BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 04/06 pp183–194
Volume 4, Number 3, September 2021
DOI: 10.26599/BDMA.2021.9020001

Effective Density-Based Clustering Algorithms for Incomplete Data

Zhonghao Xue and Hongzhi Wang*

Abstract: Density-based clustering is an important category among clustering algorithms. In real applications, many datasets suffer from incompleteness. Traditional imputation technologies or other techniques for handling missing values are not suitable for density-based clustering and decrease clustering result quality. To avoid these problems, we develop a novel density-based clustering approach for incomplete data based on Bayesian theory, which conducts imputation and clustering concurrently and makes use of intermediate clustering results. To avoid the impact of low-density areas inside non-convex clusters, we introduce a local imputation clustering algorithm, which aims to impute points to high-density local areas. The performances of the proposed algorithms are evaluated using ten synthetic datasets and five real-world datasets with induced missing values. The experimental results show the effectiveness of the proposed algorithms.

Key words: density-based clustering; incomplete data; clustering algorihtm

1 Introduction

Clustering aims to find a set of groups of similar objects within a dataset, while keeping dissimilar objects separated in different groups or the group of noise points. It is an basic area of data mining and has been vastly applied in many fields. Density-based clustering is an important category among clustering algorithms[1], which defines clusters as areas of higher density than the remainder of the dataset. Objects in these sparse areas, which are required to separate clusters, are usually considered to be noise and border points. Density-based clustering methods do not require the number of clusters as the input, and clusters do not necessarily have a convex shape, but can be arbitrarily shaped in the dataset. The most popular density-based clustering method is Density-Based Spatial Clustering of Applications with Noise (DBSCAN)[2].

In real applications, many datasets suffer from incompleteness, i.e., a dataset can contain vectors that are missing one or more of the attribute values, as a result of failure in data collection, measurement errors, missing observations, or random noise. Density-based clustering algorithms always need to compute Euclidean distance or some other forms of distance, which requires all attributions to be completed. Therefore, they are not directly applicable to such incomplete datasets.

There are mainly two strategies to handle missing values. One is imputation, and the other is to alter the algorithm to conduct clustering directly on the data with missing values.

Imputation techniques aim to replace missing values by some estimated values derived from complete data[3]. After imputation, with all data completed, we still apply the original clustering algorithm for completed data. Even though such approaches are suitable for clustering approaches, such as k-means and Expectation Maximization algorithm (EM), they could hardly be applied on density-based clustering algorithms, since k-means and EM consider relative distances of a point to clusters, while density-based clustering algorithms consider absolute distances[2,4,5]. We use the following example to illustrate this point.

As shown in Fig. 1, a dataset contains three clusters: orange, blue, and purple. Each data point has two

• Zhonghao Xue is with USC Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90007, USA. Email: zhonghao.xue@gmail.com.
• Hongzhi Wang is with the Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China. E-mail: wangzh@hit.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2020-12-13; accepted: 2021-01-13

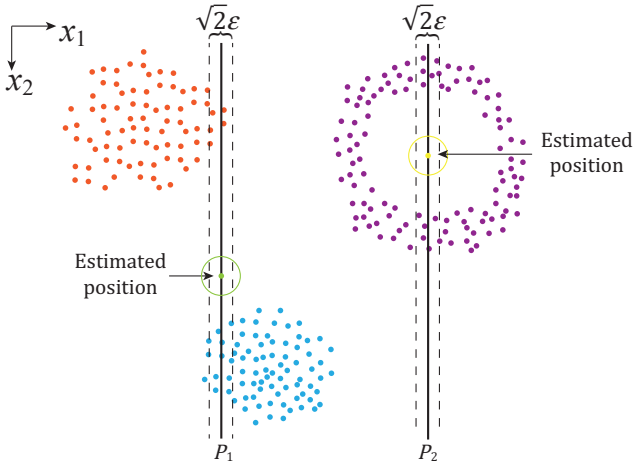


Fig. 1 An example for incomplete data clustering.

dimensions: x_1 and x_2. ε denotes the radius of the neighborhood in the density-based clustering algorithms. P_1 and P_2 indicate two incomplete data points, whose all possible positions consist straight lines, as shown in Fig. 1. In other words, the real position of the incomplete datum indicated by P_1 could be any position in the straight line. And the target of imputation is to select a position in the line to replace the incomplete datum.

For some imputation approaches, such as k-nearest-neighbor imputation, a popular imputation approach which imputes each missing attribute by the mean value of its k-nearest-neighbors[6], P_1 will be estimated somewhere near the green point. This is a reasonable imputation because the majority of its neighbors are blue points, which means that P_1 is most likely to belong to blue cluster. Obviously, k-means and EM will tend to assign it to the blue cluster, because it is relatively closer to the blue cluster.

However, for density-based clustering algorithms, such as DBSCAN, there are not any other points inside its neighborhood, which means that the green point will be treated as a noise point. This result is equivalent to drop P_1 directly, so the imputation of P_1 is meaningless.

Some imputation methods use iterations to optimize the quality of estimated missing values[7]. This eases the problem of insufficient information for clusters. However, these methods are time-costly, since they run imputation and clustering in each iteration, and some of them are sensitive to initial points.

Another way to handle missing values is to modify the clustering algorithm to directly handle incomplete data without imputation. As far as we know, only Partial Distance Strategy (PDS) adopts such strategy by extending the definition of distance. For data with missing values, it computes partial distance by using existing values, and then rescales it with the number of missing values. However, PDS cannot solve the problem in imputation, but even worse, it will lead to much more serious problems to density-based algorithms. We use an example to illustrate its drawbacks.

As shown in Fig. 1, the two dotted lines on each side of P_1 indicate the neighborhood of the incomplete data P_1. When we implement DBSCAN, because of enough points inside its neighborhood, P_1 will be considered as a core point. Since all points within a core point's neighborhood will be assigned to the same cluster, P_1 will connect the orange (upper left) and blue (lower) clusters together, leading to a serious decrease of clustering result quality.

Motivated by these problems, alternatively, we choose to conduct imputation and clustering concurrently by locating incomplete points to high-density area. It is inspired by Bayesian theory[8]. The theory is that a point is located in a density area in a high probability when its real location could be hardly determined. With this idea, by using intermediate clustering results, we impute proper missing values to prevent the missing values to be a noisy point as described before. As a result, the quality of clusters is increased. To support changing clustering results, we use union-find sets to represent clusters to improve efficiency of the proposed clustering algorithm.

In the proposed algorithm, we avoid low-density areas between clusters with intermediate clustering results. However, low-density areas inside clusters could not be avoided for non-convex clusters. As shown in Fig. 1, the purple cluster is annular shaped, and no data are in its center. However, the proposed algorithm will estimate P_2 to somewhere near the yellow point. For the same reason, among the green point, the yellow point will be considered as a noise point, making the imputation meaningless.

Motivated by this observation, we propose Concurrently Imputation clustering algorithm (CI-clustering) and Local Imputation clustering algorithm (LI-clustering). Such approaches could successfully exclude low-density areas inside clusters by locating incomplete points to high-density local areas. For example, in Fig. 1, there are two locally density areas on the straight line with respect to P_2, which are the top and bottom areas of the purple (band shaped) cluster. If the proposed algorithm could impute P_2 into one of these areas, P_2 will be correctly assigned to the purple cluster, rather than be treated as the noise data.

The contributions of this paper are summarized as follows.

• We propose density-based clustering algorithms for incomplete data. The proposed algorithms conduct imputation and clustering concurrently to ensure high accuracy. To the best of our knowledge, this is the first density-based clustering algorithms for incomplete data.

• To minimize the negative impact of incomplete data, we estimate the location of an incomplete point according to the idea of Bayesian theory, such that an incomplete point is located to the relative density area. With such strategy, the quality of clustering is increased. For various excepted cluster shapes, we propose CI- and LI-clustering algorithms.

• Experimental results show that both algorithms outperform traditional clustering algorithms with imputation and each of them has its own suitable scenarios. CI-clustering performs better on high-dimensional data, while it will cost more time and be affected by non-convex clusters. LI-clustering is efficient and robust to the shape of clusters, but its performance on high-dimensional data is not as good as CI-clustering.

This paper is organized as follows. In Section 2, we give some basic notations and briefly introduce algorithm DBSCAN. In Section 3, we discuss the motivation, method, and some implementation details about CI-clustering. In Section 4, to avoid some shortages of CI-clustering, we develop LI-clustering algorithm. We discuss the experimental results in Section 5 and draw conclusions in Section 6.

2 Preliminary

2.1 Notations

We denote a set of K-dimensional data points by $D_{origin} = \{p_i | i = 1, 2, \ldots, n\}$, where n is the number of points. To make the attributes on a similar scale, they were meanly normalized into approximately a range of $[-1, 1]$ by

$$p_{ik} \leftarrow \frac{p_{ik} - \mu_k}{s_k}, k = 1, 2, \ldots, K, i = 1, 2, \ldots, n,$$

where $\mu_k = (\sum_{i=1}^{n} p_k)/n$, and $s_k = \max_i\{p_{ik}\} - \min_i\{p_{ik}\}$. We denote mean normalized set by $D = \{p_i | i = 1, 2, \ldots, n\}$, and D_0 denotes complete points. The distance of two points is defined to be $dist(p_i, p_j) = (\sum_{k=1}^{K} (p_{ik} - p_{jk})^2)^{\frac{1}{2}}$. If p_j is incomplete, let $dist(p_i, p_j) = partial\_dist(p_i, p_j)$, which is the distance between points with incomplete dimensions, and $partial\_dist\ (p_i, p_j) = (\sum_{\{k|k \notin missAttr\}} (p_{ik} - p_{jk})^2)^{\frac{1}{2}}$, which can be also understood as considering $p_{jk} = p_{ik}$, if p_{jk} is missing. The reason for defining *partial_dist*( ) in this way will be discussed in Section 4.3.

2.2 Definitions in DBSCAN

In this section, we introduce some fundamental definitions of DBSCAN algorithm. The Eps-neighborhood of a point p, denoted by $N_{Eps}(p)$, is the set of points with the distance to p smaller or equal to a threshold *Eps* and is defined as $N_{Eps}(p) = \{q \in D | dist(p, q) \leqslant Eps\}$. A point p is called a core point if $|N_{Eps}(p)| \geqslant MinPts$, where *MinPts* is a threshold for core point definition. A point p is directly density-reachable from a point q with respect to *Eps* and *MinPts*, if (1) $p \in N_{Eps}(q)$ and (2) $|N_{Eps}(q)| \geqslant MinPts$ (core point condition). A point p is density-reachable from a point q with respect to *Eps* and *MinPts*, if there is a chain of points $p_1, \ldots, p_n, p_1 = q, p_n = p$, such that p_{i+1} is directly density-reachable from p_i. p and q are density-connected if there is a point o from which both p and q are density-reachable with respect to *Eps* and *MinPts*. A cluster C with respect to *Eps* and *MinPts* is a non-empty subset of D satisfying the following conditions: (1) $\forall p, q$, if $p \in C$ and q is density-reachable from p with respect to *Eps* and *MinPts*, then $q \in C$ (maximality); (2)$\forall p, q \in C$, p is density-connected to q with respect to *Eps* and *MinPts* (connectivity)[2].

2.3 DBSCAN algorithm

In this section, we introduce the algorithm DBSCAN designed to discover the cluster defined before. Firstly, DBSCAN finds the point set with each point p satisfying $N_{Eps}(p) \geqslant MinPts$, and saves their directly density-reachable points $N_{Eps}(p)$. Then, DBSCAN uses depth-first search to discover all clusters.

2.3.1 Find Eps-neighborhood

The Eps-neighborhood of one point can be obtained with time complexity $O(kN^{1-1/k})$ by using k-dimensional tree (kd-tree)[9], where k is the dimensions of data. kd-tree is a binary tree in which each node is a k-dimensional point[10]. Each non-leaf node generates a hyperplane, which divides the space into two parts. The points to the left of this hyperplane are represented by the left subtree of that node, and the points to the right of the hyperplane are represented by the right subtree.

To find the Eps-neighborhood with respect to a point p, $N_{Eps}(p)$ is set to $\varnothing$ initially. Then the algorithm

performs a recursive process named *FindNeighbor*( ), which takes p and a node o in kd-tree as the input, and returns $N_{Eps}(p)$ in the subtree with respect to o. The process firstly sets $N_{Eps}(p) \leftarrow N_{Eps}(p) \cup \{o\}$, if $dist(p, o) \leqslant Eps$. It then judges whether p is to the left or right of the hyperplane with respect to o. In the case of left, let $N_{Eps}(p) \leftarrow N_{Eps}(p) \cup FindNeighbor(p, o.left\_son)$. If the distance between p and the hyperplane is greater than *Eps*, which means the points belonging to $N_{Eps}(p)$ may exist on the right subtree, let $N_{Eps}(p) = N_{Eps}(p) \cup FindNeighbor(p, o.right\_son)$. By invoking $FindNeighbor(p, kd\text{-}tree.root)$ for all points, we obtain all Eps-neighborhoods.

2.3.2 Find cluster

A cluster is uniquely determined by any of its points[2]. Therefore, to find a cluster, the algorithm can start with an arbitrary point. In order to make the algorithm more concise, it starts with an arbitrary core point p. Each point in $N_{Eps}(p)$ is directly density-reachable from p. Hence $N_{Eps}(p) \subseteq C$ with respect to p. If there is a chain of core points $p_1, \ldots, p_n, p_1 = p$, and $p_n = q$, such that $p_{i+1} \in N_{Eps}(p_i)$, each point in $N_{Eps}(q)$ is density-reachable from p. Hence $N_{Eps}(q) \subseteq C$ with respect to p.

To find the cluster with respect to p, the algorithm uses a stack, namely *stack*, to record core points. Let $stack \leftarrow \{p\}$ firstly. It enumerates all points q in $N_{Eps}(p)$ for each point in *stack*. Let $C \leftarrow C \cup \{q\}$, and push q into *stack* if q is a core point.

3 CI-Clustering

In this section, we introduce our improved DBSCAN algorithm for clustering incomplete data. As mentioned in Section 1, traditional imputation technologies sometimes just impute a point relatively close to a cluster. However, this is not enough for density-based clustering methods, which highly rely on absolute distance, to make them the same cluster. Instead, such kind of points will be considered as noise points because of too few points nearby.

To avoid such problem, we propose CI-clustering, which makes sufficient use of intermediate clustering results to increase the clustering quality.

Firstly, we construct a kd-tree with respect to D_0 and perform DBSCAN on it. Then, we constantly predict the cluster of incomplete data, impute them with respect to the cluster, insert newly completed point into the kd-tree, and update the clustering result.

In this section, we firstly discuss the method and its theoretical basis, and then briefly introduce the implementations.

3.1 Method

Suppose that $X \in \mathbf{R}^N$ is an incomplete tuple with k missing values, $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ denotes the set of missing values, and $\{x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}}\}$ denotes the set of existing values. The prediction of missing values in X is equivalent to the following optimization problem:

$$\arg\max_{x_{i_1}, x_{i_2}, \ldots, x_{i_k}} P(x_{i_1}, x_{i_2}, \ldots, x_{i_k} | x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}}) \tag{1}$$

Due to the locality of DBSCAN, this optimization problem will lead to a problem that the imputed data may become noise data as illustrated in Section 1. To avoid this problem, we can firstly determine the cluster for the incomplete data, and then perform the imputation according to points belonging to the same cluster. Consequently, the incomplete data will be imputed inside a cluster, and will not be considered as noisy data.

Concretely, the determination of the cluster based on incomplete data is equivalent to the following optimization problem:

$$\arg\max_{l} P(p \in C_l | x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}}) \tag{2}$$

where l is the number of the clusters.

For the convenience of calculation, we apply Bayesian inference to the following optimization problem:

$$P(p \in C_l | x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}}) = \frac{P(x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}} | p \in C_l) \times P(p \in C_l)}{P(x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}})} \tag{3}$$

where $P(x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}})$ is irrelevant to l. The optimization problem is converted to the following problem:

$$\arg\max_{l} P(x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}} | p \in C_l) \times P(p \in C_l) \tag{4}$$

Suppose that the points of a cluster is normal distributed in a local area. We estimate $P(x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}} | p \in C_l) \sim \mathcal{N}(\mu, \Sigma)$, where $\mathcal{N}$ is the normal distribution function, μ and Σ are estimated by the set of neighbor points $\{q \in C_l | partial\_dist(p, q) \leqslant Eps\}$, and $P(p \in C_l)$ is defined as

$$P(p \in C_l) = \frac{n_k}{n} \tag{5}$$

where n denotes the amount of points inside the neighborhood of p, and n_k denotes the amount of points belonging to C_l and inside the neighborhood.

By solving this problem, we can determine the cluster for p. Then, we need to impute the missing values based on C_l, which could be formalized as the following optimization problem:

$$\arg\max_{x_{i_1},x_{i_2},\ldots,x_{i_k}} P(x_{i_1}, x_{i_2}, \ldots, x_{i_k} | x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}}, p \in C_l) \tag{6}$$

Such formula is a conditional probability. We consider that the points in a cluster are normally distributed in a local area, which means

$$P(x) \sim \mathcal{N}(\mu, \Sigma) \tag{7}$$

Therefore,

$$P(x_{i_1}, x_{i_2}, \ldots, x_{i_k} | x_{j_1}, x_{j_2}, \ldots, x_{j_{(N-k)}}, p \in C_l)$$

subjects to a conditional normal distribution. Supposing X, μ, and Σ are partitioned as follows:

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{11} \Sigma_{12} \\ \Sigma_{21} \Sigma_{22} \end{bmatrix} \tag{8}$$

Then, suppose a point misses x_1 and has $x_2 = a$, the distribution of x_1 will be $\mathcal{N}(\bar{\mu}, \bar{\Sigma})$[11],

$$\bar{\mu} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(a - \mu_2) \bar{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{9}$$

This is also a normal distribution. Therefore, the optimal solution is to impute $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ by $\bar{\mu}$.

3.2 Algorithm description

Based on above discussions, we use traditional DBSCAN algorithm to cluster completed datasets. And then we use these clusters to predict incomplete data and insert them into completed data.

As shown in Algorithm 1, we firstly normalize both $X_{complete}$ and $X_{incomplete}$ using the same parameters with the function *Normalize* ( ). Then, based on the complete data, we build a kd-tree, save range search results with function *KD-TREE* ( ), and apply traditional DBSCAN algorithm for clustering with function *RangeSearch* ( ) and *getClusters* ( ). For each incomplete point, we predict its missing values with the function *predict* ( ) in Algorithm 2 and insert it into the kd-tree with function *insert* ( ). Then the neighborhoods and clusters are updated with functions *update_neighborhoods* ( ) and *update_clusters* ( ).

Note that the insertion of new points may make two clusters merge into one cluster. The reason is that the insertion of new points may change some non-core points to core points. That is why we should update neighborhoods and clusters after each insertion. However, updating operations will not consume much time, since those points must satisfy two conditions before the insertion: $|N_{Eps}| = MinPts - 1$ and within $N_{Eps}(p)$. After the insertion of point p, $|N_{Eps}|$ changes to *MinPts*, and thus becomes core points. That is to say, only points inside the neighborhood of p have a possibility to become a core point, and cause a merging operation. Therefore, while updating the clustering result, we do not have to scan all points again. Instead, we should only concentrate on points inside the neighborhood.

In traditional algorithms, we give each point a label with respect to its cluster during depth-first search. By using labels, once we merge two clusters, we have to change the label for all points into the same, which may consume much time. To reduce the time of merging operation, we use union-find sets[12] to represent cluster rather than index, which makes the amortized time complexity of merge operation to be $\textcircled{H}(\alpha(n))$, which is extraordinarily close to constant time.

Algorithm 1 DBSCAN algorithm for incomplete data

Input: $X_{complete}$ and $X_{incomplete}$
Output: Vector y denoting the clustering result

```
Normalize(X_complete)
Normalize(X_incomplete)
tree ← KD-TREE(X_complete)
neighborhoods ← RangeSearch(X_complete, tree)
clusters ← getClusters(X_complete, neighborhoods)
for x in X_incomplete do
    x ← predict(x, tree, clusters)
    insert(tree, x)
    update_neighborhoods(neighborhoods)
    update_clusters(clusters)
end for
```

Algorithm 2 Predict algorithm

Input: x, *tree*, *clusters*
Output: $x_{predicted}$

```
neighbors ← find_neighbor(x, tree)
max_p ← 0
for c in {c ∈ clusters | ∃x_0 ∈ neighbors, x_0 ∈ c} do
    X_c ← {x ∈ neighbors | x ∈ c}
    μ, Σ ← estimate_param(X_c, complete_attributions)
    p ← P(x, μ, Σ, |c|, |neighbors|)
    if max_p < p then
        max_p ← p
        x_predicted ← predict(μ, Σ, x)
    end if
end for
return x_predicted
```

4 LI-Clustering

4.1 Problem

CI-clustering avoids low-density areas between clusters with intermediate clustering result. However, low-

density areas inside clusters could not be avoided for non-convex clusters.

As show in Fig. 2, blue points are completed points. The straight line represents all possible positions of a point p with one missing value. Blue points within two dotted lines are the neighborhoods of p. Since we have used the average value of a conditional normal distribution to predict its missing values, p will be estimated near the red point, which means that it will be considered as noisy data. If this point is indeed belonging to the blue cluster, estimating its position near either orange points will avoid this problem.

Motivated by this observation, we propose LI-clustering. Such approach could successfully exclude low-density areas inside clusters by locating incomplete points to high-density local areas (orange points).

4.2 Method

Supposing that the point p has a missing value, the new method aims to maximize the number of points in $N_{Eps}(p)$, since $N_{Eps}(p)$ can be considered as an indicator of local density.

Firstly, we will discuss the case that the incomplete point p has one missing attribute. In the K-dimensional space, all of the possible positions of p form a straight line, denoted as l_p. For a point q, the ranges of points belonging to $N_{Eps}(q)$ represent a sphere (including inner space), denoted as S_q. If p belongs to $N_{Eps}(q)$, which means $p \in S_q$, since $p \in l_p$, we have $l_p \cap S_q \neq \varnothing$, which means S_q intersects into l_p. Equivalently, there is a same possibility that q belongs to $N_{Eps}(p)$ because of the symmetry of distance.

Many points have possibilities to belong to $N_{Eps}(p)$, so we can obtain a series of intervals.

Concretely, suppose that if $p_i = a$ or $p_i = b(a < b)$, p will be located on the intersections. Then, q belongs to $N_{Eps}(p)$ if and only if $p_i \in [a, b]$. Many points have a possibility to belong to $N_{Eps}(p)$, so we can obtain a series of intervals $[a, b]$.

If we predict $p_i = x$, $|N_{Eps}(p)|$ will be equal to the number of intervals containing x. Consequently, the process of prediction is equivalent to find the value x contained by the maximum intervals. Since the possible value of x is not a point but an interval, intuitively, we let x be equal to the mid-value of this interval. Actually, this is unimportant, because as long as x is within this interval, the neighborhood of this incomplete data will be the same. If the intervals are not unique, we choose the longer one, because those points with respect to the longer interval are either closer to the incomplete point or close to each other.

An example is shown in Fig. 3. In this example, $D = \{p, q_1, q_2, q_3, q_4\}$, and p is an incomplete point. To predict the position of p, we drew four circles (sphere in 2-dimensional space) with respect to q_1, q_2, q_3, and q_4, and obtained six intersections. Each interval has a number to identify the number of neighbors if the missing value is in this interval. As shown in Fig. 3, if we let p in $[a_2, b_1]$, there will be 2 points in $N_{Eps}(p)$, which is the maximum amount.

In the case that an incomplete point p has more than one missing attribute, we predict these attributes successively. Suppose that p has k missing attributions ($k \geqslant 2$). Different from previous cases, in the K-dimensional space, all of the possible positions of p represent a k-dimensional hyperplane, and the intersection of this hyperplane with a sphere represented by q is a $(k-1)$-dimensional sphere.

Nevertheless, we can predict the position of p in a similar way. When we predict the i-th attribute, the

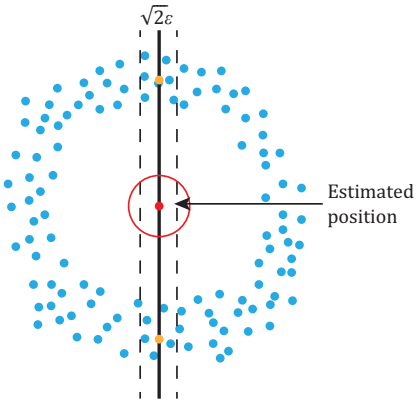


Fig. 2 An arbitrarily shaped cluster.

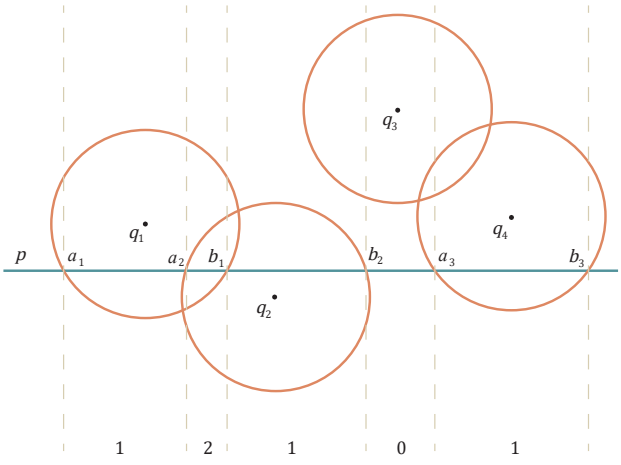


Fig. 3 Predicting 1-dimensional missing points.

k-dimensional hyperplane represented by p must go through the i-th axis. Because those intersections with respect to other points are in this hyperplane, we can project them on the i-th axis and obtain some segments. Take these segments as intervals, then we can predict the i-th attribute in the same way as before. After the prediction, the hyperplane represented by p has been changed to be $(k-1)$-dimensional. After k times iterations, p will be a complete point.

An example is shown in Fig. 4, where $K = 3$ and $k = 2$. Three spheres intersect with the hyperplane represented by p. By projecting these intersections on the corresponding axis, we can obtain three intervals. By cutting each other, these intervals are divided into 5 small intervals. To maximize $N_{Eps}(p)$, p should be in the small interval $[a_2, b_1]$ (as shown in Fig. 4, $N_{Eps}(p) = 2$).

4.3 Algorithm

Different from the CI-clustering algorithm, LI-clustering algorithm only uses neighborhood information without considering intermediate clustering results. Therefore, in this algorithm, we firstly build a kd-tree based on complete dataset. Then for each incomplete point, we predict its missing values and insert it into the kd-tree. Finally, after all points are imputed, we run traditional DBSCAN algorithm to obtain the clustering result.

In the first step, the judgement of whether a sphere intersecting the hyperplane is equivalent to the computation of the distance between the hyperplane and q, which is equivalent to find the minimum distance between q and any point p' in the hyperplane. Thus,

$$\begin{aligned}
&\underset{\{p'_i | i \in missAttr\}}{\arg\min} \quad dist(p', q) = \\
&\underset{\{p'_i | i \in missAttr\}}{\arg\min} \sqrt{\sum_{k=1}^{K} (p'_k - q_k)^2} = \\
&\underset{\{p'_i | i \in missAttr\}}{\arg\min} \sum_{k \in i} (p'_k - q_k)^2 = \\
&p'_i \leftarrow q_i
\end{aligned} \tag{10}$$

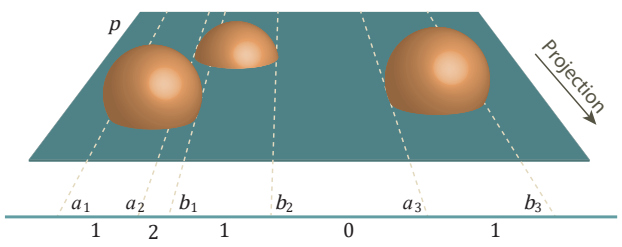


Fig. 4 Predicting multi-dimensional missing points.

where *missAttr* denotes the set of attributes with missing values.

This result is the same to the definition of *partial_dist*. Hence the distance between the hyperplane and q is $partial\_dist(p, q)$. If and only if $partial\_dist(p, q) \leqslant Eps$, the sphere intersects the hyperplane. This is the reason why we defined $partial\_dist(p, q)$ in that way.

In the second step, supposing that q' is a point in the intersection with respect to q, we predict the i-th attribute. Finding the interval is equivalent to find the maximum and minimum value of q'_i. Since q' is on the intersection, we have

$$dist^2(p', q') + dist^2(p', q) = Eps^2 \tag{11}$$

Since q' is in the hyperplane with respect to p, we have

$$dist^2(p', q') = \sum_{j \in missAttr} (p'_j - q'_j)^2 \tag{12}$$

Hence,

$$\begin{aligned}
(p'_i - q'_i)^2 = Eps^2 - dist^2(p', q) - \sum_{\substack{j \in missAttr \\ j \neq i}} (p'_j - q'_j)^2 = \\
Eps^2 - partial_dist^2(p, q) - \sum_{\substack{j \in missAttr \\ j \neq i}} (p'_j - q'_j)^2
\end{aligned} \tag{13}$$

Since $(p'_j - q'_j)^2 \leqslant 0$,

$$\min\{q'_i\} = q_i - \sqrt{Eps^2 - dist^2(p', q)} = p_i - \sqrt{Eps^2 - dist^2(p', q)} \tag{14}$$

$$\max\{q'_i\} = p_i + \sqrt{Eps^2 - dist^2(p', q)} \tag{15}$$

In the third step, we store all the intervals in a maximum heap, namely *bound*. After finding all the intervals, we traverse the heap in the order of values. If it is a right boundary, which means a new interval starting, we set $n = n + 1$. If it is a left boundary, which means a new interval ending, we set $n = n - 1$. The small interval with the maximum n is the interval with the maximum $|N_{Eps}(p)|$.

We illustrate this method with the example in Fig. 3, where $bound = \{a_1, a_2, a_3, b_1, b_2, b_3\}$. Initially, we set $n = 0$,

b_3 is a right boundary, let $n \leftarrow n + 1 = 1$;
a_3 is a left boundary, let $n \leftarrow n - 1 = 0$;
b_2 is a right boundary, let $n \leftarrow n + 1 = 1$;
b_1 is a right boundary, let $n \leftarrow n + 1 = 2$;
a_2 is a left boundary, let $n \leftarrow n - 1 = 1$;
a_1 is a left boundary, let $n \leftarrow n - 1 = 0$.

Hence the maximum $|N_{Eps}(p)|$ is 2, and the corresponding interval is $[a_2, b_1]$. We predict $p_i = (a_2 + b_1)/2$.

5 Experiment

5.1 Setup

(1) Datasets. Experiments were conducted on 15 various test datasets, including 10 synthetic datasets and 5 real-world datasets.

The details of the synthetic datasets are shown in Table 1. There are three values in the feature column, independent, dependent, and hollow. As the shapes of clusters shown in Fig. 5, independent clusters are the clusters in which data are not intersecting in any dimension, dependent clusters are the clusters that overlap with each other in some dimensions, and hollow clusters are those with low-density area inside.

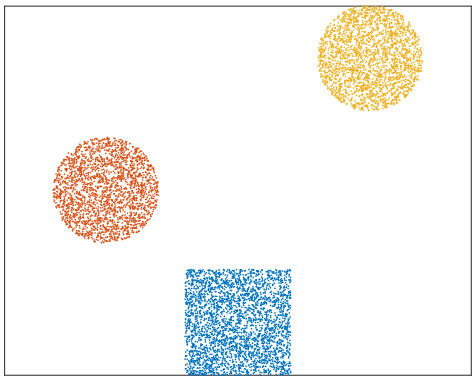
The details of the real-world datasets are shown in Table 2, where *MinPts* is the minimum number of points required to form a dense region in DBSCAN. RGB-D is a dataset first used in Ref. [13]. Such RGB-D data can be converted to 3D real-world coordinates of each depth pixel. In our experiment, we convert two depth images into real-world coordinates, and use those coordinates for clustering. This kind of data are most suitable for density-based clustering. Besides, we have also conducted experiments on Absenteeism at work[14], BuddyMove[15], Gesture Phase Segmentation[16], and Anuran Calls.

(2) Algorithms. Besides CI-clustering and LI-clustering, we have also implemented an algorithm, named Mean, which uses the average values of points in neighborhood to impute missing values.

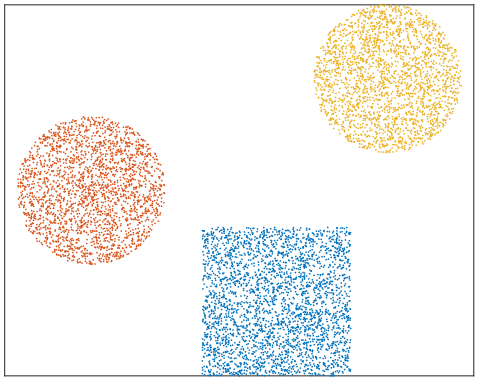
We have also implemented the popular K-Nearest Neighbor (KNN) imputation, but did not show the experiment results of this algorithm, since this algorithm always imputes incomplete points between clusters. As a result, different clusters are connected by these imputed points. An illustration of cluster shapes is shown in Fig. 6, with 10% missing values induced into synthetic

Table 1 Synthetic dataset.

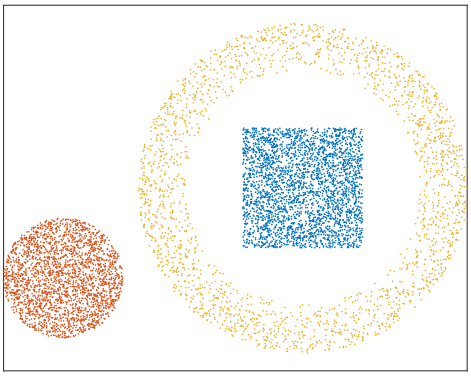
ID	#Points	#Dimensions	#Clusters	Feature
1	3000	2	3	Independent
2	3000	2	3	Dependent
3	3000	2	3	Hollow
4	1000	2	3	Independent
5	10 000	2	3	Independent
6	30 000	2	3	Independent
7	3000	4	3	Independent
8	3000	6	3	Independent
9	3000	8	3	Independent
10	3000	10	3	Independent



(a) Independent clusters



(b) Dependent clusters



(c) Hollow clusters

Fig. 5 Three features of synthetic datasets.

Table 2 Real-world dataset.

Name	#Points	#Dimensions	ε	*MinPts*
RGB-D(1)	640×480	3	0.015	4
RGB-D(2)	640×480	3	0.015	4
Absenteeism at work	740	21	1.000	4
BuddyMove	249	7	0.180	4
Gesture Phase Segmentation	9900	50	0.150	4
Anuran Calls	7195	22	0.200	4

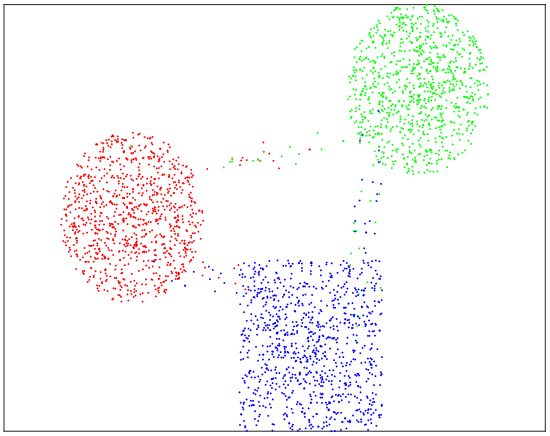


Fig. 6 KNN imputation result (K =3).

Dataset 2. After imputation, three clusters have been connected as a whole cluster.

(3) Implementation. All algorithms were implemented in C++ library *Libigl*[17] which has been used in the implementation of linear algebra related part in CI-clustering. The experiments were run on a Windows laptop with a 3.4 GHz processor and 16 GB main memory.

(4) Quality measures. Since our algorithms aim to implement DBSCAN on incomplete data, their clustering results should be as close as possible to those on complete data. Thus, we use original DBSCAN clustering results on the completed data as the baseline, and then use F-measure[18] to determine how close a result is to the baseline.

5.2 Experimental results

Effectiveness. We use synthetic Datasets 1, 2, and 3 to verify the effectiveness of CI-clustering and LI-clustering. Each dataset was induced 10% missing values. Experimental results are shown in Table 3, the numbers in which are F1-measure of the experimental results. For independent dataset, results of three algorithms have similar quality. This is because the value ranges of clusters do not overlap in any dimension. Thus, these clusters do not interact with each other. For the dependent dataset, where three clusters overlap with each other, the quality of the clustering result of Mean is reduced significantly. For the hollow dataset, because of the blank area inside the cluster, the quality of the clustering result of CI-clustering is reduced significantly, while the quality of LI-clustering is reduced relative slightly.

Table 3 Experiment result of synthetic Datasets 1, 2, and 3.

Feature	Mean	CI-clustering	LI-clustering
Independent	0.989 790	0.988 110	0.991 780
Dependent	0.878 380	0.950 283	0.974 400
Hollow	0.825 039	0.739 776	0.893 936

From above experiments, we have verified the effectiveness of CI-clustering and LI-clustering, and for datasets with low-density area inside clusters, LI-clustering will reach to a better result. Then, we analyze the algorithm from the perspective of quantity and dimension through more experiments.

Scalability. In Figs. 7 and 8, experimental results show that the data size does not have significant effect on the quality of clustering result. The running time of CI-clustering grows faster than LI-clustering, because we need to calculate the inverse matrix while calculating the probability density of the normal distribution. Besides, the running time of CI-clustering is not linearly related to missing rate. This is mainly caused by two reasons. One is that with the growth of missing rate, there will be many blank points with all attributes missing. The partial distance between these points from all other points is 0. Thus, all points are inside their neighborhood. The calculation will cost much time. The other reason is that with too many incomplete points, there will be many small clusters in the early stage, and the inverse matrix need to be calculated for each cluster.

Impact of #dimensions. For dimensions, because of the information redundancy, clustering results are better on high-dimensional data. This conclusion is obvious for CI-clustering. While for LI-clustering, experimental results are fluctuating. This is because LI-clustering imputes each missing value individually without considering all dimensions at the same time. Note that, for the running time of CI-clustering, the gap between 0% loss and 10% loss is wider with the growth of dimensions. This is because the quantity of incomplete points is higher for high-dimension datasets for a fixed missing rate. For example, with 10% missing values induced, the quantity of incomplete points of Dataset 2 will not exceed 600, while the quantity of the Dataset 6 is always over 1900.

Real-world data. Finally, we test the effectiveness of the algorithm through real-world data. Experimental results in Fig. 9 show that both CI-clustering and LI-clustering have good performance even, when the missing rate is 30%, which is much better than Mean. The number in the bracket in the legend of Fig. 10 shows the number of the dataset, i.e., RDG-D(1) or RDG-D(2). Figure 9 shows the experimental results on other real

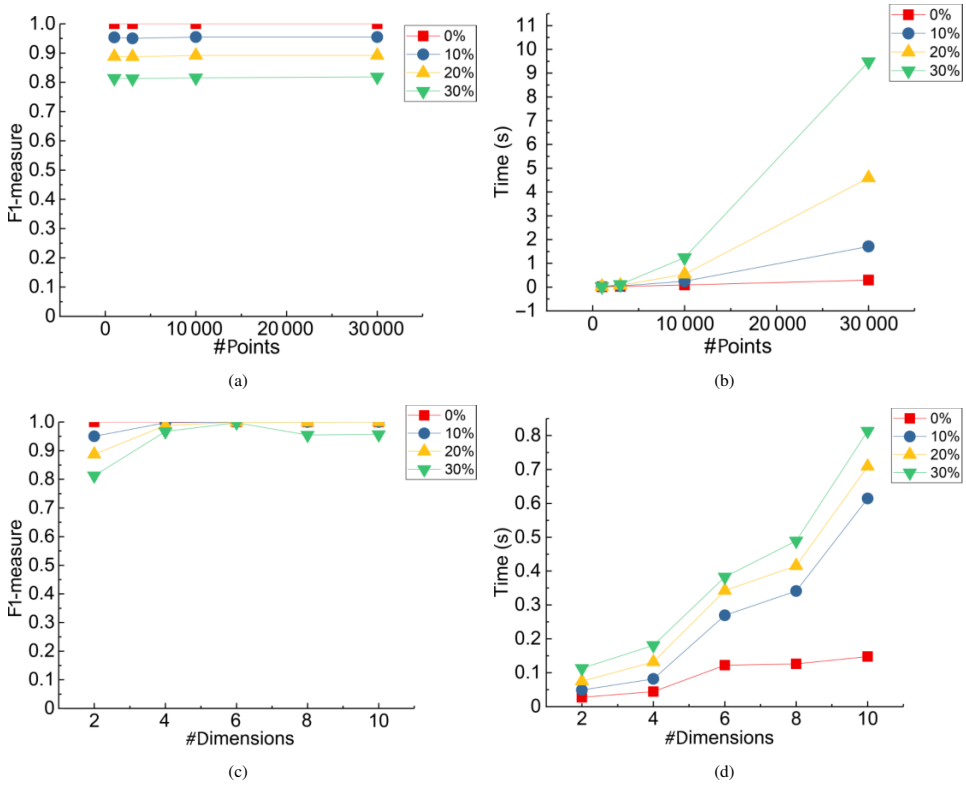


Fig. 7 Experimental results of CI-clustering with different missing rates (0%, 10%, 20%, and 30%).

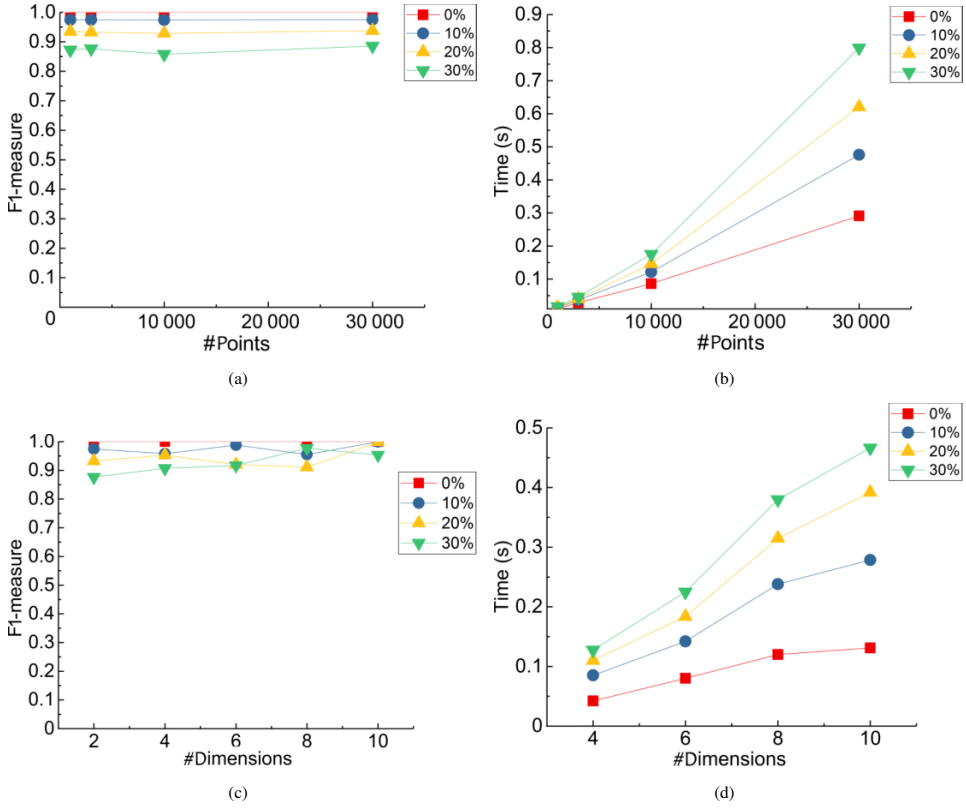


Fig. 8 Experimental results of LI-clustering with different missing rates (0%, 10%, 20%, and 30%).

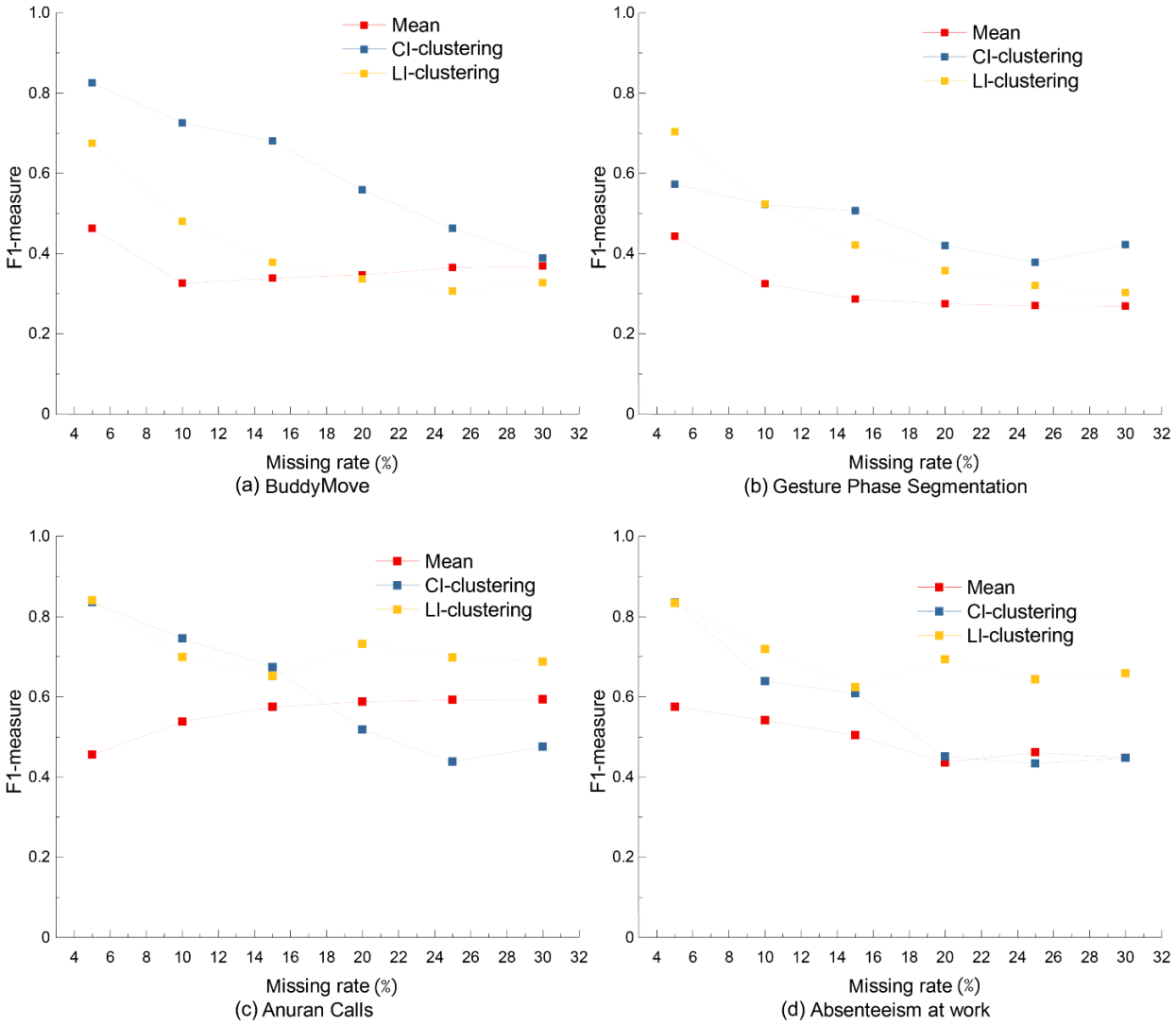


Fig. 9 Experimental results of real-world datasets.

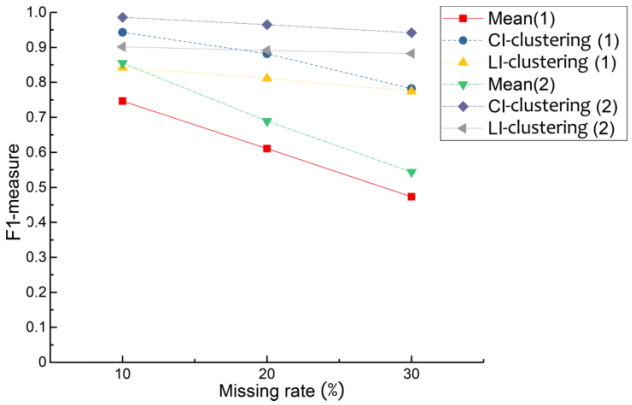


Fig. 10 Experimental results of RGB-D.

world datasets. When the missing rate is low, CI-clustering and LI-clustering have much higher F1-measure values, which means that their clustering result are closer to the standard clustering result on the complete data using original DBSCAN. As the missing rate increases, different datasets are in different trends, but the optimal result is always achieved by CI-clustering or LI-clustering. Thus, we have verified the effectiveness of CI-clustering and LI-clustering on real-world datasets.

6 Conclusion

In this paper, we discussed two effective density-based clustering algorithms for incomplete data. We showed that traditional strategies to handle missing values are unsuitable for the density-based clustering algorithm. Motivated by the problems occurred in traditional imputation methods and based on Bayesian theory, we introduced CI-clustering, which conducts imputation and clustering concurrently with the help of intermediate clustering results. For various excepted cluster shapes, we propose LI-clustering. The experimental results show

that our proposed algorithms are effective, and each of them has its own characteristics. Our further work will focus on the clustering for large incomplete datasets and incomplete data stream.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. U1866602 and 71773025) and the National Key Research and Development Program of China (No. 2020YFB1006104).

References

[1] R. J. G. B. Campello, P. Kröger, J. Sander, and A. Zimek, Density-based clustering, *Wiley Interdisciplinary Reviews*: *Data Mining and Knowledge Discovery*, vol. 10, no. 2, p. e1343, 2020.

[2] X. W. Xu, M. Ester, H. P. Kriegel, and J. Sander, A distribution-based clustering algorithm for mining in large spatial databases, in *Proc. 14th Int. Conf. Data Engineering*, Washington, DC, USA, 1998, pp. 324–331.

[3] H. O. Hartley and R. R. Hocking, The analysis of incomplete data, *Biometrics*, vol. 27, no. 4, pp. 783–823, 1971.

[4] J. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proc. 5th Berkeley Symp. Mathematical Statistics and Probability*, Oakland, CA, USA, 1967, pp. 281–297.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society. Series B* (*Methodological*), vol. 39, no. 1, pp. 1–38, 1977.

[6] E. Acuña and C. Rodriguez, The treatment of missing values and its effect on classifier accuracy, in *Classification, Clustering, and Data Mining Applications*, D. Banks, F. R. McMorris, P. Arabie, and W. Gaul, eds. Berlin, Germany: Springer, 2004, pp. 639–647.

[7] R. J. Hathaway and J. C. Bezdek, Fuzzy c-means clustering of incomplete data, *IEEE Transactions on Systems, Man, and Cybernetics*, *Part B* (*Cybernetics*), vol. 31, no. 5, pp. 735–744, 2001.

[8] M. G. Kendall, *Advanced Theory of Statistics Vol.-I*. London, UK: Charles Griffin, 1943.

[9] D. T. Lee and C. K. Wong, Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees, *Acta Informatica*, vol. 9, no. 1, pp. 23–29, 1977.

[10] J. L. Bentley, Multidimensional binary search trees used for associative searching, *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[11] L. J. Gleser, Multivariate statistics: A vector space approach, *Journal of the American Statistical Association*, vol. 80, no. 392, pp. 1069–1070, 1985.

[12] B. A. Galler and M. J. Fisher, An improved equivalence algorithm, *Communications of the ACM*, vol. 7, no. 5, pp. 301–303, 1964.

[13] K. Lai, L. F. Bo, X. F. Ren, and D. Fox, A large-scale hierarchical multi-view RGB-D object dataset, in *Proc. 2011 IEEE Int. Conf. Robotics and Automation*, Shanghai, China, 2011, pp. 1817–1824.

[14] A. Martiniano, R. P. Ferreira, R. J. Sassi, and C. Affonso, Application of a neuro fuzzy network in prediction of absenteeism at work, in *Proc. 7th Iberian Conf. Information Systems and Technologies* (*CISTI 2012*), Madrid, Spain, 2012, pp. 1–4.

[15] S. Renjith and C. Anjali, A personalized mobile travel recommender system using hybrid algorithm, in *Proc. 2014 1st Int. Conf. Computational Systems and Communications* (*ICCSC*), Trivandrum, India, 2014, pp. 12–17.

[16] R. C. B. Madeo, C. A. M. Lima, and S. M. Peres, Gesture unit segmentation using support vector machines: Segmenting gestures from rest positions, in *Proc. 28th Annu. ACM Symp. Applied Computing*, New York, NY, USA, 2013, pp. 46–52.

[17] A. Jacobson, D. Panozzo, C. Schüller, O. Diamanti, Q. N. Zhou, S. Koch, J. Dumas, A. Vaxman, N. Pietroni, S. Brugger, et al., libigl: A simple C++ geometry processing library, http://libigl.github.io/libigl/, 2018.

[18] Y. Sasaki, The truth of the F-measure, *Teach Tutor Mater*, vol. 1, no. 5, pp. 1–5, 2007.

Hongzhi Wang received the BEng, MEng, and PhD degrees from Harbin Institute of Technology, China in 2001, 2003, and 2008, respectively. He is a professor and PhD supervisor at Harbin Institute of Technology, the secretary general of ACM SIGMOD China, a CCF outstanding member, and a member of CCF databases and big data committee. His research fields include big data management and analysis, database, knowledge engineering, and data quality.

He was a "starring track" visiting professor at MSRA and a postdoctoral fellow at University of California, Irvine. He has been the PI or co-PI for more than 10 national or international projects, including NSFC key projects, NSFC projects, and national technical support projects. He also serves as a member of ACM Data Science Task Force. He has won the First Natural Science Prize of Heilongjiang Province, MOE Technological First Award, Microsoft Fellowship, IBM PHD Fellowship, and Chinese Excellent Database Engineer. His publications include over 200 papers, including VLDB, SIGMOD, and SIGIR, 6 books, and 3 book chapters. His PhD thesis was elected to be outstanding PhD dissertation of CCF and Harbin Institute of Technology. He severs as a reviewer of more than 20 international journals, including *IEEE TKDE*, and a PC member of over 30 internal conferences. His papers were cited more than 1500 times. His personal website is http://homepage.hit.edu.cn/wang.

Zhonghao Xue received the BEng degree from Harbin Institute of Technology in 2019. He is a master student at University of Southern California, USA . His research interests include analysis and processing of dirty data, and conventional machine learning algorithm.