A Novel Clustering Algorithm based on Directional Propagation of Cluster Labels

1st Na Xiao
School of Information Science and Engineering
Hunan University
Changsha, China
naxiao@hnu.edu.cn

2nd Kenli Li
School of Information Science and Engineering
Hunan University
Changsha, China
lkl@hnu.edu.cn

3rd Xu Zhou
School of Information Science and Engineering
Hunan University
Changsha, China
happypanda2006@126.com

4th Keqin Li
Department of Computer Science
State University of New York
New Paltz, NY 12561, USA
lik@newpaltz.edu

***Abstract*—Clustering is an important topic in the field of machine learning. There are abundant algorithms which are proposed for clustering, and they are mainly based on two basic physical metrics, distance and density. However, it is difficult to reflect the orientation relationship between data by distance and density alone, while the orientation relationship can be easily expressed by direction. Inspired by this, we regard direction as the main basic physical metric, and propose a clustering algorithm based on directional propagation of cluster labels, namely DBC. DBC doesn't need to make specific assumptions about the density of data, but uses the orientation relationship between data to help clustering. Similar to the density-based clustering algorithms DBSCAN (Density Based Spatial Clustering Of Applications with Noise) and DPC (Density Peak Clustering), DBC can also find clusters of arbitrary shapes. It requires two parameters, but its parameter selection is easier than DBSCAN. Moreover, it is less affected by uneven density distribution than DBSCAN and DPC. We compare DBC with four well-known clustering algorithms, including DPC, DBSCAN, AP (Affinity Propagation) and K-means++. Experiments on artificial data sets show that DBC performs better than DBSCAN and DPC on data sets with uneven density distribution, and can effectively identify clusters with overlapping regions. Experiments on real-world data sets show that DBC has advantages over DPC, AP and K-means++ in clustering effect.**

I. Introduction

Clustering, as a data analysis technology, plays an important role in machine learning. Its goal is to divide a collection of data into meaningful clusters based on their similarity so that the data in the same cluster have a high degree of similarity while the data in different clusters shows a high degree of dissimilarity [1]. So far, many kinds of clustering algorithms have been proposed, including hierarchical algorithms [2], partition-based algorithms [3], density-based algorithms [4], and graph-based algorithms [5].

Despite the wide variety of clustering algorithms, most of them rely on two basic physical metrics, distance and density. Through these two physical metrics, they can obtain the necessary information to help clustering. Although distance and density play an important role in clustering algorithms, they also have certain limitations. Distance and density alone are difficult to reflect the orientation relationship between data, while the orientation relationship is an important factor to reflect the distribution of data. For example, in two-dimensional space, we can roughly describe the orientation relationship between two points in four directions: east, south, west, and north. If we only know the distance between the two points and their density, we cannot know which direction one point is located in of the other point. In order to effectively utilize the orientation relationship between data in the clustering process, we regard direction as the major basic physical metric, as the orientation relationship can be easily expressed by direction. And based on the idea that the propagation of cluster labels has directionality, we propose a new direction-based clustering algorithm, namely DBC.

Like DBSCAN (Density Based Spatial Clustering Of Applications with Noise) [4] and DPC (Density Peak Clustering) [6], DBC has the ability to discover clusters of arbitrary shapes. It doesn't make specific assumptions about the density of data, and it is less affected by uneven density distribution than DBSCAN and DPC. It requires two parameters, but its parameters are easier to select than those of DBSCAN. In addition, it is based on a simple clustering strategy. The rest of the paper is divided into four sections. In Section 2, we give a brief introduction to DBSCAN and DPC, mainly from the perspective of principles and existing problems. During the experiments, our algorithm will make an important comparison with these two algorithms. In Section 3, we describe the direction-based clustering algorithm DBC in detail. Section 4 is the experimental part. We test the clustering performance of DBC through different types of data sets. In section 5, we conclude the paper and discuss the direction of our future work.

II. Related Work

In this section, we present a brief introduction to the density-based clustering algorithms DBSCAN and DPC. DBSCAN and DPC are two well-known representatives of density-based clustering algorithms. They have some similarities with DBC. Firstly, all three algorithms are based on simple clustering strategies. Secondly, all three algorithms are based on two basic physical metrics. DBSCAN and DPC are based on distance and density, while DBC is based on distance and direction. Finally, all three algorithms can find clusters of arbitrary shapes. Therefore, comparing DBC with DBSCAN and DPC in the experiment is helpful for evaluating the clustering effect of DBC.

A. Density Based Spatial Clustering Of Applications with Noise

DBSCAN [4] was proposed by Ester et al. in 1996. It divides data into core points and border points according to the density threshold. It has two parameters, Eps and $Minpts$. Eps refers to the neighborhood radius, and $MinPts$ refers to the minimum number of points a core point should have in its neighborhood, which is the density threshold of DBSCAN. By randomly selecting a core point and traversing all points which are density-reachable [4] from it, DBSCAN can find a cluster. In this way, DBSCAN discovers all clusters one by one. Its problems are mainly reflected in four aspects. First, DBSCAN has difficulty in selecting parameters [6]. Second, subtle changes in parameters may cause large changes in clustering results [1]. Third, in some data sets with uneven density distribution, it cannot get the correct clustering results [7]. Fourth, it is not suitable for high dimensional data space [8].

B. Density Peak Clustering

DPC [6] is a new clustering algorithm proposed by Rodriguez et al. in 2014 and published on science. DPC is based on two simple assumptions. First, it believes that each cluster has a cluster center surrounded by lower density points and far away from higher density points. Second, it believes that a point should be in the same cluster as its nearest higher density point. The algorithm flow of DPC can be divided into the following steps:

1) Calculate the density ρ of each point using either (1) or (2).

$$\rho_i = \sum_{j \neq i} \chi\left(d_{ij} - d_c\right) \tag{1}$$

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

   In (1) and (2), ρ_i represents the density of point i, d_{ij} represents the distance between point i and point j, and d_c is the cutoff distance. $\chi(x)$ is such a function that if $x < 0$, its value is 1, otherwise, its value is 0.
2) Calculate the distance between each point and its nearest higher density point, denoted by δ. It is worth noting that the δ value of the highest density point is the furthest distance from it to other points in the dataset.
3) Create a decision graph based on ρ and δ.
4) Select the points with unusually large ρ and δ as cluster centers, which usually appear in the upper right corner of the decision graph. In DPC, a cluster center represents a cluster.
5) Assign each remaining point to the cluster to which its nearest higher density point belong.

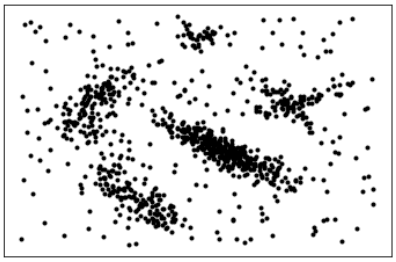


Fig. 1: A simple data set

Compared with DBSCAN, DPC's parameter selection is easier and the clustering results are more robust [6]. Its problems are mainly reflected in two aspects. First, it is difficult for DPC to obtain the correct clustering results in some data sets with uneven density distribution [1]. Second, it may divide a cluster into multiple clusters, if there are multiple points in the cluster that satisfy the condition of becoming a cluster center [9].

III. DBC: A Direction Based Clustering Algorithm

A. Basic Idea

In data sets, a common phenomenon is that a point has neighbors distributed in multiple directions. For a point, if its neighbors are unevenly distributed in direction, it means that there are more neighbors in some directions and fewer neighbors in other directions. This situation is especially evident for boundary data. Fig. 1 shows a simple data set [6]. In the data set, there are five clusters surrounded by noise points. Through observation, we can find that the data in the clusters are densely distributed, while the data outside the clusters are sparsely distributed. Therefore, for a boundary point, its neighbors are unevenly distributed in direction, with dense intra-cluster data on one side and sparse noise points on the other side. Assuming that neighbors on both sides can pass their cluster labels to the boundary point, how should the boundary point handle these cluster labels to make the received cluster label most reliable? By the common sense, it should discard cluster labels from the side where neighbors are sparsely distributed and receive cluster labels from the side where neighbors are densely distributed.

Similarly, for any point with unevenly distributed neighbors, we believe that cluster labels from the direction where neighbors are densely distributed are more reliable than cluster labels from the direction where neighbors are sparsely

distributed. Based on this idea, DBC is proposed. In DBC, each point has a receiving direction, pointing to the direction with the densest neighbor distribution. Cluster labels from this direction are by default the most trusted. The direction is represented by a vector. In addition, we believe that a neighbor whose position deviates from this direction at a small angle is still a credible neighbor, and the point can still receive the cluster label of this neighbor. Therefore, we can use an angle to determine the receiving range of the cluster labels in the neighborhood. Each point can only accept cluster labels from the neighbors within the receiving range.

In DBC, different clusters can be naturally separated by boundary data. As we know, a typical feature of boundary data is the unevenness of neighbor distribution. They have densely distributed neighbors on one side and sparsely distributed neighbors on the other side. Therefore, according to the idea of DBC, boundary data can only accept cluster labels from the internal data of clusters, because the data outside cluster boundary are sparsely distributed. As long as the neighborhood range is set properly, the data within the cluster cannot receive cluster labels from the data between cluster boundaries. Thus, the cluster label of one cluster do not pass through the cluster boundary of another cluster when propagating. Therefore, in DBC, the cluster boundary act as a barrier. With the help of cluster boundaries, all clusters can be naturally separated.

B. Concepts related to the Propagation of Cluster Labels

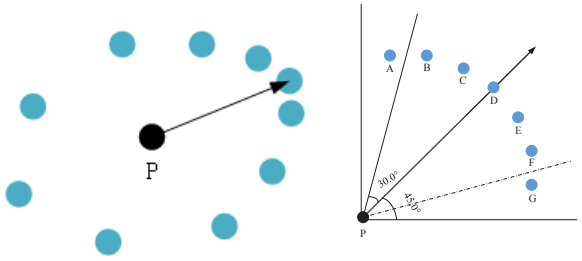
Definition 1: (Receiving direction) In DBC, each point has a direction in which the neighbors are most densely distributed. This direction is called the receiving direction. As shown in Fig. 2a, the direction indicated by the black arrow is the direction in which the neighbors of point P are most densely distributed. Therefore, we can take this direction as the receiving direction of point P. In order to obtain the receiving direction of each point, we uniformly use a vector expression to represent the receiving direction of each point, as shown in (3):

$$\vec{v} = \arg\max_{\vec{v_i} \in V} \sum_{\vec{v_j} \in V : \vec{v_i} \cdot \vec{v_j} > 0.5} \vec{v_i} \cdot \vec{v_j} \tag{3}$$

In (3), the receiving direction $\vec{v}$ is determined by the distribution of neighbors. Each neighbor corresponds to a unit vector called a neighbor vector whose direction points from the point to the neighbor and whose length is 1. All these unit vectors are stored in the vector set V. Suppose a point P has m neighbors, denoted by n_1, n_2, ..., n_m, the corresponding unit vectors are $\vec{v_1}$, $\vec{v_2}$, ..., $\vec{v_m}$, and $V = \{\vec{v_1}, \vec{v_2}, \ldots, \vec{v_m}\}$. Therefore, $\vec{v_i} \cdot \vec{v_j}$ is the dot product of two neighbor vectors $\vec{v_i}$ and $\vec{v_j}$. Equation (3) selects a unit vector from V as the receiving direction $\vec{v}$. The method for obtaining neighbor vectors is very simple. As we know, in a dataset, each point is represented by a multidimensional vector, and each dimension represents an attribute. Suppose that point P is the i-th point in the data set, denoted as $\vec{X_i}$, and the k-th neighbor of point P is the j-th point in the data set, denoted as $\vec{X_j}$, then the k-th neighbor vector of point P should be expressed as:

$$\vec{v_k} = \frac{\vec{X_j} - \vec{X_i}}{\left\| \vec{X_j} - \vec{X_i} \right\|}, \tag{4}$$

where $\left\| \vec{X_j} - \vec{X_i} \right\|$ is the Euclidean norm of the vector $\vec{X_j} - \vec{X_i}$.



(a) The receiving direction (b) The maximum deviation angle

Fig. 2: Illustration of two concepts (the left image is related to the receiving direction, and the right image is related to the maximum deviation angle)

Definition 2: (Maximum deviation angle) After the receiving direction is determined, an angle is required to determine the receiving range of the cluster labels. This angle is called the maximum deviation angle. It refers to the maximum angle at which a neighbor vector can deviate from the receiving direction, so that the point can receive the cluster label of the corresponding neighbor.

As we know, the receiving direction and a neighbor vector can form an angle, where the receiving direction determines one edge and the neighbor vector determines the other edge. If this angle is smaller than the maximum deviation angle, the neighbor can pass its cluster label to the point, otherwise it cannot. As shown in Fig. 2b, the arrow represents the receiving direction and also represents one edge of the maximum deviation angle. When the maximum deviation angle is set to 30°, B, C, D, E and F can pass their cluster labels to P, while A and G cannot. When the maximum deviation angle is set to 45°, all points except P can pass their cluster labels to P. For ease of description, we introduce the definitions of $sender$ and $receiver$.

Definition 3: (Sender) For a point P, a sender is a neighbor which can pass its cluster label to P. As shown in Fig. 2b, when the maximum deviation angle is set to 30°, B, C, D, E and F are senders of P.

Definition 4: (Receiver) For a point P, if a neighbor of P can pass its cluster label to P, P is a receiver of the neighbor. As shown in Fig. 2b, when the maximum deviation angle is set to 30°, P is a common receiver of B, C, D, E and F.

C. Required Parameters

Two parameters need to be set in DBC, one for determining the range of the neighborhood and the other for determining

the receiving range of cluster labels in the neighborhood. We use K-nearest neighbors to determine the neighborhood range. Therefore, the first parameter is K, which ensures that each point has K nearest neighbors. Generally, the values of K can vary in a wide range, and good clustering results can be obtained by adjusting the second parameter. Experimental experience shows that for data sets with less than 1500 points, the reasonable value of K is usually between 8 and 25, while for data sets with more than 1500 points, the reasonable value of K is usually between 30 and 60. For small data sets, when K is 15 or 20, good clustering results are usually available. For large data sets, slight change in the value of K will not have a great impact on the clustering results. Therefore, we tend to test only four values: 30, 40, 50 and 60. The second parameter is the angle value of the maximum deviation angle. We believe that the reasonable value of the angle is between $30°$ and $90°$. In our experiments, the angular values used are all included in the set G, and $G = \{30°, 35°, 40°, 45°, 50°, 55°, 60°, 65°, 70°, 75°, 80°, 85°, 90°\}$. In fact, for a data set, when the neighborhood range is determined, the reasonable range of the maximum deviation angle is a subset of G, and the angle values in the subset are usually continuous. Therefore, we can roughly determine the reasonable range of the maximum deviation angle by simply selecting several values from G for testing. For example, we can choose four angle values for testing, and they are $30°$, $50°$, $70°$, and $90°$. Among the obtained clustering results, we can determine the clustering result closest to the real cluster distribution based on some prior knowledge, such as the actual number of clusters or the ratio of the data amount of clusters. Next, we only need to explore the angle values around the corresponding angle to get the most suitable angle value in set G, so as to obtain the optimal clustering result.

D. Algorithm description

After the parameters are determined, each point can acquire its senders and receivers according to its receiving direction and the maximum deviation angle. Then, the propagation of cluster labels begins. The algorithm first selects the point with the largest number of receivers as the starting point of clustering among the unassigned points. Then it assigns a new cluster label to the starting point and the point becomes a sender with the function of sending cluster labels. It sends the cluster label to its unassigned receivers. These receivers are immediately in the allocated state, and become new senders with the function of sending cluster labels. The propagation process continues until no new sender with the function of sending cluster labels is generated. That is, the points at which the cluster label finally arrives have either no receiver or only the assigned receivers. When the process is over, a cluster is discovered. At this point, the process of finding the next cluster begins again. Algorithm 1 describes the whole process of finding all clusters.

In this code, $SetOfPoints$ is the collection of all points in the dataset, and $ReceiverDict$ is a dictionary that stores the receivers of all points in the dataset. First, the cluster labels for all points are initialized to 0, which means that all points are in an unassigned state. This process is described from line 1 to line 4. Num records the number of unassigned points in the data set. Since all points are not allocated at the beginning, the initial value of Num is set to the number of points in the data set on line 5. $ClusterID$ records the value of the cluster label of the newly discovered cluster. Before the clustering process has begun, its value is 0. After these preparations are done, the clustering process begins. From line 7 to line 41 is a description of the entire clustering process. The clustering process can be divided into two sub-processes. The first sub-

Algorithm 1 FindClusters(SetOfPoints, ReceiverDict)

```
1:  for i = 0; i < SetOfPoints.size; i++ do
2:      Point = SetOfPoints.get(i)
3:      Point.label = 0
4:  end for
5:  Num = SetOfPoints.size
6:  ClusterID = 0
7:  while Num > 0 do
8:      Maximum = −1
9:      for i = 0; i < SetOfPoints.size; i++ do
10:         Point = SetOfPoints.get(i)
11:         if Point.label == 0 then
12:             Receivers = getReceivers(ReceiverDict, Point)
13:             NumberOfReceivers = Receivers.size
14:             if NumberOfReceivers > Maximum then
15:                 Maximum = NumberOfReceivers
16:                 Start = Point
17:             end if
18:         end if
19:     end for
20:     ClusterID = ClusterID + 1
21:     Start.label = ClusterID
22:     SenderList = [ ]
23:     SenderList.append(Start)
24:     while SenderList.size > 0 do
25:         NewSenderList = [ ]
26:         for i = 0; i < SenderList.size; i++ do
27:             Sender = SenderList.get(i)
28:             Receivers = getReceivers(ReceiverDict, Sender)
29:             for j = 0; j < Receivers.size; j++ do
30:                 Receiver = Receivers.get(j)
31:                 if Receiver.label == 0 then
32:                     Receiver.label = ClusterID
33:                     NewSenderList.append(Receiver)
34:                 end if
35:             end for
36:         end for
37:         SenderList.clear()
38:         SenderList = NewSenderList.copy()
39:     end while
40:     Num = NumberOfUnassignedPoints(SetOfPoints)
41: end while
```

process is to find the starting point of clustering, and the second sub-process is the propagation of a new cluster label.

From line 8 to line 19 is the first sub-process. Through line 11, DBC checks the allocation status of each point and gets the unassigned points. On line 12, DBC gets the receivers of each unassigned point from the dictionary $ReceiverDict$. Through the comparison process from line 14 to line 16, the unassigned point with the largest number of receivers is regard as the starting point of clustering $Start$. On line 20, the value of $ClusterID$ is incremented by 1 to represent a new cluster label. And on line 21, $Start$ gets this new cluster label.

The second sub-process is from line 22 to line 39. On line 22, a new list $SenderList$ is defined for storing the newly discovered senders with the function of sending cluster labels during the propagation of cluster labels. Only when $senderList$ is not empty can the propagation of a cluster label continue. Therefore, on line 23, $Start$ is added to $SenderList$. On line 25, a new list $NewSenderList$ is defined for temporarily storing the newly discovered senders with the function of sending cluster labels. On line 28, DBC gets the receivers of a sender. On line 32, the sender's cluster label is passed to each unassigned receiver. And on line 33, these receivers immediately become new senders and are added to $NewSenderList$. After all the senders in $SenderList$ have performed the function of propagating the cluster label, $SenderList$ is emptied on line 37. On line 38, the new senders in $NewSenderList$ are copied to $SenderList$, and the algorithm goes back to line 24. If there is no sender in $SenderList$, the propagation of the cluster label is terminated. At this point, a cluster is found, and the algorithm counts the unassigned points in the data set on line 40. If there are unassigned points in the dataset, the algorithm will go back to line 7 and start looking for the next cluster.

E. Inferences

Although the receiving direction of each point is different, two points in the same cluster can always be connected by one path, and all points on the path share a cluster label. In order to describe the relationship between two points in the same cluster, we introduce the following definitions.

Definition 5: (Direct target) If point B is a receiver of point A, that is, point A can directly pass the cluster label to point B; then point B is a direct target of point A.

Definition 6: (Indirect target) If point A has to go through multiple points to pass the cluster label to point B; then point B is an indirect target of point A.

Definition 7: (Common source) If point A is a direct or indirect target of point C, and point B is also a direct or indirect target of point C; then point C is a common source of point A and point B. It is stipulated that the starting point of clustering is its own source.

Based on these definitions, we can draw the following inferences.

Inference 1: Each point has at least one common source with the points that have the same cluster label as one of its senders.

According to definition 7, the starting point is a common source of the points which have the same cluster label as it. For a point A, if it has the same cluster label as a sender B, A and the points that have the same cluster label as B are in the same cluster, so they have a common source—the starting point. Otherwise, according to definition 5 and 6, A is an indirect target of the starting point of B, and the points that have the same cluster label as B are direct or indirect targets of the starting point of B. Therefore, the starting point of B is a common source of A and these points. Thus, we can get the above inference.

Inference 2: Two different points in the same cluster must have at least a common source, but two different points which have at least a common source may not be in the same cluster.

Since the starting point of clustering must be a common source of two different points in the same cluster, thus they must have at least a common source. However, if two different points have at least a common source, it doesn't mean they are in the same cluster because they may get cluster labels from different starting points, if one of them is in the transition region between cluster boundaries. As we know, the neighbor distribution of points in the transition region between cluster boundaries is unpredictable. There is no guarantee that all senders of a point in the transition region will be in the same cluster. When a point's senders exist in multiple clusters, it can only belong to one of the clusters. So although it has at least a common source with the points that have the same cluster label as one of its senders, it may not in the same cluster as some of them. Thus, we can get the above inference.

IV. Experiments

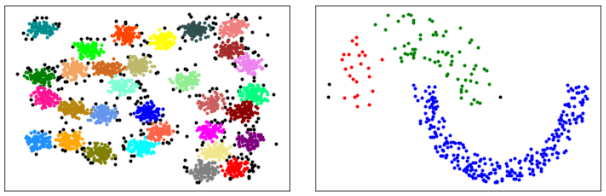
The clustering effect of DBC is evaluated in this section. First, the clustering effect of DBC on artificial data sets is compared with two well-known density-based clustering algorithms, DBSCAN and DPC. Second, the clustering effect of DBC on real world data sets is compared with three state of the art clustering algorithms, K-means++, AP (Affinity Propagation) and DPC. The information of all data sets is presented in Table I, including name, number of instances and source.

A. Clustering effect on artificial data sets

There are eight artificial data sets used in experiment 1, including Hard, D31, Jain, Flame, Spiral, Aggregation, Lsun and Chainlink. The eight data sets can be divided into four groups, each with two data sets, which evaluate the clustering performance of the algorithms from different aspects. Flame and D31 are used to evaluate the clustering effect of the algorithms on data sets with overlapping regions between clusters. Aggregation and Spiral are used to evaluate the adaptability of the algorithms to clusters of different shapes. Jain and Hard are used to evaluate the clustering effect of the algorithms on data sets with large differences in density between clusters. Lsun and Chainlink are used to evaluate the clustering effect of the algorithms on data sets containing clusters with uneven density distribution. First, experiments

TABLE I: The data sets used in experiments

Name	No instances	No attributes	No clusters	Source
Hard	1501	3	2	[10]
D31	3100	2	31	[11]
Jain	373	2	2	[12]
Flame	240	2	2	[13]
Spiral	312	2	3	[14]
Aggregation	788	2	7	[15]
Lsun	400	2	3	[16]
Chainlink	1000	3	2	[16]
Iris	150	4	3	[17]
Wine	178	13	3	[17]
Seeds	210	7	3	[17]
WDBC	569	30	2	[17]
Breast	683	9	2	[17]
Ionosphere	351	34	2	[17]
Dermatology	358	34	6	[17]
Banknote	1372	4	2	[17]



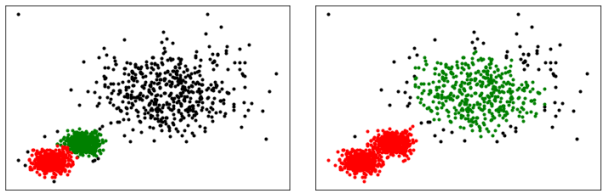
(a) D31 (b) Jain

Fig. 3: The clustering results obtained by DBSCAN in the D31 and Jain data sets

on DBSCAN and DPC are carried out. In DPC, we use (2) to calculate the density of points. For these eight data sets, neither DBSCAN nor DPC can get the correct clustering results on all data sets. DBSCAN cannot achieve good clustering results in the D31, Jain, and Hard data sets. And DPC cannot achieve good clustering results in the Jain, Lsun, and Chainlink data sets.

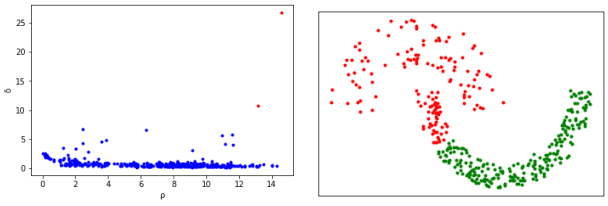
DBSCAN generates noise during clustering. Therefore, in the figures, noise is represented by black, while different clusters are represented by different colors. As shown in Fig. 3a, although DBSCAN can identify 31 clusters in the D31 dataset, it generates a lot of noise, resulting in waste of data. This means that when many clusters in the data set overlap with each other, DBSCAN is difficult to achieve good clustering effect. DBSCAN cannot get the correct clustering result in the Jain dataset. There are two clusters in the Jain dataset, and DBSCAN divides the cluster above into two clusters, as shown in Fig. 3b.

In the Hard data set [10], two high-density clusters are close together, and the minimum density of data in the junction area is higher than the maximum density of data in the low density cluster. Therefore, when the global density threshold is set high enough to distinguish the two high-density clusters, the data in the low-density cluster will all be regarded as noise, as shown in Fig. 4a. If the global density threshold is set low enough to recognize the low-density cluster, the two high-density clusters cannot be separated, as shown in Fig. 4b.



(a) The density threshold is set to a high value. (b) The density threshold is set to a low value.

Fig. 4: Two clustering results obtained by DBSCAN in the Hard data set.

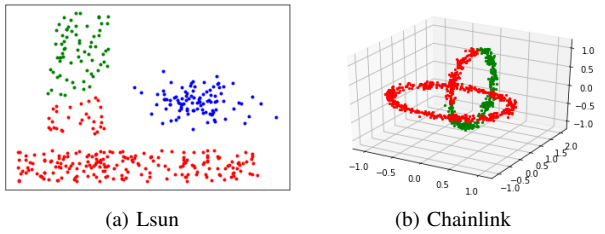


(a) Decision graph (b) Clustering result

Fig. 5: The decision graph and clustering result obtained by DPC in the Jain dataset

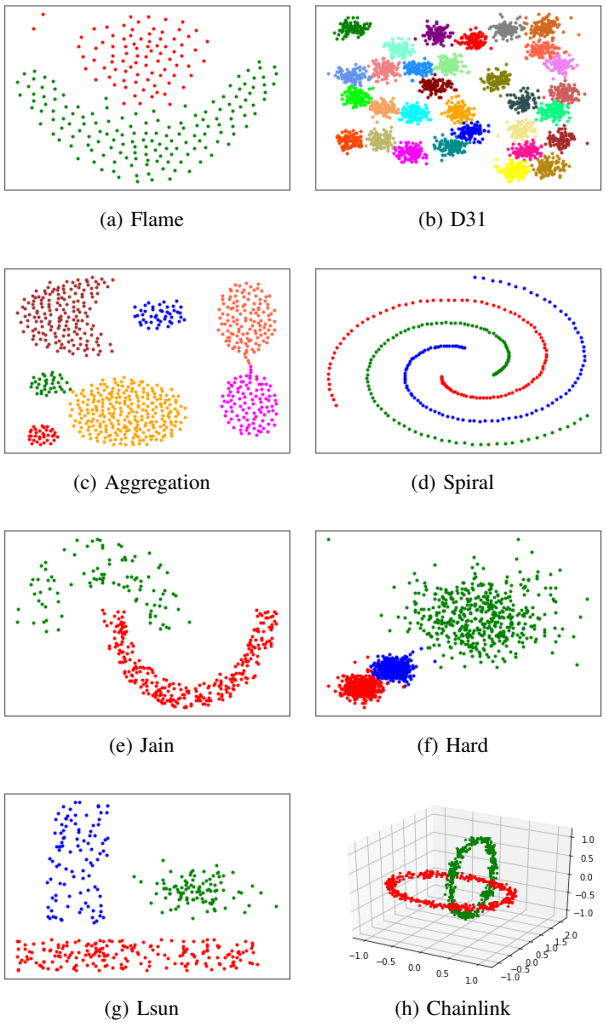
DPC also fails to get the correct clustering result in the Jain dataset. As shown in Fig. 5a, the two cluster centers represented by red dots are both from the high-density cluster, as they are with a high value of ρ. Therefore, the high-density cluster is divided into two parts, and all data of the low density cluster are misclassified to the cluster below, as shown in Fig. 5b.

When a cluster with uneven density distribution is close to another cluster, DPC may not get the correct clustering result, because error propagation may occur. If a point in the cluster with uneven density distribution has its nearest higher density point in another cluster, it will be misclassified. And those points that have it as the nearest higher density point will also be misclassified. In this way, the error will spread, causing a significant portion of the data in one cluster to be misclassified into another cluster. For this reason, DPC cannot obtain correct clustering results in the Lsun and Chainlink data sets, as shown in Fig. 6a and Fig. 6b, respectively.



(a) Lsun (b) Chainlink

Fig. 6: The clustering results of DPC in the Lsun and Chainlink data sets



(a) Flame (b) D31

(c) Aggregation (d) Spiral

(e) Jain (f) Hard

(g) Lsun (h) Chainlink

Fig. 7: Clustering results obtained by DBC on the eight data sets

Fig. 7 shows the clustering effect of DBC on these eight data sets. It can be seen from the figure that DBC can achieve good clustering results in all data sets. The data sets presented in Fig. 7 contains clusters of different shapes, which indicates that DBC has strong adaptability to clusters of various shapes. Compared to DBSCAN, it performs better in data sets with many clusters overlapping with each other, as shown in Fig. 7b. In the D31 dataset, it successfully identifies 31 clusters without losing data. In addition, compared with DBSCAN and DPC, it is less affected by uneven density distribution. DBC can get the correct clustering results in the four data sets Jain, Hard, Lsun and Chainlink, which DBSCAN and DPC cannot do.

B. Clustering effect on real world data sets

In this section, we evaluate the clustering effect of DBC through real-world data sets. In the experiment, we also compare DBC with three state-of-the-art clustering algorithms: DPC, AP (Affinity Propagation) [20] and K-means++ [21].

TABLE II: Comparison of clustering effects on real data sets

Dataset	Criteria	K-means++	AP	DPC	DBC
Iris	Acc	0.8866	0.9066	**0.9600**	**0.9600**
	NMI	0.7419	0.7608	0.8622	**0.8705**
Wine	Acc	0.9550	0.9606	0.8932	**0.9775**
	NMI	0.8529	0.8609	0.7261	**0.9119**
Seeds	Acc	0.8904	**0.9238**	0.9047	**0.9238**
	NMI	0.6742	0.7391	0.7193	**0.7398**
WDBC	Acc	0.9279	**0.9349**	0.8629	0.8664
	NMI	0.6231	**0.6409**	0.5009	0.5203
Breast	Acc	0.9604	0.9604	0.9355	**0.9692**
	NMI	0.7478	0.7492	0.6562	**0.7936**
Dermatology	Acc	**0.9553**	0.9245	0.8687	0.8826
	NMI	**0.9289**	0.8613	0.8263	0.8102
Ionosphere	Acc	0.7122	0.7094	0.7492	**0.8632**
	NMI	0.1349	0.1320	0.1626	**0.3970**
Banknote	Acc	0.5758	0.6501	**0.9905**	0.9555
	NMI	0.0173	0.0761	**0.9316**	0.7467

We evaluate the clustering performance of the four algorithms using two evaluation metrics, Acc (Clustering Accuracy) [18] and NMI (Normalized Mutual Information) [19]. For both evaluation metrics, their values range from 0 to 1. A value of 0 means that the clustering effect is the worst, and a value of 1 means that the clustering effect is perfect. When the value is between 0 and 1, the larger the value, the better the clustering effect. Before the experiment, we preprocessed the data. The processing steps can be divided into two steps:

1) Remove incomplete data from the data sets.
2) Normalize the data using min-max normalization [22] so that each attribute plays the same role.

Different algorithms use different parameters, so the operation for each algorithm is different. For K-means++, we take the actual number of clusters as its input, and run the program 100 times. For AP, the value of the parameter $preference$ traverses all integers between -100 and -1. For both algorithms, the optimal clustering results are selected from the obtained clustering results for comparison. For DPC, we first determine the reasonable range of dc in each data set, and then approach the ideal value of dc at the pace of 0.01. Each dc corresponds to a decision graph, and we select the cluster centers according to the principle that the cluster centers have unusually large ρ and δ. Among the obtained clustering results, we select the optimal clustering result for comparison. For our algorithm DBC, the value of K traverses every integer between 8 and 25. Considering that some data sets may require large K values, the value set of K also includes 30, 40, 50 and 60. For each value of K, we adjust the maximum deviation angle to obtain the optimal clustering result corresponding to this value. From the obtained clustering results, we select the optimal clustering result for comparison. Table II is a comparison of the optimal clustering results obtained by these algorithms on each data set.

There are eight real-world data sets, and the best clustering effect on each dataset is marked in bold font. Each algorithm has its advantages and application scenarios, so it can always achieve good clustering results on some data sets in Table II. Overall, DBC performs best. It shows the best clustering effect

on the five data sets Iris, Wine, Seeds, Breast and Ionosphere. And it shows near-optimal clustering effect on the Banknote data set. Compared with DPC, DBC performs better on 6 data sets. Compared with AP, DBC performs better on 5 data sets. Compared with K-means++, DBC performs better on 6 data sets. Therefore, on the whole, DBC performs better in clustering effect than DPC, AP and K-means++.

V. Conclusions

A novel clustering algorithm DBC is proposed in this paper. Different from the previous clustering algorithms, it is based on the idea that the propagation of cluster labels has directionality. Like DBSCAN and DPC, it has the ability to find clusters of arbitrary shapes. Therefore, we compared it with DBSCAN and DPC in the experiments. Experiments on artificial data sets show that DBC is less susceptible to uneven density distribution than DBSCAN and DPC. And experiments on real world data sets show that DBC has advantages over k-means++, AP and DPC in clustering effect. In future work, we will apply it to some specific fields so that its application value can be developed.

References

[1] M. Du, S. Ding , and H. Jia . "Study on Density Peaks Clustering Based on k-Nearest Neighbors and Principal Component Analysis." Knowledge Based Systems, vol. 99, 2016, pp. 135–145.

[2] S. Guha, R. Rastogi , and K. Shim . "CURE: An Efficient Clustering Algorithm for Large Databases." Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, vol. 27, no. 2, 1998, pp. 73–84.

[3] J.B. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations." Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, vol. 1, 1967, pp. 281–297.

[4] M. Ester, H.P. Kriegel, J. Sander and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." KDD, 1996, pp. 226–231.

[5] J. Shi, and J. Malik. "Normalized Cuts and Image Segmentation." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, 2000, pp. 888–905.

[6] A. Rodriguez, and A. Laio . "Clustering by Fast Search and Find of Density Peaks." Science, vol. 344, no. 6191, 2014, pp. 1492–1496.

[7] L. Lelis, and J. Sander. "Semi-Supervised Density-Based Clustering." 2009 Ninth IEEE International Conference on Data Mining, 2009, pp. 842–847.

[8] K.M. Kumar, and A.R.M. Reddy. "A Fast DBSCAN Clustering Algorithm by Accelerating Neighbor Searching Using Groups Method." Pattern Recognition, vol. 58, 2016, pp. 39–48.

[9] Z. Liang, and P. Chen. "Delta-Density Based Clustering with a Divide-and-Conquer Strategy." Pattern Recognition Letters, vol. 73, 2016, pp. 52–59.

[10] Y. Zhu, K.M. Ting , and M.J. Carman . "Density-Ratio Based Clustering for Discovering Clusters with Varying Densities." Pattern Recognition, vol. 60, 2016, pp. 983–997.

[11] C.J. Veenman, M.J.T. Reinders, and E. Backer. "A Maximum Variance Cluster Algorithm." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 9, 2002, pp. 1273–1280.

[12] A.K. Jain, and M.H.C. Law. "Data Clustering: A User's Dilemma." Pattern Recognition and Machine Intelligence, 2005, pp. 1–10.

[13] L. Fu, and E. Medico. "FLAME, a Novel Fuzzy Clustering Method for the Analysis of DNA Microarray Data." BMC Bioinformatics, vol. 8, no. 1, 2007, pp. 3–3.

[14] H. Chang, and D. Yeung. "Robust Path-Based Spectral Clustering." Pattern Recognition, vol. 41, no. 1, 2008, pp. 191–203.

[15] A. Gionis, H. Mannila, and P. Tsaparas. "Clustering Aggregation." ACM Transactions on Knowledge Discovery from Data (TKDD), 2007.

[16] A. Ultsch. "U*C: Self-Organized Clustering with Emergent Feature Maps." LWA, 2005, pp. 240–244.

[17] A.U. Asuncion, and D.J. Newman. UCI Machine Learning Repository. 2007.

[18] C. Aytekin, X. Ni, F. Cricri, and E. Aksu. "Clustering and Unsupervised Anomaly Detection with L2 Normalized Deep Auto-Encoder Representations." 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–6.

[19] D. Cheng, Q. Zhu, J. Huang and L. Yang. "Natural Neighbor-Based Clustering Algorithm with Density Peeks." Neural Networks (IJCNN), 2016 International Joint Conference On, 2016, pp. 92–98.

[20] B.J. Frey, and D. Dueck. "Clustering by Passing Messages Between Data Points." Science, vol. 315, no. 5814, 2007, pp. 972–976.

[21] D. Arthur, and S. Vassilvitskii."K-Means++: The Advantages of Careful Seeding." Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1027–1035.

[22] 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." Information Sciences, vol. 354, 2016, pp. 19–40.