



An improved probability propagation algorithm for density peak clustering based on natural nearest neighborhood

Wendi Zuo^a, Xinmin Hou^{a,b,c,*}

^a School of Data Science, University of Science and Technology of China, Hefei, Anhui 230026, China

^b School of Mathematical Sciences, University of Science and Technology of China, Hefei, Anhui 230026, China

^c CAS Key Laboratory of Wu Wen-Tsun Mathematics, University of Science and Technology of China, Hefei, Anhui 230026, China

ARTICLE INFO

Keywords:

Clustering

Density peaks

Natural nearest neighborhood

Probability propagation

ABSTRACT

Clustering by fast search and find of density peaks (DPC) (Since, 2014) has been proven to be a promising clustering approach that efficiently discovers the centers of clusters by finding the density peaks. The accuracy of DPC depends on the cutoff distance (d_c), the cluster number (k) and the selection of the centers of clusters. Moreover, the final allocation strategy is sensitive and has poor fault tolerance. The shortcomings above make the algorithm sensitive to parameters and only applicable for some specific datasets. To overcome the limitations of DPC, this paper presents an improved probability propagation algorithm for density peak clustering based on the natural nearest neighborhood (DPC-PPNNN). By introducing the idea of natural nearest neighborhood and probability propagation, DPC-PPNNN realizes the nonparametric clustering process and makes the algorithm applicable for more complex datasets. In experiments on several datasets, DPC-PPNNN is shown to outperform DPC, K-means and DBSCAN.

1. Introduction

Clustering, also known as unsupervised classification, aims to divide datasets into subsets or clusters according to the similarity measure of the data sample (physical or abstract) such that the data samples within the subset or cluster have a high degree of similarity and that the data samples belonging to different subsets or clusters have a high degree of dissimilarity [1]. Currently, cluster analysis plays an important role in many fields such as social sciences, biology, pattern recognition, information retrieval and so on [2]. It is so useful in machine learning and data mining that many researchers have paid much attention to it. Over the past few decades, a number of excellent clustering algorithms have been developed for different types of applications. Typical algorithms include K-means [3] and K-medoids [4] based on partitioning, CURE [5] and BIRCH [6] based on hierarchy, DBSCAN [7] and OPTICS [8] based on density, WaveCluster [9] and STING [10] based on grids and statistical clustering [11] based on models.

In 2014, a density-based clustering algorithm DPC was given by Rodriguez and Laio [12] (Clustering by fast search and find of density peaks, Science, 344 (2014) 1492). The main idea of the algorithm DPC is as follows: For each data point i , we compute two quantities: its local density ρ_i and its distance δ_i from points of higher density. By mapping all the data points to the decision graph which takes ρ and δ as the two axes, we can recognize density peak points (cluster centers)

as points for which the values of ρ_i and δ_i 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 DPC algorithm is simple and efficient, and it can quickly find the high density peak points (cluster centers) without iteratively calculating the objective function. Moreover, it is suitable for cluster analysis on large-scale data. Although the DPC algorithm has obvious advantages over other clustering algorithms, it also has some shortcomings: the accuracy of DPC depends on the cutoff distance (d_c), the cluster number (k) and the selection of the centers of clusters. Moreover, the final allocation strategy is sensitive and has poor fault tolerance. Last, the algorithm has its basis on the assumptions that cluster centers are surrounded by neighbors with lower local density and that they are at a relatively large distance from any point with higher local density.

To avoid the deficiencies of DPC, in this paper, we present an improved probability propagation algorithm for density peak clustering based on natural nearest neighborhood (DPC-PPNNN). By introducing the concept of natural nearest neighborhood, we can avoid setting d_c manually. By calculating $\gamma = \rho \times \delta$ and $\theta = \frac{\rho}{\delta}$, we can select cluster centers automatically. Finally, the clustering algorithm based on probability propagation can help us allocate all the remaining data points. By doing all these, DPC-PPNNN is suitable for more complex datasets and can distinguish two clusters that are close to each other.

* Corresponding author at: School of Data Science, University of Science and Technology of China, Hefei, Anhui 230026, China.

E-mail addresses: zuowendi@mail.ustc.edu.cn (W. Zuo), xmhou@ustc.edu.cn (X. Hou).

The rest of this paper is organized as follows. In Section 2, we introduce the research progress related to the DPC algorithm. In Section 3, we briefly review the basic definitions and processes of the traditional DPC algorithm and reveal some problems within them. In Section 4.1, we introduce the basic concept and the algorithm of the natural nearest neighborhood. In Section 4.2, we show the improvement on the local density and the method of selecting cluster centers automatically. In Section 4.3, we propose the allocation algorithm based on probability propagation. In Section 5, we compare the DPC-PPNNN algorithm with other classical clustering algorithms using various synthetic and real-world datasets. In Section 6, we summarize the advantages and disadvantages of the DPC-PPNNN algorithm and point out the direction of our future research.

2. Related works

DPC has been proven to be an effective clustering strategy but it also has many limitations such as the arbitrary selection of density estimation metrics, clustering centers and the risk of error propagation. Over the past few years, many optimized variants of DPC have been proposed considering the following aspects:

The first aspect is improving the density measure of the DPC algorithm. Du, Ding and Jia [13] proposed DPC-KNN, which introduces the concept of K-nearest neighbors (KNN) to DPC and provides another option for computing the local density. They also employ PCA to reduce the dimensionality of data. However, the method still suffers from the limitations of DPC because it applies the same procedure in determining the cluster centers and assigning the rest of the points. Liu, Wang, and Yu [2] proposed SNN-DPC, which estimates the local density based on the idea of shared nearest neighbor similarity. However, it still suffers from the limitation of manually selecting the number of cluster centers. Mehmood et al. [14] proposed CFSFDP-HD, which applies a nonparametric density estimator (Kernel Density Estimation, KDE) to eliminate the reliance of DPC on the cut-off distance d_c .

The second aspect is to automatically recognize the numbers of clusters and cluster centers. Liang and Chen [15] proposed 3DC, which introduces a divide-and-conquer strategy to determine the ideal number of clusters. However, it ignores the local structure of the datasets which may cause missing clusters. Xu, Wang and Deng [16] proposed DenPEHC, which could automatically detect all possible centers and build a hierarchy presentation for the dataset. Nevertheless, both 3DC and DenPEHC will aggravate the propagation of errors due to the hierarchical clustering strategy. Li, Ge, and Su [17] proposed an automatic clustering algorithm for determining the density of clustering centers. In this algorithm, it is considered that if the shortest distance between a potential cluster center and a known cluster center is less than the cutoff distance d_c , then the potential cluster center is a redundant center. Otherwise, it is regarded as the actual center of another cluster.

The third aspect is to improve the assignment strategy to reduce error propagation. In most of the DPC variants, the idea of K-nearest neighbors is hybridized in the aggregation strategies. For instance, Zhou et al. [18] constructed the veins of clusters by connecting pairs with the highest similarity from the high-density regions to the cluster borders. The rest of the points are then assigned to the nearest veins. Xie et al. [19] proposed a two-step procedure for label propagation. The first strategy assigns non-outliers based on the search of the K-nearest neighbors starting from the density peaks. The second strategy assigns the other points using the fuzzy KNN technique. Geng et al. [20] proposed a KNN graph-based label propagation strategy to assign the remaining points. Liu, Wang and Yu [2] introduced a two-step allocation method based on inevitably and possibly subordinate allocation of the noncenter points.

3. DPC algorithm and analysis

3.1. DPC algorithm

The DPC algorithm presented by Rodriguez and Laio [12] has its basis on the assumption that cluster centers are surrounded by neighbors with lower local density and that they are at a relatively larger distance from any point with higher local density. There are two quantities to describe each data point i : its local density ρ_i and its distance δ_i from points of higher density.

The DPC algorithm provides two methods for calculating the local density for a data point i : the cutoff distance method and the kernel distance method. For a data point i , its local density ρ_i is defined as

$$\rho_i = \sum_{j \neq i} \chi(d_{ij} - d_c), \quad \chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x > 0 \end{cases} \quad (1)$$

with the cutoff distance method, or

$$\rho_i = \sum_{j \neq i} \exp \left[-\left(\frac{d_{ij}}{d_c} \right)^2 \right] \quad (2)$$

with the kernel distance method, where d_{ij} is the Euclidean distance between data points i and j , $d_c (> 0)$, the cutoff distance, is the radius of a point for scanning its neighborhood, which is set by the user. Thus, the local density ρ_i is positively correlated with the number of points with a distance from i less than d_c . The most obvious difference between the two methods is that for Eq. (1), ρ_i is a discrete value, whereas for Eq. (2), it is a continuous value. However, for both methods, ρ_i is sensitive to d_c .

Subsequently, δ_i in DPC is defined as

$$\delta_i = \min_{j: \rho_j > \rho_i} (d_{ij}) \quad (3)$$

For the data point i that has the highest local density ρ_i , δ_i is conventionally defined as

$$\delta_i = \max_j (d_{ij}) \quad (4)$$

As shown in Eq. (3), δ_i is the minimum distance between point i and points j with local densities $\rho_j > \rho_i$.

After calculating the two quantities: ρ and δ , we can recognize density peak points (cluster centers) as points for which the values of ρ_i and δ_i are anomalously large by mapping all the data points to the decision graph which takes ρ and δ as the two axes. However, sometimes we cannot select cluster centers from the decision graph correctly because they are too close to each other. Instead, we can sort all the data points by γ defined in Eq. (5) and choose the largest k (the cluster number) data points as the cluster centers.

$$\gamma_i = \rho_i \times \delta_i \quad (5)$$

Since the cluster centers have been found, each remaining point is assigned to the same cluster as its nearest neighbor of higher density.

3.2. Analysis

Although the experimental results obtained with DPC have shown that DPC performs well in many instances, the following drawbacks are obvious.

First, the accuracy of DPC depends on the setting of the cutoff distance (d_c). As shown in Fig. 1, there is a two-moon dataset. The red points are the cluster centers we select and the same for all the other pictures in this article. When we change the value of d_c , the clustering result of DPC can be greatly affected even for this simple dataset.

Second, the cluster number (k) is difficult to determine. Actually, we have little idea about the distribution of the dataset most of the time, not to mention choosing an ideal cluster number (k) for DPC. Fig. 2 shows the results of the traditional DPC algorithm on the Donut3

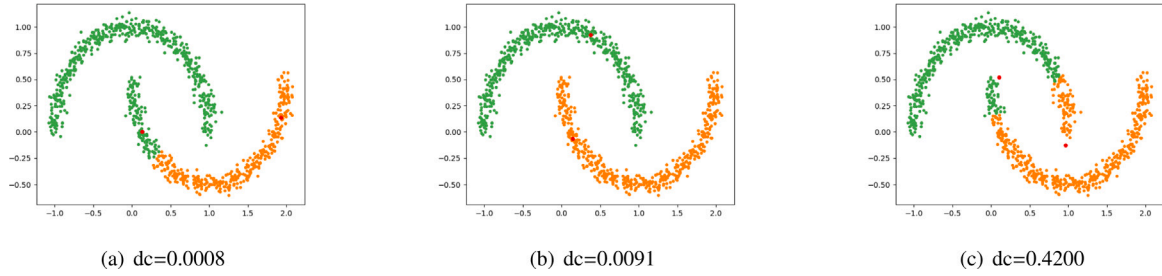


Fig. 1. Results of the traditional DPC algorithm on the two-moon dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

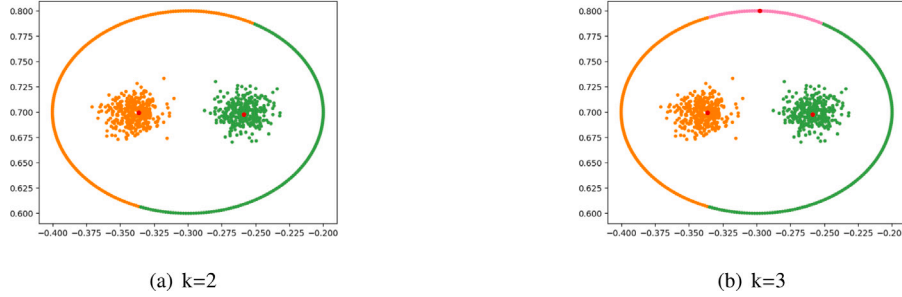


Fig. 2. Results of the traditional DPC algorithm on the Donut3 dataset.

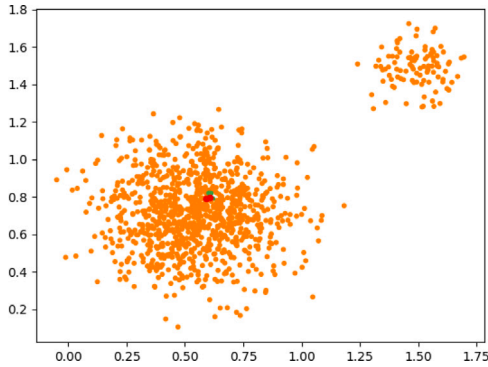


Fig. 3. Results of the traditional DPC algorithm on the dataset with two Gaussian clusters with very different densities.

dataset. In Fig. 2(a), we recognize the outer ring as noise and let the cluster number be 2. However, clearly, the result cannot satisfy us. In Fig. 2(b), we recognize the outer ring as a cluster and let the cluster number be 3. Even when we choose the correct cluster center in the outer ring, we cannot obtain the ideal result.

Third, it is difficult to recognize the centers of clusters. As shown in Fig. 3, there is a dataset that has two clusters based on a Gaussian distribution with one of the densities much larger than the other. Even when we choose the two data points with the largest γ_i as the cluster centers, both of them will belong to the same cluster, which has a larger density. The reason for this phenomenon is that the difference in densities between the two clusters is so large that we can hardly recognize the centers of clusters regardless of no matter by the decision graph or by sorting all the data points by γ .

Last, the final allocation strategy is sensitive and has poor fault tolerance. As shown in Fig. 2(b), even when we choose the three cluster centers correctly, we cannot obtain the ideal result. There are still some data points that cannot be allocated to the correct cluster.

To overcome the deficiencies mentioned above, in this article, we present an improved probability propagation algorithm for density peak clustering based on natural nearest neighbors (DPC-PPNNN). By

introducing the concept of natural nearest neighbors, we can avoid setting d_c manually. By calculating $\gamma = \rho \times \delta$ and $\theta = \frac{\rho}{\delta}$, we can select cluster centers automatically. Finally, the clustering algorithm based on probability propagation can help us allocate all the remaining data points. By doing all these, DPC-PPNNN can be suitable for more complex datasets and distinguish two clusters that are close to each other.

4. DPC-PPNNN

4.1. Natural nearest neighborhood

The natural nearest neighborhood (NNN) [21] is very different from traditional k -nearest neighbors. The NNN is a scale-free concept and does not require parameters in the selection of neighbors. The size of NNN of every data point may be different according to the distribution of the dataset. The following is a precise description of the natural neighborhood method through the definition of the relevant concepts.

Let $X = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$ be a dataset. We use the Euclidean distance $d(x_i, x_j)$ to measure the distance between points x_i and x_j . For a given integer $k > 0$, let $mn_k(x_i)$ be the k th nearest neighbor of x_i in $X \setminus \{x_i\}$. Define the k -nearest neighborhood of x_i ($NN_k(x_i)$) as the set of k nearest neighbors of x_i , i.e.

$$NN_k(x_i) = \cup_{j=1}^k \{mn_j(x_i)\}.$$

The k -reverse nearest neighborhood of x_i ($RNN_k(x_i)$) is the set of points $x_j \in X \setminus \{x_i\}$ with $x_i \in NN_k(x_j)$, i.e.

$$RNN_k(x_i) = \{x_j \in X \setminus \{x_i\} \mid x_i \in NN_k(x_j)\}.$$

The k -natural nearest neighborhood of x_i ($NNN_k(x_i)$) is defined as

$$NNN_k(x_i) = NN_k(x_i) \cap RNN_k(x_i).$$

Clearly, $NNN_{n-1}(x_i) = X \setminus \{x_i\} \neq \emptyset$ for any $x_i \in X$.

Definition 1 (Natural Nearest Neighborhood (NNN)). The least integer λ with $NNN_\lambda(x_i) \neq \emptyset$ for every $x_i \in X$ is called the natural eigenvalue of X . We define $NNN_\lambda(x_i)$ as the natural nearest neighborhood of x_i , denoted by $NNN(x_i)$, for every $x_i \in X$.

Let $NNN_k(X) = \{NNN_k(x_i) \mid x_i \in X\}$. Note that $NNN_k(X)$ maybe a multiset. Let

$$NNN_k^0(X) = \{x_j \mid x_j \in X \text{ with } NNN_k(x_j) = \emptyset\}.$$

By Definition 1, the natural eigenvalue λ represents the number of iterations in the process of searching the natural nearest neighborhood $NNN(X) = NNN_\lambda(X)$. The natural eigenvalue λ is generally small when there is no outlier or noise (the data points make $\lambda \in \Omega(\ln n)$). In the extremal case, λ can be the largest possible value $n - 1$ if there is an outlier $x_j \notin NNN_{n-2}(x_i)$ for any $x_i \neq x_j$ and $x_i \in X$.

To decrease the effect of outliers or noise in dataset X , we introduce $\ln n + \ln \lambda$ as the threshold to control the number of iterations when $NNN_k^0(X)$ remains unchanged in its subsequent iteration, i.e. $NNN_k^0(X) = NNN_{k+1}^0(X)$. Formally, we define the logarithmic natural nearest neighborhood as follows:

Definition 2 (Logarithmic Natural Nearest Neighborhood). We call the least integer λ with

$$|\{i \mid i \leq \lambda - 1 \text{ with } NNN_i^0(X) = NNN_{i+1}^0(X)\}| \geq \ln n + \ln \lambda$$

the logarithmic natural eigenvalue and define $NNN_\lambda(x_i)$ as the logarithmic natural nearest neighborhood of x_i , denoted by $NNN(x_i)$, for $x_i \in X$.

By Definition 2, the logarithmic natural eigenvalue λ represents the number of iterations in the process of searching the logarithmic natural nearest neighborhood $NNN(X)$, which will be bounded by $\ln n + \ln \lambda$. The logarithmic natural nearest neighborhood $NNN(X)$ may contain an empty set. The robust (logarithmic) natural nearest neighborhood searching algorithm proposed in this paper, which is based on the NNN searching algorithm [21], is shown in Alg. 1.

Algorithm 1 (Logarithmic) Natural Nearest Neighborhood Search Algorithm (NNN)

Input: A set of points $X = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$

Output: The (logarithmic) natural eigenvalue $\lambda = r$ and $NNN(x_i) = NNN_r(x_i)$ for every $x_i \in X$

```

1:  $\forall x_i \in X$ , initially set  $NN_0(x_i) = \emptyset$ ,  $RNN_0(x_i) = \emptyset$ ,  $NNN_0(x_i) = \emptyset$ 
2:  $r = 1$ ,  $\text{flag} = 0$ ,  $T = 0$ 
3: while  $\text{flag} = 0$  do
4:   for  $\forall x_j \in X$  do
5:     calculate  $nn_r(x_j) = x_i$ 
6:     Reset  $NN_r(x_i) = NNN_{r-1}(x_i) \cup \{x_j\}$ 
7:     Reset  $RNN_r(x_j) = RNN_{r-1}(x_j) \cup \{x_i\}$ 
8:     Reset  $NNN_r(x_i) = NNN_{r-1}(x_i) \cap RNN_r(x_i)$ 
9:   end for
10:  calculate the set  $NNN_r^0(X)$ 
11:  if  $NNN_r^0(X)$  remains unchanged then
12:     $T := T + 1$ 
13:  end if
14:  if  $T \geq \ln r + \ln n$  or  $\emptyset \notin NNN_r(X)$  then
15:     $\text{flag} = 1$ 
16:  else
17:     $r := r + 1$ 
18:  end if
19: end while

```

Remrk. NNN searching algorithm proposed by Zhu et al. [21] does not take outliers or noise into consideration. In contrast, by introducing $\ln n + \ln \lambda$ as the threshold, the robust natural nearest neighborhood searching algorithm we propose can recognize outliers and eliminate their effect.

4.2. Local density estimation and selection of cluster centers

As shown in Section 4.1, NNN is a scale-free concept and does not require parameters in the selection of neighbors. The parameter d_c in DPC can be avoided by changing the definition of the local density based on the NNN. For any data point x_i , the new local density is defined as

$$\rho_i = \sum_{x_j \in NNN(x_i)} \exp \left[-\left(\frac{d_{ij}}{\sigma_i} \right)^2 \right], \quad (6)$$

where $\sigma_i = \max\{d(x_i, x_j) \mid x_j \in NNN(x_i)\}$. We define

$$\delta_i = \min_{j: \rho_j > \rho_i} (d_{ij})$$

and for data point i which has the highest local density ρ_i , define

$$\delta_i = \max_j (d_{ij}),$$

which are the same as in DPC.

Let $\gamma_i = \rho_i \times \delta_i$ as defined in Eq. (5). We can choose the candidate centers based on a standard normal distribution with a one-sided 95% confidence interval. The candidate centers are defined as

$$\text{CanCens} = \{x_i \in X \mid \gamma_i > E(\gamma) + 1.65\text{Var}(\gamma)\}, \quad (7)$$

where $E(\gamma)$ is the expectation of γ , $\text{Var}(\gamma)$ is the variance of γ , 1.65 is the 95% quantile of the standard normal distribution.

By the definition of CanCens, it does not contain the data points with low local density ρ_i and small distance δ_i , but it may have some points with large ρ and small δ or small ρ and large δ which is not suitable for being cluster centers. To eliminate those data points, we define

$$\theta_i = \frac{\rho_i}{\delta_i} \quad (8)$$

for every $x_i \in \text{CanCens}$. Note that the gap between ρ_i and δ_i is large enough for those data points with large ρ and small δ or small ρ and large δ . Therefore, we can select the final centers from Cancens based on standard normal distribution of θ with a two-sided 95% confidence interval as shown in Eq. (9).

$$\text{Cens}(X) = \{x_i \in \text{CanCens} \mid |\theta_i - E(\theta)| < 1.96\text{Var}(\theta)\}, \quad (9)$$

where 1.96 is the 97.5% quantile of the standard normal distribution.

In the Probabilistic Propagation Clustering Algorithm provided in Section 4.3, one cluster may have more than one center point in $\text{Cens}(X)$, so the allocation strategy of DPC cannot be used here. We need a new allocation strategy to cluster all the data points using the centers in $\text{Cens}(X)$.

4.3. Probabilistic propagation clustering algorithm

The idea of the probabilistic propagation clustering algorithm is based on the spread of the epidemic in recent years. An infected man will infect the persons he contacts. Therefore, we first choose the data point with the largest local density ρ in $\text{Cens}(X)$ as patient zero. In addition, he will infect the persons he contacts with, that is his natural nearest neighborhood (NNN). These neighbors will infect their natural nearest neighbors and the propagation will continue until there is no neighbor that can be infected. At this time, we recognize all the infected data points as a cluster. We call the processes of forming a cluster a round of propagations. Second, we select the data point that has the largest local density among those data points not infected in $\text{Cens}(X)$, and repeat the process above until all the data points in Cens have been infected.

In the real world, some people will not become infected even when they contact with infected persons because they have antibodies or other reasons. Therefore, in the algorithm, a data point $x \in X$ is infected with a certain probability $p_x = C \cdot (p'_x + p''_x)$, where p'_x (and p''_x)

denotes the rank of x in the current round of propagation. Formally, in the current round of propagation, if point x (to be checked) and the infected points can be ordered $y_1, \dots, y_{i-1}, x, y_{i+1}, \dots, y_k$ with $\rho(y_1) \leq \dots \leq \rho(y_{i-1}) \leq \rho(x) \leq \rho(y_{i+1}) \leq \dots \leq \rho(y_k)$, then we define

$$p'_x = \frac{i}{k};$$

if point x and the unchecked points in $\cup_{j=1, j \neq i}^k NNN(y_j)$ can be ordered $z_1, \dots, z_{j-1}, x, z_{j+1}, \dots, z_\ell$ with $\rho(z_1) \leq \dots \leq \rho(z_{j-1}) \leq \rho(x) \leq \rho(z_{j+1}) \leq \dots \leq \rho(z_\ell)$, then we define

$$p''_x = \frac{j}{\ell}.$$

If point x is not infected, we consider it having antibodies or dead and that it will no longer be infected. At the beginning of the propagation process, we increase the probability p_x of being infected to ensure that the dissemination process will continue. At the end of the propagation process, we decrease the probability p_x of being infected to guarantee that the outliers or noise will not be classified into any cluster.

Algorithm 2 Probabilistic propagation clustering algorithm (PP)

Input: A set of points $X \subset \mathbb{R}^d$, $\rho = \rho(X)$, $NNN(X)$

Output: Clustering results CLU

```

1: Initially set  $Cens = Cens(X)$ 
2:  $n = 1$ 
3:  $x = find(\max \rho(Cens)); Cens = Cens \setminus \{x\};$ 
    $CLU(n) = \emptyset; CLU_{can} = NNN(x);$ 
4: while  $CLU_{can} \neq \emptyset$  do
5:   for  $\forall y \in CLU_{can}$  do
6:     calculate the probability of being infected for  $y$ :  $p_y$ 
7:     if  $y$  is infected (with probability  $p_y$ ) then
8:       Reset  $CLU(n) = CLU(n) \cup \{y\}$ 
9:       Reset  $CLU_{can} = CLU_{can} \cup NNN(y)$ 
10:    if  $y \in Cens$  then
11:      Reset  $Cens = Cens \setminus \{y\}$ 
12:    end if
13:  else
14:     $y$  does not belong to any cluster
15:  end if
16:  Reset  $CLU_{can} = CLU_{can} \setminus \{y\}$ 
17: end for
18: end while
19: if  $Cens \neq \emptyset$  then
20:    $n = n + 1$ 
21:   goto Step 3
22: end if
23: Assign each remaining data point to the cluster that has the largest
   sum of local densities among its natural nearest neighborhood

```

When the propagation process is over, there will be some data points that are not infected and are not outliers. We assign each remaining data point to the cluster that has the largest sum of local densities among the natural nearest neighborhoods containing it.

By introducing the idea of probabilistic propagation, we can make the cluster algorithm closer to reality. Moreover, we can distinguish two clusters that are close to each other. The probabilistic propagation clustering algorithm is shown in Alg. 2. Denote $\rho(A) = \{\rho(x) \mid x \in A\}$.

5. Experiment

We call the algorithms 1 and 2 together the DPC-PPNNN algorithm. To test the performance of the DPC-PPNNN algorithm, we use classical synthetic datasets and real-world datasets. Moreover, we take traditional DPC, K-means and DBSCAN as the control group, where

Table 1
Synthetic datasets.

Dataset	No. of records	No. of attributes	No. of clusters
2d-4c-no9	876	2	4
3-spiral	312	2	3
Aggregation	788	2	7
Cassini	1000	2	3
Complex9	3031	2	9
Compound	399	2	6
Dartboard1	1000	2	4
Jain	373	2	2
R15	600	2	15
Shapes	1000	2	4

Table 2
Real-world datasets.

Dataset	No. of records	No. of attributes	No. of clusters
Ecoli	336	7	8
Glass	214	9	7
Heart-statlog	270	13	2
Iono	351	34	2
Iris	150	4	3
Thy	215	5	3
Wdbc	569	30	2
Wine	178	13	3

the K-means and DBSCAN algorithms are implemented in the sklearn library of Python and the DPC algorithm, without the ‘‘Halo’’ part, is based on the source code provided by the author since our datasets do not contain noise. All the results shown are the optimal results after argument tuning.

The synthetic datasets and real-world datasets used in the experiments are presented in Tables 1 and 2, respectively.

5.1. Preliminaries

To evaluate the performance of the clustering algorithm, three evaluation indices are introduced, namely, Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI) and Fowlkes–Mallows index (FMI). The upper bound of the three indicators is 1, in other words, larger values of the indicators indicate better clustering results.

Before the tests, all of the real-world datasets should be adjusted by min–max normalization to eliminate the differences in the ranges of different dimensions, as shown in Eq. (10).

$$x'_{ij} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} \quad (10)$$

where x_{ij} is the original data in the i th position of x_j .

To more objectively reflect the actual results of various algorithms, we perform argument tuning on each of the algorithms, thereby ensuring that their best performances are compared. For K-means, we simply give the correct number of clusters. For DBSCAN, we use a grid search to find the best parameter configuration. For traditional DPC, we choose the optimal d_c from 1% to 2% and apply the Gaussian kernel in the density estimation process.

5.2. Synthetic datasets

In this part, we select a number of synthetic datasets that are widely used to test the performance of clustering algorithms. These datasets are different in terms of the distribution and numbers of points and clusters. They can simulate different situations to compare the performance of various clustering algorithms in different scenarios.

Table 3 shows the clustering results in terms of the ARI, AMI and FMI scores on all synthetic datasets listed in Table 1. The evaluation criterion with the highest score is marked in bold, and we can see that in most cases, the DPC-PPNNN obtains the highest score, except for

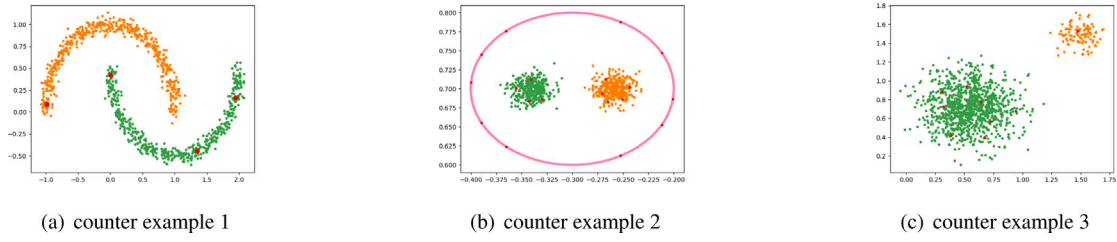


Fig. 4. The re-clustering results of the three counter examples in Section 3.2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

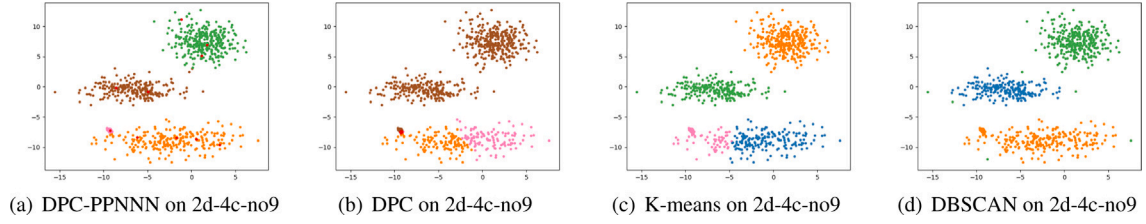


Fig. 5. The clustering results on 2d-4c-no9 by 4 algorithms.

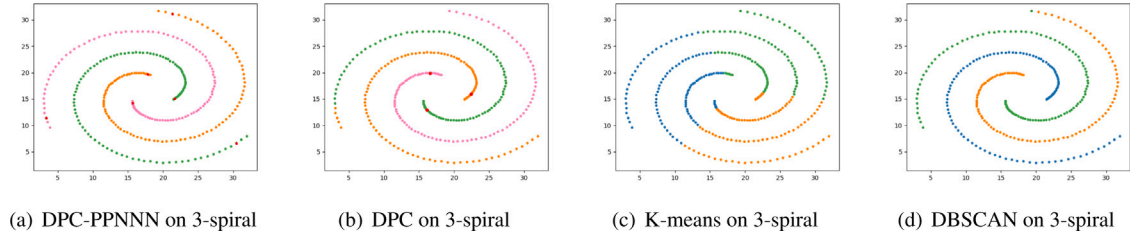


Fig. 6. The clustering results on 3-spiral by 4 algorithms.

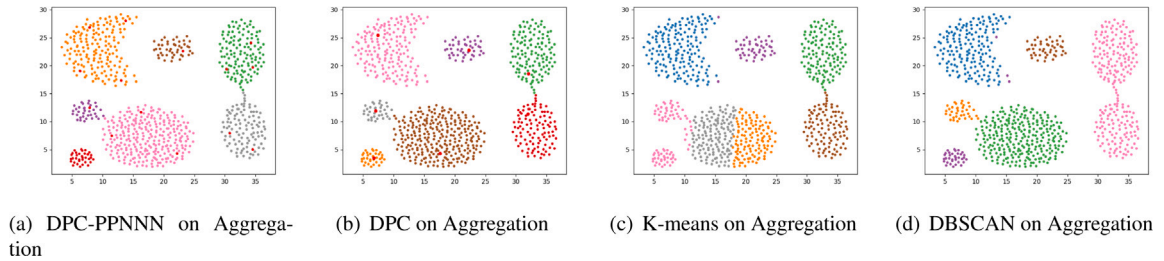


Fig. 7. The clustering results on Aggregation by 4 algorithms.

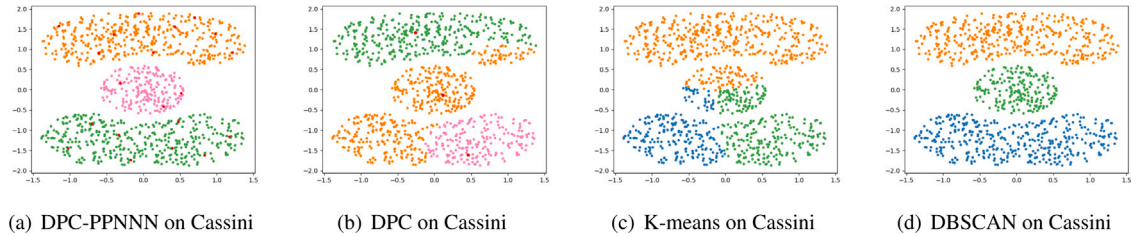


Fig. 8. The clustering results on Cassini by 4 algorithms.

the Complex9 (where the score of DPC-PPNNN is a bit less than the DBSCAN).

In Fig. 4, we recluster the three counter examples in Section 3.2 by DPC-PPNNN. Clearly, the results are much better than the original DPC which implies that our algorithm can overcome the deficiencies of DPC.

Next, we will show some typical clustering results on the synthetic datasets by DPC-PPNNN and the algorithms of the control group. The points with different colors in the figures are assigned to different

clusters. The cluster centers obtained from DPC and DPC-PPNNN are marked in red.

In Fig. 5, we can see that the 2d-4c-no9 dataset has four clusters, of which one cluster has a very high density. DPC-PPNNN succeeds in detecting all of them, while the other algorithms of the control group fail to do so.

Fig. 6 shows the results of each algorithm on the 3-spiral dataset. DPC-PPNNN can identify the four clusters correctly. Furthermore, DPC

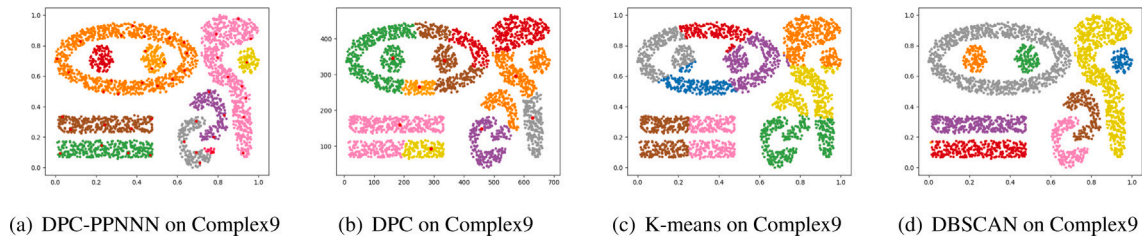


Fig. 9. The clustering results on Complex9 by 4 algorithms.

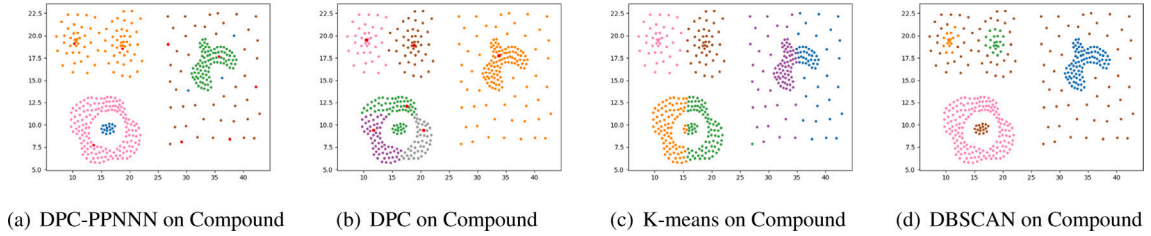


Fig. 10. The clustering results on Compound by 4 algorithms.

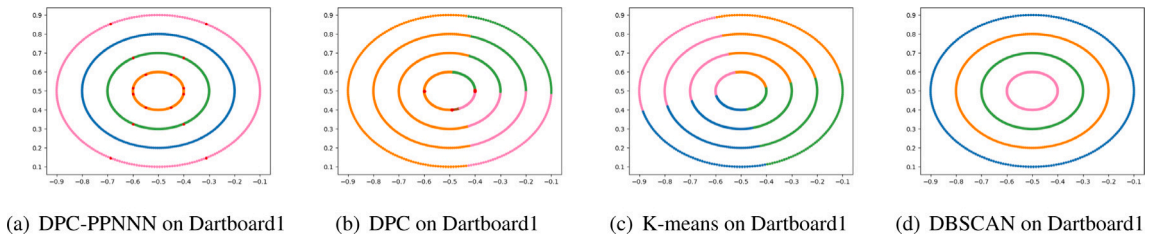


Fig. 11. The clustering results on Dartboard1 by 4 algorithms.

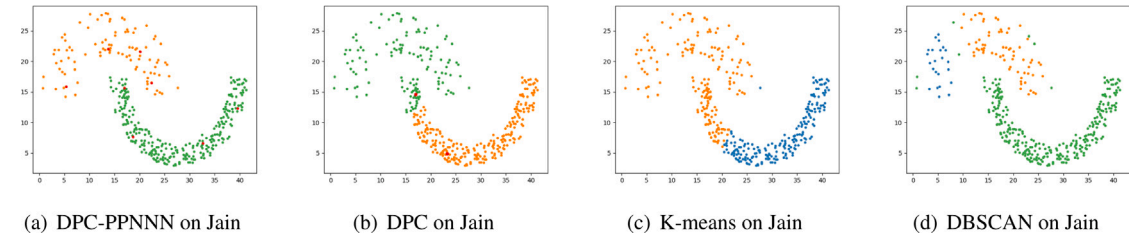


Fig. 12. The clustering results on Jain by 4 algorithms.

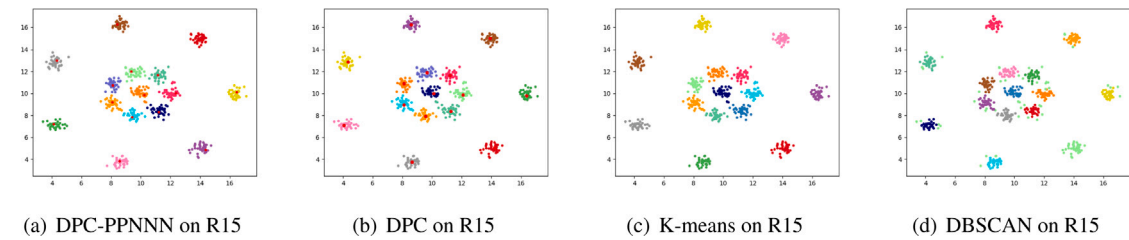


Fig. 13. The clustering results on R15 by 4 algorithms.

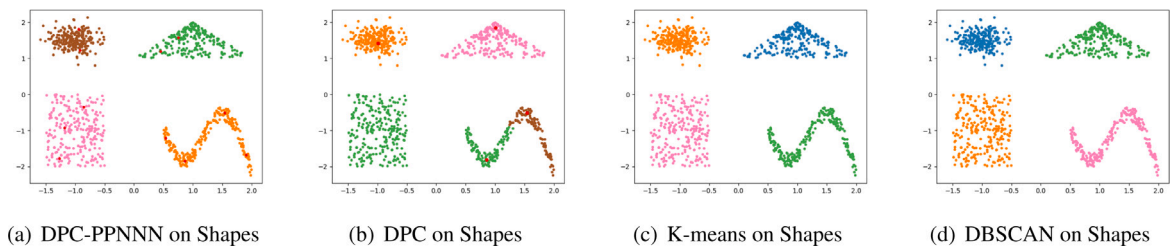


Fig. 14. The clustering results on Shapes by 4 algorithms.

and DBSCAN detect the four clusters mostly correctly with some wrong data points at the tail, perhaps because the arguments are not set correctly, while KNN cannot recognize them at all.

The clustering results of the DPC-PPNNN and the other three algorithms of the control group on the Aggregation dataset are shown in Fig. 7. The dataset can be detected correctly by DPC-PPNNN and DPC. K-means fails to do so because the dataset is somewhat complex for K-means. DBSCAN cannot distinguish the two clusters on the right.

From the clustering results in Fig. 8, we can see that DPC-PPNNN and DBSCAN can detect the clusters in the Cassini dataset. Although DPC can find the correct cluster centers, it fails to allocate the other remaining data points correctly. The three clusters are not uniform in shape which leads to the wrong cluster results by K-means.

The Fig. 9 shows the results of the four algorithms on the Complex9 dataset. DBSCAN can recognize all the clusters successfully. While for DPC-PPNNN, there are some misclassified points at the tail of a cluster because these points are not connected tightly enough. DPC finds the correct cluster centers but it fails to allocate the other remaining data points correctly. K-means has poor performance.

Fig. 10 displays the results on the Compound dataset. DPC-PPNNN can detect most of the clusters correctly and can also recognize noise. DBSCAN is good at recognizing noise but it also misclassifies many data points as noise. DPC and K-means fail to recognize the clusters with noise.

As shown in Fig. 11, the Dartboard1 dataset has four concentric rings. Both DPC-PPNNN and DBSCAN have perfect performance. DPC cannot find the correct cluster centers because the densities of the data points have little difference. Moreover, K-means has poor performance.

For the Jain dataset shown in Fig. 12, only DPC-PPNNN correctly identifies all clusters. DPC has not found the correct cluster centers because the difference between the densities of the two clusters is too large. The reason is the same for the poor performance of DBSCAN. K-means is not suitable for the datasets with streamline shapes.

Fig. 13 displays the results on the R15 dataset. The distribution of points makes it the most straightforward dataset for all the algorithms. Although there are some small defects among them, all the algorithms can recognize both the clusters and centers.

The clustering shown in Fig. 14 demonstrates the ability to address the datasets with clusters of different shapes. DPC-PPNNN, K-means and DBSCAN all perform well. Only DPC cannot detect the clusters because there is a cluster with manifold structure.

5.3. Read-world datasets

In this section, 8 UCI datasets, as shown in Table 2, are used to demonstrate the performance of the DPC-PPNNN clustering algorithm. These datasets are different in terms of sample number, feature number and cluster number. As shown in Table 4, DPC-PPNNN performs almost the best among the test cases.

6. Conclusions

In this paper, we proposed an improved probability propagation algorithm for density peak clustering based on natural nearest neighbors (DPC-NNN). The new algorithm does not require any parameters and can recognize cluster centers automatically. The final clustering process of the DPC-PPNNN motivated by the epidemic spread performs well especially for distinguishing two clusters that are close to each other.

However, since the algorithm is based on density and probability, its performance is not good and stable enough, especially when all the data points have similar δ or similar densities ρ . For future work, we will work on solving these problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 3

Performances of different clustering algorithms on different synthetic datasets.

Algorithm	ARI	AMI	FMI	ARI	AMI	FMI
2d-4c-no9			Compound			
DPC-PPNNN(ours)	0.9931	0.9898	0.9951	0.9375	0.9345	0.9543
DPC [12]	0.3336	0.5261	0.6108	0.5099	0.7542	0.6235
K-means [3]	0.8952	0.9007	0.9241	0.5364	0.7128	0.6410
DBSCAN [7]	0.8412	0.8810	0.8940	0.8774	0.8673	0.9103
3-spiral			Dartboard1			
DPC-PPNNN(ours)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
DPC [12]	0.9521	0.9406	0.9680	0.0006	0.0130	0.3213
K-means [3]	−0.0060	−0.0055	0.3274	−0.0030	−0.0033	0.2470
DBSCAN [7]	0.9953	0.9918	0.9969	1.0000	1.0000	1.0000
Aggregation			Jain			
DPC-PPNNN(ours)	0.9927	0.9881	0.9943	1.0000	1.0000	1.0000
DPC [12]	0.9978	0.9956	0.9983	0.6438	0.5951	0.8502
K-means [3]	0.7624	0.8776	0.8159	0.3241	0.3677	0.7005
DBSCAN [7]	0.8719	0.9223	0.9051	0.9350	0.8476	0.9742
Cassini			R15			
DPC-PPNNN(ours)	1.0000	1.0000	1.0000	0.9928	0.9938	0.9932
DPC [12]	0.4886	0.5879	0.6736	0.9857	0.9885	0.9866
K-means [3]	0.5308	0.5400	0.7012	0.9928	0.9938	0.9932
DBSCAN [7]	1.0000	1.0000	1.0000	0.8307	0.8905	0.8416
Complex9			Shapes			
DPC-PPNNN(ours)	0.9375	0.9345	0.9543	1.0000	1.0000	1.0000
DPC [12]	0.3929	0.6666	0.4940	0.7439	0.8410	0.8145
K-means [3]	0.3461	0.6161	0.4517	1.0000	1.0000	1.0000
DBSCAN [7]	0.9998	0.9994	0.9998	1.0000	1.0000	1.0000

Table 4

Performances of different clustering algorithms on different real-world datasets.

Algorithm	ARI	AMI	FMI	ARI	AMI	FMI
Ecoli			Iris			
DPC-PPNNN(ours)	0.5761	0.5614	0.7160	0.8680	0.8446	0.9114
DPC [12]	0.2660	0.3179	0.4361	0.5414	0.6780	0.7539
K-means [3]	0.4115	0.5921	0.5482	0.7163	0.7387	0.8112
DBSCAN [7]	0.1004	0.1275	0.5282	0.5681	0.7316	0.7715
Glass			Thy			
DPC-PPNNN(ours)	0.2595	0.3450	0.5534	0.7522	0.6108	0.8827
DPC [12]	0.0651	0.0984	0.3516	0.2105	0.2301	0.7530
K-means [3]	0.1662	0.2942	0.3945	0.6283	0.5909	0.8546
DBSCAN [7]	0.2382	0.3577	0.5500	0.6715	0.4911	0.8606
Heart-statlog			Wdbc			
DPC-PPNNN(ours)	0.2380	0.2186	0.5432	0.7193	0.6054	0.8671
DPC [12]	0.0003	0.0002	0.7050	0.4705	0.4614	0.7860
K-means [3]	0.3488	0.2692	0.6755	0.7302	0.6226	0.8770
DBSCAN [7]	0.0861	0.1242	0.3417	0.2201	0.2618	0.6091
Iono			Wine			
DPC-PPNNN(ours)	0.5287	0.4114	0.7935	0.7421	0.7268	0.8277
DPC [12]	0.0019	0.0013	0.7302	0.6990	0.7233	0.8006
K-means [3]	0.1776	0.1330	0.6053	0.8537	0.8400	0.9026
DBSCAN [7]	0.7214	0.6343	0.8779	0.4229	0.5209	0.6482

Acknowledgments

The work was supported by National Natural Science Foundation of China (No. 12071453), the National Key R and D Program of China (2020YFA0713100), Anhui Initiative in Quantum Information Technologies, China (AHY150200), and the Innovation Program for Quantum Science and Technology, China (2021ZD0302904).

References

- [1] Tan PN, Steinback M, Kumar V. Introduction to data mining. Data mining. 2011.
- [2] Liu R, Wang H, Yu X. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Inform Sci 2018;450:200–26.

- [3] Macqueen JB. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967. 1967.
- [4] Kaufmann L, Rousseeuw PJ. *Clustering by means of medoids*. North-Holland; 1987.
- [5] Guha S, Rastogi R, Shim K. Cure: An efficient clustering algorithm for large databases. *Inf Syst* 1998;26:35–58.
- [6] Zhang T, Ramakrishnan R, Livny M. Birch: An efficient data clustering method for very large databases. *ACM SIGMOD Rec* 1996;25:103–14.
- [7] Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. *AAAI Press*; 1996.
- [8] Ankerst M, Breunig MM, Kriegel HP, Sander J. Optics: Ordering points to identify the clustering structure. In: *SIGMOD 1999, Proceedings ACM SIGMOD international conference on management of data*, June 1-3, 1999. Philadelphia, Pennsylvania, USA; 1999.
- [9] Sheikholeslami G, Chatterjee S, Zhang A. Wavecluster: A wavelet based clustering approach for spatial data in very large databases. *VLDB J* 2000;8:289–304.
- [10] Wang W, Yang J, Muntz R. Sting: A statistical information grid approach to spatial data mining. *VLDB J* 1997;25:186–95.
- [11] Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the em algorithm. *Proc R Stat Soc* 1977;39:1–22.
- [12] Rodriguez A, Laio A. Clustering by fast search and find of density peaks. *Science* 2014;344:1492.
- [13] Du M, Ding S, Jia H. Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowl-Based Syst* 2016;99:135–45.
- [14] Mehmood R, Zhang G, Bie R, Dawood H, Ahmad H. Clustering by fast search and find of density peaks via heat diffusion. *Neurocomputing* 2016;208:210–7.
- [15] Liang Z, Chen P. Delta-density based clustering with a divide-and-conquer strategy. *Pattern Recognit Lett* 2016;73:52–9.
- [16] Xu J, Wang G, Deng W. DenPEHC: Density peak based efficient hierarchical clustering. *Inform Sci* 2016;373:200–18.
- [17] Tao LI, Hongwei GE, Shuzhi SU. Density peaks clustering by automatic determination of cluster centers. *Front Comput Sci Technol* 2016;10:1614–22.
- [18] Zhou Z, Si G, Zhang Y, Zheng K. Robust clustering by identifying the veins of clusters based on kernel density estimation. *Knowl-Based Syst* 2018;159:309–20.
- [19] Xie JY, Grant WX, Philip W, Liu XH, Gao HC. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors. *Inform Sci* 2016;354:19–40.
- [20] Geng YA, Li Q, Zheng R, Zhuang F, He R. Recome: a new density-based clustering algorithm using relative knn kernel density. *Inform Sci* 2018;436–437:13–30.
- [21] Zhu Q, Huang J, Feng J, Zhou X. A clustering algorithm based on natural nearest neighbor. *J Comput Inf Syst* 2014;10:5473–80.