta Q$  is less than or equal to zero. The detailed algorithm is elaborated in Algorithm 2, where *Bisection* (*Graph*) is implemented by the method in [11].

---

#### Algorithm 2: M-Partition

---

**Input:** *Graph*: the constructed graph on the data set  
**Output:** *subGraphs*  
 $(\text{subGraph1}, \text{subGraph2}) = \text{Bisection}(\text{Graph});$   
 $\Delta Q = \text{computeDeltaQ}(\text{Graph}, \text{subGraph1}, \text{subGraph2});$   
**if**  $\Delta Q \leq 0$  **then**  
     $\text{subGraphs} = \text{subGraphs} \cup \text{Graph};$   
    **return**;  
**end**  
*M-Partition*(*subGraph1*);  
*M-Partition*(*subGraph2*);

---

M-Partition is a recursive method. Through introducing modularity, we avoid setting the minimum size of sub-clusters and get initial sub-clusters internal closely connected. In each iteration, we need to compute the weight of cut edges. Since the number of edges in SNG is less than  $n \times \text{sup}_k / 2$ , its time complexity is less than  $n \times \text{sup}_k / 2$  (where  $\text{sup}_k$  is a constant value). Therefore, the time complexity of the recursive M-Partition algorithm is  $O(n \log n)$ .

SNG depicts the distribution of a data set. Points in dense region are more closely linked, so there will be more edges. When a closely linked cluster is divided, its modularity will be reduced. So it is reasonable to use  $\Delta Q \leq 0$  as the termination condition. For example, Fig. 2a is the SNG of a data set from [22], and Fig. 2b is the corresponding result of M-partition. It shows that using  $\Delta Q \leq 0$  as the termination condition of the partitioning method correctly divides the SNG into internal closely connected sub-graphs, and each sub-graph becomes a sub-cluster.

### 4.3 Similarity between clusters

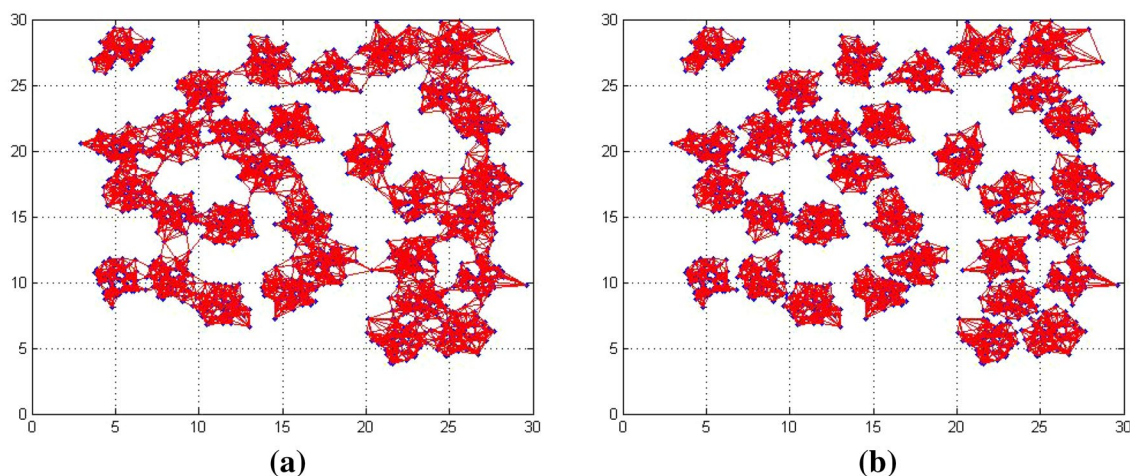
In order to discover clusters with complex structures, a good idea is to divide the data set into many small clusters and merge the most similar clusters. How to evaluate the similarity between clusters is a crucial problem. Different from single-link [21] or complete-link [14] which just takes the shortest or longest distance between clusters into consideration when modeling cluster similarity, we take the connectivity and closeness between sub-clusters into consideration, they are defined as follows.

**Definition 6** (Connectivity) As for cluster  $C_i$  and cluster  $C_j$ , the connectivity is the number of cut edges between  $C_i$  and  $C_j$ . It is computed as:

$$\text{Conn}(C_i, C_j) = |\text{EC}(C_i, C_j)| \quad (6)$$

where  $\text{EC}(C_i, C_j)$  is the set of cut edges.





**Fig. 2** **a** The SNG of a data set. **b** The result of M-Partition

**Definition 7** (Closeness) As for cluster  $C_i$  and cluster  $C_j$ , the closeness is the average weight of cut edges between cluster  $C_i$  and cluster  $C_j$ . It is computed as:

$$Close(C_i, C_j) = \frac{\sum_{e(v_i, v_j) \in EC(C_i, C_j)} w(i, j)}{|EC(C_i, C_j)|} \quad (7)$$

where  $w(i, j)$  is the weight between  $v_i$  and  $v_j$ .

Our algorithm uses a function that combines the connectivity and closeness to compute the similarity between clusters, so the similarity between a pair of clusters  $C_i$  and  $C_j$  is computed as:

$$Sim(C_i, C_j) = Conn(C_i, C_j) \times Close(C_i, C_j)^2 \quad (8)$$

#### 4.4 A hierarchical clustering algorithm based on noise removal

The steps of HCBNR are: (1) compute the natural neighbor-based density and remove noise points; (2) construct a saturated neighbor graph (SNG) for the normal points, search for connected sub-graphs from SNG, and remove sub-graphs with fewer points; (3) if the number of connected sub-graphs is greater than or equal to the expected number of clusters, the sub-graphs are the final clustering results, otherwise, we employ M-Partition to divide the sub-graphs into initial clusters and repeatedly merge initial clusters until the desired number of clusters is obtained. The detailed algorithm of HCBNR is described in Algorithm 3.

#### Algorithm 3: HCBNR

---

**Input:**  $D$ : the data set,  $n_c$ : the expected number of clusters  
**Output:** clusters  
 Initializing:  $D' = \phi$ ;  $clusters = \phi$ ;  
 $(NN_r, sup_k) = NaN\text{-Searching}(D)$ ;  
 $density = \text{computeDen}(NN_r)$ ;  
 Obtain the density threshold  $denThreshold$ ;  
**for each point**  $i$  **in**  $D$  **do**  
   **if**  $density(i) > denThreshold$  **then**  
      $D' = D' \cup i$ ;  
   **end**  
**end**  
 $Graphs = \text{constructSNG}(D')$ ;  
 $n_G = \text{length}(Graphs)$ ;  
**if**  $n_G \geq n_c$  **then**  
    $clusters = Graphs$ ;  
   **return**;  
**end**  
 $subGraphs = \text{M-Partition}(Graphs)$ ;  
 $clusters = subGraphs$ ;  
 $c = \text{length}(clusters)$ ;  
 Compute similarity matrix  $Sim$  according to Formula (8);  
**while**  $c > n_c$  **do**  
    $(i, j) = \arg \max_{1 \leq i, j \leq N} Sim(i, j)$ ;  
    $clu = \text{merger}(C_i, C_j)$ ;  
   Remove  $C_i$  and  $C_j$  from  $clusters$  and add  $clu$  in  $clusters$ ;  
    $c = c - 1$ ;  
   update the similarity matrix  $Sim$ ;  
**end**

---

HCBNR algorithm contains three main components: NaN-Searching, M-Partition and the merge process. The time complexities of NaN-Searching and M-Partition are both  $O(n \log n)$ . Assuming that the number of sub-clusters obtained by M-Partition is  $n_s$  ( $n_s \ll n$ ) and the expected number of clusters is  $n_c$ , since we need to count the cut edges between clusters in each merge process and the number of cut edges is less than  $n \times sup_k / 2$ , the time complexity of the merge process is  $O((n_s - n_c) \times n \times sup_k / 2)$ . Thus the overall time complexity of HCBNR is  $O(n \log n + n_s \times n)$ .

## 5 Experiments and performance evaluation

### 5.1 Evaluation metrics

We use ACC and NMI scores [2] to evaluate the clustering performance. They are defined as follows.

$$ACC = \frac{1}{n} \sum_{i=1}^n \delta(y_i, \text{map}(c_i)) \quad (9)$$

where  $y_i$  is the real cluster label,  $c_i$  is the serial number obtained by clustering, and  $\delta(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$  is a discriminate function.  $ACC \in [0, 1]$ , the larger the value of  $ACC$  is, the better the clustering performance of the algorithm means.

$$NMI = \frac{\sum_{i=1}^k \sum_{j=1}^k n_{ij} \log\left(\frac{n_{ij}}{n_i \cdot n_j}\right)}{\sqrt{\left(\sum_i n_i \log \frac{n_i}{n}\right) \left(\sum_j n_j \log \frac{n_j}{n}\right)}} \quad (10)$$

where  $n$  is the number of points in a data set,  $n_i$  and  $n_j$  denote the number of points in category  $i$  and cluster  $j$ , respectively, and  $n_{ij}$  denotes the number of points in category  $i$  as well as in cluster  $j$ . NMI measures how good the clustering result is, with respect to the ground-truth information.  $NMI=1$  means the clustering result is perfect and  $NMI=0$  means the clustering result is useless. Other value between 1 and 0 measures the quality of the clustering result.

### 5.2 Clustering on synthetic data sets

In order to evaluate the performance of our method, we compare HCBNR with DBSCAN [4], Chameleon [12], CURE [7] and DP in [20] on three synthetic data sets depicted in Fig. 3. Dataset 1, taken from [8], has three ball-shaped

clusters and a manifold cluster. Dataset 2, taken from [20], is drawn from a probability distribution with non-spherical and strongly overlapping peaks and has 5 clusters. There are 9 clusters with different shapes in Dataset 3 which is used in Chameleon algorithm [12]. The configuration of the computer used in our experiment is as follows: processor is Intel Core i5 2.80GHz; memory size is 4 GB. To show the results of clustering algorithms, we visualize each cluster by a different color. The black points in each figure are noise points discovered by the algorithms.

To gain the best effect, we need to adjust the parameters to achieve the best classification accuracy for DBSCAN. The best performances are depicted in the first row of Fig. 4, and the corresponding parameters are set as  $MinPts = 4$  and  $Eps = 0.2$  for Dataset 1,  $MinPts = 3$  and  $Eps = 0.015$  for Dataset 2,  $MinPts = 3$  and  $Eps = 7$  for Dataset 3. For DP, we select the most appropriate cluster centers in the corresponding decision graph, and the second row of Fig. 4 shows the results of DP. In order to better display the result, the noise points are not depicted in Dataset 3. We set cluster number  $n_c=4$  for Dataset 1, 5 for Dataset 2 and 9 for Dataset 3 in Chameleon, CURE and our method. For Chameleon, the  $k$  value for  $k$  nearest neighbor graph is set to 13 for Dataset 1, 8 for Dataset 2 and 11 for Dataset 3. Third row of Fig. 4 is the results of Chameleon. For CURE, the number of representatives  $n_r$  is set to 10 for Dataset 1, 25 for Dataset 2 and 50 for Dataset 3, the shrink factor  $\alpha$  is set to 0.2 for Dataset 1 and Dataset 3 and 0.5 for Dataset 2. The results of CURE are shown in the fourth row of Fig. 4. For our method HCBNR, according to the analysis of density increasing curve, we set the noise percentage to 5% for Dataset 1 and Dataset 3, and 12.5% for Dataset 2, and the clusters discovered by HCBNR are shown as the last row of Fig. 4. The comparison result of ACC and NMI on synthetic data sets is listed in Table 1. The bold values are the best results. The running time of clustering algorithms is shown in Table 2.

DBSCAN discovers clusters and detects the noise points in Dataset 1, but it does not apply to Dataset 2 and 3, which

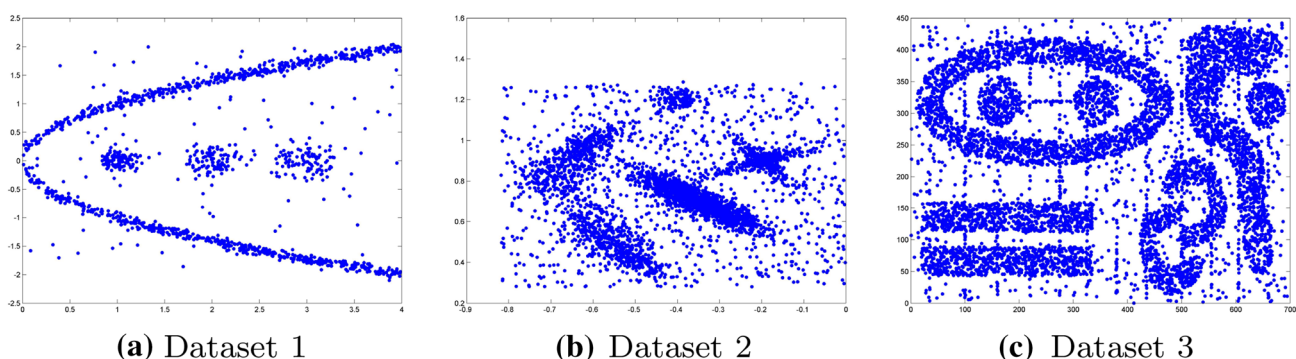
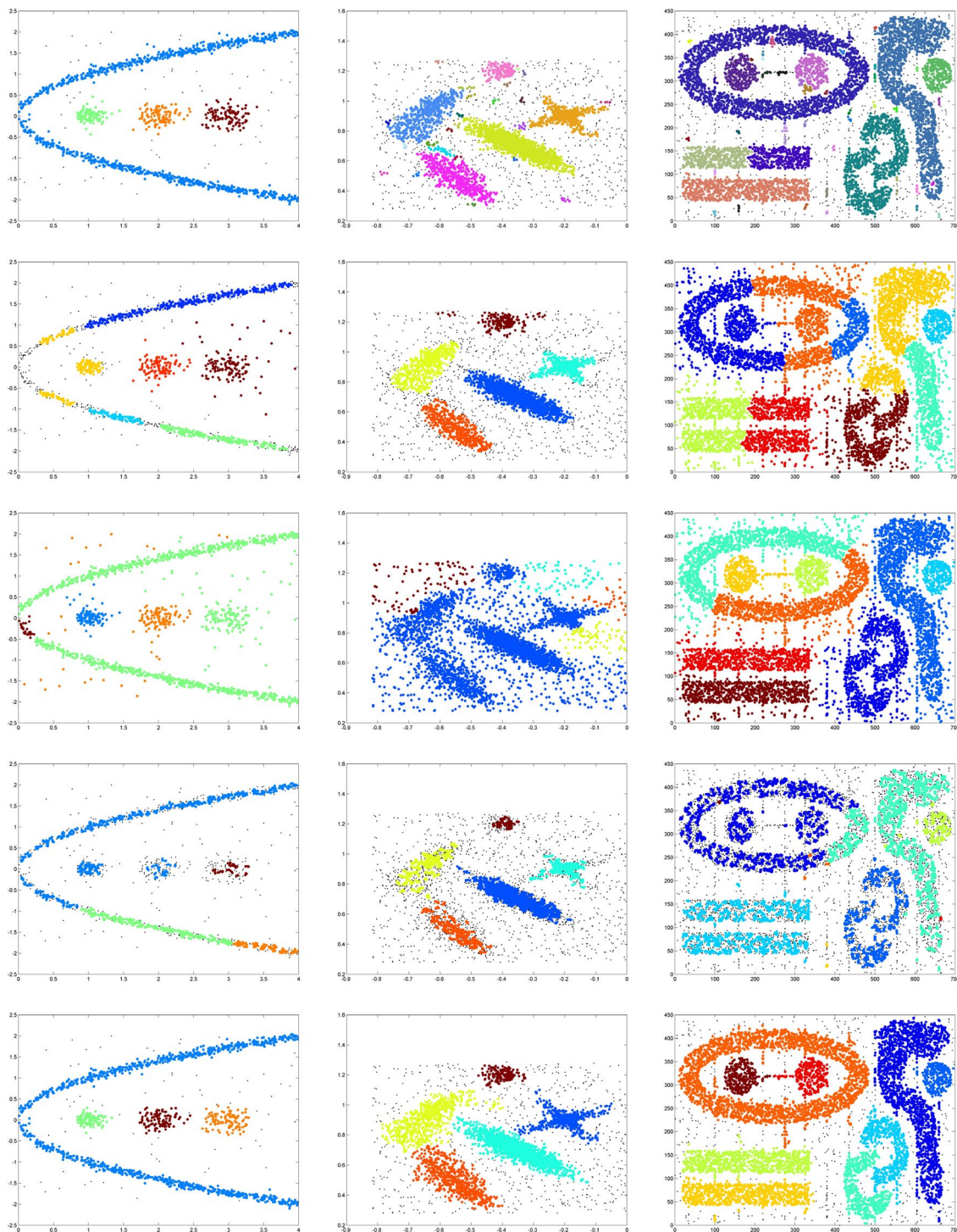


Fig. 3 The three datasets



**Fig. 4** Clustering results of these algorithms. The black points are noise points detected by these algorithms. The first row is the results of DBSCAN, the second row is the results of DP, the third row is the

results of Chameleon, the forth row is the results of CURE and the last row is the results of HCBNR

illustrates that DBSCAN cannot effectively find clusters with different densities. DP is an efficient algorithm, which can fast and correctly find clusters of the second data set, because decision graph apparently exhibits cluster centers.

After clustering, points on the border region of clusters are recognized as noise points. However, as for manifold clusters, it will not produce satisfactory results. The reason is that as for these data sets, it is hard to correctly select cluster



**Table 1** The comparison of ACC and NMI scores on synthetic data sets

Datasets	DBSCAN	DP	Chameleon	CURE	HCBNR
Dataset 1					
ACC	0.949	0.474	0.873	0.426	<b>0.960</b>
NMI	0.861	0.523	0.619	0.206	<b>0.889</b>
Dataset 2					
ACC	0.795	0.780	0.479	0.637	<b>0.857</b>
NMI	0.725	0.723	0.172	0.581	<b>0.784</b>
Dataset 3					
ACC	0.802	0.534	0.783	0.482	<b>0.941</b>
NMI	0.831	0.694	0.895	0.466	<b>0.913</b>

**Table 2** The running time of these methods (s)

Datasets	DBSCAN	DP	Chameleon	CURE	HCBNR
Dataset 1	48.593	1.189	259.296	71.838	0.950
Dataset 2	56.125	5.541	466.006	565.315	19.952
Dataset 3	121.922	20.007	905.688	3172.210	110.691

centers from decision graphs. Therefore, DP does not apply to manifold data sets like Dataset 1 and 3. In Chameleon algorithm, all points are assigned to a certain cluster. However, the clustering results are affected by the noise points. CURE can find clusters in Dataset 2. However, it fails to process data sets with complex patterns, like Dataset 1 and Dataset 3. Besides, CURE incorrectly identifies many normal points as noise points. HCBNR correctly discovers the clusters and noise points for all the data sets. The ACC and NMI scores in Table 1 also demonstrate that HCBNR is much better than other algorithms. From the running time shown in Table 2, DP is the fastest and HCBNR is faster than the other three. In terms of time and clustering performance, HCBNR has more advantages on discovering clusters with arbitrary shapes.

### 5.3 Clustering on real data sets

To further demonstrate the effectiveness of our algorithm, we also compare HCBNR with DBSCAN, DP and Chameleon on several benchmarking real data sets from UCI.

The characteristics of these data sets are shown in Table 3. The expected number of clusters in these data sets are set to the real number of clusters. For DBSCAN algorithm, we adjust the parameter to achieve the best clustering results. For HCBNR algorithm, we determine the noise percentage according to the density increasing curve.

**Table 3** The characteristics of real data sets from UCI

Datasets	Instances	Attributes	Clusters
Iris	150	4	3
Wine	178	13	3
Control	600	60	6
Segment	2310	19	7
Pageblocks	5473	10	5

**Table 4** The comparison of the algorithms on real data sets from UCI

Datasets	Chameleon	DBSCAN	DP	HCBNR
Iris				
ACC	0.693	0.667	0.907	<b>0.960</b>
NMI	0.723	0.761	0.806	<b>0.871</b>
Wine				
ACC	0.382	0.691	0.882	<b>0.955</b>
NMI	0.040	0.527	0.710	<b>0.848</b>
Control				
ACC	0.608	0.285	0.557	<b>0.820</b>
NMI	<b>0.819</b>	0.094	0.746	0.803
Segment				
ACC	0.532	0.530	0.520	<b>0.609</b>
NMI	<b>0.696</b>	0.591	0.511	0.629
Pageblocks				
ACC	0.672	0.790	0.899	<b>0.904</b>
NMI	0.164	0.093	0.110	<b>0.318</b>

The comparison of ACC and NMI is illustrated in Table 4. The best results are shown in bold. From the results, we can learn that accuracy of HCBNR are much higher than other algorithms. The ACC and NMI scores of Chameleon and DBSCAN are not high for most of the data sets. Chameleon is susceptible to noise points and parameters. DBSCAN is robust against noise points, but it still suffers from the problem of parameters selection. The results of DP are better than Chameleon and DBSCAN for it makes good use of decision graph to get cluster centers. However, the density might be affected by large statistical errors, and the clustering process cannot effectively discover clusters with arbitrary shapes. HCBNR gets the best ACC and NMI scores for most data sets. For control and segment, HCBNR gets the second best NMI scores. The clustering results of HCBNR are significantly better than other algorithms.

### 5.4 The discussion on noise removal

In order to demonstrate the effectiveness of our noise removal method, we compare our method with LOF [1]

on synthetic data sets. LOF is a well-known noise removal method, but it has to set parameter  $k$  for  $k$  nearest neighbor. Here,  $k$  is set to the same as that in Chameleon algorithm. Specifically, according to the experiment in Sect. 5.2,  $k$  is set to 13 for Dataset 1, 8 for Dataset 2 and 11 for Dataset 3, and the noise percentage is set to 5% for Dataset 1 and Dataset 3, and 12.5% for Dataset 2. The results are shown in Fig. 5. The running time of LOF and our noise removal method is shown in Table 5. From the results, we can see that the results of Dataset 1 are almost the same. However, for Dataset 2 and Dataset 3, LOF fails to detect most noise points and recognizes some normal points as noise points. Besides, the running time of LOF is far more than that of our method.

We also compare the performance of our clustering algorithm with and without noise removal on synthetic data sets. The original version of HCBNR is with noise removal. We remove the noise removal step in HCBNR and compare this version with the original one on synthetic data sets. The results are shown in Fig. 6 and the ACC and NMI scores are displayed in Table 6. The best results are shown in bold. We can learn from Fig. 6 that

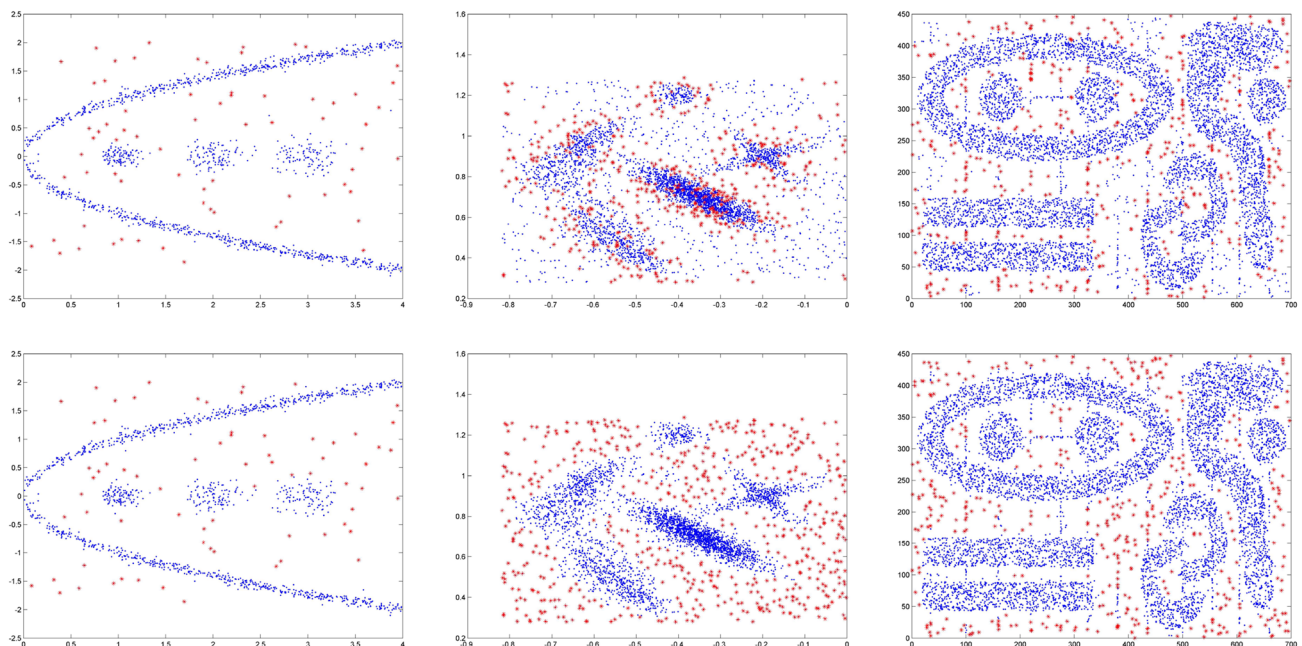
**Table 5** The running time of LOF and our noise removal method(s)

Datasets	LOF	our method
Dataset 1	2.149	0.750
Dataset 2	10.052	5.730
Dataset 3	34.637	24.252

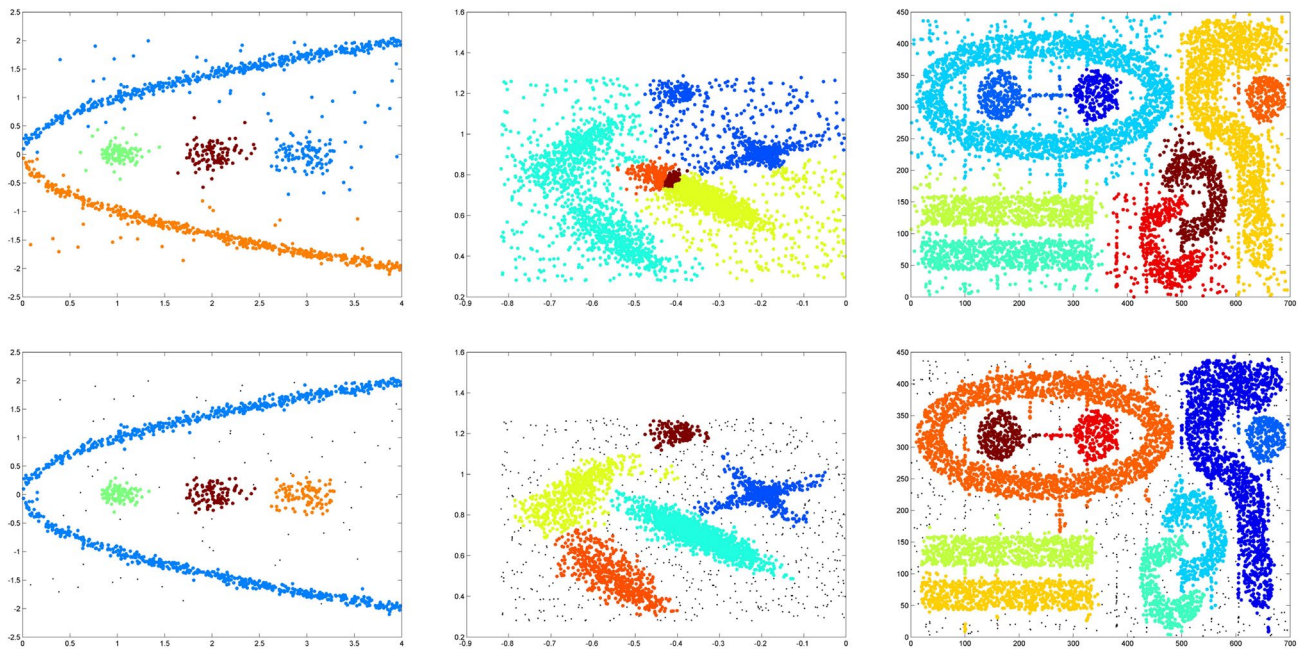
**Table 6** The comparison of HCBNR without and with noise removal

Datasets	Without noise removal	With noise removal
Dataset 1		
ACC	0.614	<b>0.949</b>
NMI	0.599	<b>0.861</b>
Dataset 2		
ACC	0.651	<b>0.857</b>
NMI	0.640	<b>0.784</b>
Dataset 3		
ACC	<b>0.989</b>	0.941
NMI	<b>0.970</b>	0.913

the method without noise removal fails to obtain the correct clusters of Dataset 1 and Dataset 2. Correspondingly, the ACC and NMI scores are smaller than that of with noise removal. Since noise points are considered as one cluster when computing ACC and NMI scores, the ACC and NMI scores of the method without noise removal for Dataset 3 are a little higher than that with noise removal. However, the ACC and NMI scores of both versions are larger than 0.9, which shows that most points are correctly partitioned by both versions. Moreover, the results shown in Fig. 6 also tells that both versions get the correct clusters of Dataset 3. Therefore, we can conclude that with noise removal, HCBNR excludes the interference of noise



**Fig. 5** The comparison of noise removal based on LOF and our method. The red star points are noise points. The first row is the results of LOF and the second row is the results of our method



**Fig. 6** The comparison of HCBNR without and with noise removal. The first row is the result of the method without noise removal and the second row is the results of the method with noise removal

points and holds or even improves the quality of clustering after noise removal.

## 6 Conclusions and future work

The noise points in the data set hinder most types of data analysis and it is quite crucial to remove noise points. In this paper, we propose a novel hierarchical clustering algorithm based on noise removal. First, according to the density curve, points with lower density are marked with noises and removed from the data set. Then a modularity-based graph partition method is employed to divide the data set into small compact clusters without any parameters. We obtain the final result by repeatedly merging the small clusters according to a new defined cluster similarity which improves the efficiency of HCBNR. The results of our experiments demonstrate that HCBNR is significantly better than DBSCAN, DP, Chameleon and CURE when discovering clusters with arbitrary shapes. Therefore, HCBNR has a broader application prospects, and it can be used to process some practical problems, such as data analysis in 3D reconstruction, pattern recognition and image segmentation.

In the future, we are interested in investigating a good way to automatically determine the density threshold of

noise points by regression analysis or other methods and exploring the application of HCBNR on different domains.

**Acknowledgements** This work is supported by National Natural Science Foundation of China, No. 61702060 and No. 61502060, Science and Technology Project of Chongqing Municipal Education Commission, No. KJ15012014 and Project of Chongqing Education Commission, No. KJZH17104.

## References

1. Breunig MM, Kriegel HP, Ng RT, Sander J (2000) Lof: identifying density-based local outliers. *Acm Sigmod Record* 29(2):93–104
2. Chen WY, Song Y, Bai H, Lin CJ, Chang EY (2011) Parallel spectral clustering in distributed systems. *IEEE Trans Pattern Anal Mach Intell* 33(3):568–586
3. Cheng D, Zhu Q, Huang J, Yang L, Wu Q (2017) Natural neighbor-based clustering algorithm with local representatives. *Knowl Based Syst* 123C:238–253
4. Ester M, Kriegel HP, Xu X (1996) A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: *International Conference on Knowledge Discovery and Data Mining*, pp 226–231
5. Frey BJ, Dueck D (2007) Clustering by passing messages between data points. *Science* 315(5814):972–976
6. Guha S, Rastogi R, Shim K (2000) Rock: a robust clustering algorithm for categorical attributes. *Inf Syst* 25(5):345–366
7. Guha S, Rastogi R, Shim K (2001) Cure: an efficient clustering algorithm for large databases. *Inf Syst* 26(1):35–58

8. Ha J, Seok S, Lee JS (2014) Robust outlier detection using the instability factor. *Knowl Based Syst* 63(2):15–23
9. 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
10. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. *Acm Comput Surv* 31(3):264–323
11. Karypis G, Aggarwal R, Kumar V, Shekhar S (2002) Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Trans Very Large Scale Integr Syst* 7(1):69–79
12. Karypis G, Han EH, Kumar V (1999) Chameleon: hierarchical clustering using dynamic modeling. *IEEE Computer Society Press*
13. Kaufman L, Rousseeuw PJ (2009) Finding groups in data: an introduction to cluster analysis. John Wiley, Hoboken
14. King B (1967) Step-wise clustering procedures. *J Am Stat Assoc* 62(317):86–101
15. Lv Y, Ma T, Tang M, Cao J, Tian Y, Al-Rodhaan M (2015) An efficient and scalable density-based clustering algorithm for datasets with complex structures. *Neurocomputing* 171C:9–22
16. Macqueen J (1967) Some methods for classification and analysis of multivariate observations. In: *Proceedings of Berkeley Symposium on Mathematical Statistics and Probability*, pp 281–297
17. Newman ME, Girvan M (2004) Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlinear Soft Matter Phys*. <https://doi.org/10.1103/PhysRevE.69.026113>
18. Newman MEJ (2004) Analysis of weighted networks. *Phys Rev E Stat Nonlinear Soft Matter Phys* 70(5):1–9
19. Newman MEJ (2006) Modularity and community structure in networks. *Proc Natl Acad Sci USA* 103(23):8577–8582
20. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. *Science* 344(6191):1492
21. Sneath PH, Sokal RR (1962) Numerical taxonomy. *Nature* 193:855–860
22. Veenman CJ, Reinders MJT, Backer E (2002) A maximum variance cluster algorithm. *IEEE Trans Pattern Anal Mach Intell* 24(9):1273–1280
23. Wang G, Song Q (2016) Automatic clustering via outward statistical testing on density metrics. *IEEE Trans Knowl Data Eng* 28(8):1971–1985
24. Wang X, Wang XL, Chen C, Wilkes DM (2013) Enhancing minimum spanning tree-based clustering by removing density-based outliers. *Digital Signal Process* 23(5):1523–1538
25. 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
26. Xiong H, Pandey G, Steinbach M, Kumar V (2006) Enhancing data analysis with noise removal. *IEEE Trans Knowl Data Eng* 18(3):304–319
27. Zhang T, Ramakrishnan R, Livny M (1996) Birch: an efficient data clustering method for very large databases. In: *ACM SIGMOD International Conference on Management of Data*, pp 103–114
28. Zhu Q, Feng J, Huang J (2016) Natural neighbor: a self-adaptive neighborhood method without parameter k. *Pattern Recognit Lett* 80:30–36

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.