Mathematical Biosciences and Engineering

MBE, 20(7): 11875–11894.
DOI: 10.3934/mbe.2023528
Received: 28 February 2023
Revised: 28 April 2023
Accepted: 03 May 2023
Published: 10 May 2023

http://www.aimspress.com/journal/MBE

Research article

A novel density peaks clustering algorithm for automatic selection of clustering centers based on K-nearest neighbors

Zhihe Wang, Huan Wang*, Hui Du, Shiyin Chen and Xinxin Shi

The School of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, China

* **Correspondence:** Email: 1164136724@qq.com; Tel: +8618060613073.

Abstract: The density peak clustering algorithm (DPC) requires manual determination of cluster centers, and poor performance on complex datasets with varying densities or non-convexity. Hence, a novel density peak clustering algorithm is proposed for the automatic selection of clustering centers based on K-nearest neighbors (AKDPC). First, the AKDPC classifies samples according to their mutual K-nearest neighbor values into core and non-core points. Second, the AKDPC uses the average distance of K nearest neighbors of a sample as its density. The smaller the average distance is, the higher the density. Subsequently, it selects the highest density sample among all unclassified core points as a center of the new cluster, and the core points that satisfy the merging condition are added to the cluster until no core points satisfy the condition. Afterwards, the above steps are repeated to complete the clustering of all core points. Lastly, the AKDPC labels the unclassified non-core points similar to the nearest points that have been classified. In addition, to prove the validity of AKDPC, experiments on manual and real datasets are conducted. By comparing the AKDPC with classical clustering algorithms and excellent DPC-variants, this paper demonstrates that AKDPC presents higher accuracy.

Keywords: density peak; clustering; vary densities; cluster centers; K-nearest neighbors; labels

1. Introduction

Clustering is a powerful technique for data analysis, with an irreplaceable role in data mining. It essentially exploits the similarities between samples to divide them into several different clusters. Its application is found in numerous popular areas, including image processing [1,2], medicine [3,4], text

segmentation [5], community network analysis [6], bioinformatics [7,8], etc. So far, researchers have developed multiple algorithms with satisfactory performance, including the division-based method K-means [9], the layer structure-based method BIRCH [10], the density-based method DBSCAN [11], the grid-based method WaveCluster [12] and the graph theory-based spectral clustering [13]. Among them, DBSCAN is a well-performing density-based clustering algorithm that determines clusters based on density connectivity relationships. It has the advantage of being able to cluster arbitrarily shaped datasets and presents a high resistance to noise. However, the presence of the two parameters ϵ and MinPts in the algorithm significantly impacts the final clustering results, and it shows poor performance on datasets with varying densities.

In 2014, Alex Rodriguez et al. [14] published the DPC (density peak clustering algorithm). Due to its high efficiency, robustness and simplicity of understanding, an increasing number of researchers are emphasizing the aforementioned algorithm. DPC uses two criteria when selecting cluster centers. First, the centers of the different clusters are far from each other, and secondly, the clustering center will be enclosed by some low densities of points. Considering the rest of the points, the DPC employs a one-step strategy to assign each point in a cluster to the nearest high-density cluster. While it presents multiple obvious advantages over other classical algorithms, the DPC suffers from a few problems: 1) It requires manual involvement when selecting cluster centers, and this is extremely difficult in datasets where the boundaries between clusters are not clear. 2) The DPC tends to fail to find peaks in sparse areas; consequently, it does not achieve satisfactory clustering results when large differences in data density between different clusters are present. 3) The assignment of non-peak points by the DPC tends to ignore the real situation in the area where the sample is located. Hence, when a misclassification of one point occurs during the assignment process, a domino effect is triggered. In light of the aforementioned problems, scholars have suggested various improved clustering algorithms on the basis of DPC to address its drawbacks.

In this context, ADPC-KNN [15] used K-nearest neighbors to determine the cut-off distance, with a novel method proposed for automatically selecting cluster centers, but it still underperformed on datasets with varying densities. The NDPC [16] reduced the density gap between samples from sparse areas and those from dense areas through a new method that calculated local densities so that peaks could also be found in sparse regions. Unfortunately, the NDPC was unable to discover all peaks in datasets with large density differences. The AmDPC [17] employed density deviation to replace the original local density, and the authors proposed a density deviation multi-peak auto-clustering approach that overcame the poor performance of the original DPC on non-convex datasets with low-density peaks. Nevertheless, its parameter selection process was complex. The FKNN-DPC [18] solved the sample assignment error problem in the DPC assignment strategy using the fuzzy weighted K-nearest neighbor technique. Although it was more robust than the DPC, the FKNN-DPC still required manual intervention for the selection of clustering centers. Meanwhile, the SNN-DPC [19] adopted a two-step distribution method of necessary and possible subordination to overcome the consequences of the domino effect caused by the shortcomings of the DPC allocation rules. However, it required human involvement and was more complex compared to the DPC. 3W-DPET [20] suggested a three-way density peak clustering approach on the basis of evidence theory, which solved the problem of mislabel propagation in the DPC. However, the 3W-DPET could not automatically determine the number of clusters. On the other hand, the DPC-KNN [21] joined the notion of K-nearest neighbors for distance calculation and sample assignment to improve the clustering effect in non-spherical datasets. However, the algorithm still required a manual selection of clustering centers. Through the analysis of multiple algorithms that have improved the DPC in recent years, it is clear that the above algorithms generally still do not address the problem that the DPC performs poorly on

datasets with varying density differences or non-convexity; and despite the aforementioned improvement of individual algorithms to tackle this issue, manual involvement is still required when selecting clustering centers.

The primary contribution of this study is the proposal of a novel density-peak clustering algorithm (AKDPC) that automatically selects clustering centers and adaptively completes the corresponding clusters. The suggested method divides the samples into core and non-core points by mutual K-nearest neighbor values, where non-core points are placed in the second assignment, reducing the impact on the final clustering. Simultaneously, only the average distance between the K-nearest neighbors of a sample is used as an indicator to select the cluster center. Specifically, the one with the lowest average distance among the K-nearest neighbors in remaining core points will be selected as a cluster center. Afterwards, the AKDPC adds core points that satisfy the merging condition to the corresponding clusters. Finally, the non-core points are clustered. The clustering of the AKDPC is fully automatic and is better suited to complex datasets with large density differences or non-convexity.

2. DPC

The DPC is an extremely efficient density clustering algorithm. It considers that the locations surrounded by some low local density data samples should be the cluster centers; furthermore, the clustering centers of different clusters should not be too close to each other. The DPC constructs decision diagrams to select centers from the sample local densities ρ and relative distances δ.

The original DPC uses both cut-off distance and kernel distance to compute the densities of the samples, and these two methods might apply to different datasets. The local density of sample i is defined as Eq (1).

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

Alternatively, obtained by Gaussian kernels defined as Eq (2),

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

where d_{ij} is the distance between two different samples, and d_c is the cut-off distance.

δ denotes the distance from a sample to the nearest sample with a density bigger than it, and the formula is defined as follows:

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

The relative distance for the sample with the highest density is specified to be the highest value among all samples, and it is defined as Eq (4).

$$\delta_i = \max_{i \neq j} (\delta_j) \tag{4}$$

After getting the ρ and δ values for all samples, the DPC considers the points with simultaneously larger ρ and δ to be the clustering centers. Finally, the DPC makes the labels for the leftover points consistent with the labels of the closest high-density points.

3. AKDPC

To overcome the issue that the DPC requires manual intervention to choose clustering centers and poor performance on complex datasets with varying densities or non-convexity, we propose AKDPC. The AKDPC consists of three main parts: 1) classifying samples into core and non-core points based on the sizes of their mutual K-nearest neighbor values; 2) Clustering the core points; 3) Allocation of remaining non-core points.

3.1. Classification of core and non-core points

The classification method determines whether to classify each sample point as core or non-core by its mutual K-nearest neighbor value.

The mutual K-nearest neighbor value MK_i of sample x_i is the sum of points in the sample x_i's K-nearest neighbor collection that incorporates sample x_i into its K-nearest neighbor collection, and MK_i is defined as Eq (5).

$$MK_i = \sum_{j \in KNN_i} f(x_j),\ f(x_j) = \begin{cases} 1 & x_i \in KNN_j \\ 0 & otherwise \end{cases} \tag{5}$$

where KNN_i is a collection of K nearest neighbors of sample *i*, defined as Eq (6), and where *K* is the first parameter to be specified in our algorithm.

$$KNN_i = \{x_j \mid \min_K(d_{ij}),\ x_i, x_j \in X, i \neq j\} \tag{6}$$

The *MK* values of samples have the following characteristics:

1) The *MK* value of a point represents its spatial relationship with its neighbors. If the value is large, the sample point is closely distributed with its neighbors or is in the same distribution area, so it is treated as a core point. If the value is smaller, the sample point is isolated from the surrounding points and treated as a non-core point.

2) The *MK* value of a sample point is not affected by the fact that the sample is located in a different density area because the *MK* value reflects the proximity of the sample to its neighbors.

We classify all points into core and non-core points by defining a classification threshold *MKT*, which is defined by Eq (7).

$$MKT = \frac{\sum_{j=0}^{n} MK_j}{N} \tag{7}$$

where *N* is the sum of samples.

Definition 1 (Core point). If the mutual K-nearest neighbor value MK_i of sample *i* is larger than or equal to *MKT*, then it is a core point.

Definition 2 (Non-core point). If the mutual K-nearest neighbor value MK_i of sample *i* is less than the *MKT*, then it is a non-core point.

With our proposed method above some non-core points can be accurately identified to prevent the possibility of a large impact on the final clustering, and Algorithm 1 explains in detail the classification of the sample points.

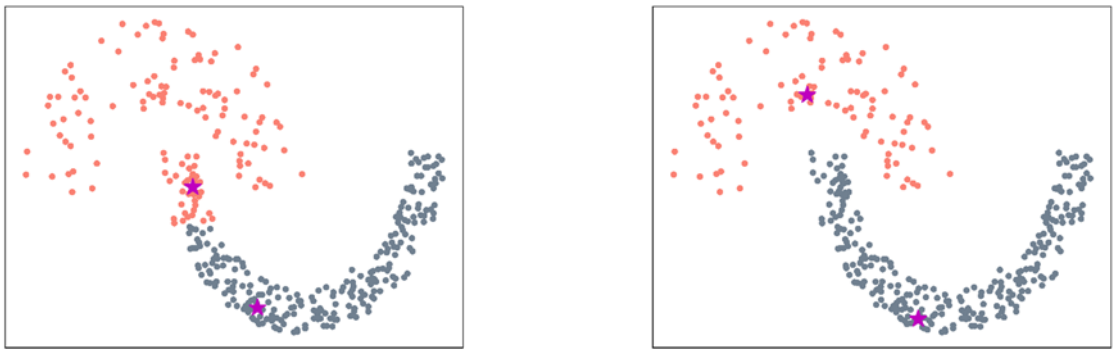
Algorithm 1: Classification of core and non-core points

Input: Dataset $D = \{x_1, x_2, ..., x_n\}$, K

Output: core points $CG = \{g_1, g_2, ..., g_m\}$, non_corepoints $NG = \{n_1, n_2, ..., n_j\}$, distance matrix $\{d_{ij}\}^{n\times n}$, Sorting Matrix $SD^{n\times n}$

Calculate distance matrix $D^{n\times n} = \{d_{ij}\}^{n\times n}$
Sort the distance matrix $D^{n\times n}$ in ascending order and record it as the $SD^{n\times n}$
Calculate the *MK* value for all samples based on Eq (5)
Calculate the *MKT* based on Eq (7)
Create set $CG = \emptyset$, $NG = \emptyset$
For each sample x in Data **Do**
    If $MKx >= MKT$ **Then**
        $CG = CG \cup x$
    If $MKx < MKT$ **Then**
        $NG = NG \cup x$
    End if
End for

3.2. Clustering of core points



(a) (b)

Figure 1. The selection of clustering centers. (a) Original DPC; (b) AKDPC.

To prevent the old DPC algorithm from requiring manual intervention to select centers of all clusters and the problem of easily ignoring clustering centers that exist in low-density areas, we redefine the local density of a sample based on the distance relationship from sample points to their nearest neighbors and suggest a new approach for merging core points. Our method does not need human involvement in the selection of clustering centers, and the whole clustering process is fully automatic. Figure 1 displays the clustering results of the original DPC algorithm and our algorithm for clustering on the classical dataset Jain [22]. As shown in the figure, the original DPC selected clustering centers located in the lower branches with higher density, no matter how the distance is

chosen up to the end. The correct classification is not completed in the end, while our method accurately identifies the clustering centers in two different density regions.

First, we define the D_i^k of core sample point *i* as the average distance from core sample *i* to its K-nearest neighbor set of points, and D^k is defined as Eq (8).

$$D_i^k = \frac{1}{k} \sum_{j \in N_i^k} d_{ij} \tag{8}$$

where N_i^k denotes the collection of k nearest neighbors of the core point *i*, defined as Eq (9), and where *k* is the second parameter to be specified in our algorithm.

$$N_i^k = \{x_j \mid \min_k(d_{ij}),\ x_i \in N, i \neq j\} \tag{9}$$

The nearest neighbor mean D^k of a sample point has the following properties:

1) If the D^k of the core point is particularly small, it is a good indication that the core point is in a very dense area, and its surrounding neighbors are all very close to it. Conversely, if the D^k value is very large, it means that the core is far away from its neighbors.

2) There is no doubt that the smaller the D^k value of a core point is, the greater the indicated density and the more qualified it is to be a cluster center. However, some clusters have a low overall density, which results in their overall D^k values being larger. It would be unwise for us to select cluster centers directly based on their D^k values. Therefore, we adopted a sequential clustering process. We first selected the point with the smallest D^k value among all core points as the center of a new cluster, and then after completing the clustering of the new cluster, we selected the core point with the smallest D^k value from the remaining core points for the center of another new cluster and then completed the clustering of the corresponding cluster. The above procedure was repeated until each core point has been allocated to different clusters. By using this method, we can correctly select the cluster centers even in data sets with uneven density distributions.

To better understand the process of clustering, we propose a definition of connectivity.

Definition 3 (Connectivity). A core point *i* is joinable to cluster C_j by connectivity if the distance from the core point *i* to the cluster C_j is smaller than the maximum value of D^k for core point *i* and any core point in cluster C_j. The connectivity of core point *i* to cluster C_j satisfies Eq (10).

$$\left\|X_i - C_j\right\| < \max\left(D_i^k, D_t^k\right),\ \ t \in C_j \tag{10}$$

where $\|X_i - C_j\|$ denotes the minimum distance between core point i and any core point in cluster *Cj*, and $max\left(D_i^k, D_t^k\right)$ denotes the maximum D^k value in core point i and cluster C_j.

As in Figure 2, the nearest distance between core point 5 and cluster 1 is d_{45} = 0.32. Also, according to Table 1, the largest nearest neighbor mean D^k in core point 5 and the core points in cluster 1 is D_5^k = 0.38. Under this condition, d_{45} is less than D_5^k satisfies connectivity, therefore core point 5 is added to cluster 1. The closest distance between core point 6 and cluster 1 is d_{56} = 0.33, and the maximum D^k value in core point 6 and the core points in cluster 1 is D_6^k = 0.41. d_{56} is less than D_6^k, satisfying the Connectivity, and thus core point 6 can join cluster 1. For the core point 7, the closest distance between it and cluster 1 is d_{67} = 0.38, and the maximum D^k value in core point 7 and the core points in cluster 1 is D_7^k = 0.43. d_{67} is less than D_7^k, and thus core point 7 can join cluster 1. In contrast, the closest distance between core point 9 and cluster 1 is d_{69} = 0.73, and the maximum D^k value in core point 9 and the core points in the cluster is D_9^k = 0.67, because d_{69} is greater than D_9^k.

The Connectivity cannot be satisfied, so core point 9 cannot be joined to cluster 1. Likewise, core point 8 cannot be joined to cluster 1.

Algorithm 2 shows the whole procedure of clustering the core points, where $SD^{n\times n}$ is the distance matrix after sorting in ascending order, and $\|g - C_i\|$ is the smallest distance from the core point g to cluster C_i. First, we create a new cluster and select the cluster center with the smallest D^k value among all core points. Second, depending on the distance of the remaining core points from the center of that cluster, we traverse from near to far and add the core points that satisfy the connectivity condition to that cluster (note that the merging of cores with clusters is an adaptive process, as each traversal adds cores to the cluster that satisfy the condition, so the maximum D^k value of the core points in the cluster will change, i.e., in the process of clustering Eq (6), the value on the right-hand side changes continuously), until no more cores can be added to the cluster. We repeat the above procedure until all core points have been grouped into clusters.

Algorithm 2: Clustering of core points

Input: core points $CG = \{g_1, g_2, ..., g_m\}$, $SD^{n\times n}$, $\{d_{ij}\}^{n\times n}$, k

Output: cluster set $C = \{c_1, c_2, ..., c_o\}$, Center set $T = \{t_1, t_2, ..., t_o\}$

1: Calculate the D_i^k based on Eq (8).
2: Sorting the core points array *CG* in descending order according to D_i^k
3: Create set $C = \emptyset, T = \emptyset$
4: Create a new cluster c_1, $c_1 = c_1 \cup CG[0], T = T \cup CG[0]$, $i = 1$ /* Create the first new cluster and set the point with the smallest D_i^k value to be the center of the cluster */
5: **while** existing g in *CG* **Do** /* Start Clustering */
6: FLAG == 0 /* Used to judge whether a cluster has joined a new core point */
7: **For** each core point g in SD[T[i]] /* Traversing the core points in order from near to far from the center of the ith cluster */
8: **If** $\|g - C_i\| < max(D_g^k, D_{ci}^k)$ **Then** /* Connectivity judgment */
9: FLAG == 1
10: $C_i = C_i \cup g$, $CG = CG \setminus g$
11: **End if**
12: **End for**
13: **If** FLAG == 0 **Then** /* If the current cluster is no longer changing, a new cluster is created and the point with the smallest D_i^k value among the remaining core points is used to be the center of clustering */
14: Creating a new cluster(c), $c = c \cup CG[0]$, $CG = CG \setminus CG[0]$, $T = T \cup \mathbf{CG}[0]$
15: $i = i + 1$ /* The clustering of the ith+1st new cluster is started */
16: **End if**
17: **End While**

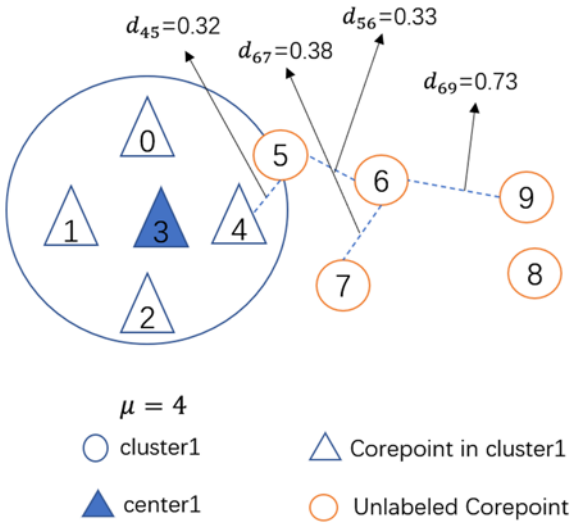


Figure 2. Example of merging between core samples and clusters.

Table 1. The D^k value of core points in the example in Figure 2.

Sample	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
D_i^k(cm)	0.35	0.34	0.31	0.33	0.38	0.41	0.43	0.68	0.67

3.3. Allocation of remaining non-core points

Once we had finished clustering the core points, to determine the labels of the non-core samples, we first sorted the non-core samples in ascending order based on their D^k value. Empirically, we found that most of the neighboring samples were in the same cluster, so we label the unclassified non-core points with the same labels as the nearest points that have been classified. The above method is explained in detail in Algorithm 3.

Algorithm 3: Allocation of remaining non-core points

Input: cluster set C = $\{c_1,c_2,...,c_o\}$, non_corepoints, NG = $\{n_1,n_2,...,n_j\}$ Sorting Matrix $SD^{n\times n}$
Output: cluster set C = $\{c_1,c_2,...,c_o\}$

```
For each n in NG do
For x in SD[n] do
    If x ∈ cj ( j = 1, 2, …, o) Then
        cj = cj ∪ x
        Break
    End if
End for
End for
```

3.4. Example

To understand our algorithm more intuitively, we show the whole process of our algorithm using the manual dataset Aggregation [23] as an example. Aggregation has a total of 788 samples, and there

were 7 correctly classified clusters. The results of each step in the clustering procedure are clearly shown in Figure 3 (algorithm parameters $K = 11$, $k = 11$).

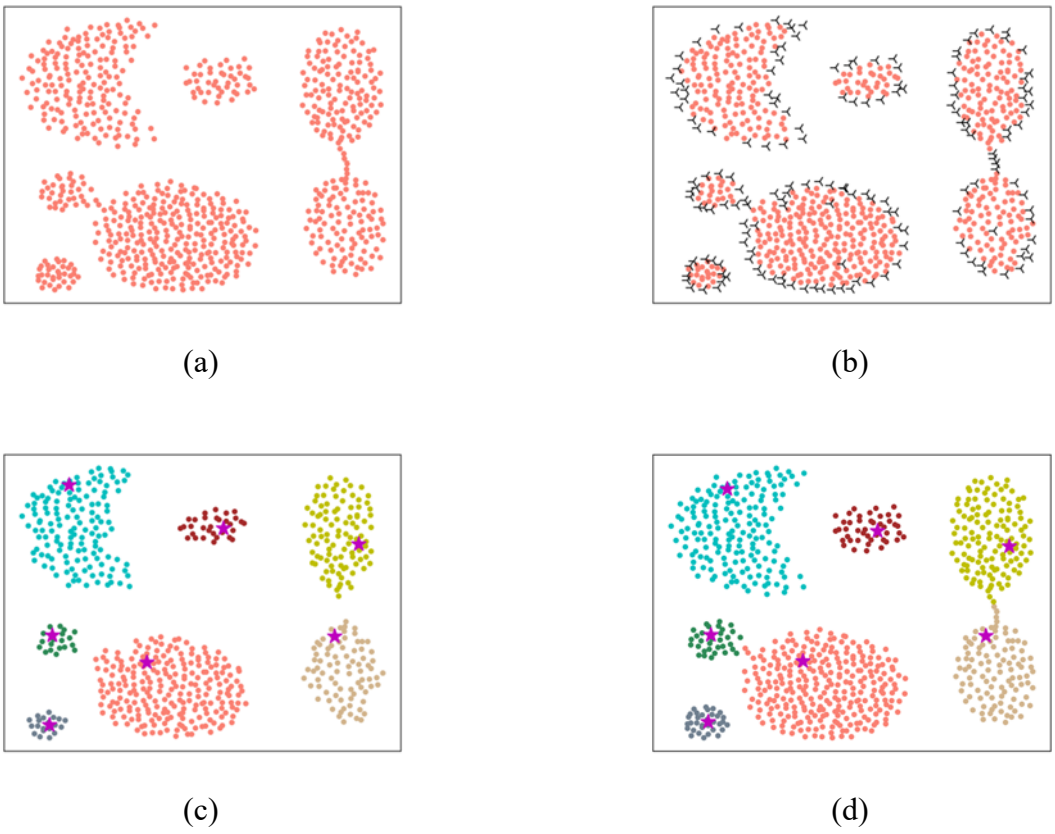


Figure 3. Clustering process on the aggregation dataset. (a) Original distribution of data; (b) Core and sample points; (c) Clustering results on core points; (d) Final results of clustering on AKDPC.

3.5. Time complexity analysis of AKDPC

Assume that there are n samples in the dataset, and then the n samples are divided into a core and b non-core points. The complexity of our method depends mainly on the following six aspects.

1) The complexity of computing *MK* values for each sample is O(kn);
2) The complexity of classifying samples based on *MKT* values is O(n);
3) The complexity of the calculation of the D^k value for each sample is O(n);
4) The complexity of defining clustering centers and clustering the core points is $O(a^2)$;
5) The complexity of distributing non-core points is O(ab);
6) The AKDPC has an overall time complexity of $O(n^2)$ based on the analysis above, which is the same as the original DPC.

4. Experiments

To prove the feasibility and validity of our suggested AKDPC algorithm, we used DPC and its improved algorithm DPC-KNN and DBSCAN for comparison. Since DBSCAN is a classical density-

based clustering algorithm, and DPC-KNN is similar to this paper in that it is based on KNN to improve the original DPC, using them as a comparison can demonstrate the superiority of the AKDPC algorithm. Experiments were performed on eight classical manual datasets and eight real datasets obtained from [24,25]. To verify the performance of the algorithms for different density distributions and shapes, we used seven datasets that can represent different situations such as Aggregation, Jain, Flame [26], ThreeCircles, Pathbased [27], D9 [28], T4, etc., while the number of clusters and features of the samples are also different for each of the eight real datasets, for which experiments can be conducted to verify the generalizability of AKDPC. The detailed attribute tables for the data are given in Table 2. The source code and datasets used in this section are available at https://github.com/wanghuani/AKDPC.

Table 2. Details of the manual dataset and real dataset.

Dataset	Instances	Dimensions	Clusters
Manual			
Aggregation	788	2	7
Jain	373	2	2
Flame	240	2	2
ThreeCircles	299	2	3
Pathbased	300	2	3
D9	1400	2	4
T4	8000	2	6
Real			
Zoo	101	16	7
Thyroid	215	6	3
Wine	178	13	3
Wdbc	569	30	2
Vote	299	2	2
Pima	768	9	2
Diabetes	768	8	2
Ecoli	336	8	8

We uniformly used adjusted rand index (ARI [29]), normalized mutual information (NMI [30]), and clustering accuracy (ACC) as performance evaluation metrics throughout the experiments, with an upper limit of 1 for the metrics and higher values indicating better performance. Among them, ACC is one of the most commonly used clustering performance evaluation metrics, defined as Eq (11).

$$ACC = \frac{\sum_{i=1}^{N} \delta(y_i, map(z_i))}{N} \tag{11}$$

where y_i is the true label, z_i is the label after clustering, and map(.) represents the reassignment of clustering labels, which is generally implemented using the Hungarian algorithm.

ARI is an adjusted RI. ARI measures the degree of agreement between two data distributions, and Eqs (12) and (13) show the calculation process of RI and ARI:

$$RI = \frac{a+b}{C_2^n} \tag{12}$$

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)} \tag{13}$$

where Y represents the actual label, Z represents the clustering result, C_2^n represents the total number of element pairs that can be composed in the data set, a represents the number of elements in both Y and Z that are of the same category, and b represents the number of elements that are not of the same category, while E represents the expectation.

NMI is also a commonly used measure of the similarity between the clustering results and the true results, as defined by Eq (14).

$$NMI(Y,Z) = 2 * I(Y,Z) / (H(Y) + H(Z) \tag{14}$$

where Y represents the true category, Z represents the result after clustering, H represents the cross entropy, and $I(Y,Z)$ represents the mutual information.

To ensure the fairness of the experiment, all datasets used before starting the experiment were mapped linearly to the range [0,1] by pre-processing.

In this paper, all parameters of the four algorithms are tuned to make sure that the overall experiment demonstrates the best possible clustering for each algorithm. AKDPC requires two parameters, K and k, taking values between 1 and 100. In addition, according to experience, for some simple low-dimensional datasets, the parameters K and k are generally chosen around 10–20, and the same values can be taken in most cases. Meanwhile, for data sets with more samples, the performance of AKDPC is generally better by choosing larger values of K and k. At the same time, we both need to choose a cut-off distance dc for the DPC and DPC-KNN, and empirically we choose a dc that is 1–2% after we sort the data points in ascending order by distance from each other. Furthermore, DPC-KNN requires manual determination of the number of the nearest neighbor K, which we choose from 4 to 15. As for the two parameters of DBSCAN, we choose ε between 0.01 and 2, and *minpts* between 1 and 50. Finally, for both DPC-KNN and DPC algorithms, we manually set the number of clusters.

4.1. Experiments with artificial datasets

In this part, we have selected the classical manual datasets which were used to test various clustering algorithms. The final clustering outcomes of the manual dataset are given in Table 3. Just as Table 3 shows, AKDPC shows near-perfect results for all types of datasets, while other algorithms are not as generalizable as AKDPC. For example, DPC is unable to handle large density differences and some complex non-convex datasets, while DPC-KNN generally does better than DPC on these datasets overall but also has the problem of being unable to handle some complex non-convex datasets. DBSCAN shows good results in processing each dataset though, but it tends to misidentify some samples as noise points. The overall effect is not as good as AKDPC.

Table 3. Clustering outcomes on manual datasets.

Dataset	Algorithm	ACC	ARI	NMI	Arg
Aggregation	AKDPC	**0.9975**	**0.9956**	**0.9924**	11/11
	DPC	0.9975	0.9956	0.9924	0.062
	DBSCAN	0.9797	0.975	0.9707	0.05/10
	DPC-KNN	0.9975	0.9956	0.9924	7/0.062
Jain	AKDPC	**1**	**1**	**1**	17/17
	DPC	0.8954	0.6183	0.577	0.0424
	DBSCAN	9732	0.9731	0.9178	0.08/5
	DPC-KNN	1	1	1	7/1.08
Flame	AKDPC	**1**	**1**	**1**	11/11
	DPC	1	1	1	0.09
	DBSCAN	0.9208	0.8607	0.7923	0.07/6
	DPC-KNN	1	1	1	7/0.1
ThreeCircles	AKDPC	**1**	**1**	**1**	11/11
	DPC	0.408	-0.001	0.1123	0.0266
	DBSCAN	1	1	1	0.09/4
	DPC-KNN	0.4916	0.1089	0.2019	8/0.0266
Pathbased	AKDPC	**0.9933**	**0.9798**	**0.9659**	12/17
	DPC	0.7333	0.453	0.539	0.0545
	DBSCAN	0.6533	0.5319	0.675	0.08/10
	DPC-KNN	0.74	0.4572	0.5425	8/0.0545
D9	AKDPC	**0.9971**	**0.99**	**0.9772**	35/55
	DPC	0.3836	0.0236	0.2701	0.0375
	DBSCAN	0.9593	0.9108	0.8844	0.05/6
	DPC-KNN	0.4871	0.2128	0.4678	6/0.0375
T4	AKDPC	**0.993**	**0.9849**	**0.9747**	70/20
	DPC	0.7013	0.6027	0.7337	0.0653
	DBSCAN	0.9156	0.8808	0.8828	0.02/15
	DPC-KNN	0.6721	0.5651	0.709	8/0.0653

We have visualized the final clustering results in Figures 4–10, and we have used asterisks to indicate the cluster centers obtained by other algorithms, except for the DBSCAN algorithm. In Figure 4, DPC, DPC-KNN and AKDPC can identify the structure of aggregated datasets consisting of arbitrarily distributed non-spherical clusters, and for DBSCAN the general shape of each cluster is correct although some points are labeled as noise.

The result of the clustering of the classic dataset Jain is shown in Figure 5. Since DPC always prefers to select clustering centers in high-density areas, it cannot detect clustering centers in sparse areas above the dataset Jain, which leads to wrong results in the end. At the same time, DBSCAN incorrectly classifies the left side of the top sparse region as a new cluster, while identifying some points on the right end as noise. In contrast, both AKDPC and DPC-KNN identified density peaks distributed over different density regions.

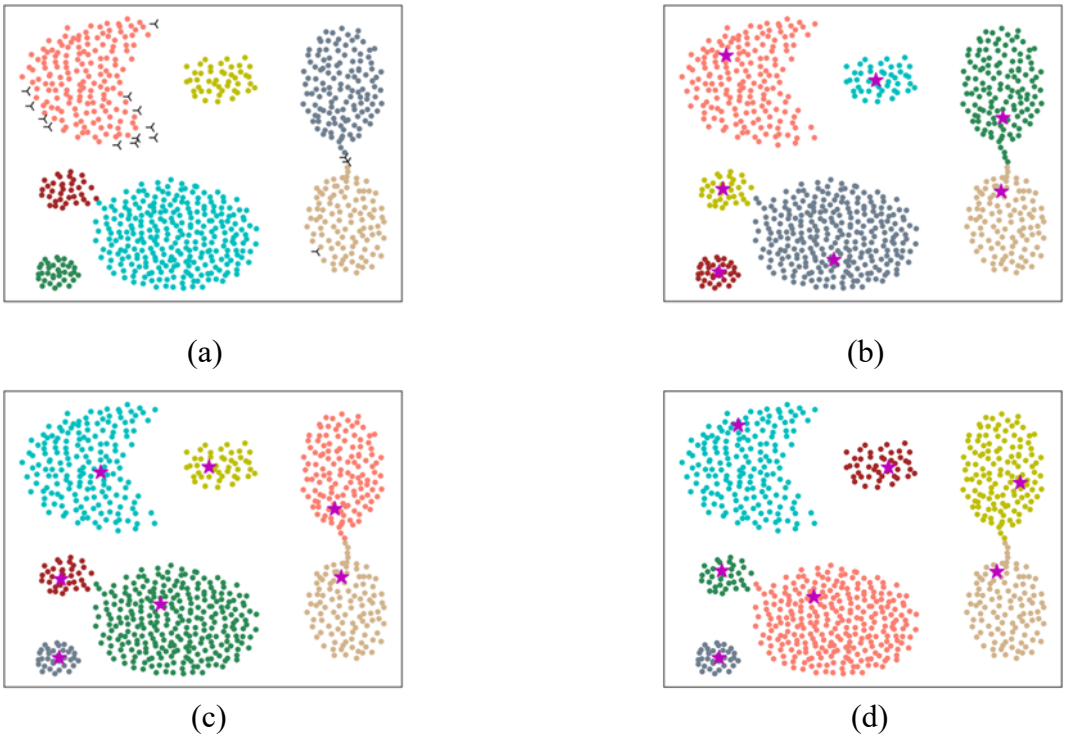


Figure 4. The clustering results over the Aggregation dataset. (a) DBSCAN; (b) DPC; (c) DPC-KNN; (d) AKDPC.

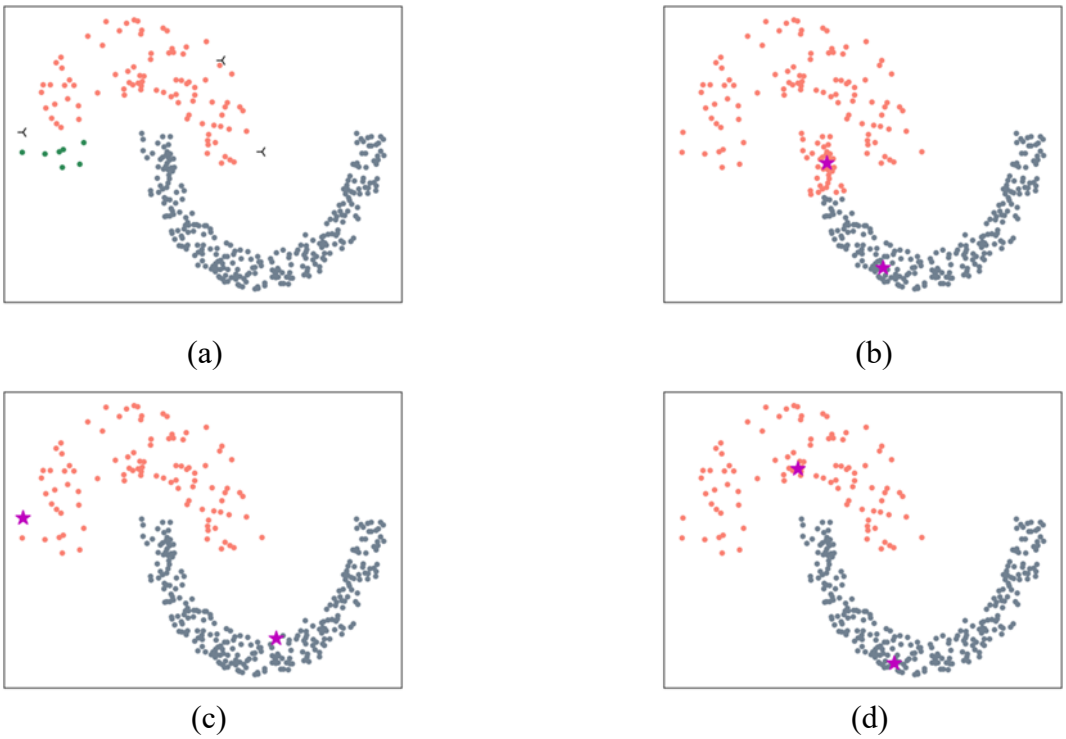


Figure 5. The clustering results over the Jain dataset. (a) DBSCAN; (b) DPC; (c) DPC-KNN; (d) AKDPC.

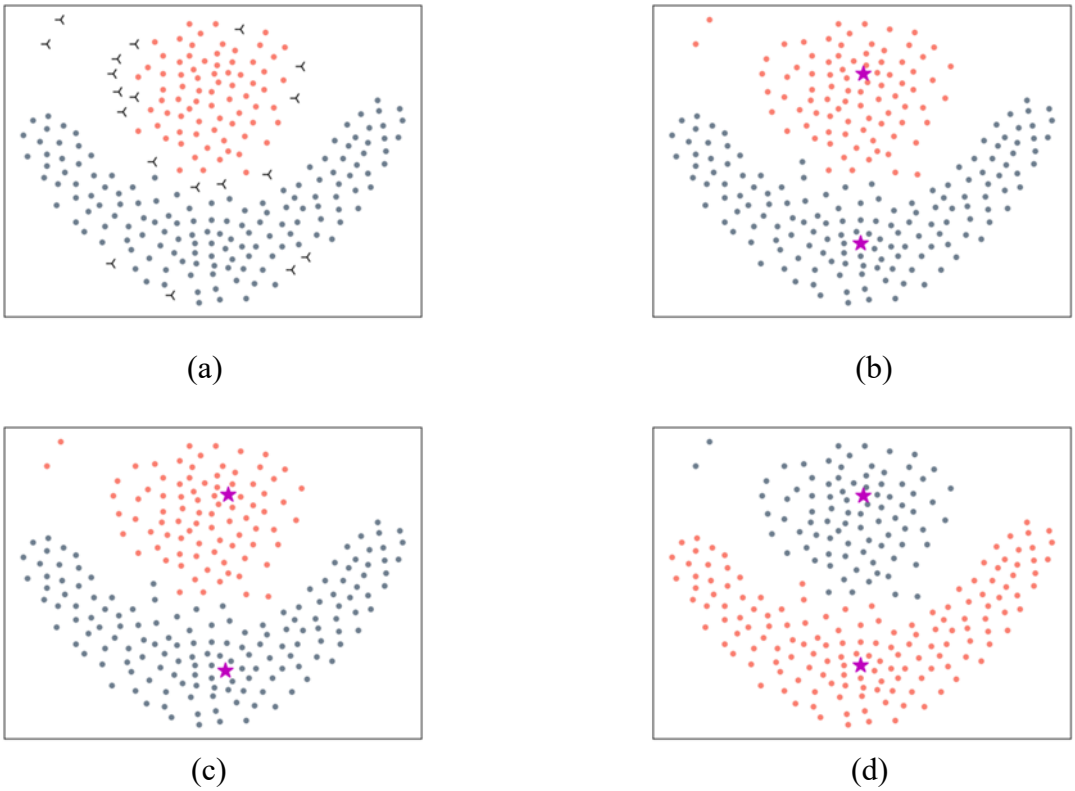


Figure 6. The clustering results over the Flame dataset. (a) DBSCAN; (b) DPC; (c) DPC-KNN; (d) AKDPC.

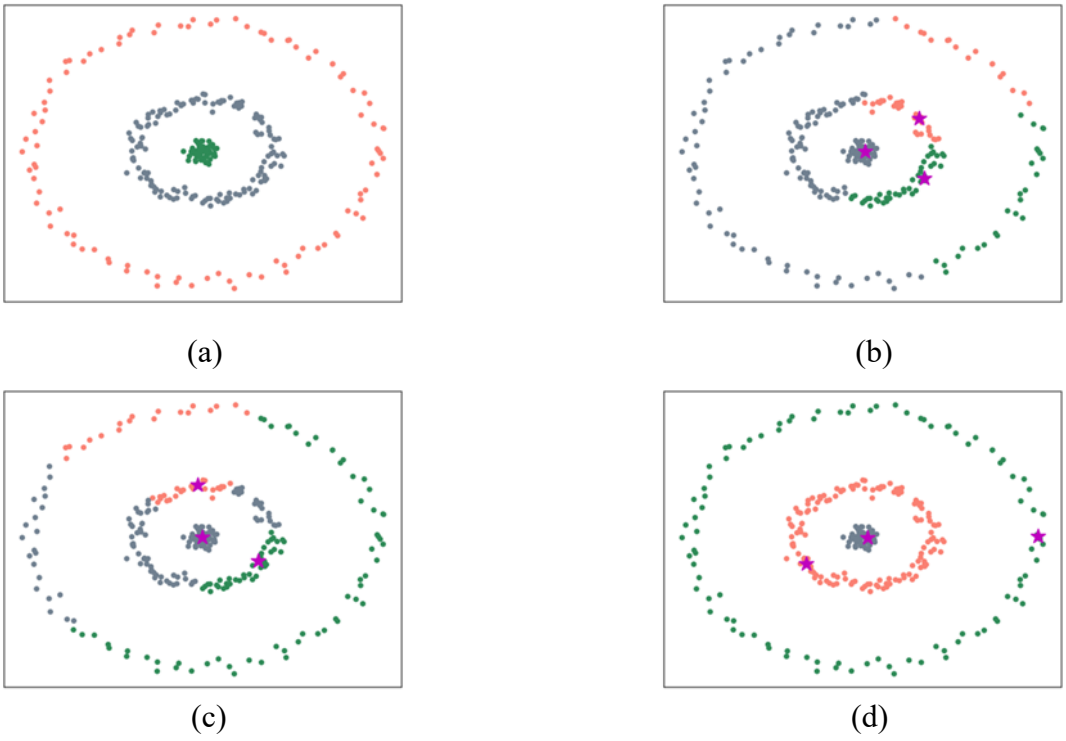


Figure 7. The clustering results over the ThreeCircles dataset. (a) DBSCAN; (b) DPC; (c) DPC-KNN; (d) AKDPC.

The results of the Flame dataset are presented in Figure 6, where DPC, DPC-KNN and AKDPC all effectively aggregate the two clusters, while DBSCAN correctly detects both clusters, but it incorrectly identifies some edge points as noise, which affects the outcome of the clustering.

The clustering results of the clustering of the ThreeCircles dataset are presented in Figure 7. As the graph shows, DPC and DPC-KNN select three clustering centers in the middle two rings, making the final outer ring without a center, resulting in the final incorrect clustering result, but DBSCAN correctly identifies three clusters, while our algorithm AKDPC not only identifies the clustering centers of the three rings. The final clustering result is also completely correct.

Figure 8 displays the clustering results for the Pathbased dataset. For this typical non-convex dataset, DPC and DPC-KNN are not performing well. DPC and DPC-KNN correctly identify the three clustering centers, but they wrongly assign the left and right sides of the dataset to the two clusters in the middle, while DBSCAN correctly identifies only two clusters, leaving the remaining samples in the outer sparse region as noise. On this dataset, only AKDPC not only succeeded in identifying three clustering centers, but the final clustering result was also relatively perfect.

As shown in Figure 9, dataset D6 has many discrete points, which often have a significant impact on algorithm performance. Not surprisingly, DPC and DPC-KNN were not up to the task of processing such a large curvature dataset, and only DBSCAN and AKDPC succeeded in identifying the four clusters, with DBSCAN having the slight flaw of identifying almost all of the discrete points as noise.

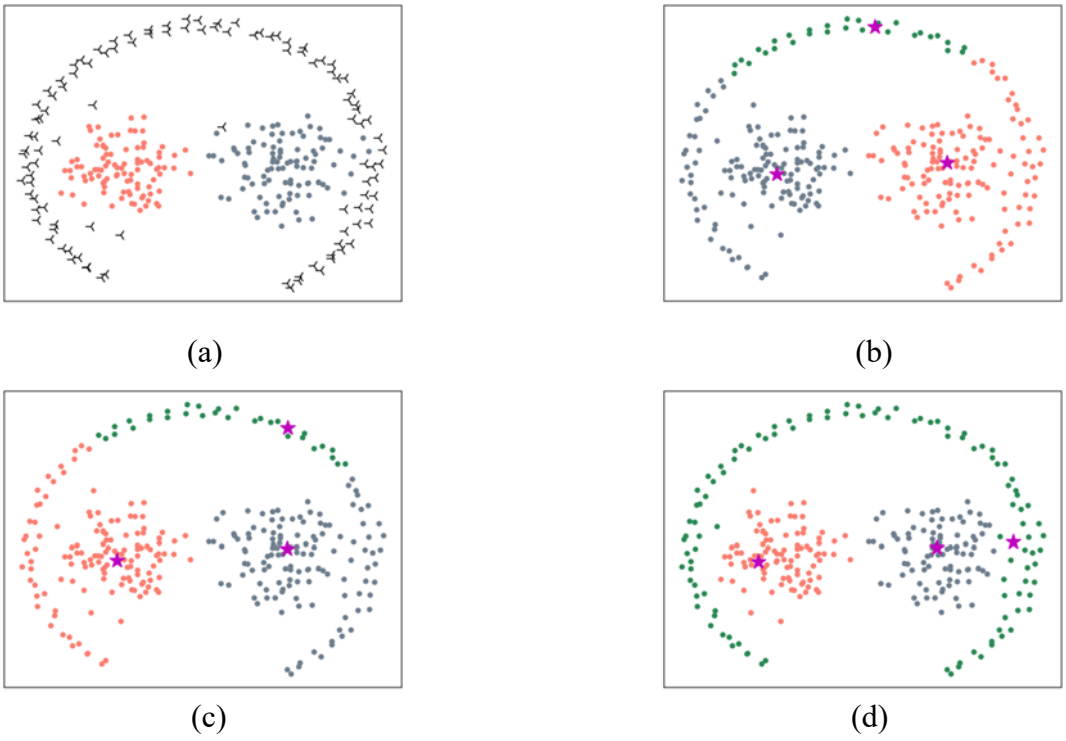


Figure 8. The clustering results over the Pathbased dataset. (a) DBSCAN; (b) DPC; (c) DPC-KNN; (d) AKDPC.

The results of the T4 dataset for all algorithms are shown in Figure 10, which contains six clusters, most of which are cross-tangled, and many discrete points at the edges of each cluster, testing the ability of the algorithms to handle complex datasets. While DBSCAN and AKDPC showed their performance in adapting to various complex situations and successfully identified six clusters of different shapes, DPC and DPC-KNN showed less satisfactory results on this complex dataset with large cross-tangles.

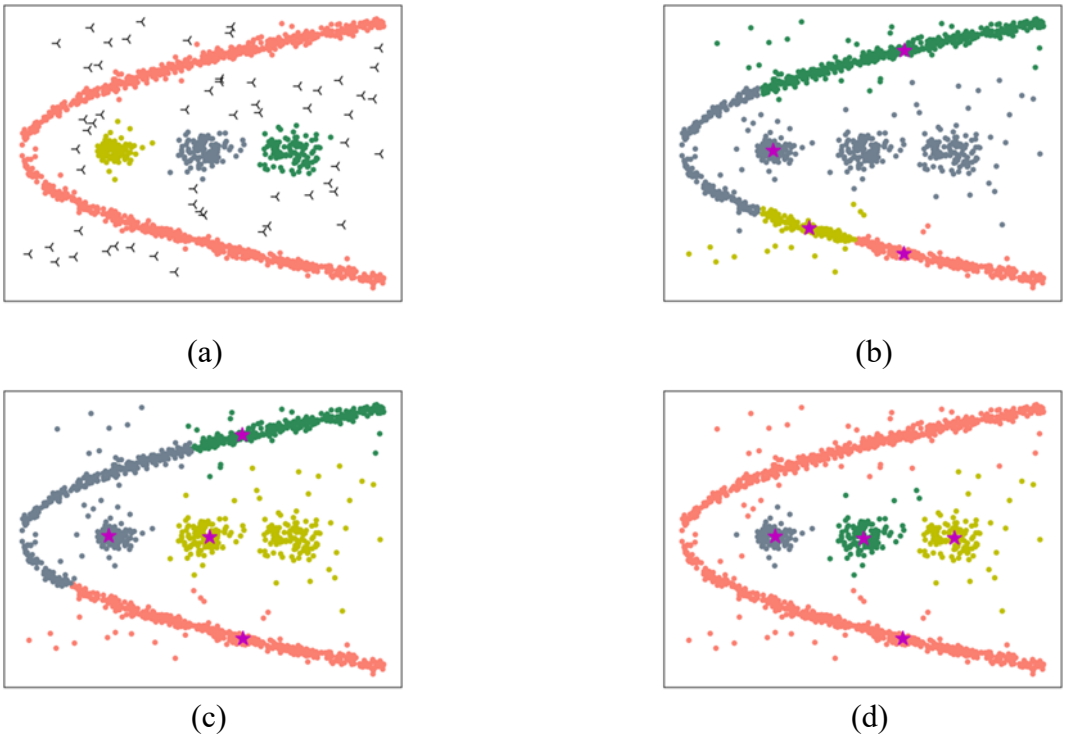


Figure 9. The clustering results over the D9 dataset. (a) DBSCAN; (b) DPC; (c) DPC-KNN; (d) AKDPC.

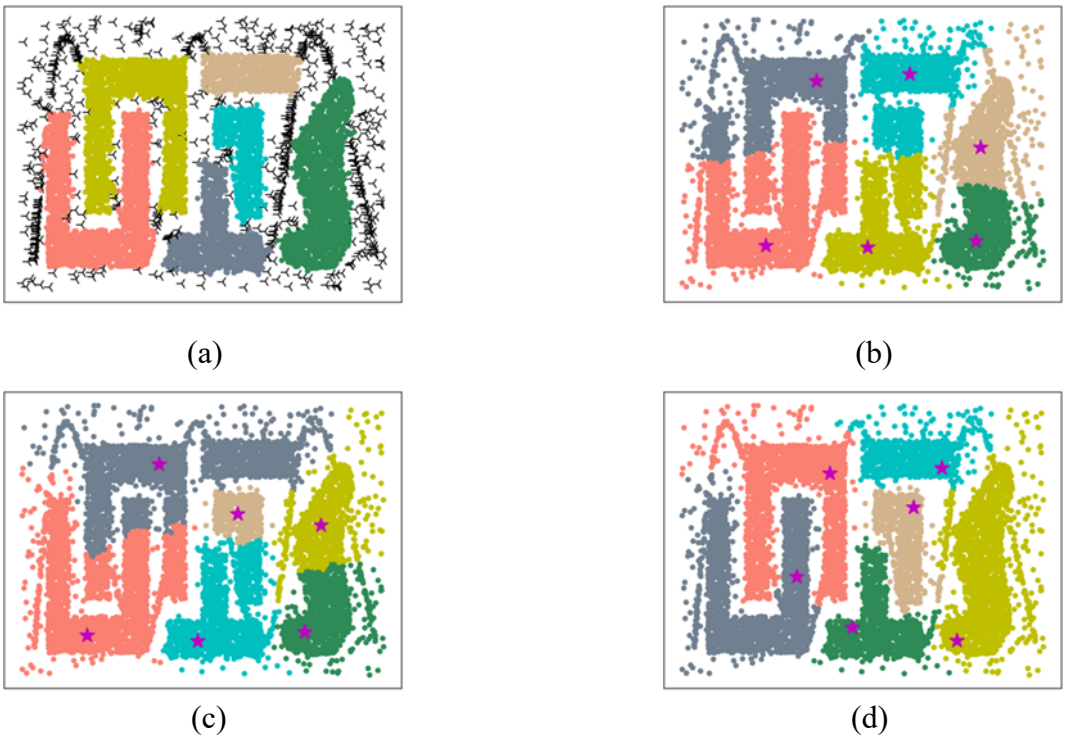


Figure 10. The clustering results over the T4 dataset. (a) DBSCAN; (b) DPC; (c) DPC-KNN; (d) AKDPC.

Table 4. Clustering outcomes on real datasets.

Dataset	Algorithm	ACC	ARI	NMI	Arg
Zoo	AKDPC	0.8614	0.9249	0.8885	11/15
	DPC	0.6337	0.4972	0.7219	1
	DBSCAN	0.8812	0.9007	0.8584	1.1/5
	DPC-KNN	0.6733	0.6043	0.7815	9/1
Thyroid	AKDPC	0.8884	0.7679	0.6688	20/12
	DPC	0.5535	0.144	0.2819	0.676
	DBSCAN	0.8372	0.7932	0.6405	0.13/32
	DPC-KNN	0.5674	0.1388	0.3035	2/0.676
Wine	AKDPC	0.8652	0.7096	0.7014	18/10
	DPC	0.882	0.6723	0.7104	0.4165
	DBSCAN	0.8146	0.5292	0.5905	0.5/21
	DPC-KNN	0.8876	0.6855	0.7181	5/0.4165
Wdbc	AKDPC	0.9244	0.7529	0.6361	58/8
	DPC	0.6203	-0.0056	0.0094	0.3528
	DBSCAN	0.8383	0.4501	0.3376	0.46/38
	DPC-KNN	0.8559	0.5047	0.3932	4/0.3528
Vote	AKDPC	0.9034	0.6502	0.5706	39/26
	DPC	0.8759	0.5641	0.5059	0.7071
	DBSCAN	0.8	0.465	0.387	1.0/24
	DPC-KNN	0.8989	0.6354	0.5534	6/0.7071
Pima	AKDPC	0.638	0.0486	0.0501	16/8
	DPC	0.6185	0.0146	0.002	0.2243
	DBSCAN	0.651	0	0	1.4/4
	DPC-KNN	0.6875	0.1226	0.0611	2/0.2243
Diabate	AKDPC	0.6263	0.0642	0.0639	17/7
	DPC	0.6185	0.0146	0.002	0.2243
	DBSCAN	0.6901	0.1184	0.0584	0.3/31
	DPC-KNN	0.6615	0.0495	0.0201	7/0.2243
Ecoli	AKDPC	0.7649	0.6958	0.6663	23/7
	DPC	0.497	0.3086	0.4973	0.1568
	DBSCAN	0.6458	0.5255	0.5055	0.2/22
	DPC-KNN	0.5417	0.4323	0.59	2/0.1568

4.2. Experiments with real datasets

We further demonstrate the excellent capability of the AKDPC clustering algorithm using eight real datasets. Table 4 displays the clustering performance for the ACC, ARI and NMI metrics on the real datasets. Bold text within the table denotes the optimum results of the same dataset. As we can see from Table 4, no algorithm can always outperform the other algorithms due to the diversity of real datasets with different data, but we can also observe from Table 4 that in most situations, AKDPC does better, indicating that the AKDPC algorithm is more generalizable and can handle datasets with different situations.

5. Discussion

To address the weaknesses of the original DPC, we propose a density peak clustering algorithm for the automatic selection of clustering centers based on K-nearest neighbors. We use relationships between samples and their surrounding neighbors to classify the sample into core and non-core samples while eliminating any possible adverse effects of non-core samples on the final clustering results by batch clustering. At the same time, we use the information on the distance of the sample from its surrounding neighbors to replace the density of samples in the original DPC. Through a novel sequential clustering method, we solve the problem of the former DPC requiring human intervention to determine the clustering centers and can also achieve better performance on complex datasets with large differences in density or non-convexity. Of course, our algorithm AKDPC has some drawbacks. First, we need to manually determine two parameters K and k. Second, we assign non-core points by labeling them as the same as the nearest points that have been classified and do not consider the effect of cluster shape, which may lead to incorrect assignments.

In future research we will try to reduce the number of parameters while maintaining the current accuracy, and for unassigned non-core points, we will propose a new method to reduce incorrect assignments. Finally, we will try to combine it with some supervised algorithms to test it with some excellent variants of the KNN algorithm used in this paper, such as CDNN [31,32] and ECDNN [33], and explore them with respect to their application in some related research [34].

Acknowledgments

The financial support for this project was provided by the National Natural Science Foundation of China [61962054].

Conflict of interest

The authors declare no conflict of interest.

References

1. Z. Chen, Z. Qi, F. Meng, L. Cui, Y. Shi, Image segmentation via improving clustering algorithms with density and distance, *Procedia Comput. Sci.*, **55** (2015), 1015–1022. https://doi.org/10.1016/j.procs.2015.07.096
2. Q. Zhao, X. Li, Y. Li, X. Zhao, A fuzzy clustering image segmentation algorithm based on hidden Markov random field models and Voronoi tessellation, *Pattern Recognit. Lett.*, **85** (2017), 49–55. https://doi.org/10.1016/j.patrec.2016.11.019
3. X. Zeng, A. Chen, M. Zhou, Color perception algorithm of medical images using density peak based hierarchical clustering, *Biomed. Signal Process. Control*, **48** (2019), 69–79. https://doi.org/10.1016/j.bspc.2018.09.013
4. J. Gao, M. T. Chang, H. C. Johnsen, S. P. Guo, B. E. Sylvester, S. O. Sumer, et al., 3D clusters of somatic mutations in cancer reveal numerous rare mutations as functional targets, *Genome Med.*, **9** (2017), 1–13. https://doi.org/10.1186/s13073-016-0393-x

5. J. W. Wu, J. C. Tseng, W. N. Tsai, A hybrid linear text segmentation algorithm using hierarchical agglomerative clustering and discrete particle swarm optimization, *Integr. Comput.-Aided Eng.*, **21** (2014), 35–46. https://doi.org/10.3233/ICA-130446
6. A. Sapountzi, K. E. Psannis, Social networking data analysis tools & challenges, *Future Gener. Comput. Syst.*, **86** (2018), 893–913. https://doi.org/10.1016/j.future.2016.10.019
7. X. Cai, X. Z. Gao, Y. Xue, Improved bat algorithm with optimal forage strategy and random disturbance strategy, *Int. J. Bio-Inspired Comput.*, **8** (2016), 205–214. https://doi.org/1504.2016/IJBIC.078666
8. Q. Zou, G. Lin, X. Jiang, X. Liu, X. Zeng, Sequence clustering in bioinformatics: an empirical study, *Briefings Bioinf.*, **21** (2020), 1–10. https://doi.org/1093.090/bib/bby
9. J. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proc. 5th Berkeley Symposium on Math., Stat., and Prob*, (1965), 281.
10. T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, *ACM Sigmod Record*, **25** (1996), 103–114. https://doi.org/10.1145/235968.233324
11. M. Ester, H. P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in *kdd*, **96** (1996), 226–231.
12. G. Sheikholeslami, S. Chatterjee, A. Zhang, WaveCluster: a wavelet-based clustering approach for spatial data in very large databases, *VLDB J.*, **8** (2000), 289–304. https://doi.org/10.1007/s007780050009
13. U. Von Luxburg, A tutorial on spectral clustering, *Stat. Comput.*, **17** (2007), 395–416.
14. A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, *Science*, **344** (2014), 1492–1496. https://doi.org/10.1126/science.1242072
15. Y. Liu, Z. Ma, F. Yu, Adaptive density peak clustering based on K-nearest neighbors with aggregating strategy, *Knowledge-Based Syst.*, **133** (2017), 208–220. https://doi.org/10.1016/j.knosys.2017.07.010
16. Z. Guo, T. Huang, Z. Cai, W. Zhu, A new local density for density peak clustering, in *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3–6, 2018, Proceedings, Part III 22*, (2018), 426–438. https://doi.org/10.1007/978-3-319-93040-4_34
17. W. Zhou, L. Wang, X. Han, M. Parmar, M. Li, A novel density deviation multi-peaks automatic clustering algorithm, *Complex Intell. Syst.*, **9** (2023), 177–211. https://doi.org/10.1007/s40747-022-00798-3
18. J. Xie, H. Gao, W. Xie, X. Liu, P. W. Grant, Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors, *Inf. Sci.*, **354** (2016), 19–40. https://doi.org/10.1016/j.ins.2016.03.011
19. R. Liu, H. Wang, X. Yu, Shared-nearest-neighbor-based clustering by fast search and find of density peaks, *Inf. Sci.*, **450** (2018), 200–226. https://doi.org/10.1016/j.ins.2018.03.031
20. H. Yu, L. Chen, J. Yao, A three-way density peak clustering method based on evidence theory, *Knowledge-Based Syst.*, **211** (2021), 106532. https://doi.org/10.1016/j.knosys.2020.106532
21. J. Jiang, Y. Chen, X. Meng, L. Wang, K. Li, A novel density peaks clustering algorithm based on k nearest neighbors for improving assignment process, *Physica A*, **523** (2019), 702–713. https://doi.org/10.1016/j.physa.2019.03.012

22. A. K. Jain, M. H. Law, Data clustering: A user's dilemma, in *Pattern Recognition and Machine Intelligence: First International Conference, PReMI 2005, Kolkata, India, December 20–22, Proceedings 1*, (2005), 1–10. https://doi.org/10.1007/11590316_1
23. A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, *ACM Trans. Knowl. Discovery Data*, **1** (2007), 4-es. https://doi.org/10.1145/1217299.1217303
24. D. Dua, C. Graff, *UCI Machine Learning Repository*, 2017. Available from: https://archive.ics.uci.edu/ml.
25. W. N. Street, W. H. Wolberg, O. L. Mangasarian, Nuclear feature extraction for breast tumor diagnosis, in *Biomedical Image Processing and Biomedical Visualization*, **1905** (1993), 861–870. https://doi.org/10.1117/12.148698
26. L. Fu, E. Medico, FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data, *BMC Bioinf.*, **8** (2007), 1–15. https://doi.org/10.1186/1471-2105-8-3
27. H. Chang, D. Y. Yeung, Robust path-based spectral clustering, *Pattern Recognit.*, **41** (2008), 191–203. https://doi.org/10.1016/j.patcog.2007.04.010
28. Q. Z. Dai, Z. Y. Xiong, J. Xie, X. Wang, Y. Zhang, J. Shang, A novel clustering algorithm based on the natural reverse nearest neighbor structure, *Inf. Syst.*, **84** (2019), 1–16. https://doi.org/10.1016/j.is.2019.04.001
29. J. M. Santos, M. Embrechts, On the use of the adjusted rand index as a metric for evaluating supervised classification, in *Artificial Neural Networks—ICANN 2009: 19th International Conference, Limassol, Cyprus, September 14–17, 2009, Proceedings, Part II 19*, (2009), 175–184. https://doi.org/10.1007/978-3-642-04277-5_18
30. A. F. McDaid, D. Greene, N. Hurley, Normalized mutual information to evaluate overlapping community finding algorithms, preprint, arXiv:11102515.
31. B. P. Nguyen, W. L. Tay, C. K. Chui, Robust biometric recognition from palm depth images for gloved hands, *IEEE Trans. Hum.-Mach. Syst.*, **45** (2015), 799–804. https://doi.org/10.1109/THMS.2015.2453203
32. A. X. Wang, S. S. Chukova, B. P. Nguyen, Implementation and analysis of centroid displacement-based k-nearest neighbors, in *Advanced Data Mining and Applications: 18th International Conference, ADMA 2022, Brisbane, QLD, Australia, November 28–30, 2022, Proceedings, Part I*, (2022), 431–443. https://doi.org/10.1007/978-3-031-22064-731
33. A. X. Wang, S. S. Chukova, B. P. Nguyen, Ensemble k-nearest neighbors based on centroid displacement, *Inf. Sci.*, **629** (2023), 313–323. https://doi.org/10.1016/j.ins.2023.02.004
34. K. Liu, Z. Li, C. Yao, J. Chen, K. Zhang, M. Saifullah, Coupling the k-nearest neighbor procedure with the Kalman filter for real-time updating of the hydraulic model in flood forecasting, *Int. J. Sediment Res.*, **31** (2016), 149–158. https://doi.org/10.1016/j.ijsrc.2016.02.002

AIMS Press