Expert Systems With Applications 227 (2023) 120195

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

PDCSN: A partition density clustering with self-adaptive neighborhoods

Shuai Xing, Qian-Min Su *, Yu-Jie Xiong *, Chun-Ming Xia

School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai, 201620, China

ARTICLE INFO

Keywords:
Density-based clustering
Optimal neighborhood size
Nearest neighbors
Arbitrary density
Self-adaptive

ABSTRACT

Density-based clustering can discover convex and non-convex clusters without specifying the number of clusters. However, its ability to handle clusters with heterogeneous densities is limited. Although various variants solve this problem to some extent, they are still powerless for adjacent clusters with similar densities. Furthermore, the clustering performance is heavily dependent on user-specified parameters. This paper proposes a partition density clustering with self-adaptive neighborhoods (PDCSN). For the parameter dependence problem, a self-adaptive approach based on the natural neighborhood is designed. This approach utilizes the differences and intrinsic properties of sample density distributions to automatically find the optimal neighborhood size λ. A partitioning strategy based on mutual λ-nearest neighbors-connectedness is proposed to distinguish clusters with large varying densities. Moreover, a core sample search approach based on shared similarity and density heterogeneity is proposed to identify adjacent clusters with similar densities. A series of experiments on 9 synthetic, 10 real-world, and 2 image datasets demonstrate that PDCSN outperforms several famous clustering algorithms.

1. Introduction

Clustering is an unsupervised learning tool aiming to divide samples into groups with specific meanings according to the similarity of the internal attributes of the data (Fahad et al., 2014). It has been universally utilized in many industries over the past 50 years, such as recommender system (Thanh, Ali, & Son, 2017), biometrics (Ye & Ho, 2019), and object recognition (Tian et al., 2022; Zhu et al., 2020). However, since the shape and distribution of data are usually complex, clustering for data with any shapes and densities has gained widespread attention (Chowdhury, Bhattacharyya, & Kalita, 2021). Besides, clustering with low complexity is also favored due to the emergence of big data (Hu, Liu, Zhang, & Liu, 2021; Song, Yao, Nie, Li, & Xu, 2021; Zhou et al., 2020).

Density-based clustering is one of the most mainstream algorithms, such as DBSCAN (Ester, Kriegel, Sander, Xu, et al., 1996), OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999), and DPC (Rodriguez & Laio, 2014). The process of density-based clustering algorithms usually contains two steps. First, the samples within dense regions (core samples) are identified based on the defined approach for evaluating the density. Second, all connected dense regions are assigned to the same cluster. In DBSCAN, the sample density is defined by two user-specified parameters, eps and $MinPts$. A sample is identified as a core sample if at least $MinPts$ samples are within an eps distance from it (Bryant & Cios, 2018). Then, the core samples are assigned to the same cluster if they are closely packed together. However, the fixed distance parameter eps limits the ability of DBSCAN to handle clusters with heterogeneous densities. Besides, the results of DBSCAN are heavily dependent on user-specified parameters. OPTICS generates an augmented ordering of the data to identify clusters with large variations in density. Although OPTICS addresses one of the shortcomings of DBSCAN (i.e., limited ability for handling clusters with heterogeneous densities), its clustering results still depend on eps and $MinPts$, to a certain extent like DBSCAN. In DPC, density peaks are taken as cluster centroids to absorb the remaining samples, but the number of cluster centroids must be manually given. Besides, DPC selects a single core sample for the cluster centroid, resulting in the inability to generate clusters with arbitrary shapes.

1.1. Limitations of existing work and proposed solutions

Considering the shortcomings of the above algorithms, limitations of the existing work and proposed solutions are summarized in this study as follows.

Parameter dependence is an important factor leading to the instability of clustering. To avoid users repeatedly trying to find the optimal number of nearest neighbors, a self-adaptive approach is constructed to estimate the optimal neighborhood size λ (i.e., the optimal number of nearest neighbors) without user intervention. The key idea is rooted

* Corresponding authors.
E-mail addresses: xshuai@sues.edu.cn (S. Xing), suqm@sues.edu.cn (Q.-M. Su), xiong@sues.edu.cn (Y.-J. Xiong), cmxia@sues.edu.cn (C.-M. Xia).

https://doi.org/10.1016/j.eswa.2023.120195
Received 1 June 2022; Received in revised form 8 March 2023; Accepted 15 April 2023
Available online 26 April 2023

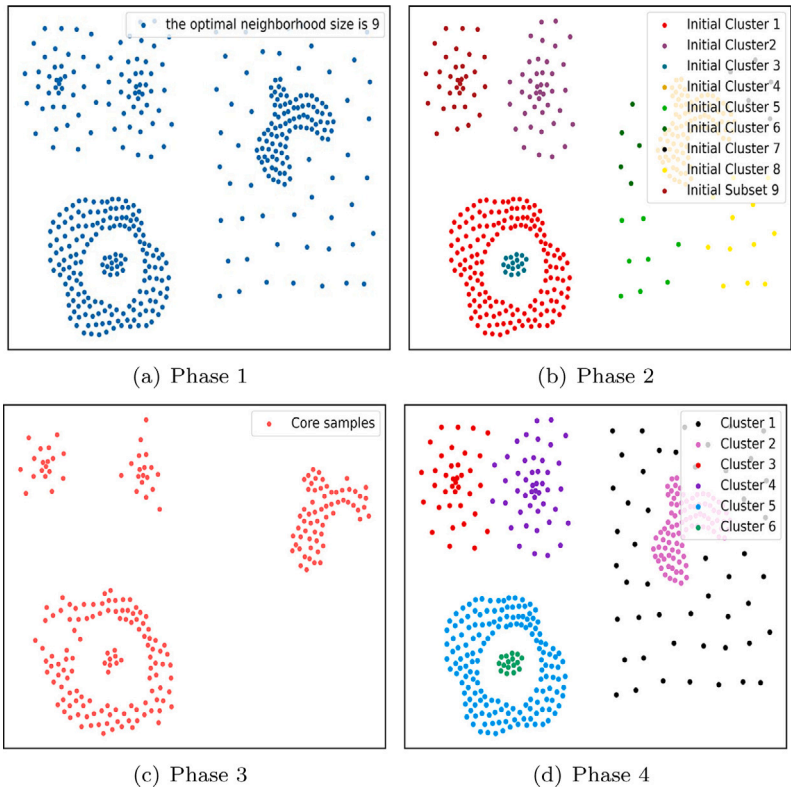


Fig. 1. Visualization of the result in each phase of PDCSN.

from that samples in dense regions contain more neighbors, while samples in sparse regions contain fewer neighbors. Thus, we utilize the differences and intrinsic properties of sample density distributions to automatically estimate the critical range of the dense/sparse neighborhood from all samples in a bottom-up iterative manner.

Density-based clustering is almost ineffective in identifying clusters with large varying densities. Although the k-nearest neighbor method breaks the traditional distance metric, it still cannot recognize the irregular distribution of clusters in some cases. To solve the above problem, we propose a partitioning strategy to partition clusters with large varying densities into different initial clusters. The samples within a cluster are all connected by mutual λ-nearest neighbors. However, if all samples are treated equally, adjacent clusters with large varying densities are partitioned together. Therefore, this strategy first discovers local high/low-density samples. Then, the local high-density sample and its mutually λ-nearest-neighbor connected samples are divided into the same initial cluster.

Traditional density-based clustering and most algorithms are powerless to deal with adjacent clusters with similar densities because distance-based and KNN-based metrics ignore the sparsity of the sample distribution in the high-dimensional space. Besides, there may be several adjacent clusters with similar densities within the initial cluster. Aiming at the above deficiencies, a novel search approach is proposed to identify the core samples near the cluster centroids. It amplifies the sparse distribution information from two aspects: (1) in the neighborhood of edge samples, local high-density samples account for a small fraction; (2) the neighborhoods of samples at the junction of adjacent clusters have greater sparsity than those near the cluster centroids. Moreover, the proposed search approach does not rely on the distance-based metric, but instead uses specific sample distribution attributes to determine the dense/sparse neighborhoods in the space. This avoids the limitations of distance metrics in high-dimensional space and enables the algorithm to handle high-dimensional data more effectively.

1.2. Contributions

Inspired by the above solutions, we propose a partition density clustering with self-adaptive neighborhoods (PDCSN). PDCSN introduces the k-nearest neighbor and statistical methods to fuse adjacent samples' density distribution information, thereby enlarging the heterogeneity and sparsity of the sample density distributions. Furthermore, rather than relying on static user-defined density, PDCSN adopts a top-down splitting strategy to identify the different cluster structures from the nearest neighbor graphs. PDCSN consists of four phases, as shown in Fig. 1. In ***Phase 1***, we propose a self-adaptive neighborhood approach to estimate the optimal neighborhood size λ (i.e., the optimal number of nearest neighbors) and construct the λ-nearest neighbor graph as input for subsequent phases. In ***Phase 2***, we propose a partitioning strategy to partition clusters with large varying densities into different initial clusters. It discovers local high/low-density samples and divides the local high-density sample and its mutual λ-nearest-neighbor-connected samples together. In ***Phase 3***, since there may be several adjacent clusters with similar densities and clusters with pseudo-high densities within the initial cluster, the core sample search approach is proposed to identify the core samples from all local high-density samples. In ***Phase 4***, the clustering result of each initial cluster is composed of core samples and their density-reachable, density-connected samples, and extended samples.

PDCSN has the following merits: (1) ability to eliminate the dependence of clustering results on user-specified parameters, (2) ability to handle clusters with varying densities, (3) ability to discover non-convex and convex shaped clusters, (4) ability to detect noise. The contribution of this paper is as follows.

(1) We propose a self-adaptive neighborhood approach to estimate the optimal neighborhood size without manual intervention.

(2) We propose a partitioning strategy to partition the initial clusters, which can handle clusters with large variations in density.

(3) We propose a core sample search approach to distinguish adjacent clusters with similar densities.

(4) Experimental results on some synthetic, real-world, and image datasets prove that PDCSN can discover clusters with arbitrary shapes and densities in noise space. Besides, PDCSN is more suitable for clustering tasks on real-world datasets compared with several state-of-the-art algorithms.

The remainder of the paper is organized as follows. Section 2 reviews the related work of clustering algorithms. Section 3 elaborates on the principle of PDCSN. Section 4 compares the performance of PDCSN with other clustering algorithms using several synthetic and real-world datasets. Finally, the conclusion is given in Section 5.

2. Related work

As an important research field of data mining, clustering algorithms are divided into different categories, such as partition-based, density-based, grid-based, integrated, and hierarchical methods. Among the partition-based methods, the most well-known clustering technique is the K-means algorithm. K-means finds convex shaped clusters efficiently, but the choice of K greatly impacts the clustering results. KMDD (Wang et al., 2018) overcomes the drawback that K-means fails to find non-convex shaped clusters by using the density-based method to merge multiple small subclusters into actual clusters.

The hierarchical clustering algorithms consider a top-down or bottom-up way to partition different hierarchical structures. CURE (Guha, Rastogi, & Shim, 1998) conducts a hierarchical agglomerate clustering on subsets that have been randomly selected. Chameleon (Karypis, Han, & Kumar, 1999) merges pairs of clusters by inter-cluster connectivity and similarity. Both of the algorithms are somehow suitable for discovering clusters with arbitrary shapes. However, they do not perform well on high-dimensional data. DWMB (Rehman & Belhaouari, 2022) resembles the hierarchal and density-based clustering algorithms, where density-based methods are used to merge small sub-clusters that have been partitioned.

Density-based clustering algorithms have been widely applied due to their ability to discover clusters with arbitrary shapes. Researchers have proposed numerous variants of DPC. Chen et al. proposed CLUB (Chen et al., 2016), which introduces the concept of mutual k-nearest neighbors-connectedness to partition initial clusters. However, samples residing in low-density regions will partition two adjacent clusters together. SNN-DPC (Liu, Wang, & Yu, 2018) estimates local densities based on shared nearest neighbor similarity. ClusterDv (Marques & Orger, 2018) overcomes the effect of user-specified parameters by automatically finding the number of clusters based on the separability index. GADPC (Xu & Jiang, 2022) automatically determines the centroids of clusters based on the turning angle and the graph connectivity of centroids. Chowdhury et al. proposed UIFDBC (Chowdhury et al., 2021), which introduces the separability of cluster centers to find the optimal number of clusters. Zhou et al. proposed ICKDC (Zhou, Si, Sun, Qu, & Hou, 2022), which utilizes a set of core samples as the center to represent a cluster and classifies the remaining samples based on the distribution information. DPC-AHS (Zhang, Miao, Tian, & Wang, 2022) assigns some samples as candidate centers to find cluster centers by residual analysis and linear regression.

Some variants of DBSCAN are proposed to improve the ability to handle clusters with heterogeneous densities. SNN (Ertöz, Steinbach, & Kumar, 2003) takes the shared similarity of the k nearest neighbors of samples as the density estimator to find core samples. However, it highly depends on the specified $Minpts$ and the minimum shared similarity. GMDBSCAN (Xiaoyun, Yufang, Yan, & Ping, 2008) is a grid partitioning-based algorithm, which determines the density definition by the number of samples in each grid. However, when it deals with high-dimensional data, the number of grid cells will increase exponentially. By combining the ant clustering algorithm with PDBSCAN, Jiang et al. construct k nearest neighbor distance graphs to determine the ε-distance of each subset (Jiang, Li, Yi, Wang, & Hu, 2011). Although this variant finds clusters with different densities, how to determine the initial parameters becomes its main limitation. Avory Bryant et al. proposed RNN-DBSCAN (Bryant & Cios, 2018), which uses the number of reverse nearest neighbors as a new method to define the sample density. Its advantage is that it only needs a specified number of nearest neighbors as a global parameter to identify clusters with varying densities. Besides, it breaks the symmetry of distance metrics and outperforms other similar algorithms (Cassisi, Ferro, Giugno, Pigola, & Pulvirenti, 2013; Lv et al., 2016). Li et al. proposed ADBSCAN (Li, Liu, Li, & Gan, 2020), which utilizes the inherent features of the k-nearest neighbor graph to find dense regions and applies statistical methods to filter samples in dense regions. Ros et al. proposed S-DBSCAN (Ros, Guillaume, Riad, & El Hajji, 2022), which hybridizes the concepts of k-nearest, distance, and density peak to assign variable clusters without depending on global density thresholds. However, RNN-DBSCAN, ADBSCAN, and S-DBSCAN are incapable of dealing with adjacent clusters with similar densities because they ignore the density distribution sparsity in high-dimensional space. KR-DBSCAN (Hu et al., 2021) introduces the concepts of the reverse nearest neighbor and the influence space to distinguish adjacent clusters with varying densities.

Some other variants of DBSCAN are proposed to address the high dependence of clustering results on user-specified parameters. Chen et al. proposed APSCAN (Chen, Liu, Qiu, & Lai, 2011), which applies the affinity propagation method to detect the local densities of all samples and generates a normalized density list as the input of the parameters to generate the final clustering result. DSets-DBSCAN (Hou, Gao, & Li, 2016) performs histogram equalization to the similarity matrices and applies DBSCAN to extend the results of dominant sets that are independent of user-specified parameters. Although both variants do not rely on user-specified parameters, their ability to handle clusters with varying densities is limited. Besides, their time overhead is very high on big datasets. Zhang et al. proposed DC-SKCG (Zhang, Du, Qu, & Sun, 2021), which optimizes the clustering process through a conflict game approach based on the adaptive cut-off distance and the shared k-nearest neighbors to reduce the parameter sensitivity of the algorithm.

Some variants of DBSCAN are proposed to process high-dimensional/large-scale datasets. Boonchoo et al. proposed GDCF (Boonchoo et al., 2019). Aiming at the defects of grid-based DBSCAN, it utilizes bitmap indexing and cluster forest to overcome the problems in the neighbor explosion and redundancies in merging. μDBSCAN (Sarma et al., 2019) is the DBSCAN algorithm based on micro-clusters, which introduces distributed methods to break the sequential access of data and uses the μR-tree and micro-clusters concept to optimize the processing time of neighborhood queries. H-DBSCAN (Weng, Gou, & Fan, 2021) reduces the clustering time in two ways. The first one is to apply HNSW technology instead of linear search and the second one is to reduce samples presented to DBSCAN. BLOCK-DBSCAN (Chen et al., 2021) first utilizes $\epsilon/2$-norm ball sphere to identify the internal core blocks and then introduces a fast approximate algorithm to determine the density-reachability between the internal core blocks. OP-DBSCAN (Hanafi & Saadatfar, 2022) takes a part of the whole samples as the operation set and adopts local samples to calculate the density of each sample, thus greatly reducing the calculation cost of clustering.

3. The algorithm: PDCSN

3.1. Clustering problem

The clustering problem solved in this paper is defined as follows: for a given dataset X, the problem is to divide C clusters $C_1, C_2, \ldots, C_c$. The clusters satisfy the following conditions: each cluster contains at least one sample (i.e., $C_i \neq \emptyset, i = 1, 2, \ldots, C$), and each sample is assigned to one and only one cluster (i.e., $C_i \cap C_j = \emptyset, i \neq j$), except for noisy samples.

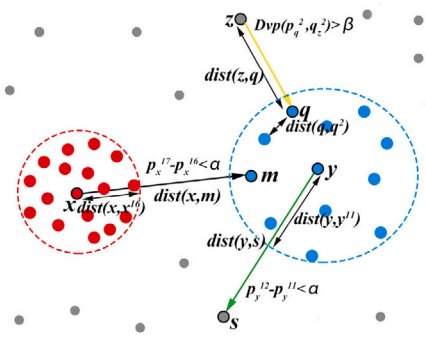


Fig. 2. Illustration of density constraints. The neighborhood searching range of sample x is expanded to 17, and $x^{17} = m$. Since x does not satisfy the density constraint ($\rho_x^{17} - \rho_x^{16} < \alpha$), the expansion terminates, i.e., $\mathrm{CN}(x) = 16$. The neighborhood searching range of sample z is expanded to 2, and $z^2 = q$. Because of $Dvp(\rho_q^2, \rho_z^2) > \beta$, $\mathrm{CN}(z) = 1$.

In order to address the clustering problem, this paper proposes a novel density-based algorithm (PDCSN) that overcomes the challenge of discovering clusters with arbitrary shapes, sizes, and densities in noise space without user-specified parameters. PDCSN contains four phases: (1) estimate the optimal neighborhood size (λ) and construct the λ-nearest neighbor graph; (2) discover local high/low-density samples and partition the initial clusters; (3) identify the core samples from all local high-density samples; and (4) generate the structure of clusters within each initial cluster.

3.2. Self-adaptive estimation of optimal neighborhood size

Let X represent a dataset of size $n = |X|$, and the dimensional space of each sample is d, $\forall x \in X : x \in R^d$. All calculations in this paper are expressed by Euclidean distance, where $d(x, y) = \sqrt{\Sigma_{i=1}^{d}(x_i - y_i)^2}$. Moreover, since PDCSN relies on the technique of KNN, the basic concepts of KNN-based density estimation include the following.

Definition 1 (*k-Nearest Neighbors*)**.** The k-nearest neighbors of sample x are the k most similar neighbors of x, denoted as $N_k(x) = \{y | d(x, y) \le d(x, x^k)\}$, where $x \in X$, $y \in X/\{x\}$, and x^k is the kth nearest neighbor of x.

Definition 2 (*k-Nearest Neighborhood Density*)**.** The k-nearest neighborhood density of sample x is the ratio of k and $d(x, x^k)$, expressed as $\rho_x^k = k/d(x, x^k)$.

Definition 3 (*Variation Index of Relative Density*)**.** The variation index of relative density between samples x and y in the k-nearest neighborhood, $Dvp(\rho_x^k, \rho_y^k)$, is defined as:

$$Dvp(\rho_x^k, \rho_y^k) = |\frac{\rho_x^k - \rho_y^k}{(\rho_x^k + \rho_y^k)/2}| \tag{1}$$

Inspired by the concept of natural neighborhood (Cheng, Zhu, Huang, Wu, & Yang, 2019), PDCSN constructs a self-adaptive neighborhood approach based on the differences and intrinsic properties of sample density distributions. Compared with other variants based on the k-nearest method, this approach can automatically estimate the optimal neighborhood size λ (i.e., the optimal number of nearest neighbors) without user specification. The intuition behind it is that samples located in dense regions contain more neighbors, whereas samples located in sparse and cluttered regions contain fewer neighbors. The estimation of λ is divided into two steps: (1) find the critical range of the dense/sparse neighborhood of each sample x, $\mathrm{CN}(x)$; (2) select the largest critical range as λ, i.e., $\lambda = \{\mathrm{CN}(x) | x \in X\}$. Specifically, continuously expand the neighborhood searching range k ($2 \le k \le n$), and each time judge whether sample x satisfies the following two density constraints: (1) $\rho_x^k - \rho_x^{k-1} > \alpha$ (α is the threshold) and (2) $Dvp(\rho_x^k, \rho_{x^k}^k) < \beta$ (β is the threshold). The $1, \ldots, k-1/k$ neighborhood of x is considered a dense/sparse neighborhood when at least one of the above two density constraints is not satisfied. At this moment, The searching range $k-1$ is $\mathrm{CN}(x)$. The two constraints effectively characterize the variation of the sample density distribution from different perspectives. Fig. 2 shows an example explaining density constraints. The definition of the critical range of the dense/sparse neighborhood is given by Definition 4.

In order to fill in the gaps with no user-specified parameters, this approach automatically determines the above thresholds, α and β. The difference set of between the k and $k+1$-nearest neighborhood densities of all samples is expressed in ascending order as $P^k = \mathrm{Ascend}(\{p | p = \rho_x^{k+1} - \rho_x^k, x \in X\})$. $Dvr^k = \mathrm{Descend}(\{v | v = Dvp(\max_{i=1}^{k}(\rho_{x^i}^k), \rho_x^k), x \in X\})$ denotes the set of density variation indices of all samples in the k-nearest neighborhood sorted in descending order, where $\mathrm{Ascend}(\cdot)/\mathrm{Descend}(\cdot)$ is the ascending/descending sort function. To estimate α, PDCSN calculates the mean of the first half of P^k ($\Delta^k = \mathrm{mean}(\Sigma_{i=1}^{n/2} P^k)$), considering that noise samples only account for a tiny portion of the dataset. Generally, $\Delta^2, \ldots, \Delta^{n-1}$ show a stable decreasing trend. Therefore, to strengthen the constraints of the density distributions and avoid the effect of overlapping samples, the first value that is not infinitesimal, Δ^τ ($\tau = \min\{k | \Delta^k \ne -\infty, 2 \le k \le n\}$), is selected as α by searching in ascending order from Δ^2. In addition, discrepancies in the density distribution of some datasets are large, causing Δ^τ to be an outlier. So, $\Delta^{\tau+1}$ and $\Delta^{\lfloor\sqrt{n}\rfloor}$ are considered to correct it. To estimate β, PDCSN calculates the mean of the first half of Dvr^k ($\Omega^k = \mathrm{mean}(\Sigma_{i=1}^{n/2} Dvr^k)$) and selects the maximum of Ω^τ and $\Omega^{\lfloor\sqrt{n}\rfloor}$ as β. α and β are given by Eq. (2).

$$\alpha = \begin{cases} \max(0, \Delta^\tau - |\Delta^{\tau+1}|) & Dvp(\Delta^\tau, \Delta^{\lfloor\sqrt{n}\rfloor}) > 0.9 \\ \max(0, \Delta^\tau + |\Delta^{\tau+1}|) & Dvp(\Delta^\tau, \Delta^{\lfloor\sqrt{n}\rfloor}) \le 0.1 \\ \max(0, \Delta^\tau) & else \end{cases} \tag{2}$$
$$\beta = \max(\Omega^\tau, \Omega^{\lfloor\sqrt{n}\rfloor})$$

Definition 4 (*Critical Range of the Dense/sparse Neighborhood*)**.** The critical range of the dense/sparse neighborhood of sample x, $\mathrm{CN}(x)$, is $k-1$ if $(\rho_x^k - \rho_x^{k-1} > \alpha) \cap (Dvp(\rho_x^k, \rho_{x^k}^k) < \beta) = 0$ and $\forall r \in \{2, \ldots, k-1\}$: $(\rho_x^r - \rho_x^{r-1} > \alpha) \cap (Dvp(\rho_x^r, \rho_{x^r}^r) < \beta) = 1$, where k is $2, 3, \ldots, n$.

3.3. Partition of the initial clusters

In order to distinguish clusters with large varying densities, PDCSN considers that samples within a cluster are all connected by mutual λ-nearest neighbors. Furthermore, to prevent two adjacent clusters with large variations in density are partitioned together, PDCSN breaks the idea of treating all samples equally. It first uses the distance between the sample and its λth nearest neighbor to discover local high-density samples, as shown in Definition 5. After traversing all samples, a set of local high-density samples LH is obtained. Figs. 3(a) and (c) display the local high-density samples on the datasets of Compound and Flame (Fränti & Sieranoja, 2018). Next, according to the concept of MλNN-connected given in Definition 6, the randomly selected sample from LH and its MλNN-connected samples are partitioned into the same initial cluster, as described in Definition 7. If a sample x is MλNN-connected to a sample y, they are incredibly similar and likely from the same cluster, as shown in Fig. 3(b).

Definition 5 (*Local High-Density Sample*)**.** A sample x is a local high-density sample if it satisfies the following conditions:

$$dist(x, x^\lambda) \le \frac{\sum_{i=1}^{\lambda} dist(x^i, (x^i)^\lambda) + \min_{1 \le i \le \lambda} dist(x^i, (x^i)^\lambda)}{\lambda} \tag{3}$$

where x^i is the ith nearest neighbor of x, and $(x^i)^\lambda$ is the λth nearest neighbor of x^i.

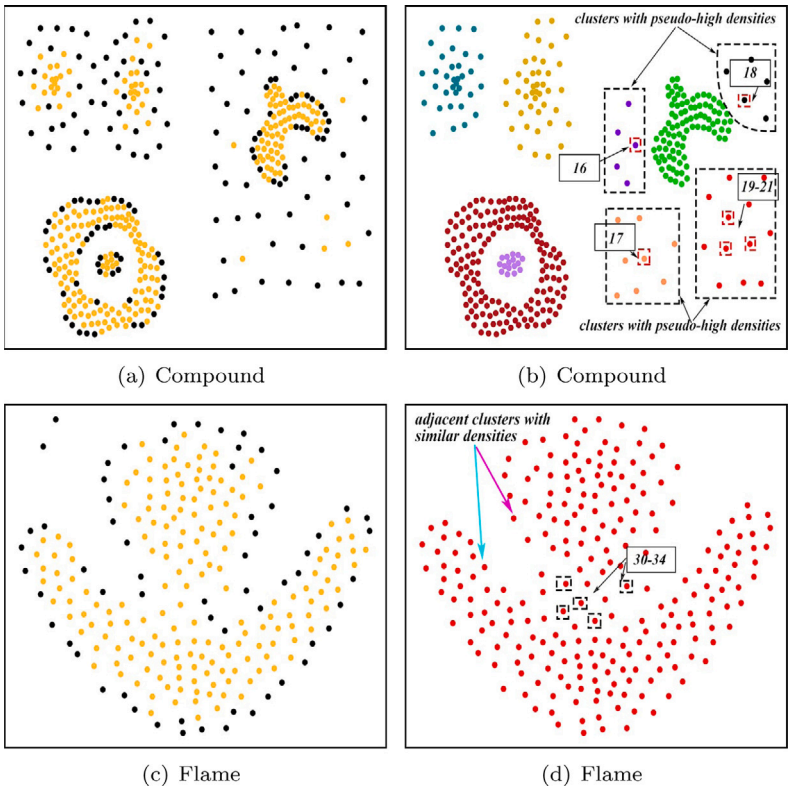


Fig. 3. Partition of initial clusters on datasets Compound and Flame. The yellow samples in (a) and (c) are local high-density samples. Different colors represent different initial clusters in (b) and (d). Besides, the clusters with pseudo-high densities are marked with black boxes in (b), and the initial cluster in (d) is composed of two adjacent clusters with similar densities.

Definition 6 (*Mutual λ Nearest Neighbors-Connected*)**.** A sample y is mutual λ nearest neighbors-connected (MλNN-connected) to a sample x if there is a chain of samples $x_1, \ldots, x_l$, $x_1 = x$, $x_l = y$ and satisfies $\forall 1 \le i \le l-1$: $x_i \in LH$, $x_i \in N_\lambda(x_{i+1})$ and $x_{i+1} \in N_\lambda(x_i)$.

Definition 7 (*Initial Cluster*)**.** An initial cluster Ic is a non-empty subset of X and satisfies the following conditions: $\forall x, y \in X$: if $x \in Ic$, y is MλNN-connected to x, then $y \in Ic$.

3.4. Identification of core samples

Although the initial clusters successfully distinguish numerous clusters with varying densities, there may be several adjacent clusters with similar densities and clusters with pseudo-high densities within the initial cluster. This is mainly because samples far from the cluster centers are misclassified as local high-density samples. As illustrated in Fig. 3(b), because samples No. 16–21 are near the edge of the cluster, the clusters composed of them are spurious. Also, two adjacent clusters with similar densities are partitioned into the same initial cluster due to samples No. 30–34 near their junction in Fig. 3(d).

To further divide the initial clusters, the two constraints in Definition 8 are used to judge whether a local high-density sample x is a core sample. Precisely, (1) density heterogeneity constraint: x's density heterogeneity $\mathrm{H}(x)$ is the ratio of the number of local high-density samples to half the number of local low-density samples in $N_\lambda(x)$. If x is near the cluster edge, then $N_\lambda(x)$ has fewer local high-density samples (i.e., $\mathrm{H}(x) < 1$). (2) shared similarity constraint: x's shared similarity set $\mathrm{S}(x)$ consists of the shared similarities between x and its λ nearest neighbors, where the shared similarity is expressed as the number of nearest neighbors the two samples share in λ-nearest neighbors. If x lies at the junction of adjacent clusters, then the λ-nearest neighbor of x shares a smaller amount with λ-nearest neighbors of the sample near the cluster center (i.e., there are tiny elements in $\mathrm{S}(x)$). When the estimated λ is small, there will be an abnormal judgment because of the severe left deviation of the data distribution in $\mathrm{S}(x)$. Thus, 3 is the lower limit of the elements in $\mathrm{S}(x)$. These two constraints are vital for the identification of clusters with heterogeneous density distributions because the sample distribution sparsity is adequately amplified. Figs. 4(a) and (b) show that samples No. 16–21 and No. 30–32 are filtered out by density heterogeneity and shared similarity constraints, respectively.

Definition 8 (*Core Sample*)**.** A sample x is a core sample if $x \in LH$ and satisfies the following conditions:

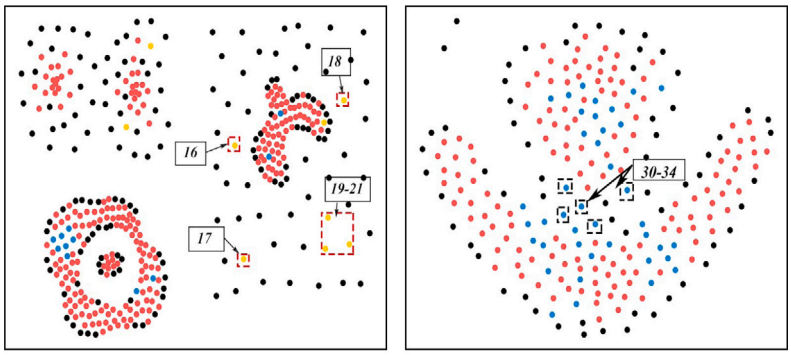
(1) $\mathrm{H}(x) \ge 1$, where $\mathrm{H}(x) = \dfrac{|\{y | y \in N_\lambda(x), y \in LH\}|}{|\{z | z \in N_\lambda(x), z \notin LH\}| / 2 + 1}$.

(2) $\max(3, \min(\mathrm{S}(x))) > (\lambda - median_x)/2$, where $median_x$ is the median of $\mathrm{S}(x)$ and $\mathrm{S}(x) = \{len | len = |N_\lambda(x) \cap N_\lambda(y)|, y \in N_\lambda(x)\}$.

3.5. Generation of clustering result

Generally, samples within a cluster consist of all density-reachable and density-connected samples from a given core sample. However, PDCSN performs a similar process to RNN-DBSCAN, i.e., it extends the identified clusters and takes them as the final clustering result. Besides, PDCSN differs from RNN-DBSCAN in that the samples in different initial clusters are not density-reachable. This prevents clusters with varying densities amongst different initial clusters from merging again.

Definitions 9–11 give the definitions for directly density-reachable, density-reachable, and density-connected. The cluster is defined as shown in Definition 12. Note that a boundary sample x in cluster C may be reachable from some core sample in another cluster C'. To this end, PDCSN depends on sample ordering to assign boundary samples. Furthermore, since the core samples are highly near the cluster centroids, more unclustered samples are generated after all clusters



(a) Compound (b) Flame

Fig. 4. Identification of core samples on datasets Compound and Flame. Core samples are shown in red. Local high-density samples filtered out by the density heterogeneity constraint are shown in yellow. Local high-density samples filtered out by the shared similarity constraint are shown in blue.

have been identified. Given a cluster C, PDCSN generates an extended cluster of C, $\bar{C}$, by inserting unclustered samples into C. That is, $\bar{C}$ is a superset of C, $\bar{C} \supseteq C$. The unclustered sample x is inserted into C if there is a clustered sample x^i closest to x in the λ-nearest neighbors of x and $\rho_x^i \geq \rho_{x^i}^i/2$, as shown in Definition 13. After cluster extension is complete, unclustered samples are considered noise samples.

Definition 9 (*Directly Density-Reachable*)**.** The sample y is directly density-reachable from the sample x if x is the core sample, $x, y \in Ic$, and $y \in N_k(x)$.

Definition 10 (*Density-Reachable*)**.** The sample y is the density-reachable from the sample x, if there is the sample chain $x_1, \ldots, x_l$, $x_1 = x$, $x_l = y$ and satisfies the conditions: $\forall 1 \leq i \leq l-1$: x_i is the core sample, x_{i+1} is directly density-reachable from x_i or x_i is directly density-reachable from x_{i+1}.

Definition 11 (*Density-Connected*)**.** The sample x is density-connected to the sample y if there is the sample z such that both x and y are density-reachable from z.

Definition 12 (*Cluster*)**.** A cluster C is a non-empty subset of X and satisfies the following conditions:

(1) $\forall x, y \in X$: if $x \in C$ and y is density-reachable from x, then $y \in C$
(2) $\forall x, y \in C$: x is density-connected to y

Definition 13 (*Extended Cluster*)**.** Given cluster C, extended cluster of C, $\bar{C}$, is equal to the union of C and all samples x that satisfy the following conditions:

(1) $x \notin C$ and there exists no other cluster $C' \neq C$ such that $x \in C'$.
(2) There exists a sample $y \in C$ such that $y = x^m \in N_\lambda(x)$, and there exists no other sample $z \in (C' \cup C)$ such that $dist(x, z) < dist(x, y)$.
(3) The previous condition holds and $p_x^m \geq p_y^m/2$.

Definition 14 (*Noise Sample*)**.** Let $\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_l$ be the clusters of dataset X, a sample x is a noisy sample if x does not belong to any cluster $\bar{C}_{1 \leq i \leq l}$.

3.6. PDCSN algorithm

Given a dataset X as the input of PDCSN, in ***Phase 1***, first traverse X in arbitrary order and estimate the critical range of dense/sparse neighborhood for each sample via Eq. (2) and Definition 4. Then choose the largest critical range as the optimal neighborhood size (λ) and construct the λ nearest neighbor graph as a parameter for subsequent operations. In ***Phase 2***, PDCSN first discovers all local high-density samples via Eq. (3). Then, a randomly selected local high-density

Algorithm 1 PDCSN(X)

Input: Dataset X **Output**: Clustering result of X
Phase 1: Estimation of λ
1: **for** each $x \in X$ **do**
2: Calculate CN(x) via Definition 4 and Eq. (2)
3: **end for**
4: $\lambda = \max(\{\mathrm{CN}(x) | x \in X\})$
Phase 2: Partition of initial clusters
1: $\{x_{ic} = 0 | x \in X\}, i = 1$
2: Calculate a set of local high-density samples LH via Definition 5
3: **while** $\exists x_{ic} = 0$ & $x \in LH$ **do**
4: $Ic_i = \{x, z | z_{ic} = 0, z \in N_\lambda(m), m \in N_\lambda(z), m \in Ic_i \cap LH\}$
5: $\{z_{ic} = i | z \in Ic_i\}, i = i + 1$
6: **end while**
Phase 3: Identification of core samples
1: $Core = \{\}$
2: **for** each $x \in LH$ **do**
3: **if** x satisfies Definition 8 **then**
4: $Core = Core \cup \{x\}$
5: **end if**
6: **end for**
Phase 4: Generation of clustering results
1: Initial cluster id $c = 1$, $\{x_{cid} = 0 | x \in X\}$, $\{x_{visit} = 0 | x \in X\}$
2: **for** each $x \in Core$ & $x_{visit} = 0$ **do**
3: $Cluster = \{x, y | y \in N_\lambda(p), y_{ic} = p_{ic}, p \in Cluster \cap Core, p_{visit} = 0\}$
4: $\{p_{visit} = 1 | p \in Cluster \cap Core\}$
5: **if** $Flag = \{p_{cid} | p_{cid} \neq 0, p \in Cluster \cap Core\} == \emptyset$ **then**
6: $\{y_{cid} = c | y \in Cluster, y_{cid} = 0\}$, $c = c + 1$
7: **else**
8: $Sample = \{y | y \in X, y_{cid} \in Flag\} \cup Cluster$
9: $\{y_{cid} = \min(Flag) | y \in Sample\}$
10: **end if**
11: **end for**
12: $UC = \{x | x \in X, x_{cid} = 0\}$
13: **for** each $x \in UC$ **do**
14: **for** each $y = x^i \in N_\lambda(x)$ **do**
15: **if** $y_{cid} \neq 0$ and $y \notin UC$ **then**
16: $\{x_{cid} = y_{cid} | \rho_x^i \geq \rho_y^i/2\}$
17: break
18: **end if**
19: **end for**
20: **end for**
21: $noise = \{x | x_{id} = 0, x \in X\}$
22: **return** X_{cid}

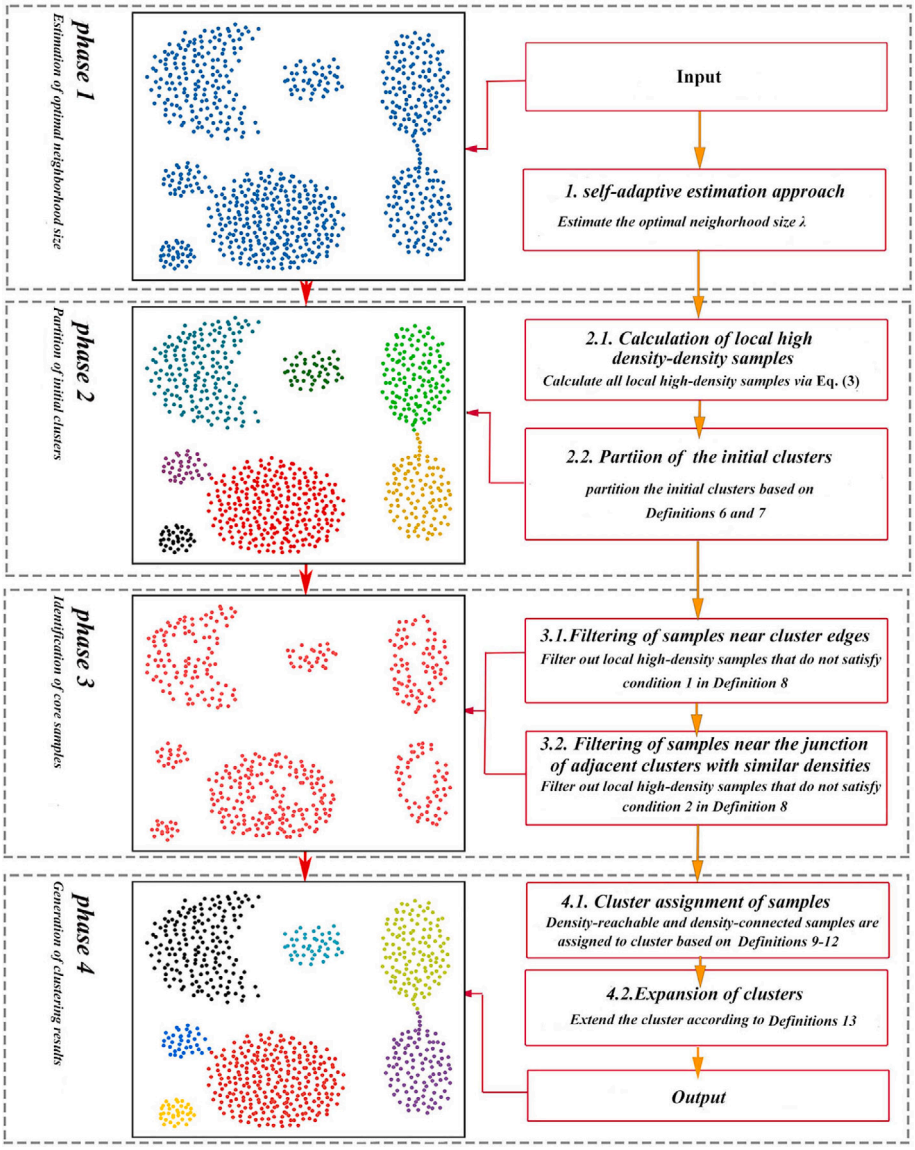


Fig. 5. The implementation of PDCSN.

sample and its MλNN-connected samples are partitioned into the same initial cluster based on Definitions 6 and 7. In ***Phase 3***, the local high-density samples that satisfy Definition 8 are identified as the core samples. In ***Phase 4***, traversed in samples order for all samples, if the current sample is the core sample, it and its density-reachable and density-connected samples are assigned to the same cluster. Finally, the extended clusters of all clusters are obtained as the algorithm's output according to Definition 13. Algorithm 1 outlines the process of PDCSN. Fig. 5 summarizes the implementation of PDCSN.

3.7. Complexity analysis

The complexity of PDCSN depends on the following five steps: (1) calculate the nearest neighbors; (2) estimate the optimal neighborhood size λ; (3) partition the initial clusters; (4) identify the core samples; (5) generate the cluster structures.

The naive solution to nearest neighbor problems in the first step is $O(n^2)$. Many existing methods solve the nearest neighbor problem, such as space-partitioning tree (k-d tree (Bentley, 1975), R-tree (Guttman, 1984)) and locality hashing techniques (Wang et al., 2014). The k-d tree considered in DBSCAN reduces the computational complexity to $O(n\times\log n)$. However, these approaches are closely related to the density variations and distributions of the data. In BLOCK-DBSCAN (Chen et al., 2021), the cover tree (Beygelzimer, Kakade, & Langford, 2006) is introduced to accelerate the calculation of nearest neighbors. All these approaches can be applied in PDCSN. Since this paper chooses the k-d tree to calculate and store the λ nearest neighbors, in the first step, the average/worst time complexity is $O(n\times\log n)/O(n^2)$, and the average/worst space complexity is $O(\lambda\times n)/O(n^2)$. In the second step, because the sample's 1,2, …, $\sqrt{n}$ nearest neighbor density and the critical range of the dense/sparse neighborhood need to be calculated and stored, the time/space complexity is $O(n^{1.5})/O(n)$. In the third step, the time/space complexity required to judge local high-density samples is $O(\lambda\times n)/O(n)$. The time/space complexity of assigning samples to the initial clusters is $O(\lambda\times n)/O(n)$. As a result, the overall time/space complexity of the third step is $O(\lambda\times n)/O(n)$. The complexity of the fourth step depends on solving the shared similarity problem and storing the core samples. The complexity of solving the number of nearest neighbors the two samples share in λ-nearest neighbors is $O(\lambda)$. Thus, the time/space complexity of the fourth step is $O(\lambda^2\times n)/O(n)$. The time/space complexity of the fifth step is $O(\lambda\times n)/O(n)$.

Combining the above five steps, the average/worst time complexity of PDCSN is $O(n\times\log n+(2\times\lambda+\lambda^2)\times n+n^{1.5})/O(n^2+(2\times\lambda+\lambda^2)\times n+n^{1.5})$. The

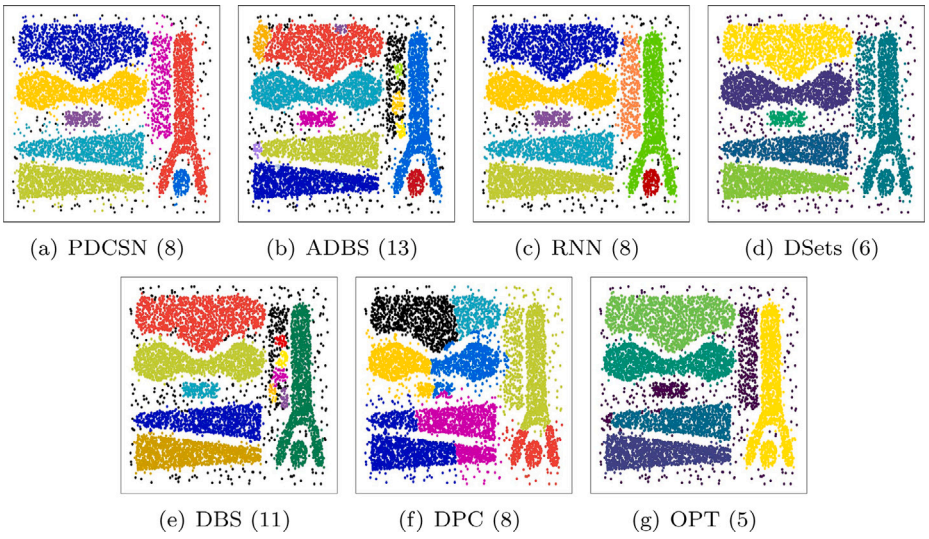


Fig. 6. Clustering results on D1. In parentheses is the number of clusters.

Table 1
Time and space complexity for PDCSN.

Step	Time complexity	Space complexity
Calculate nearest neighbors	$O(n \times \log n)$	$O(\lambda \times n)$
Estimate λ	$O(n^{1.5})$	$O(n)$
Partition initial clusters	$O(\lambda \times n)$	$O(n)$
Identify core samples	$O(\lambda^2 \times n)$	$O(n)$
Generate clusters	$O(\lambda \times n)$	$O(n)$
Average cost	$O(n \times \log n + (2 \times \lambda + \lambda^2) \times n + n^{1.5})$	$O((4 + \lambda) \times n)$
Worst cost	$O(n^2 + (2 \times \lambda + \lambda^2) \times n + n^{1.5})$	$O(4 \times n + n^2)$

average/worst space complexity of PDCSN is $O((4+\lambda)\times n)/O(4\times n+n^2)$. The space complexity and time complexity of PDCSN are shown in Table 1.

4. Experiments

In this section, to verify the clustering performance, PDCSN is compared with six typical clustering algorithms, including DBSCAN (DBS) (Ester et al., 1996), ADBSCAN (ADBS) (Li et al., 2020), RNN-DBSCAN (RNN) (Bryant & Cios, 2018), DSets-DBSCAN (DSets) (Hou et al., 2016), DPC (Rodriguez & Laio, 2014), and OPTIC (OPT) (Ankerst et al., 1999). Since the initial parameters of the algorithms influence the clustering results, all initial parameters of these algorithms are set as the suggested values given by the original references. The initial parameters of these algorithms are described in Table 2. Various public datasets are utilized, including 9 synthetic datasets (Fränti & Sieranoja, 2018; Karypis et al., 1999), 10 real-world datasets from the UCI machine learning repository (Dua & Graff, 2017), the ORL face dataset (Samaria & Harter, 1994), and the USPS digital recognition dataset (Wolf, Hassner, & Maoz, 2011). In addition, RNN, ADBS, DSets, and DPC are obtained from open-source codes that the original author provided. DBS and OPT are obtained from scikit-learn. To accurately measure the performance of the clustering algorithm, four well-known performance evaluation metrics are applied, including Accuracy (ACC), F-measure (F), Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI).

4.1. Clustering on the synthetic datasets

The 7 synthetic datasets applied in this experiment are divided into two categories. The first category is 2 synthetic datasets without ground truth, namely D1 and D2. The second category is 5 synthetic datasets with ground truth, namely Spril, Aggregation, Jain, Compound, and D31. These are classic datasets that are valuable for the evaluation of clustering performance. Table 3 lists the characteristics of these datasets, including the number of samples (n), dimensionality (d), and the standard number of clusters (c). Since the first category of synthetic datasets has no ground truth, the performance of algorithms can only be analyzed from the clustering result graph. The parameter settings of the algorithms on the second category of synthetic datasets are listed in Table 4. Moreover, the ACC, F, NMI, and ARI scores on the second category of datasets are shown in Table 5.

(1) D1 consists of clusters with small variations in density. Besides, Noise samples are located in the space between clusters. The clustering results of seven algorithms on D1 are shown in Fig. 6. Only PDCSN and RNN successfully identify the cluster structures and detect background samples as noise samples. Although DBS identified the approximate shapes of the clusters, clusters with sparse density distributions are misidentified by it. The ability of ADBS to detect high-density regions is limited, resulting in its inability to identify core samples within sparse clusters correctly. Since DSets cannot divide clusters with similar densities close to each other into different dominant sets, it merges two adjacent clusters on the right. DPC identifies no clusters due to the shortcomings of the aggregation strategy. OPT misclassifies samples in sparse clusters as noise samples.

(2) D2 contains irregular non-spherical and regular spherical clusters in the noise space. Furthermore, the significant intervals in the distance between clusters alleviate the difficulty of clustering. In Fig. 7, PDCSN, ADBS, RNN, and DBS identify the correct cluster structures. DSets incorrectly classify the two adjacent clusters with similar densities on the upper right together. Neither DPC nor OPT can identify the correct number of clusters.

(3) Spril consists of 3 spiral-shaped clusters with similar densities. Aggregation contains 7 spherical-shaped clusters with small variations in density. As shown in Figs. 8 and 9, only DSets detect the incorrect number of clusters. Because OPT, ADBS, and DBS find few core samples, they mistake the samples within the clusters to as noisy samples. In Table 5, the ACC, F, NMI, and ARI scores of PDCSN are optimal on Spril. Although the F, NMI, and ARI scores of PDCSN are not the highest on Aggregation, the differences from the highest values are negligible. The difference between the minimum value of the four metrics in PDCSN and the maximum value in RNN is only 0.5%.

(4) Jain consists of two clusters with large varyings in density. Fig. 10 shows PDCSN, ADBS, and OPT successfully identifying clusters. RNN classifies edge samples as noise samples. Because DBS and DSets cannot handle clusters with different densities, they misidentify the

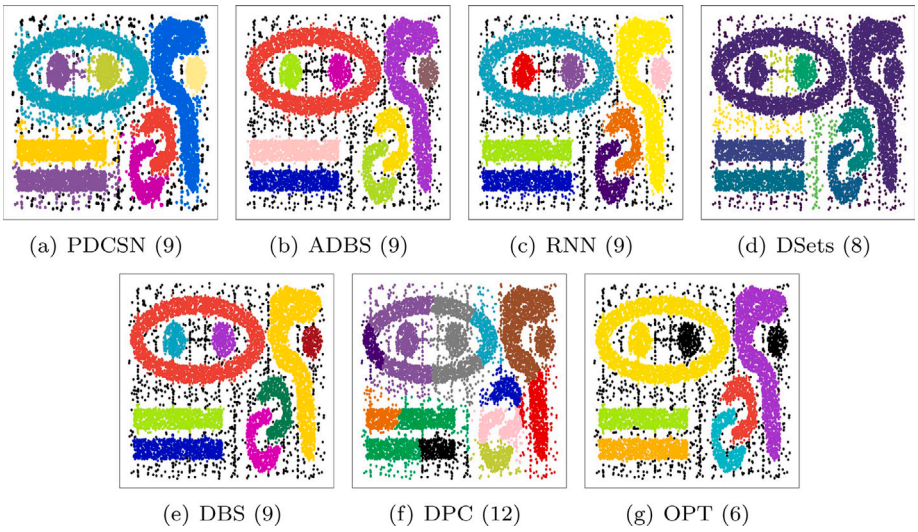


Fig. 7. Clustering results on D2. In parentheses is the number of clusters.

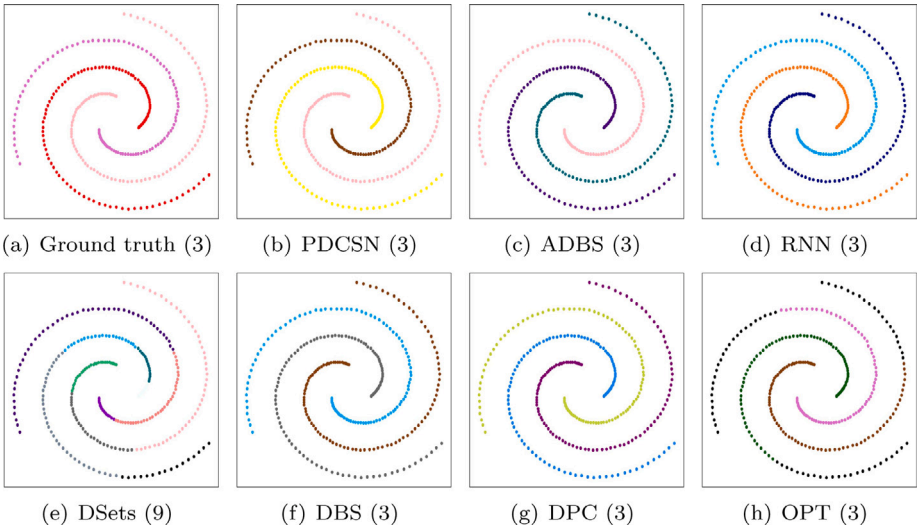


Fig. 8. Clustering results on Spril. In parentheses is the number of clusters.

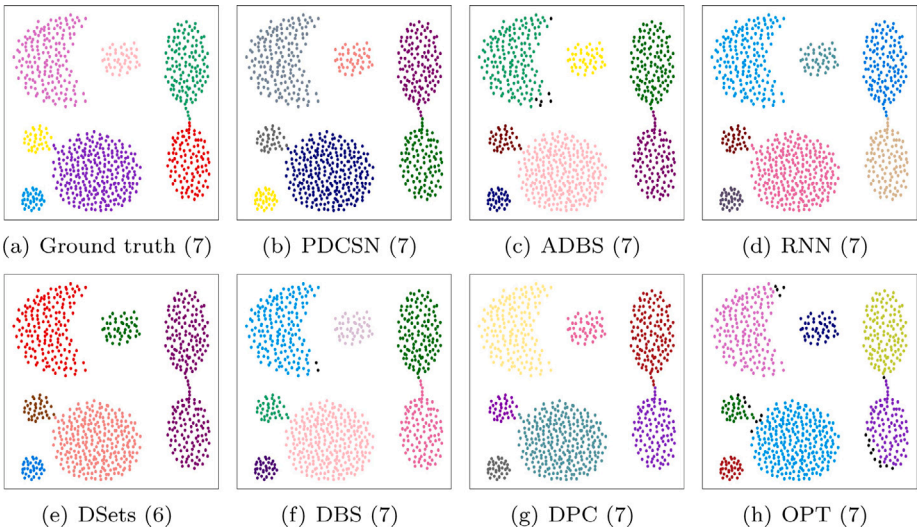


Fig. 9. Clustering results on Aggregation. In parentheses is the number of clusters.

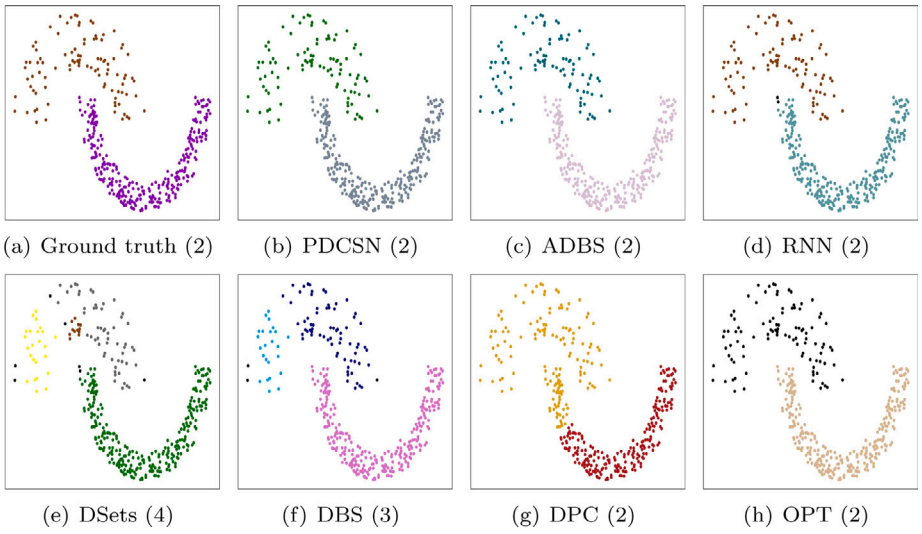


Fig. 10. Clustering results on Jain. In parentheses is the number of clusters.

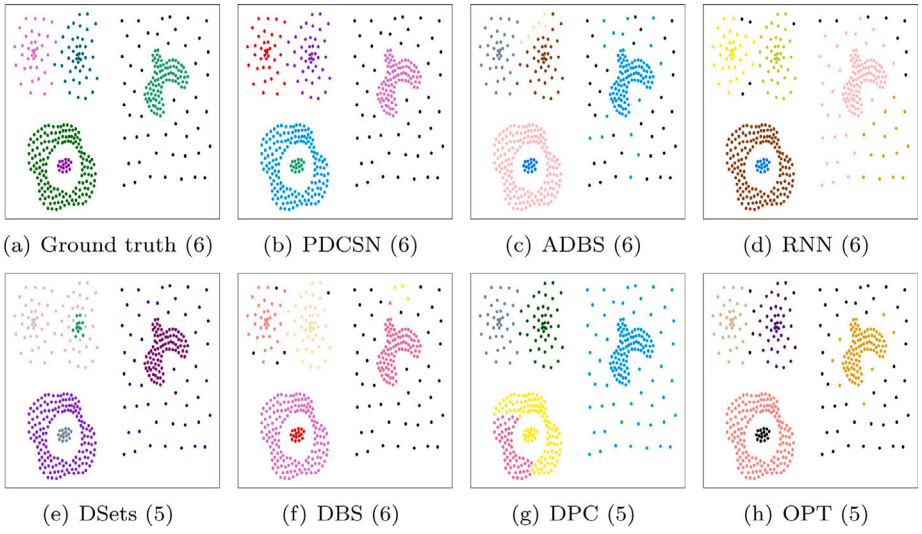


Fig. 11. Clustering results on Compound. In parentheses is the number of clusters.

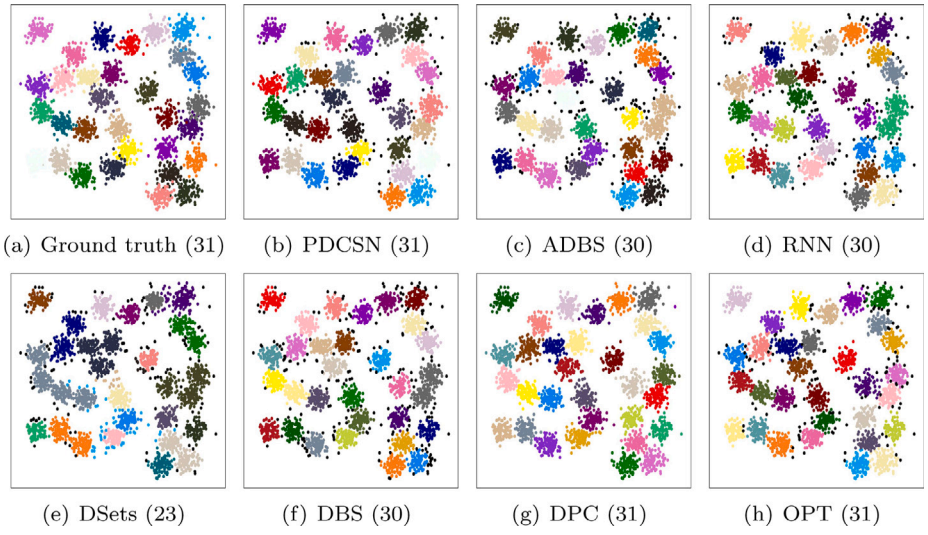


Fig. 12. Clustering results on D31. In parentheses is the number of clusters.

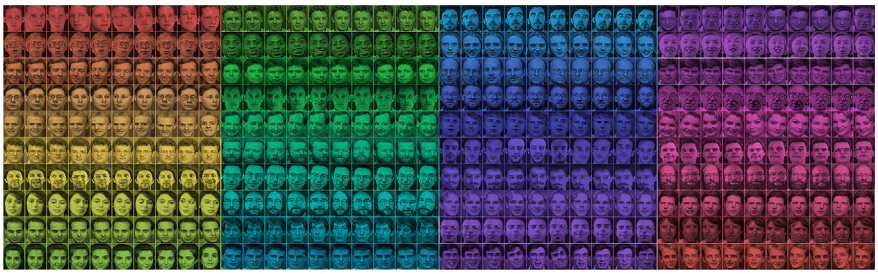


Fig. 13. The ORL face dataset.

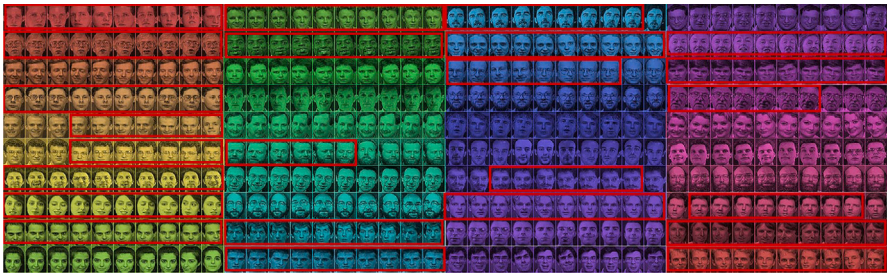


Fig. 14. Part of the initial clusters partitioned by PDCSN on the ORL face dataset.

Table 2
User-specified parameters description of algorithms.

Algorithm	Parameter description
PDCSN( )	none
ADBS($k/noisep$)	k: the number of nearest neighbors $noisep$: the proportion of noise
RNN(k)	k: the number of nearest neighbors
DSets($Minpts$)	$Minpts$: the least number of samples in the specified neighborhood
DPC(λ)	λ: the percentage of the average number of neighbors
DBS($Minpts/Eps$)	$Minpts$: the least number of samples in the Eps- neighborhood Eps: the neighborhood radius of each sample
OPT(k)	k: the number of nearest neighbors

cluster on the upper left as multiple. DPC divides samples from different clusters together. Compound contains multiple clusters with various densities, where there are two clusters with extremely unbalanced distributions on the right. In Fig. 11, only PDCSN recognizes the correct distribution of clusters. Neither ADBS, DBS, RNN, nor OPT can handle clusters with unbalanced distribution in density. The cluster on the upper left is incorrectly divided by DSets. The evaluation metrics of PDCSN in Table 5 confirm that it can handle clusters with different densities.

(5) D31 contains regular spherical clusters. Only PDCSN, PDC, and OPT correctly identify the structure of all clusters on D31, as shown in Fig. 12. ADBS, RNN, DSets, and DBS misclassify two adjacent clusters with similar densities on the middle right. Since DPC cannot identify noise samples, its ACC, F, NMI, and ARI scores are the highest on D31 from Table 5. The remaining algorithms can identify the varying number of noise samples. Although the four evaluation metrics scores of PDCSN are not the highest, they are the best among the other five algorithms.

In summary, the experimental results on synthetic datasets with different densities and shapes demonstrate the excellent performance of PDCSN on low-dimensional datasets. Specifically, the clustering results on D1 and D2 validate the ability of PDCSN to handle clusters in the noise space. The clustering results on Spril, Aggregation, Jain, and Compound demonstrate that PDCSN can handle clusters with highly heterogeneous densities and extremely unbalanced distributions. Furthermore, the ability of PDCSN to handle adjacent clusters with small variations in density is confirmed on D31.

Table 3
Characteristics of the synthetic datasets.

DataSet	n	d	c
D1	8000	2	8
D2	10000	2	9
Spril	312	2	3
Aggregation	788	2	7
Jain	373	2	2
Compound	399	2	6
D31	3100	2	31

Table 4
The parameter settings on the second category of synthetic datasets.

Algorithm	Parameter	Datasets				
		Spril	Aggregation	Jain	Compound	D31
ADBS	k	26	30	40	35	75
	$noisep$	0	0.15	0	0.1	0.45
RNN	k	2	13	17	8	45
DSets	$Minpts$	3	4	5	4	4
DPC	λ	2	1	2	2	2
DBS	Eps	2	2	2.5	1.5	1
	$Minpts$	2	14	3	3	40
OPT	k	10	25	10	12	45

4.2. Clustering on the real-world datasets

The 8 real-world datasets from UCI are iris, wine, seed, breast, banknote, digit, htru2, and ecoil. Note that wine, seed, breast, and htru2 have been standardized in advance in PDCSN. The high dimensionality

Table 5
Performances for algorithms on the second category of synthetic datasets.

DataSet		**PDCSN**	ADBS	RNN	DSets	DPC	DBS	OPT
Spril	ACC	**1.0**	**1.0**	**1.0**	0.462	**1.0**	**1.0**	0.776
	F	**1.0**	**1.0**	**1.0**	0.512	**1.0**	**1.0**	0.742
	NMI	**1.0**	**1.0**	**1.0**	0.662	**1.0**	**1.0**	0.687
	ARI	**1.0**	**1.0**	**1.0**	0.412	**1.0**	**1.0**	0.639
Aggregation	ACC	**0.998**	0.986	**0.998**	0.869	**0.998**	0.991	0.964
	F	0.997	0.983	**0.998**	0.908	**0.998**	0.988	0.967
	NMI	0.993	0.973	**0.996**	0.932	0.995	0.978	0.938
	ARI	0.996	0.978	**0.998**	0.880	0.997	0.984	0.958
Jain	ACC	**1.0**	**1.0**	0.995	0.885	0.861	0.928	**1.0**
	F	**1.0**	**1.0**	0.994	0.959	0.788	0.976	**1.0**
	NMI	**1.0**	**1.0**	0.973	0.777	0.505	0.865	**1.0**
	ARI	**1.0**	**1.0**	0.983	0.901	0.515	0.941	**1.0**
Compound	ACC	**0.997**	0.904	0.897	0.917	0.682	0.975	0.927
	F	**0.998**	0.944	0.914	0.959	0.693	0.980	0.945
	NMI	**0.992**	0.915	0.878	0.939	0.792	0.946	0.896
	ARI	**0.997**	0.926	0.884	0.945	0.592	0.974	0.926
D31	ACC	0.949	0.900	0.910	0.605	**0.967**	0.894	0.931
	F	0.915	0.850	0.870	0.604	**0.935**	0.830	0.882
	NMI	0.938	0.915	0.922	0.840	**0.957**	0.908	0.022
	ARI	0.912	0.845	0.867	0.587	**0.933**	0.824	0.878

Table 6
Characteristics of real-world datasets.

DataSet	n	d	c
iris	150	4	3
wine	178	13	3
seed	210	7	3
breast	699	10	2
banknote	1372	4	2
digit	1797	64	10
htru2	17898	8	2
ecoil	336	7	8

and authenticity of these datasets allows for a more comprehensive validation of the algorithm's performance. Table 6 lists n, d, and c on the 8 real-world datasets. The parameter settings of the algorithms on these datasets are shown in Table 7.

The ACC, F, NMI, ARI, and Clu (the number of clusters) of each algorithm on these datasets are shown in Table 8. From the perspective of the evaluation metrics: the NMI and ARI scores of PDCSN are the best on 8 real-world datasets. Except on htru2, PDCSN's ACC and F scores are the maxima on other datasets. For example, the mean of ACC, F, NMI, and ARI of PDCSN on wine is 14% higher than the second-ranked OPT. The mean of the four evaluation metrics of PDCSN on seed is 6.5% higher than the second-ranked ADBS. The mean of the four evaluation metrics of PDCSN on breast is 5.6% higher than the second-ranked DSets. For the 64-dimensional dataset digit, PDCSN still has the performance of rank one. In addition, it can be seen from the four evaluation metrics scores of 8 real-world datasets that PDCSN has a relatively stable performance compared with the other algorithms. The ARI scores of ADBS on the datasets breast and banknote differ by 80.3%, while the ARI scores of PDCSN differ by only 22.3%. The NMI scores of RNN on htru2 and ecoil differ by 35.6%, while the NMI scores of PDCSN differ by only 21.6%. From the perspective of the number of clusters: except for the datasets digit, ecoil, and htru2, PDCSN extracts the optimal number of clusters on the other 5 datasets. ADBS and RNN only determine the optimal number of clusters on 3 datasets. DBS and OPT only identify the optimal number of clusters on 2 datasets. DSets finds the wrong number of clusters on all datasets. The Clu of DPC is not convincing because the number of clusters is determined by the user.

In conclusion, satisfactory results on these datasets demonstrate that PDCSN can perform excellently on real-world datasets. Furthermore, the clustering stability of our algorithm is more reliable than other algorithms.

4.3. Clustering on the high-dimensional/large-scale datasets

The 6 high-dimensional/large-scale datasets are utilized further to verify the performance of PDCSN. The ORL face dataset contains 40 clusters, each with 10 images of faces that have different facial expressions. The ORL face dataset is shown in Fig. 13, where images of the same color correspond to a cluster. In this paper, the images are converted into a matrix that can be recognized by the machine, where each row vector represents a facial image. Since the image size is 92 × 112 pixels, the converted matrix size is 400 × 10304. The USPS digital recognition dataset contains 10 clusters, each composed of 16 × 16 pixel digital images. Letter and MNIST are real-world datasets, where Letter consists of 20,000 pictures of the alphabet, and MNIST is composed of 60,000 gray handwritten digital images with 28 × 28 pixels points. Birch2 and Worms are synthetic datasets, where Birch2 is a 2-dimensional dataset with a sinusoidal curve, and Worms is a 2-dimensional dataset with worm like shapes. Table 9 lists n, d, and c on these high-dimensional/large-scale datasets. The parameter settings of the algorithms on these datasets are displayed in Table 10. Note that the DPC and Dsets algorithms were not able to complete their operations on the Birch2 and Worms datasets within the given time limit of 3 h, and thus their results are labeled as N/A (not applicable). Overall, these datasets provide a diverse set of challenges for clustering algorithms, ranging from high-dimensional image datasets to large-scale synthetic datasets with irregular shapes.

The ACC, F, NMI, ARI, and Clu of each algorithm on high-dimensional/large-scale datasets are listed in Table 11. The average score of the four evaluation metrics of PDCSN is the highest among the seven algorithms on these datasets. For example, on the ORL, the ARI score of PDCSN is 72.9%, which is 26.4% higher than that of RNN and 26.5% higher than that of DBS. The F score of PDCSN is 49.6% higher than that of OPT. The ACC score of PDCSN ranks second, only 1.8% away from the highest value of 74.3%. the excellent performance of PDCSN on the ORL face dataset is mainly attributed to our proposed partitioning strategy. As shown in Fig. 14, the partially correct initial clusters partitioned by PDCSN (more than five samples in each initial cluster) are marked with red boxes, where there are 23 red boxes in total. PDCSN still has the best performance on the USPS dataset. Its NMI score is 53.7% higher than DPC and 17.5% higher than RNN. On the Letter dataset, only PDCSN correctly identifies half of the clusters, which is better than the other algorithms. On the MNIST dataset, PDCSN has a mean of 15.2% higher performance on the four metrics compared to ADBS and 17.4% higher compared to RNN. In addition, on datasets Birch2, Worms, Letter, and MNIST, PDCSN identifies the number of clusters closest to the true value. The optimal performance

Table 7
The parameter settings on real-world datasets.

Algorithm	Parameter	Datasets							
		iris	wine	seed	breast	banknote	digit	htru2	ecoil
ADBS	k	41	29	48	39	71	58	110	58
	$noisep$	0	0	0.45	0.05	0.05	0.35	0.2	0.45
RNN	k	5	4	5	19	37	3	3	3
DSets	$Minpts$	4	4	4	5	4	4	4	4
DPC	λ	2	2	2	2	2	2	2	2
DBS	Eps	1	2.5	1	3	2	21	1	0.163
	$Minpts$	10	10	14	25	16	4	70	9
OPT	k	25	20	21	45	50	26	10	40

Table 8
Performances for algorithms on the real-world datasets.

DataSet		**PDCSN**	ADBS	RNN	DSets	DPC	DBS	OPT
iris	ACC	**0.873**	0.680	0.747	0.660	0.833	0.667	0.673
	F	**0.807**	0.741	0.738	0.740	0.762	0.746	0.744
	NMI	**0.739**	0.717	0.674	0.720	0.717	0.734	0.723
	ARI	**0.714**	0.564	0.638	0.558	0.633	0.568	0.566
	Clu	3	2	4	2	3	2	2
wine	ACC	**0.921**	0.708	0.652	0.663	0.635	0.702	0.831
	F	**0.879**	0.652	0.669	0.641	0.686	0.638	0.746
	NMI	**0.805**	0.608	0.566	0.398	0.586	0.531	0.656
	ARI	**0.822**	0.537	0.492	0.388	0.454	0.424	0.628
	Clu	3	8	3	2	2	2	3
seed	ACC	**0.919**	0.895	0.800	0.605	0.886	0.638	0.795
	F	**0.867**	0.807	0.739	0.644	0.803	0.606	0.769
	NMI	**0.751**	0.662	0.617	0.456	0.698	0.546	0.687
	ARI	**0.804**	0.717	0.627	0.411	0.703	0.442	0.676
	Clu	3	3	4	2	3	3	3
breast	ACC	**0.889**	0.381	0.808	0.859	0.773	0.817	0.815
	F	**0.845**	0.349	0.706	0.791	0.726	0.715	0.716
	NMI	**0.541**	0.309	0.390	0.489	0.309	0.299	0.288
	ARI	**0.683**	0.184	0.402	0.592	0.275	0.400	0.395
	Clu	2	28	2	3	2	1	1
banknote	ACC	**0.972**	0.962	0.853	0.541	0.792	0.699	0.681
	F	**0.953**	0.942	0.875	0.559	0.803	0.686	0.539
	NMI	**0.834**	0.809	0.700	0.445	0.665	0.537	0.395
	ARI	**0.906**	0.887	0.771	0.345	0.660	0.511	0.271
	Clu	2	3	3	8	4	7	3
digit	ACC	**0.874**	0.850	0.769	0.110	0.590	0.788	0.644
	F	**0.844**	0.820	0.748	0.180	0.478	0.695	0.451
	NMI	**0.855**	0.837	0.807	0.014	0.736	0.797	0.691
	ARI	**0.828**	0.801	0.724	0.002	0.389	0.661	0.361
	Clu	12	13	35	3	9	17	7
htru2	ACC	0.898	0.919	0.828	0.727	**0.941**	0.930	0.907
	F	0.897	0.906	0.848	0.723	**0.928**	0.922	0.907
	NMI	**0.440**	0.358	0.194	0.180	0.376	0.296	0.004
	ARI	**0.547**	0.543	0.333	0.260	0.478	0.511	0.015
	Clu	3	2	204	8	2	2	1
ecoil	ACC	**0.750**	0.658	0.693	0.595	0.625	0.631	0.732
	F	**0.785**	0.674	0.676	0.612	0.605	0.673	0.756
	NMI	**0.656**	0.522	0.550	0.442	0.454	0.507	0.601
	ARI	**0.698**	0.506	0.526	0.437	0.376	0.520	0.655
	Clu	3	2	8	4	2	2	3

Table 9
Characteristics of the high-dimensional/large-scale datasets.

DataSet	n	d	c
ORL	400	10304	40
USPS	9298	256	10
Birch2	100000	2	100
Worms	105600	2	35
Letter	20000	16	26
MNIST	60000	768	10

Table 10
The parameter settings on high-dimensional/large-scale datasets.

Dataset	ADBS	RNN	DSets	DPC	DBS	OPT
	$k/noise\_p$	k	$minpts$	λ	$minpts/eps$	k
ORL	25/0.0	3	4	2	1/3270	4
USPS	119/0.5	2	4	2	10/3	10
Birch2	104/0.0	34	N/A	N/A	850/0.095	35
Worms	61/0.5	114	N/A	N/A	550/0.1	9
Letter	114/0.2	29	4	2	250/2.2	22
MNIST	500/0.4	10	4	2	15/12	20

Table 11
Performances for algorithms on the high-dimensional/large-scale datasets.

Dataset	Metrics	PDCSN	ADBS	RNN	DSets	DPC	DBS	OPT
ORL	ACC	0.725	**0.743**	0.678	0.15	0.622	0.628	0.613
	F	**0.735**	0.688	0.480	0.077	0.579	0.479	0.240
	NMI	**0.923**	0.909	0.837	0.366	0.868	0.853	0.789
	ARI	**0.729**	0.680	0.465	0.035	0.569	0.464	0.211
	Clu	65	45	42	7	77	61	44
USPS	ACC	**0.561**	0.515	0.330	0.239	0.165	0.343	0.167
	F	**0.523**	0.416	0.275	0.238	0.194	0.280	0.194
	NMI	**0.570**	0.568	0.395	0.165	0.033	0.386	0.009
	ARI	**0.443**	0.308	0.125	0.071	0.010	0.127	0.001
	Clu	5	16	208	4	91	5	1
Birch2	ACC	**0.971**	0.850	0.957	N/A	N/A	0.585	0.540
	F	**0.938**	0.846	0.875	N/A	N/A	0.438	0.139
	NMI	0.970	**0.974**	0.960	N/A	N/A	0.879	0.810
	ARI	**0.937**	0.844	0.873	N/A	N/A	0.429	0.123
	Clu	100	102	100	N/A	N/A	60	53
Worms	ACC	**0.369**	0.300	0.364	N/A	N/A	0.289	0.203
	F	**0.332**	0.150	0.269	N/A	N/A	0.105	0.101
	NMI	**0.570**	0.430	0.536	N/A	N/A	0.416	0.317
	ARI	**0.544**	0.298	0.226	N/A	N/A	0.046	0.272
	Clu	32	31	21	N/A	N/A	25	8
Letter	ACC	**0.535**	0.265	0.174	0.105	0.217	0.111	0.301
	F	**0.437**	0.113	0.083	0.078	0.121	0.098	0.091
	NMI	**0.702**	0.439	0.300	0.164	0.427	0.155	0.399
	ARI	**0.407**	0.047	0.011	0.004	0.154	0.029	0.021
	Clu	25	31	36	19	25	3	21
MNIST	ACC	**0.375**	0.160	0.114	0.114	0.312	0.194	0.112
	F	**0.303**	0.184	0.182	0.163	0.282	0.198	0.183
	NMI	**0.343**	0.044	0.104	0.002	0.306	0.173	0.001
	ARI	0.183	**0.210**	0.109	0.001	0.181	0.035	0.001
	Clu	5	25	5	3	19	4	1

of PDCSN on these datasets is mainly attributed to the proposed core sample search approach that breaks the shortcomings of the traditional distance metric by using specific sample distribution attributes instead of distance information to identify core samples in high-dimensional spaces. This approach avoids the failure of the distance metric in high-dimensional spaces to some extent. The experimental results on these six datasets show that PDCSN outperforms the other six algorithms in high-dimensional and large-scale datasets.

4.4. Correctness of the proposed self-adaptive neighborhood estimation

In order to avoid the effect of user-specified parameters, this paper proposes a self-adaptive neighborhood approach to estimate the critical range of the dense/sparse neighborhood of each sample x, $\mathrm{CN}(x)$, iteratively and select the maximum value of $\{\mathrm{CN}(x)|x \in X\}$ as the optimal neighborhood size λ (i.e., the optimal number of nearest neighbors). Two hyperparameters, 0.9 and 0.1, are given in the estimation process to determine numerical similarity.

To verify the rationality and effectiveness of our proposed self-adaptive neighborhood approach, we compare the performance of PDCSN with different λ on 4 datasets (e.g., wine, Compound, banknote, ORL). As shown in Fig. 15, the following trends can be found for 4 datasets: when λ is small or large, the samples cannot form clusters correctly. In addition, ARI and NMI are at high levels when clusters are correctly identified, and conversely, both ARI and NMI are at low levels. The above trends reveal that ARI and NMI positively correlate with suitable λ. In Fig. 15, the λ of the triangles mark is obtained by our proposed self-adaptive neighborhood approach. The λ values estimated by this approach are approximate optimal solutions on datasets wine, Computer, banknote, and ORL. Experimental results on several two-dimensional, high-dimensional, and ultra-high-dimensional datasets validate the correctness and robustness of our proposed approach.

4.5. Running efficiency

To verify the running efficiency of PDCSN, its running time is compared with that of the other six algorithms on all experimental datasets. For a fair comparison, each algorithm is executed in Python on a PC with $Intel\ Xeon\ W-2223\ CPU$ and 32 GB of RAM. Furthermore, PDCSN uses the kd-tree to speed up the computation of nearest neighbors. Table 12 lists the running time (unit: s) of the seven algorithms on all datasets, where the running time is the average of ten runs.

As shown in Table 12, PDCSN is the second fastest method on small-scale datasets ($n<$ 8000), which outperforms ADBS, DSets, DPC, OPT, and DBS in terms of running time. In addition, the difference between the running time of PDCSN and the optimal running time is negligible. For large-scale datasets ($n>$ 8000), since PDCSN integrates more neighbor information to describe the sample density, it undoubtedly adds considerable time overhead in computing the nearest neighbor and shared similarity problems. For example, on dataset D1, the estimated λ of PDCSN is 48, which is 11 larger than the optimal k of RNN and 6 higher than the optimal k of ADBS. On dataset hutr, the estimated λ of PDCSN is 397 larger than the optimal k of RNN and 290 higher than that of ADBS. Nevertheless, the running time of PDCSN is still better than that of DSets, DPC, DBS, and OPT on all experimental datasets.

To further investigate the scalability of PDCSN on high-dimensional and large-scale datasets, 12 different types of datasets were generated, i.e., experiments were performed on 12 synthetic datasets each of size from 100,000 to 600,000 and dimensions ranging from 2 to 60. To ensure variability between clusters, each dataset has three clusters with variances of 1.0, 2.0, and 3.0. This paper uses the size and dimension of datasets to represent each of them. For example, a two-dimensional dataset with 200,000 samples is represented as D20-2. In addition, this paper also compares the speedup effect of different nearest neighbor searches (kd-tree, R-tree) on the proposed algorithm. Figs. 16(a) and (b) show the running times of kd-tree-based PDCSN, R-tree-based PDCSN, DBS, and RNN on different size datasets (D10-2,

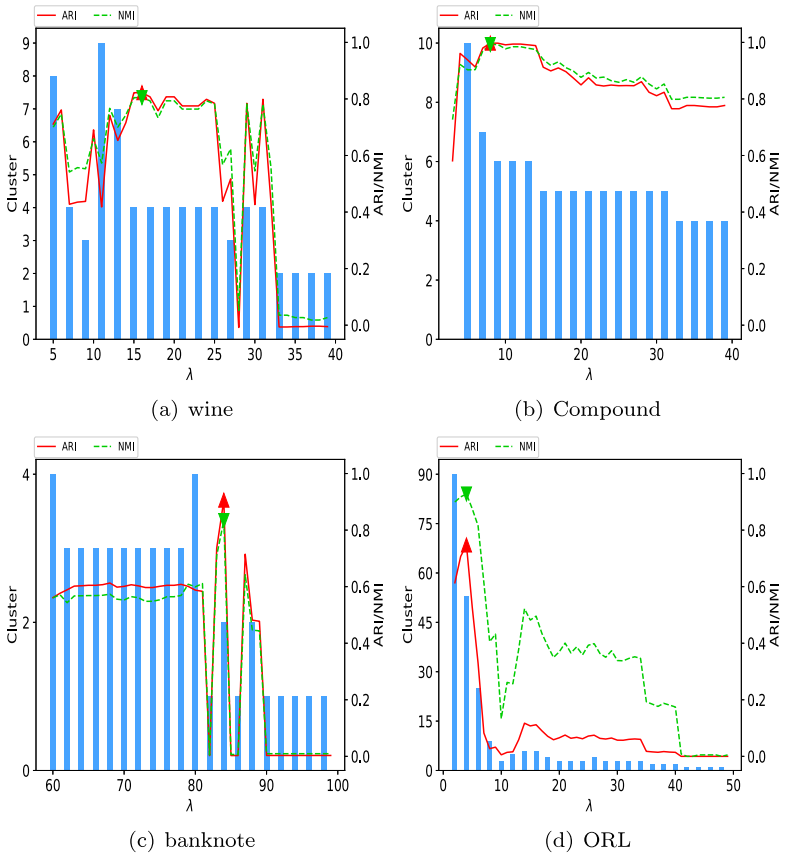


Fig. 15. ARI/NMI scores and the number of clusters of PDCSN with different λ on wine (a), Compound (b), banknote (c), and ORL (d).

Table 12
The running times of the seven algorithms on the experimental datasets (unit: s).

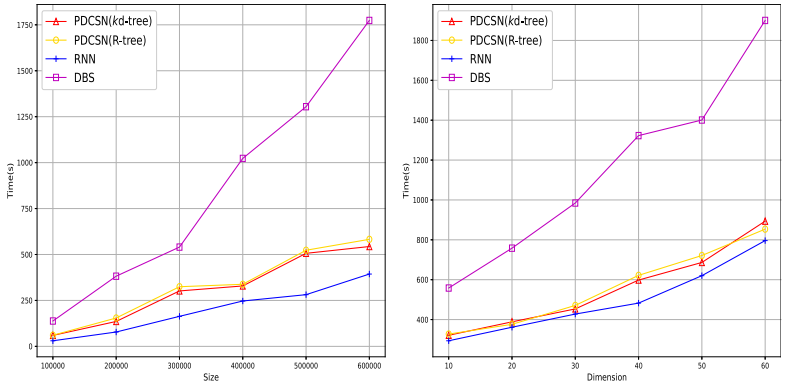
DataSet	PDCSN	ADBS	RNN	DSets	DPC	DBS	OPT
D1	5.183	3.065	2.580	806.209	48.882	10.464	11.636
D2	7.702	3.293	3.047	528.449	72.927	12.216	14.425
Spril	0.024	0.051	0.010	1.708	0.179	0.158	0.144
Aggregation	0.092	0.097	0.046	3.126	0.566	0.503	0.404
Jain	0.039	0.054	0.037	1.205	0.300	0.211	0.174
Compound	0.029	0.065	0.014	1.187	0.235	0.236	0.199
D31	0.503	0.711	0.630	47.805	16.075	15.539	12.063
iris	0.010	0.020	0.005	0.240	0.046	0.041	0.069
wine	0.015	0.016	0.006	0.261	0.092	0.074	0.088
seed	0.017	0.025	0.007	0.410	0.055	0.045	0.100
breast	0.091	0.096	0.086	2.029	0.222	0.157	0.365
banknote	0.466	0.285	0.180	1.952	1.292	0.902	0.841
digit	0.454	0.227	0.240	16.959	2.254	0.817	2.316
htru2	22.692	5.378	1.591	1712.951	220.239	64.163	79.865
ecoil	0.038	0.050	0.010	0.750	0.078	0.065	0.158
ORL	0.081	0.099	0.072	4.647	3.768	2.629	5.074
USPS	13.892	5.033	2.505	1206.429	110.119	90.617	100.779
Birch2	35.816	21.716	13.088	N/A	N/A	222.451	232.627
Worms	47.667	40.150	39.568	N/A	N/A	198.018	152.581
Letter	20.412	12.226	14.351	2540.139	724.763	98.452	102.365
MNIST	175.575	163.236	158.129	6742.387	2620.486	622.689	558.161

D20-2, . . . , D60-2) and different dimension datasets (D10-10, D10-20, . . . , D10-60), respectively. It is observed from Figs. 16(a) and (b) that the execution time of the algorithms is proportional to the size of the dataset. Moreover, compared with low-dimensional datasets, the algorithm needs more time to process high-dimensional datasets. On these datasets, RNN is found to be the fastest algorithm, while DBS is the slowest. Although PDCSN is not the fastest, its time overhead is within an acceptable range compared to RNN. Additionally, since kd-tree-based PDCSN is slightly better than R-tree-based PDCSN, all experiments in this paper introduce kd-tree to accelerate the nearest neighbor search.

To explore the specific factors affecting the time overhead of PDCSN, the running times of each phase and nearest neighbor search on datasets D10-2, D20-2, D30-2, D10-10, D10-20, and D10-30 are shown in Table 13. From Table 13, as the amount of data increases, the real impact on time overhead is for phases 1 and 3. For example, on dataset D30-2, although the running time of the nearest neighbor search is 30 s, the running times of phase 1 and phase 3 significantly increase the overall time overhead. For phase 1, the large number of computations increases its time overhead because it requires multiple traversals of the dataset to calculate the optimal neighborhood size λ. For phase 3, it needs to calculate the intersection of a large number of nearest

Table 13
The running times of PDCSN in each phase on the experimental datasets (unit: s).

DataSet	Nearest neighbor search		Phase 1	Phase 2	Phase 3	Phase 4
	k-d tree	R-tree				
D10–2	7.320	8.124	25.267	6.093	19.034	1.887
D20–2	16.437	35.939	54.851	17.084	41.347	5.906
D30–2	30.282	53.393	105.995	32.055	127.449	5.915
D10–10	186.798	193.587	49.846	22.622	56.763	5.151
D10–20	233.233	221.957	63.344	18.920	59.536	13.544
D10–30	280.340	304.572	68.190	23.903	67.580	13.412



(a) Different size datasets (b) Different dimensional dataset

Fig. 16. Compare the running times of *k*d-tree-based PDCSN, R-tree-based PDCSN, DBS, and RNN on different size datasets (a) and different dimensional datasets (b), respectively.

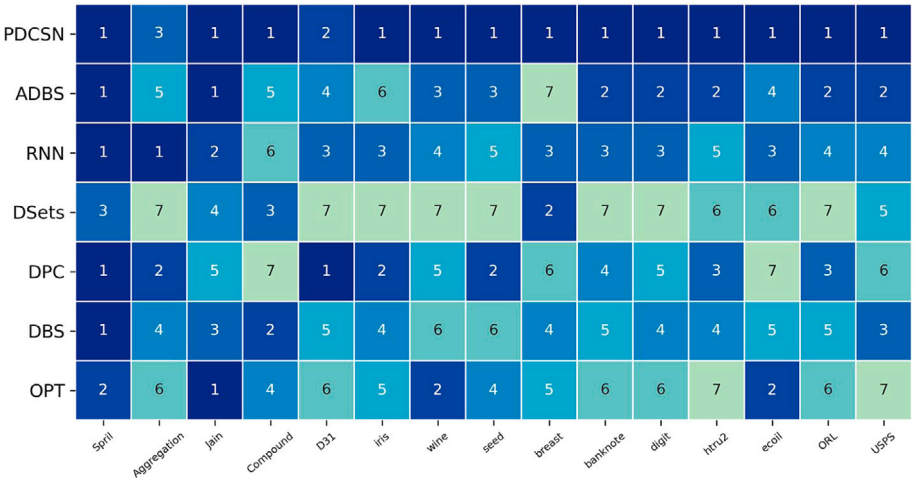


Fig. 17. The ranking matrix of the average of ACC, NMI, F, and ARI, where the numbers on the blocks indicate the ranking of the algorithm performance.

neighbor lists, where the speed of finding the intersection is positively related to the λ. Furthermore, with the increase of data dimension, it is the nearest neighbor search that affects the time overhead. On dataset D10-30, the searching speedups of *k*d-tree and R-tree are 280 s and 300 s, respectively, accounting for 61.8% and 63.7% of the total running time, respectively. This indicates that the nearest neighbor search becomes more time-consuming as the data dimension increases.

By comparing the running efficiency of PDCSN with the other six algorithms, it can be concluded that the running efficiency of PDCSN is acceptable and feasible because PDCSN fills the gap in user-input-free, i.e., it does not require multiple attempts to determine the optimal performance.

4.6. Discussion

Parameter dependence: ADBS, RNN, and OPT take the number of nearest neighbors as user input, while DBS requires the user to input $Minpts$ and Eps. Besides, DPC requires the user to set the number of clusters. The lack of visual reference in high-dimensional datasets makes finding the optimal parameters difficult. Compared with these algorithms, PDCSN proposes a self-adaptive neighborhood approach to estimate the optimal neighborhood size (i.e., the optimal number of nearest neighbors) without user input as a prior knowledge. The effectiveness of this approach is also confirmed in Section 4.4. Thus, PDCSN can discover clusters without user input.

Algorithm accuracy: Fig. 17 shows the ranking of the average of evaluation metrics on the 16 datasets. Fig. 18 depicts the difference between the number of obtained and actual clusters. It is observed from Figs. 17 and 18 that the first-ranked PDCSN has the optimal metrics performance on 13 datasets. PDCSN is the first-best algorithm in finding the optimal number of clusters on 12 datasets. In addition, the clustering performances of PDCSN on datasets Compound, Jain, and D31 verify its ability to handle adjacent clusters with arbitrary density. Thus, the number of clusters and the cluster evaluation metrics

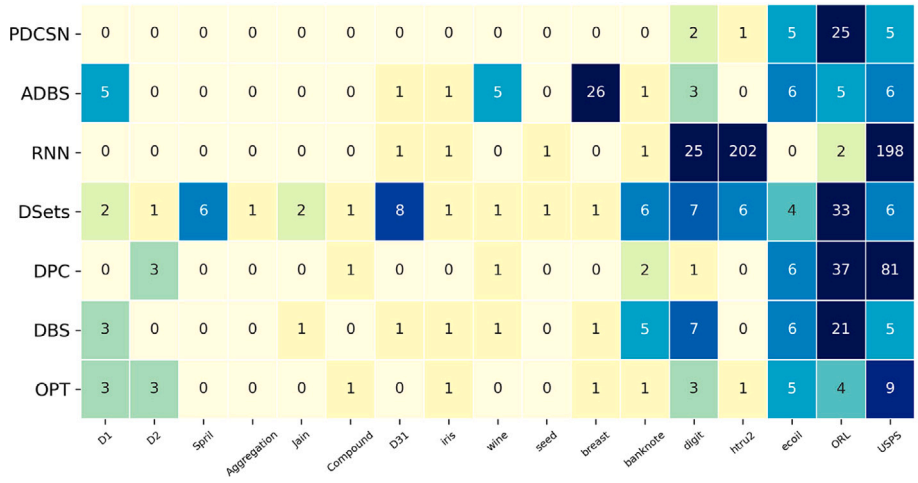


Fig. 18. The matrix of the error in the number of clusters, where the number on the block indicates the L1 distance between the actual number and the obtained number.

demonstrate that PDCSN has better accuracy and robustness compared with other algorithms.

Data dimensionality: as shown in Tables 5, 8, and 11, PDCSN achieves excellent performance on 2-dimensional synthetic datasets while still carrying over to real-world datasets. In addition, the clustering performance of PDCSN still does not lose out to other algorithms on high dimensional/large scale datasets. The data distribution is usually sparse for high-dimensional datasets, so simple distance metrics are vulnerable to dimensional catastrophes. PDCSN ensures robustness of clustering on high-dimensional data by integrating the neighbor information and taking advantage of the sparsity of the sample density distribution. Therefore, the results on various dimensional datasets show that PDCSN breaks the limitation of dimensional catastrophe to reach the optimal solution.

Running efficiency: Table 12 shows that the running time of PDCSN is much faster than DSets, DPC, DBS, and OPT on all datasets. Although the running efficiency of PDCSN is not optimal, the running time on the 10 datasets is comparable to the optimal value. However, these datasets are small-scale datasets. On large-scale datasets, as shown in Fig. 16, the running time of PDCSN differs significantly from that of the fastest RNN. Table 13 explains the main reason for this discrepancy, i.e., more nearest neighbors and shared similarities need to be computed as the size of the dataset increases. Moreover, the difficulty of setting the algorithm parameters results in the need for multiple executions to find the best performance of the algorithm. Since PDCSN does not require multiple attempts to find the optimal solution, it runs within an acceptable efficiency range. However, The running time on large datasets is still a limitation of PDCSN, which is a direction for our future work.

5. Conclusions

This paper proposes a partition density clustering with self-adaptive neighborhoods (PDCSN) to discover clusters with arbitrary shapes and densities in noise space without user-specified parameters. Compared with other density-based algorithms, PDCSN has the following advantages. First, the optimal neighborhood size can be estimated automatically, thus breaking the dependency of user-specified parameters. Secondly, the partition of the initial clusters can distinguish clusters with large varying densities. Finally, the proposed core sample search approach can distinguish adjacent clusters with similar densities, which other variants cannot overcome. PDCSN is evaluated on 9 synthetic, 10 real-world, and 2 image datasets. The experimental results on these datasets show that our algorithm can produce better clustering performance. However, our algorithm increases the time overhead in computing nearest neighbors and shared similarities. Therefore, in our future work, we will consider reducing the time overhead while maintaining the clustering performance.

CRediT authorship contribution statement

Shuai Xing: Conceptualization, Methodology, Software, Writing – original draft. **Qian-Min Su:** Supervision, Final review. **Yu-Jie Xiong:** Supervision, Writing – review & editing, Funding acquisition. **Chun-Ming Xia:** Supervision, Funding acquisition.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgments

This work is jointly sponsored by the Science and Technology Commission of Shanghai Municipality, China (21DZ2203100), National Natural Science Foundation of China (62006150), Shanghai Young Science and Technology Talents Sailing Program (19YF1418400).

References

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on management of data* (pp. 49–60). Association for Computing Machinery, http://dx.doi.org/10.1145/304182.304187.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*(9), 509–517. http://dx.doi.org/10.1145/361002.361007.

Beygelzimer, A., Kakade, S., & Langford, J. (2006). Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on machine learning* (pp. 97–104). Association for Computing Machinery, http://dx.doi.org/10.1145/1143844.1143857.

Boonchoo, T., Ao, X., Liu, Y., Zhao, W., Zhuang, F., & He, Q. (2019). Grid-based DBSCAN: Indexing and inference. *Pattern Recognition*, *90*, 271–284. http://dx.doi.org/10.1016/j.patcog.2019.01.034.

Bryant, A., & Cios, K. (2018). RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates. *IEEE Transactions on Knowledge and Data Engineering*, *30*(6), 1109–1121. http://dx.doi.org/10.1109/TKDE.2017.2787640.

Cassisi, C., Ferro, A., Giugno, R., Pigola, G., & Pulvirenti, A. (2013). Enhancing density-based clustering: Parameter reduction and outlier detection. *Information Systems*, *38*(3), 317–330. http://dx.doi.org/10.1016/j.is.2012.09.001.

Chen, M., Li, L., Wang, B., Cheng, J., Pan, L., & Chen, X. (2016). Effectively clustering by finding density backbone based-on kNN. *Pattern Recognition*, *60*, 486–498. http://dx.doi.org/10.1016/j.patcog.2016.04.018.

Chen, X., Liu, W., Qiu, H., & Lai, J. (2011). APSCAN: A parameter free algorithm for clustering. *Pattern Recognition Letters*, *32*(7), 973–986. http://dx.doi.org/10.1016/j.patrec.2011.02.001.

Chen, Y., Zhou, L., Bouguila, N., Wang, C., Chen, Y., & Du, J. (2021). BLOCK-DBSCAN: Fast clustering for large scale data. *Pattern Recognition*, *109*, Article 107624. http://dx.doi.org/10.1016/j.patcog.2020.107624.

Cheng, D., Zhu, Q., Huang, J., Wu, Q., & Yang, L. (2019). A novel cluster validity index based on local cores. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(4), 985–999. http://dx.doi.org/10.1109/TNNLS.2018.2853710.

Chowdhury, H. A., Bhattacharyya, D. K., & Kalita, J. K. (2021). UIFDBC: Effective density based clustering to find clusters of arbitrary shapes without user input. *Expert Systems with Applications*, *186*, Article 115746. http://dx.doi.org/10.1016/j.eswa.2021.115746.

Dua, D., & Graff, C. (2017). UCI machine learning repository. URL http://archive.ics.uci.edu/ml.

Ertöz, L., Steinbach, M. S., & Kumar, V. (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 2003 SIAM international conference on data mining* (pp. 47–58). http://dx.doi.org/10.1137/1.9781611972733.5.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the second international conference on knowledge discovery and data mining* (pp. 226–231).

Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., et al. (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, *2*(3), 267–279. http://dx.doi.org/10.1109/TETC.2014.2330519.

Fränti, P., & Sieranoja, S. (2018). K-means properties on six clustering benchmark datasets. (pp. 4743–4759). URL http://cs.uef.fi/sipu/datasets/.

Guha, S., Rastogi, R., & Shim, K. (1998). CURE: An efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on management of data* (pp. 73–84). Association for Computing Machinery, http://dx.doi.org/10.1145/276304.276312.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *SIGMOD Record*, *14*(2), 47–57. http://dx.doi.org/10.1145/971697.602266.

Hanafi, N., & Saadatfar, H. (2022). A fast DBSCAN algorithm for big data based on efficient density calculation. *Expert Systems with Applications*, *203*, Article 117501.

Hou, J., Gao, H., & Li, X. (2016). Dsets-DBSCAN: A parameter-free clustering algorithm. *IEEE Transactions on Image Processing*, *25*(7), 3182–3193. http://dx.doi.org/10.1109/TIP.2016.2559803.

Hu, L., Liu, H., Zhang, J., & Liu, A. (2021). KR-DBSCAN: A density-based clustering algorithm based on reverse nearest neighbor and influence space. *Expert Systems with Applications*, *186*, Article 115763. http://dx.doi.org/10.1016/j.eswa.2021.115763.

Jiang, H., Li, J., Yi, S., Wang, X., & Hu, X. (2011). A new hybrid method based on partitioning-based DBSCAN and ant clustering. *Expert Systems with Applications*, *38*(8), 9373–9381. http://dx.doi.org/10.1016/j.eswa.2011.01.135.

Karypis, G., Han, E.-H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, *32*(8), 68–75. http://dx.doi.org/10.1109/2.781637.

Li, H., Liu, X., Li, T., & Gan, R. (2020). A novel density-based clustering algorithm using nearest neighbor graph. *Pattern Recognition*, *102*, Article 107206. http://dx.doi.org/10.1016/j.patcog.2020.107206.

Liu, R., Wang, H., & Yu, X. (2018). Shared-nearest-neighbor-based clustering by fast search and find of density peaks. *Information Sciences*, *450*, 200–226. http://dx.doi.org/10.1016/j.ins.2018.03.031.

Lv, Y., Ma, T., Tang, M., Cao, J., Tian, Y., Al-Dhelaan, A., et al. (2016). An efficient and scalable density-based clustering algorithm for datasets with complex structures. *Neurocomputing*, *171*, 9–22. http://dx.doi.org/10.1016/j.neucom.2015.05.109.

Marques, J. C., & Orger, M. B. (2018). Clusterdv: A simple density-based clustering method that is robust, general and automatic. *Bioinformatics*, *35*(12), 2125–2132.

Rehman, A. U., & Belhaouari, S. B. (2022). Divide well to merge better: A novel clustering algorithm. *Pattern Recognition*, *122*, Article 108305. http://dx.doi.org/10.1016/j.patcog.2021.108305.

Rodriguez, A., & Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, *344*(6191), 1492–1496. http://dx.doi.org/10.1126/science.1242072.

Ros, F., Guillaume, S., Riad, R., & El Hajji, M. (2022). Detection of natural clusters via S-DBSCAN a self-tuning version of DBSCAN. *Knowledge-Based Systems*, *241*, Article 108288. http://dx.doi.org/10.1016/j.knosys.2022.108288.

Samaria, F., & Harter, A. (1994). Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE workshop on applications of computer vision* (pp. 138–142). http://dx.doi.org/10.1109/ACV.1994.341300.

Sarma, A., Goyal, P., Kumari, S., Wani, A., Challa, J., Islam, S., et al. (2019). μDBSCAN: An exact scalable DBSCAN algorithm for big data exploiting spatial locality. In *2019 IEEE international conference on cluster computing* (pp. 1–11). Los Alamitos, CA, USA: IEEE Computer Society, http://dx.doi.org/10.1109/CLUSTER.2019.8891020.

Song, K., Yao, X., Nie, F., Li, X., & Xu, M. (2021). Weighted bilateral K-means algorithm for fast co-clustering and fast spectral clustering. *Pattern Recognition*, *109*, Article 107560. http://dx.doi.org/10.1016/j.patcog.2020.107560.

Thanh, N. D., Ali, M., & Son, L. H. (2017). A novel clustering algorithm in a neutrosophic recommender system for medical diagnosis. *Cognitive Computation*, *9*(4), 526–544. http://dx.doi.org/10.1007/s12559-017-9462-8.

Tian, F., Gao, Y., Fang, Z., Fang, Y., Gu, J., Fujita, H., et al. (2022). Depth estimation using a self-supervised network based on cross-layer feature fusion and the quadtree constraint. *IEEE Transactions on Circuits and Systems for Video Technology*, *32*(4), 1751–1766. http://dx.doi.org/10.1109/TCSVT.2021.3080928.

Wang, J., Wang, N., Jia, Y., Li, J., Zeng, G., Zha, H., et al. (2014). Trinary-projection trees for approximate nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(2), 388–403. http://dx.doi.org/10.1109/TPAMI.2013.125.

Wang, J., Zhu, C., Zhou, Y., Zhu, X., Wang, Y., & Zhang, W. (2018). From partition-based clustering to density-based clustering: Fast find clusters with diverse shapes and densities in spatial databases. *IEEE Access*, *6*, 1718–1729. http://dx.doi.org/10.1109/ACCESS.2017.2780109.

Weng, S., Gou, J., & Fan, Z. (2021). *h*-DBSCAN: A simple fast DBSCAN algorithm for big data. In *Asian conference on machine learning* (pp. 81–96). PMLR.

Wolf, L., Hassner, T., & Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011* (pp. 529–534). http://dx.doi.org/10.1109/CVPR.2011.5995566.

Xiaoyun, C., Yufang, M., Yan, Z., & Ping, W. (2008). GMDBSCAN: Multi-density DBSCAN cluster based on grid. In *2008 IEEE international conference on E-business engineering* (pp. 780–783). http://dx.doi.org/10.1109/ICEBE.2008.54.

Xu, T., & Jiang, J. (2022). A graph adaptive density peaks clustering algorithm for automatic centroid selection and effective aggregation. *Expert Systems with Applications*, *195*, Article 116539. http://dx.doi.org/10.1016/j.eswa.2022.116539.

Ye, X., & Ho, J. W. (2019). Ultrafast clustering of single-cell flow cytometry data using FlowGrid. *BMC Systems Biology*, *13*(2), 35. http://dx.doi.org/10.1186/s12918-019-0690-2.

Zhang, R., Du, T., Qu, S., & Sun, H. (2021). Adaptive density-based clustering algorithm with shared KNN conflict game. *Information Sciences*, *565*, 344–369. http://dx.doi.org/10.1016/j.ins.2021.02.017.

Zhang, R., Miao, Z., Tian, Y., & Wang, H. (2022). A novel density peaks clustering algorithm based on Hopkins statistic. *Expert Systems with Applications*, *201*, Article 116892. http://dx.doi.org/10.1016/j.eswa.2022.116892.

Zhou, Z., Si, G., Sun, H., Qu, K., & Hou, W. (2022). A robust clustering algorithm based on the identification of core points and KNN kernel density estimation. *Expert Systems with Applications*, *195*, Article 116573. http://dx.doi.org/10.1016/j.eswa.2022.116573.

Zhou, J., Sun, J., Cong, P., Liu, Z., Zhou, X., Wei, T., et al. (2020). Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT. *IEEE Transactions on Services Computing*, *13*(4), 745–758. http://dx.doi.org/10.1109/TSC.2019.2963301.

Zhu, J., Zeng, H., Huang, J., Liao, S., Lei, Z., Cai, C., et al. (2020). Vehicle re-identification using quadruple directional deep learning features. *IEEE Transactions on Intelligent Transportation Systems*, *21*(1), 410–420. http://dx.doi.org/10.1109/TITS.2019.2901312.