doi:10.3772/j.issn.1006-6748.2018.02.005

A heuristic clustering algorithm based on high density-connected partitions①

Yuan Lufeng (苑鲁峰)②*** , Yao Erlin* , Tan Guangming*
(*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China)
(**University of Chinese Academy of Sciences, Beijing 100049, P. R. China)

Abstract

Clustering data with varying densities and complicated structures is important, while many existing clustering algorithms face difficulties for this problem. The reason is that varying densities and complicated structure make single algorithms perform badly for different parts of data. More intensive parts are assumed to have more information probably, an algorithm clustering from high density part is proposed, which begins from a tiny distance to find the highest density-connected partition and form corresponding super cores, then distance is iteratively increased by a global heuristic method to cluster parts with different densities. Mean of silhouette coefficient indicates the cluster performance. Denoising function is implemented to eliminate influence of noise and outliers. Many challenging experiments indicate that the algorithm has good performance on data with widely varying densities and extremely complex structures. It decides the optimal number of clusters automatically. Background knowledge is not needed and parameters tuning is easy. It is robust against noise and outliers.

Key words: heuristic clustering, density-based spatial clustering of applications with noise (DBSCAN), density-based clustering, agglomerative clustering, machine learning, high density-connected partitions, optimal clustering number

0 Introduction

Clustering is a classical problem in machine learning, and has been used widely in image analysis, information retrieval and data mining. Clustering algorithms can be sorted into six classes based on different features: partitioning, hierarchical, grid-based, density-based, model-based and graph-based[1]. However, each class of clustering has its own potential shortcomings. For example, K-means[2] and its variations, belonging to partitioning clustering, need a specified number of clusters as an input parameter, which often influences performance largely and is hard to decide. The primary disadvantage of hierarchical clustering[3] is that it doesn't have back-tracking ability and false partitions can't be undone. Affinity Propagation[4], belonging to graph-based clustering, has also another two important parameters, preference and damping factor, specified by users influence performance greatly.

Density-based clustering assumes the overall distribution of the data is mixture of several distributions and points of different clusters obey different specific distributions[5]. Clusters are defined as areas of higher density surrounded by that of lower density. Objects in those sparse areas required to separate clusters, are usually considered to be noise and border points[6]. Due to this general motivation, density-based clustering can find clusters of arbitrary shape. The other feature is that density-based clustering doesn't specify the number of clustering. Some representative algorithms include DBSCAN, DBCLASD, DENCLUE, OPTICS, Mean Shift. DBSCAN(density-based spatial clustering of applications with noise)[7] require two input parameters: radius, *eps*, to define the neighborhood of each object, and the minimum number, *minPts*, of objects to form a cluster. DBSCAN starts with an arbitrary starting point that has not been visited. This point's

① Supported by the National Key Research and Development Program of China (No. 2016YFB0201305), National Science and Technology Major Project (No. 2013ZX0102-8001-001-001) and National Natural Science Foundation of China (No. 91430218, 31327901, 61472395, 61272134, 61432018).

② To whom correspondence should be addressed. E-mail: yuanlufeng@ncic.ac.cn

Received on Sep. 27, 2016

neighbors in *eps* range are retrieved. If it contains more points than *minPts*, a cluster is formed. Otherwise, the point is labeled as noise. If a point is found to be a dense part of a cluster, its neighbors are also part of that cluster. Hence, all points found within the *eps* range are added. This process continues until the density-connected cluster is completely found. OPTICS[8] can be seen as a generalization of DBSCAN that outputs the points in particular ordering and any clusters can be extracted by this cluster ordering. DBCLASD[9] assumes the points of a cluster are uniformly distributed and according distribution is known. This assumption limits DBCLASD's using in practice. DENCLUE[10] identifies density attractors, local maxima of the overall density function in certain areas, and determines clusters mathematically. Mean Shift[11] counts mean shift vector of a candidate centroid and locates the local maxima of a density function by updating candidates for centroids iteratively. Clusterdp[12] also computes two quantities for each data point, which are local density and point's distance from points of higher density. The points of high distance and high local density are chosen relatively to be the cluster centers. Then each remaining point is assigned to the same cluster as its nearest neighbor of higher density. A common problem of these density-based clustering methods is that they usually perform badly if the data objects have widely varying densities and extremely complex structures. Besides density and distribution problems, noise and outliers are also serious to these density-based cluster methods.

In this paper, the data is clustered iteratively and clustering parameters are adjusted heuristically to the basis of the data themselves. First the clustering begins from the highest density area in the data. Then clustering scope is spread to all the data from the highest density area to the lowest density area gradually. Density-connected methods are used similar to DBSCAN[7] to find the highest density partitions. In the algorithm, DBSCAN is applied as a density-connected partition detector. At the beginning, DBSCAN is used with extremely small *eps* and *minPts* to find some density-connected partitions. These density-connected partitions are the highest density parts of the data. Then, a representative named super core is extracted for each density-connected partition by the pooling method in deep learning. The reason of extracting super core is to increase degree of separation of different density parts and avoid adhesion between them. Some super cores with little objects or noise will form in this procedure. Because these super cores are probably meaningless and influence the accuracy of clustering, the approach performs a function to eliminate meaningless super cores. After generating super cores, a global heuristic approach is proposed to update *eps* and new density-connected partitions are detected. The value of *minPts* in our algorithm is often set to 2 for finding the high density parts. That means parameters in DBSCAN are not specified by users. The algorithm will reach convergence after several iteration.

The rest of the paper is organized as follows. In Section 1, the algorithm is presented. In Section 2, the experimental evaluations are shown. Conclusion is given in Section 3.

1 The proposed algorithm

1.1 Basic definitions

Because DBSCAN is a basic operation in the algorithm, the definitions in DBSCAN[7] such as eps, minPts, directly density-reachable, density-reachable, density-connected are also utilized in the algorithm and will not be repeated. In the following, some basic definitions are introduced:

Definition 1: (nearest neighbor) Let p be a point in set D, the nearest neighbor of p in D is point q which has the minimum distance to p in set $D - \{p\}$.

Definition 2: (nearest neighbor distance) Let p be a point in set D, the nearest neighbor distance of p in D is the distance between p and the nearest neighbor of p in set D.

Definition 3: (density-connected partition) Given *eps* and *minPts*, a density-connected partition is a local maximal set of density-connected points in set D.

Definition 4: (super core) Given a density-connected partition, super core s is an aggregation of all the points in the density-connected partition.

1.2 Motivation

The motivation of the algorithm comes from solving the difficulty that DBSCAN clusters the objects with widely varying density and complex structure, especially including outliers and noise which make DBSCAN merge into two and more different clusters together. In Fig. 1(a), cluster 1 and cluster 2 have obviously different densities and some outliers are compose of a "bridge" to connect two clusters. DBSCAN has some difficulties to cluster this data. Setting smaller *eps* and *minPts*, DBSCAN only identifies dense cluster 1 and regards sparse cluster 2 as noise like Fig. 1(b). Setting bigger *eps*, DBSCAN also identifies sparse cluster 2. But it is possible that DBSCAN merges cluster 1 and 2 as one cluster if *eps* is too big like Fig. 1(c). Furthermore, if outliers of bridge satisfy density-connect

with corresponding *eps* and *minPts* of DBSCAN, cluster 1 and 2 will merge together definitely. For overcoming this difficulty, performing DBSCAN by incremental *eps* is considered. When *eps* is very small, the most intensive density-connected partitions in cluster 1 and 2 will be detected first. Then these partitions are aggregated to super cores. Aggregating super cores can increase the degrees of cohesion in the same clusters and separation between different clusters. As increasing *eps*, more and more super cores appear and small super cores merge to big super cores. Finally, cluster 1 and cluster 2 aggregate into a super core separately and outliers of bridge are also identified as a super core like Fig. 1(d).

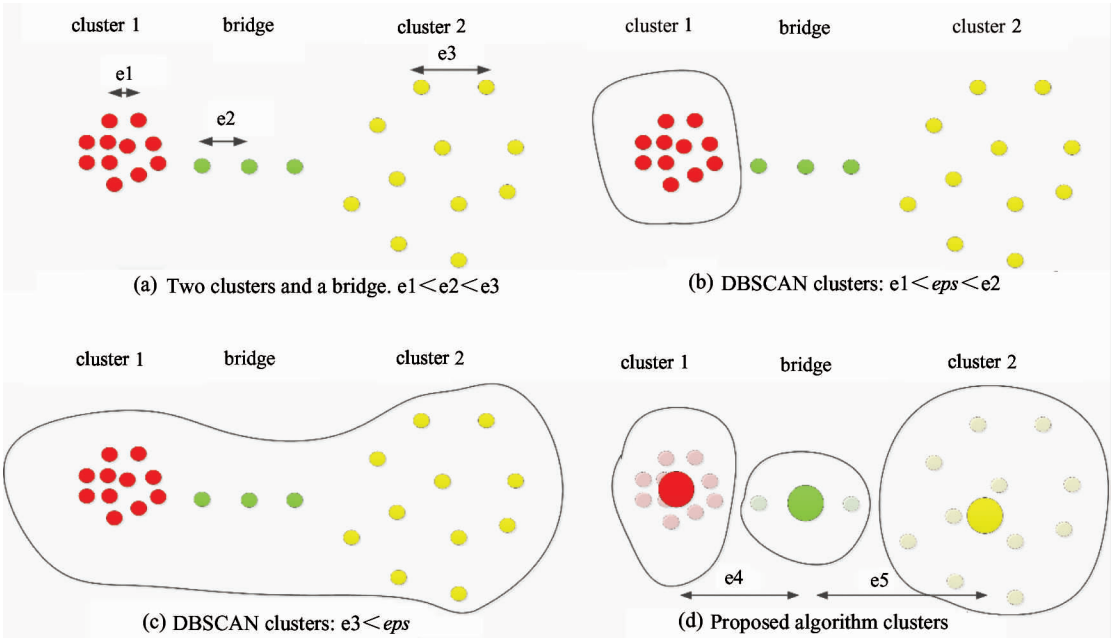

Fig. 1 Average distance between points in different groups is e1, e2 and e3

1.3 Description of algorithm

The whole data set is denoted as S. At first, the algorithm analyzes all the points in set S to determine a reasonable initial *eps*. Every nearest neighbor distance for every point is computed to get a nearest neighbor distance set D. Then *eps* is computed by

$$eps = \begin{cases} \min(\boldsymbol{D}) + \alpha \times \mathrm{std}(\boldsymbol{D}), & S \text{ has overlap} \\ \mathrm{mean}(\boldsymbol{D}) + \beta \times \mathrm{std}(\boldsymbol{D}), & \text{otherwise} \end{cases} \quad (1)$$

In Eq. (1), $\min(\boldsymbol{D})$ is the minimum of $\boldsymbol{D}$, mean (D) is the mean value of D and std(D) is the standard deviation of D. $\alpha \geqslant 0.01$ and $-3 \leqslant \beta \leqslant 3$ are both two constants. The ranges of α and β which often distribute most of meaningful points come from probability theory. Note that Eq. (1) stands for two strategy computing *eps*. When S has overlap, *eps* increases to a very tiny extent, data are classified very carefully and running time is much. Oppositely, if there isn't overlap in S, *eps* will increase with a large value, classification is quick and running time is reduced. The aim of using simple formulas, not complex distribution is to enlarge universality of algorithm.

After *eps* is determined, the algorithm makes use of DBSCAN as a density-connected detector. In the proposed algorithm, *minPts* is often set as 2. That means any two points whose distance is smaller than *eps* are directly density-reachable and density-connected partitions spread until distance between two points are larger than *eps*. When *minPts* equals 2, borders of clusters in DBSCAN disappear and all the points are only classified into core points and noise. Core points are composed of density-connected partitions which are the highest density part of set S in the current *eps*. After generating density-connected partitions, pooling of deep learning[13] is applied to extract super cores from density-connected partitions. As mentioned before, the aim of extracting super cores is to increase the degree of cohesion in the same partitions and separation between different partitions. For standing for the whole density-connected partition better, mean-pooling is applied that super core is the average value of all the points in corresponding density-connected partition.

During aggregating super cores, some super cores including little points are generated. If these super cores locate in sparse area, they are noise or outliers in fact and won't grow up. So there is a function to check whether some super cores are meaningless and should

be decomposed. After new super cores are added into S, *the number* n_i of points in every super core i is recorded and a super core quantity set N is gotten. Set N is measured to judge whether some super cores are outliers. If recover conditions, $\mathrm{std}(N) \geqslant \mathrm{mean}(N)$ and $\mathrm{median}(N) \geqslant \mathrm{mean}(N)$, are satisfied, that means some discrete super cores appear. Here $\mathrm{mean}(\cdot)$ and $\mathrm{std}(\cdot)$ are the same as before, $\mathrm{median}(N)$ is the median of N. A super core will be recovered to original points, then these points are put back to set S. Note that this process is probabilistic, rather than deterministic. The probability that a super core i is recovered is inversely proportional to its volume ratio which is defined in the Eq. (2).

$$volume_i = n_i / \left(\sum_{n_j \in N} n_j \right) \tag{2}$$

Intuitively, the super core with little points has low volume ratio and its probability recovered is high. After a super core is recovered, super core quantity set N is updated and recover conditions are computed again. If recover conditions aren't satisfied, recovering super cores will stop. Otherwise, a new super core will be recovered.

After recovering meaningless super cores, new aggregated super cores are added into S and corresponding points are removed from S. At the same time, nearest neighbor distance set D of S is also updated. Next *eps* is computed and a new procedure of detecting density-connected partitions begins. Two keys about super core should be noticed. First key being a super core is regarded as an ordinary point in computing nearest neighbor distance and generating density-connected partitions. Secondly, in pooling of density-connected partitions, a super core is regarded as the points in it. That means if density-connected partitions include super cores, corresponding super cores will be recovered to points in it and the average value of all these points generates super core of this partition.

In each iteration mean of silhouette coefficient[14] of every point is used as an internal index to indicate the performance of clustering. Silhouette coefficient is bounded from -1 to 1 and greater value is better. The reason why silhouette coefficient is chosen as index is that silhouette value is correlated better with the F1 score, F2 score, Jaccard index so that it would be a better measure for un-labeled datasets[15].

As algorithm runs, more and more super cores appear, more and more original points are removed from S, *eps* becomes larger and larger and the total points in S becomes less and less. Finally, the algorithm will converge to a balance situation. Points in the same super core are regarded as a cluster and the points that don't belong to any super cores are identified as noise in the convergence situation. The whole algorithm is described in Tabel 1.

Tabel 1 Description of algorithm

Algorithm: Input: Dataset S, α, β

Output: A clustering solution
repeat
1. Calculate the nearest neighbor distance set $\boldsymbol{D}$, determine *eps*.
2. Generate density-connected partition p using DBSCAN in *eps* and $minPts = 2$.
3. Reduce p to super core c by pooling. If there are some points which are super cores in p, recover super core to corresponding original points.
4. examine whether some super cores including little points are needed to recover. If existing, recover these super cores to original points.
5. Update S by adding new super cores and removing corresponding original points.
6. Compute mean of silhouette coefficient and record current situation

until S remain unchanged
return the solution whose silhouette coefficient is maximum.

For n objects in data S, the average complexity of DBSCAN is $O(n\log n)$ and the worst complexity is $O(n^2)$ if data are degenerated or *eps* is unreasonable. The complexity of computing nearest neighbor distance set D is $O(n^2)$. The complexity of extracting super cores is $O(n)$. The complexity of decomposing super cores including little points or noise is $O(n)$. The complexity of computing silhouette coefficient is $O(n^2)$. The number of iteration is defined as t. In the worst situation, t is $n-1$. In normal situation, $t << n$. In fact, the maximal value of t in the experiments is just 270. So the average complexity of the algorithm is $O(t \times n^2)$.

2 Performance evaluation

Performance of the algorithm is evaluated with three experiments and the algorithm in Python is implemented. Compared clustering algorithms including DB-

SCAN, K-means, average linkage, complete linkage and ward's clustering come from scikit-learn package in Python.

In the first experiment, the working procedure of the algorithm is shown and its ability to cluster the dataset with excessive unbalanced density and distribution is evaluated. The dataset is called Unbalance from[16]. This dataset contains 6500 vectors in 8 Gaussian clusters. Each cluster of left three clusters consists of 2000 vectors but each cluster of right five clusters has only 100 vectors. The density ratio between left dense cluster and right sparse cluster is more than 20. Fig. 2 shows the main steps of the proposed approach. Note that bigger colored dots mean super cores and the same colored parts or points as super cores are vectors which are aggregated by super cores. In the first iteration of Fig. 2(a), the proposed approach just constructs one super core because of small *eps*. Then 369 super cores, most of which concentrate on left three clusters and just three super cores are in the right sparse parts, are generated in the second iteration of Fig. 2(b). As *eps* increases, left three clusters have aggregated to one super core separately and right five clusters also consiste of several super cores in the tenth iteration of Fig. 2(c). Finally, all the vectors form eight super cores as shown in Fig. 2(d).

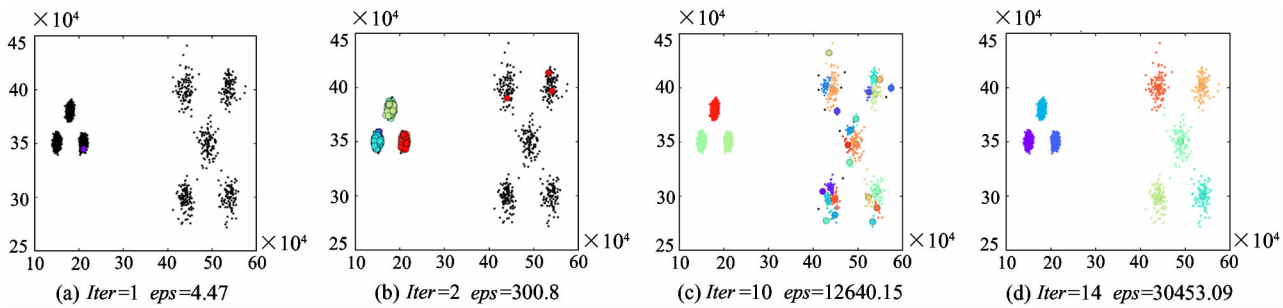

Fig. 2 Unbalance clustering (a)(b)(c)(d) clustering procedure of the proposed algorithm to Unbalance

In the second experiment, the approach's ability is evaluated to determine the number of clusters automatically. Dataset from Ref. [17] is used in the work. This dataset consists of 4 subsets of synthetic 2-d data points. Each subset containing 5000 vectors in 15 Gaussian clusters has different degree of cluster overlapping. Strategy $eps = \min(D) + \alpha \times \mathrm{std}(D)$ is employed to heuristically update *eps* in each iteration. In practice, α is increased 0.01 each time from 0.01 to 1 because too large α is meaningless. This makes tuning parameters of our algorithm easy and don't need background knowledge of dataset. In fact, although α has 100 values, there are only a little clustering results because the algorithm is robust to tuning parameters. Finally, the values of α in S_1, S_2, S_3, S_4 are decided to 0.8, 0.6, 0.4, 0.05 respectively. The result is described in Fig. 3. As shown in Fig. 3, the proposed algorithm obtains the correct number (15) of clusters and near-perfect structure on each of the four subsets. In the bottom row, the number of super cores varies from few to many and back to few, corresponding silhouette value has an incremental trend and becomes stable finally. Vertical line in the figure shows maximum of silhouette value indicates the correct clustering number. Note that there are some separated black vectors which are labelled noise because of the function of recovering super core in the proposed algorithm. These vectors lying in the edge of clusters are only in a very small portion of data and have no influence to final result.

In the third experiment, the algorithm's ability is evaluated to some datasets with special shape and different degree of connection, compare it with some other popular clustering algorithms including K-means, DBSCAN and three types of agglomerative clustering: average, complex and ward linkage. The performance is evaluated on three datasets: Aggregation[18], Flame[19] and Jain[20]. Aggregation contains 788 points from seven different groups with different shapes and sizes. The most difficulties of Aggregation are two "bridges" between different groups which perhaps connect them together in clustering. Flame contains 240 points from three groups. Upper left two outliers belong to one group. Another two groups have compact combination in the large area. Jain contains 373 points from two groups. Although two groups do not combine directly, there is an overlapping part of two groups with adjacent distance.

In the third experiment, correct number of clusters is used as input parameter for K-means, average linkage, complex linkage and ward linkage. For DBSCAN, *minPts* is varied from 2 to 10 and *eps* is traversed from the minimal distance to the maximal distance between all the points in the dataset . The strategy *eps*

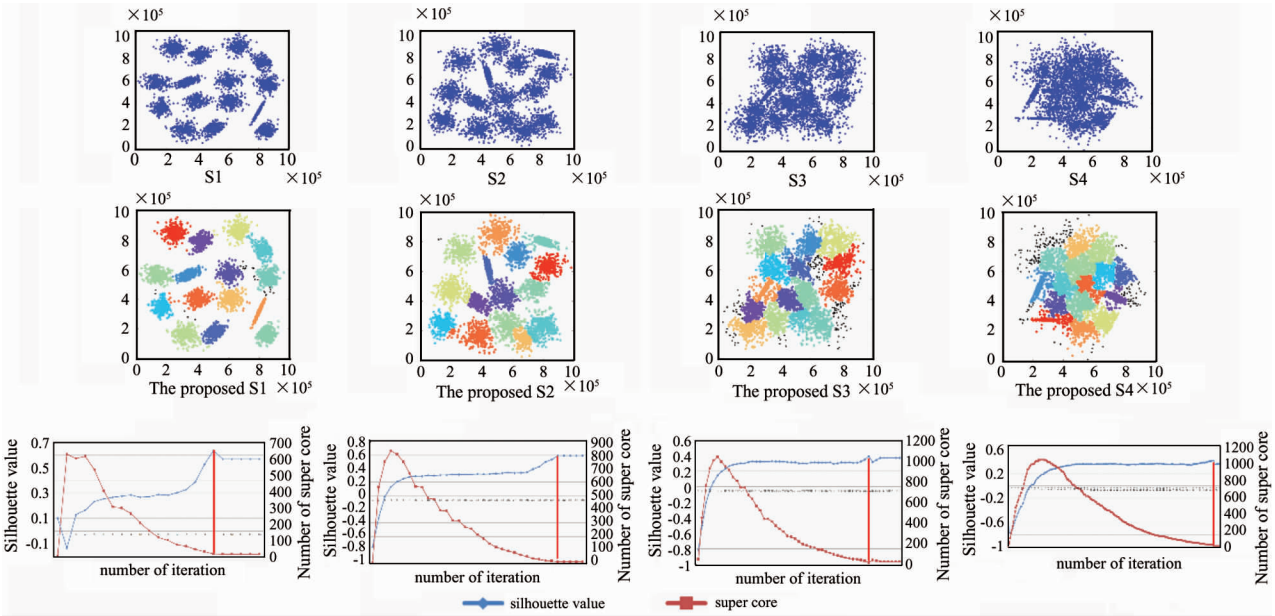

Fig. 3 Clustering for determining the number of cluster automatically. (top row) Original data. (middle row) Clustering results of proposed algorithm. (bottom row) Relationship between silhouette coefficient and number of super cores in clustering that vertical line shows the location of maximum of silhouette coefficient and corresponding number of super core

$= \text{mean}(D) + \beta \times \text{std}(D)$, $eps > 0$ and $-3 \leqslant \beta \leqslant 3$ is applied to search optimal performance of the algorithm.

In the result of the algorithm, dots which are bigger than ordinary points are super cores. Super cores are retained to show final convergence of each cluster. As demonstrated in Fig. 4, the algorithm can identify the shapes of clusters and have excellent performance in three datasets. In the Aggregation, the approach mistakes six points in the left bridge, which is three more than that in Ref. [21]. Ref. [21] utilizes four underlying algorithms to get that result, however the algorithm achieves an approximate performance independently. Note that average linkage clustering algorithm has an outstanding clustering that mistakes three points in right bridge like[21], and DBSCAN also performs well at $eps = 1.18$ and $minPts = 6$ by searching *eps* and *minPts* carefully. Another two datasets, the performance of the proposed algorithm is much better than remaining algorithms. For flame, the algorithm clusters ten points wrongly in the combination of two groups and remaining algorithms identify wrong structure. Furthermore, the algorithm clusters Jain exactly but other algorithms make mistakes in the overlap part.

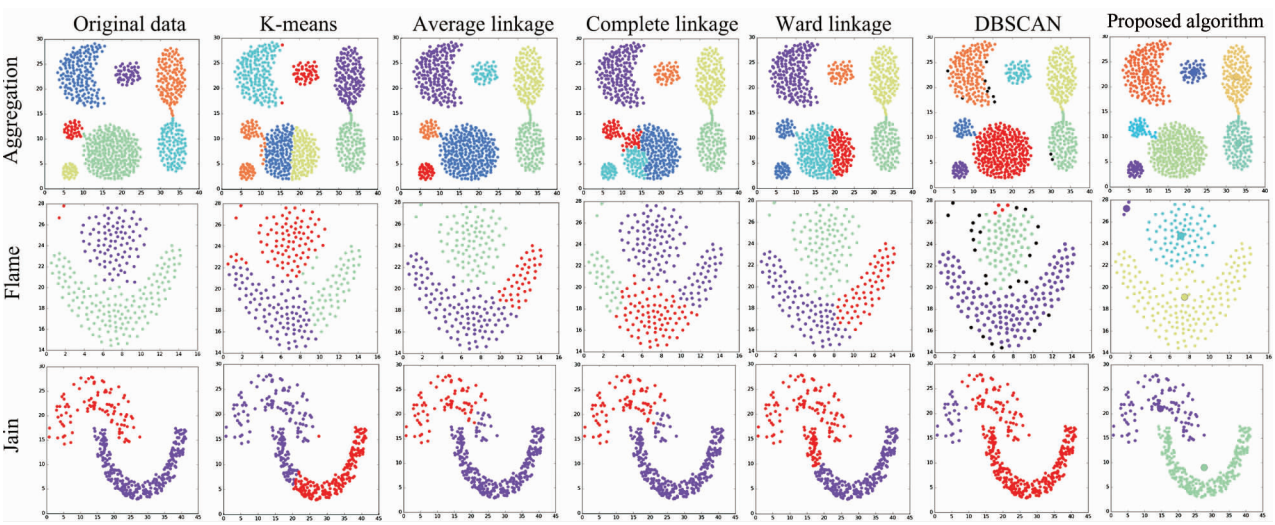

Fig. 4 Clustering in data with different shapes

3 Conclusion

A new density-based clustering algorithm is presented which is clustered from the high density-connected partition, to solve the clustering problem of the dataset with varying density and complicated structure. In essence, this algorithm is a mixture of agglomerative clustering and DBSCAN that combine their advantages and overcome their disadvantages. A lot of experiments prove that: The approach has good performance to the data with widely varying densities and extremely complex structures; It decides the optimal number of clusters automatically; Background knowledge is not needed and tuning parameters is easy and fixed; It is robust against noise and outliers.

References

[1] Andreopoulos B, An A, Wang X, et al. A roadmap of clustering algorithms: finding a match for a biomedical application[J]. *Briefings in Bioinformatics*, 2009, 10(3), 297-314

[2] Lloyd S P. Least squares quantization in PCM[J]. *IEEE Transactions on Information Theory*, 1982, 28(2):129-137

[3] Rokach, Lior, Oded M. Clustering Methods. Data Mining and Knowledge Discovery Handbook[M]. USA: Springer, 2005. 321-352

[4] Frey B J, Dueck D. Clustering by passing messages between data points[J]. *Science*, 2007, 315(5814): 972 - 976

[5] Banfield J D, Raftery A E. Model-based Gaussian and non-Gaussian clustering[J]. *Biometries*, 1993, 49(3): 803-821

[6] Kriegel H P, Kröger P, Sander J, et al. Density-based clustering[J]. *WIREs Data Mining and Knowledge Discovery*, 2011, 1(3): 231-240

[7] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, USA, 1996. 226-231

[8] Mihael A, Markus M B, Kriegel H P, et al. OPTICS: ordering points to identify the clustering structure[C]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Philadelphia, USA, 1999. 49-60

[9] Xu X, Ester M, Kriegel H P, et al. A distribution-based clustering algorithm for mining in large spatial databases[C]. In: Proceedings of the 14th ICDE, Orlando, USA, 1998. 324-331

[10] Hinneburg A, Keim D. An efficient approach to clustering large multimedia databases with noise[C]. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD 98), NewYork, USA, 1998. 58-65

[11] Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002

[12] Rodriguez A, Laio A. Clustering by fast search and find of density peaks[J]. *Science*, 2014, 344(6191), 1492-1496

[13] Boureau Y, Ponce J, LeCun Y. A theoretical analysis of feature pooling in visual recognition[C]. In: International Conference on Machine Learning, 2010, 32(4): 111-118

[14] Rousseeuw P J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis[J]. *Journal of Computational & Applied Mathematics*, 1987, 20(20), 53-65

[15] Christian W, Jan B, Richard R. Comparing the performance of biomedical clustering methods[J]. *Nature Methods*, 2015,12(11): 1033-1038

[16] Clustering benchmark datasets, http://cs.joensuu.fi/sipu/datasets/

[17] Fränti P, Virmajoki O. Iterative shrinking method for clustering problems[J]. *Pattern Recognition*, 2006, 39(5):761-765

[18] Gionis A, Mannila H, Tsaparas P. Clustering aggregation[J]. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2007, 1(1):1-30

[19] Fu L, Medico E. FLAME: a novel fuzzy clustering method for the analysis of DNA microarray data[J]. *Bmc Bioinformatics*, 2007, 8(1), 3

[20] Jain A, Law M. Data Clustering: A User's Dilemma[M]. Berlin: Springer, 2005, 3776, 1-10

[21] Li N, Latecki L J. Clustering aggregation as maximum-weight independent set[J]. *Advances in Neural Information Processing Systems* 2013. 782-790

Yuan Lufeng, born in 1982. He is a Ph. D candidate in Institute of Computing Technology, Chinese Academy of Sciences in Beijing. He received his B. S. degree from National University of Defense Technology in 2005. His research interests include the design and optimization of algorithms for machine learning, parallel processing and high performance computing.