Fuzzy and crisp clustering methods based on the neighborhood concept: A comprehensive review

Gözde Ulutagay* and Efendi Nasibov

Department of Industrial Engineering, Izmir University, Izmir, Turkey
Department of Computer Science, Dokuz Eylül University, Izmir, Turkey

Abstract. The aim of this paper has twofold: i) to explore the fundamental concepts and methods of neighborhood-based cluster analysis with its roots in statistics and decision theory, ii) to provide a compact tool for researchers. Since DBSCAN is the first method which uses the concept of neighborhood and it has many successors, we started our discussion by exploring it. Then we compared some of the successors of DBSCAN algorithm and other crisp and fuzzy methods on the basis of neighborhood strategy.

Keywords: Neighborhood-based cluster analysis, spatial data, DBSCAN, FDBSCAN, LDBSCAN, NRFJP, Bifurcated FJP, FN-DBSCAN, SDBSCAN, Scalable FN-DBSCAN

1. Introduction

Although a huge number of methods have been presented to the clustering literature in recent years, we cannot point out any of them as suitable for all kinds of purposes [8, 15, 29, 31]. Clustering algorithms could be grouped into five main classes such as hierarchical, partition-based, grid-based, density-based (or neighborhood-based), and model-based [9]. Of course, such a classification is not strict. A method can be counted in more than one category according to its structural processing. Furthermore, the above classification is not the only one. Different classifications are also possible. For instance, an approach for categorization of clustering methods can be made by distinguishing among crisp, fuzzy (possibilistic), and probabilistic clustering [12, 14].

Among the premier classification of clustering methods, density-based clustering has been accepted to be very influential to handle huge amounts of spatial data. DBSCAN is the first method which uses the concept of neighborhood [7]. Since its proposal, DBSCAN attracts much attention due to its intuitive meaning and good performance with the large spatial data sets. Since outliers could be the points with unique properties, the information about them is invaluable. For all of the aforementioned reasons, DBSCAN has a growing popularity among researchers [1–3, 9, 26, 27]. In this work, we focus to give a general viewpoint and a compact tool for researchers for density-based methods which take their roots from DBSCAN method.
2. Density-based spatial clustering of applications with noise (DBSCAN)

DBSCAN algorithm is able to detect clusters in a spatial database with noise [7]. The running principle of the algorithm is based on the fact that a cluster is formed by the density-reachable points from an arbitrary core point in the cluster. By gathering directly density-reachable points iteratively, the density-reachable points are determined. For each point in the database, the algorithm searches its ε-neighborhood. To initialize a new cluster, there must be a point o which has at least MinPts neighbors within its ε-neighborhood. If a point o has more than MinPts neighbors within ε-neighborhood Nε(o), the point o is called a core point, and the new cluster is generated with the points in Nε(o). After that, for all points p in C which have not been handled yet, the ε-neighborhood is inspected. If Nε(p) has more than MinPts points, the neighbors of p which are not already put in C are inserted to the cluster and their ε-neighborhood is inspected in the following step. This iterative procedure continues until no new point can be put in to the current cluster C. Then, in order to create a new cluster, the algorithm handles another point which has not yet been processed. Note that in DBSCAN algorithm, the user has two kinds of requirements for the resulting clusters, both of which can be met by setting ε and MinPts and re-running the algorithm until a satisfactory result is acquired: i) obtaining high-density clusters, ii) obtaining low-density clusters. On very large databases the cost of the algorithm is high, and since no a priori knowledge is used, too much processing time is wasted.

Consider the dataset X = {x1, x2, ..., xn} where each object xi, i = 1, ..., n has m attributes. Hence, each object xi could be perceived as a point of m-dimensional space, i.e., xi = (x(i1), x(i2), ..., x(im)). In this sense, distance d(x, y) between any points x, y ∈ X could be obtained.

For a point x ∈ X, by using any membership, the neighborhood set for a point x ∈ X with a fixed ε parameter is:

\[ N_\varepsilon(x) = \{ y \in X \mid d(x, y) \leq \varepsilon \} \]  \hspace{1cm} (2.1)

Note that the neighborhood set obtained by the above formula will be composed of the points within an ε-radius of point x which is given below [20]:

\[ N_\varepsilon(x) = \begin{cases} 1, & \text{if } d(x, y) \leq \varepsilon, \\ 0, & \text{otherwise.} \end{cases} \]  \hspace{1cm} (2.2)

where \( N_\varepsilon(x) \) is the membership degree of the point y to the neighborhood set of point x. The fuzzy version of this function will be given in Section 7.

As mentioned previously, a point \( x \in X \) with minimum density criterion, in other words with at least MinPts points around it ε distance, is a core point. Mathematically, a core point must hold \( |N_\varepsilon(x)| \geq \text{MinPts} \) where \( |N_\varepsilon(x)| \) is the cardinality of the set \( N_\varepsilon(x) \). All the points within the ε distance of a core point are perceived as non-core elements of the cluster and points outside any cluster are called noise points. If there is a point in the ε neighborhood of a core point q, then this point is called a directly density reachable point from q and by the chain of such points, the first and the last elements of the chain are called density reachable points. Accordingly, two points are density connected to each other by the help of the common core point with parameters ε and MinPts. Two core points within an ε radius form a cluster together with the points around their neighborhood. The main advantage of the DBSCAN algorithm is that by chaining the core points large and irregularly shaped clusters are detected in a spatial database.

DBSCAN algorithm:

Step 1. Specify ε and MinPts.
Step 2. Label all the points in the database as unclassified.
Step 3. Find an unclassified core-point p with ε and MinPts. Label p as classified. Let a new cluster be the current cluster and add p to this cluster.
Step 4. Find all the unclassified points in the ε-neighborhood.
of \( p \). Construct a set of seeds and put all these points into the set.

Step 5. Take a point \( q \) in the seeds, label \( q \) as classified, add \( q \) to the current cluster, and eliminate \( q \) from the seeds.

Step 6. If \( q \) is a core-point with \( \varepsilon \) and \( \text{MinPts} \), add all the unclassified points in the \( \varepsilon \)-neighborhood of \( q \) to the set of seeds.

Step 7. Repeat step 5 through 6 until the set of seeds get empty.

Step 8. Start a new cluster and repeat step 3 through 7 until no more core-points can be found.

Step 9. Label all the clusters found so far, and label the points as noise if they do not belong to any cluster.

As mentioned in the beginning of this section, it might be troublesome to estimate the density threshold beforehand. A higher value of \( \text{MinPts} \) will result in too many little, high-densed clusters while a smaller value will produce a few number of big clusters or just one cluster which is made up of all the points of the data set. DBSCAN needs help to decide the global parameter \( \varepsilon \). To help decrease the computational complexity, the parameter \( \text{MinPts} \) is fixed to 4 [7]. Before deciding on the value of \( \varepsilon \), the distance between a point and its \( k \)-th (usually \( k = 4 \)) nearest neighbors for all points is calculated. Then, by sorting all points according to their distance calculated a priori, the sorted \( k \)-dist graph is plotted [27]. This process is time consuming. Furthermore, a user must survey through the graph and find the first “valley”. The corresponding distance is determined as the value of \( \varepsilon \) and the final clustering quality depends highly on the \( \varepsilon \) parameter. When the dataset is the response set of objects satisfying some qualification, then the trimming of \( \varepsilon \) should be performed each time and the cost of DBSCAN will increase.

3. **Fuzzy DBSCAN (FDBSCAN)**

In some areas of application like distributed clustering or clustering of moving objects, a point is expressed by a region in which all points in this region represent the point equally rather than by a unique feature vector. The similarity between two fuzzy objects is expressed by distance probability functions which characterize a probability value to each possible distance value in the work of Kriegel and Pfeifle [16]. All the information carried out by these fuzzy functions is used just by incorporating these functions into data mining algorithms directly.

Let \( d : D \times D \rightarrow IR_{+}^{} \) be a distance function. The probability density function \( p_d : D \times D \rightarrow IR_{+}^{} \) between two objects could be calculated accurately, i.e., \( p_d(o, o') = \delta(x - x') \) [16]. The probability distribution function \( p_d := O \times O \rightarrow IR_{+}^{} \rightarrow [0, 1] \) is called a distance distribution function if the following condition holds where \( P_d(o, o') \leq b \) is the probability that \( d(o, o') \) is between \( a \) and \( b \):

\[
P(a \leq d(o, o') \leq b) = \int_{a}^{b} p_d(o, o') dx. \quad (3.1)
\]

The probability density function \( p_d \) is equal to the Dirac-delta function if the distance \( \tau = d(o, o') \) between two objects could be calculated accurately, i.e., \( p_d(o, o') = \delta(x - x') \) [16]. The probability distribution function \( p_d := O \times O \rightarrow IR_{+}^{} \rightarrow [0, 1] \) is directly:

\[
P(d(o, o') \leq b). \quad (3.2)
\]

Obtaining an aggregated distance value by fuzzy distance functions could be more practical instead of dealing distance functions which provide a unique distance value like in traditional algorithms. The distance expected value \( E_d : O \times O \rightarrow IR_{+}^{} \) could be used which can be tackled as the average distance between the fuzzy objects, i.e.,

\[
E_d(o, o') = \int_{-\infty}^{\infty} x \cdot p_d(o, o')(x) dx. \quad (3.3)
\]

FDBSCAN algorithm is based on an improved version of the core object definition of DBSCAN algorithm. The core object probability of an object \( o \) which indicates the likelihood that \( o \) is a core object and the reachability probability of \( p \) with respect to \( o \) which indicates the probability that \( p \) is directly density-reachable from \( o \), are calculated as follows, respectively:

\[
P_{\text{core}}^{\text{FDBSCAN}}_{d \rightarrow D}(o) = \sum_{A \in D} \prod_{p \in A} P_d(p, o)(\epsilon) \prod_{p' \in A, p' \neq p} (1 - P_d(p', o)(\epsilon)). \quad (3.4)
\]

\[
P_{\text{reach}}^{\text{FDBSCAN}}_{d \rightarrow D}(p, o) = P_{\text{core}}^{\text{FDBSCAN}}_{d \rightarrow D}(p) \cdot P_d(p, o)(\epsilon). \quad (3.5)
\]
where \(P_a = D \times D \rightarrow (IR^2 \rightarrow [0, 1])\) is the distance distribution function.

The traditional DBSCAN algorithm clusters a data set by always adding directly reachable objects from the current query object to the cluster. The fuzzy version of DBSCAN runs very similar to the traditional approach. The object \(p\) is included to the current cluster if the value \(p_{reach}\) is more than 0.5 where \(o\) is the current query object [6]. Note that for object \(p\), if the value of core object probability is less than 0.5, i.e., \(pcore(p) < 0.5\), then the value of reachability probability, i.e., \(p_{reach}\), could not be more than 0.5. Thus, \(p\) will not be included to the current cluster. This is also a generalization of the traditional approach.

4. Local DBSCAN (LDBSCAN)

In some cases, using a global neighborhood threshold might not be convenient since the local densities differ. For this reason, different local density thresholds could be required to determine clusters in some regions of the data set. In order to solve this adversity, local-density-based cluster analysis is proposed [6]. In this method, each point in the data set is represented by a local outlier factor (LOF) which indicated the degree of being an outlier. Also, for each point local reachability density (LRD), which indicates local density, is defined to identify clusters in the dataset.

In LDBSCAN method, the distance \(d(p, o)\) between \(p\) and a point \(o \in D\) is called \(k\) distance of point \(p\) or \(k_{dist}(p)\) and it holds the following conditions:

\[
\begin{align*}
    i) \quad & d(p, o') \leq d(p, o), \text{ for at least } k \text{ points } o' \in D(p), \\
    ii) \quad & d(p, o') < d(p, o), \text{ for at most } k - 1 \text{ points } o' \in D(p),
\end{align*}
\]

\(k\)-nearest neighbors of a point \(p\) includes each point within a defined \(k_{dist}\). Hence,

\[
N_{k_{dist}}(p) = \{q \in D(p)|d(p,q) \leq k_{dist}(p)\}
\]

The reachability distance of object \(p\) with respect to point \(o\) is formulated as follows:

\[
reach_{dist}(p, o) = \max\{k_{dist}(o), d(p, o)\}.
\] (4.1)

For each point \(p\), the local reachability density and local outlier factor are calculated by the following formulas:

\[
LRD_{MinPts}(p) = \frac{|N_{MinPts}(p)|}{\sum_{o \in N_{MinPts}(p)} \text{reach}_{dist}(p, o)}.
\] (4.2)

\[
LOF_{MinPts}(p) = \frac{\sum_{o \in N_{MinPts}(p)} LRD_{MinPts}(o)/LRD_{MinPts}(p)}{|N_{MinPts}(p)|}.
\] (4.3)

It is obvious that the local outlier factor of a point \(p\) is the average of the ratio of the local reachability density of a point \(p\) and local reachability density of \(MinPts\)-nearest neighbors of point \(p\). Local reachability density obtains the degree for which point \(p\) is called an outlier. Since the point \(p\) becomes further from its nearest cluster, the ratio of the local reachability density of \(p\) and local reachability density of \(MinPts\)-nearest neighbors of point \(p\) gets larger and also the value of local outlier factor of \(p\) becomes higher. It might be noted that an \(p\) should be in a particular cluster if \(LOF(p) \leq LOFUB\). Because local outlier factor indicates the degree that a point is an outlier and local outlier factor of most of the points \(p\) within the same cluster is about 1. For LDBSCAN algorithm if \(LOF(p) \leq LOFUB\) holds than the point \(p\) is called a core point.

In order to identify a cluster, LDBSCAN gets a random point \(p\) and consider any other point local-density reachable from \(p\) with respect to \(LOFUB\), \(p\), and \(MinPts\). If \(p\) is a core point, this procedure obtains a cluster. Otherwise, the next point is controlled in the data set [6].

**LDBSCAN algorithm:**

**Step 1.** Specify parameters \(LOFUB\), \(p\), and \(MinPts\). Label all points unclassified.

**Step 2.** Calculate LRD and LOF of each point \(p\) in the dataset. Set \(ClusterID = 0\).

**Step 3.** For each point \(p\) in the data set Do

* if \(p\) is unclassified then begin
  * if \(LOF(p) < LOFUB\) then begin
    * \(ClusterID := ClusterID + 1\); Run Procedure (Expand_Cluster)
  * end if;
* end if;}
Procedure (Expand_Cluster)
Assign ClusterID to the Point;
for i:=1 to MinPts Do
    currentP := Point.Neighbor(i);
    if currentP is [UNCLASSIFIED or NOISE] then
        CurrentP add to the TempVector;
        Assign ClusterID to the currentP;
        CurrentP := Point.Neighbor(i);
        if currentP is [UNCLASSIFIED or NOISE] then
            CurrentP add to the TempVector;
            Assign ClusterID to the currentP;
        end if
    end for
end if
while TempVector is not Empty Do
    Point := TempVector.firstElement();
    Remove Point from TempVector;
    for i:=1 to MinPts Do
        currentP := Point.Neighbor(i);
        if currentP is [UNCLASSIFIED or NOISE] then
            CurrentP add to the TempVector;
            Assign ClusterID to the currentP;
        end if
    end for
end while

Procedure Direct_Reachability (currentP, Point)
if LRD(currentP) \geq LRD(Point) \cdot (1+\varepsilon) then
    return True
else return False;
end

5. Fuzzy joint points (FJP) and noise robust FJP (NRFJP)
Fuzzy Joint Points is a density-based algorithm which perceives the neighborhood concept from a level-based viewpoint which means that the points are considered in how much detail in the formation of non-heterogenous classes [19]. In this method, the fuzzier the points, more similar they are. The points with more similarity will be in the same class while the points with less similarity will be in different classes with membership values between [0, 1]. The points which form the \( \alpha \)-level sets are detected at each iteration of the algorithm.

Let \( X = \{x_1, x_2, ..., x_n\} \) be a data set. The radius of the fuzzy points is computed as

\[
R = \frac{\max\{d(x_i, x_j) | x_i, x_j \in X\}}{2}
\]  

(5.1)

For \( A, B \in X \), the degree of the fuzzy relation \( T(A, B) \) is defined as follows:

\[
T(A, B) = \frac{d(a, b)}{d_{\text{max}}}
\]

(5.2)

If \( a \in E^1 \) is the center of fuzzy point \( A \) and \( R \in E^1 \) is the radius of its support supp \( A \), the conical fuzzy point \( A = (a, R) \in E(E^1) \) is a fuzzy set with the following membership function:

\[
\mu_A(x) = \begin{cases} 
1 - \frac{d(x, a)}{R}, & \text{if } d(x, a) \leq R, \\
0, & \text{otherwise.}
\end{cases}
\]

(5.3)

In FJP algorithm, any two fuzzy points \( A \) and \( B \) from \( X \subseteq F(E^1) \) are called \( \alpha \)-neighbor fuzzy points if \( d(T(A, B)) \geq \alpha \) for a constant \( \alpha \in (0, 1] \) and if these points are joined by the help of \( \alpha \)-neighbor fuzzy points \( C^1, ..., C^k \), \( k \geq 0 \), then these are \( \alpha \)-joint fuzzy points. FJP algorithm is very sensitive to noise points. This problem is alleviated by the Noise-Robust Fuzzy Joint Points (NRFJP) algorithm. Before starting the clustering process, if \( C \), fuzzy neighborhood cardinality, and \( T(x, y) \), of each point is lower than an \( \epsilon_t \) threshold, this point is considered as noise. After the cleaning process, clustering is performed with core points, and at the end of the clustering, the noise points are assigned to the nearest clusters. Note that, if \( \epsilon_t = 0 \), NRFJP algorithm transforms into FJP algorithm.

For the fuzzy relation \( T \), fuzzy neighborhood set of given point \( x \in X \), \( N(x) \), and the \( \epsilon_t \)-level set of \( N(x) \), \( N(x, \epsilon_t) \), are defined as follows, respectively:

\[
N(x) = \{y \in X \mid T(x, y) \geq 1\}
\]

(5.4)

\[
N(x, \epsilon_t) = \{y \in X \mid T(x, y) \geq \epsilon_t\}
\]

(5.5)
NRFJP Algorithm

Step 1. Compute: \( d_{ij} := d(x_i, x_j), \ i, j = 1, \ldots, n; \)
\( d_{\max} := \max_{i, j \in \{1, \ldots, n\}} d_{ij}; \) \( \varepsilon := 0.01 \cdot \min_{i \neq j} d_{ij}; \) Set up the values \( \varepsilon_1 \) and \( \varepsilon_2 \); Let \( a_0 := 1; \)
Step 2. Compute the fuzzy relation \( T_{ij} := 1 - \frac{d_{ij}}{\varepsilon_1}, \ i, j = 1, \ldots, n; \)
Step 3. Run Procedure \( \text{NoiseFilter}(e_1, e_2) \)
and calculate transitive closure \( T \) of \( T \) for \( X_{\text{core}} \);
Step 4. Let \( n_i \) be the number of points of \( X_{\text{core}} \); \( \varepsilon_1 \)
Step 5. Compute: \( d(y_i, y_j) = \min\{d(y_i, y_j), d(y_j, y_i)\}, \ i, j = 1, \ldots, n; \)
\( d := \min\{d(y_i, y_j), a_2 := \max\{1 - \frac{d_{ij}}{\varepsilon_2}, 0\}; \)
Step 6. Run Procedure \( \text{Clusters}(a) \)
to form fuzzy \( a \)-joint sets \( X_1, X_2, \ldots, X_k \) with fuzzy points \( (y_i, \frac{d_{ij}}{\varepsilon_2}), i \in \{1, \ldots, n\} \), and to determine the number \( k \) of these sets with current value \( a_0; \)
Step 7. If \( k > 1 \), then denote \( y_i := X_i, i = 1, \ldots, n \);
Step 8. Compute: \( \alpha_0 := a_2 - a_0 + 1, i = 1, \ldots, n; \)
Step 9. Run Procedure \( \text{Clusters}(\alpha) \)
with parameter \( \alpha; \)
Step 10. \( \alpha \) is the optimal membership degree of clustering; \( \bar{X} \) is the optimal cluster number; \( X_1, \ldots, X_k \) is a partition of the set \( X \).
Step 11. For each point \( x \in X_{\text{noise}} \)
repeat the step 12;
Step 12. Compute \( k^* := \arg\min\{|\text{dist}(x, X_i)|, i = 1, \ldots, k \}; \)
Assign \( x \) to the \( X_{k^*} \).

Procedure \( \text{Clusters}(a) \)

Input parameter: \( a; \)
Output parameter: \( a \)-fuzzy joint sets \( X_1, X_2, \ldots, X_k \) \( k \)-cluster number
Step 1. \( S := X = \{x_1, x_2, \ldots, x_n\}; \) \( k := 1; \)
Step 2. Get the first element \( A \in S \) of the set \( S; \)
Create sets: \( X^k := \{B \in S | \text{dist}(A, B) \geq a\}; \) \( S := \)
Step 3. If \( S \neq \emptyset \), then let \( k := k + 1 \) and go to Step 2;
Otherwise go to Step 4;
Step 4. Provide the sets \( X^1, X^2, \ldots, X^k \)
and the number of these sets.

By using the following procedure \( \text{NoiseFilter}(e_1, e_2) \), the original data set is partitioned into two disjunctive sets, \( X_{\text{core}} \) and \( X_{\text{noise}} \) such that \( X = X_{\text{core}} \cup X_{\text{noise}}, X_{\text{core}} \cap X_{\text{noise}} = \emptyset. \)

Procedure \( \text{NoiseFilter}(e_1, e_2) \)

Input parameters: \( e_1 \) and \( e_2 \);
Output parameter: The sets \( X_{\text{core}} \) and \( X_{\text{noise}} \);
Step 1. Let \( X = \{x_1, x_2, \ldots, x_n\} \) be the set of initial points, \( X_{\text{core}} = \emptyset; \)
Step 2. For each point \( x \in X; \)
i. Compute and \( N(x, e_1) = \sum_{y \in N(x, e_1)} T(y, y); \)
ii. If \( \text{count}(N(x, e_1)) < e_2 \), then mark \( x \) as noise point: \( X_{\text{noise}} = X_{\text{noise}} \cup \{x\}; \)
Step 3. Let \( X_{\text{core}} = X \setminus X_{\text{noise}}; \)
Step 4. Label the sets \( X_{\text{core}} \) and \( X_{\text{noise}}. \)
Note that this procedure is used only with the NRFJP algorithm.

6. Bifurcated FJP

Computing the transitive closure matrix is the most complicated step of NRFJP algorithm since the fuzzy relation matrix is multiplied \( n - 1 \) times by itself to guarantee the transitive closure matrix. By realizing the multiplication process only \( \log_2 n \) times, computational complexity could be reduced [30].

For a fuzzy neighborhood relation \( A : X \times X \to [0, 1] \), a relation matrix \( R_{n \times n} \) can be obtained by max-min composition [24]:
\[
\tilde{R}_j = \max_k \{a_{ik}, a_{kj}, i, j = 1, \ldots, n, (6.1)\}
\]
and its transitive relation \( T \) is obtained by combining its successive compositions such as
\[
T = A \cup A^2 \cup \ldots \cup A^n \cup A^{n+1} \cup \ldots \ (6.2)
\]
Note that for reflexive and symmetric relation, the transitive closure is obtained in at most \( n - 2 \) compositions. If \( A : X \times X \to [0, 1] \) is any neighborhood
relation, there is such a positive integer \( k \) that after a finite number of compositions

\[
A \to A^2 \to A^4 \to \cdots \to A^{2^k} \to \cdots .
\]  

(6.3)

the following equations could be written [5]:

\[
A^{2^k} = A^{2^{k+1}} = A^{2^{k+2}} = \cdots .
\]  

(6.4)

Then transitive closure \( T = A^k \) of the relation matrix \( A \) is obtained after \( k = \lfloor \log_2(n-1) \rfloor \) compositions

\[
A \to A^2 \to A^4 \to \cdots \to A^k = T.
\]  

(6.5)

**Bifurcated FJP Algorithm.**

*Step 1.* Compute pairwise distances between elements: \( d_{ij} = d(x_i, x_j), i, j = 1, ..., n \) and denote \( d_{\text{max}} := \max_{i \neq j} \{ d_{ij} \}. 

*Step 2.* Compute the fuzzy neighborhood relation

\[
N_{ij} := 1 - \frac{d_{ij}}{d_{\text{max}}}, i, j = 1, ..., n.
\]

*Step 3.* Compute the transitive closure matrix \( T = \tilde{N} \).

*Step 4.* Sort the elements of the transitive closure matrix in decreasing order and form an array \( a_i, i = 0, 1, 2, ... \) by getting only one of the equal elements. Determine the \( a_i \)-level appropriate to the maximum change interval:

\[
z := \arg \max_i a_i,
\]

where \( \Delta a_i := a_i - a_{i-1}, i = 0, 1, ..., \).

*Step 5.* Call the procedure \( \text{Clusters}(z) \) in order to construct the clusters convenient to the \( a_i \)-level.

**7. Fuzzy neighborhood DBSCAN (FN-DBSCAN)**

It is functional to use fuzzy neighborhood function since the neighborhood membership values of the points with varying distances from core point are also different [18]. Actually, there is evidently a difference between the points concerned. On the other hand, for crisp case the points within the same neighborhood radius of core point are the same (Fig. 2).

Accordingly, using fuzzy neighborhood function instead of crisp neighborhood function would be more beneficial.

The mathematical formulation of such a view is given below and is plotted in Fig. 3 [18]:

\[
N_k(x, y) = \begin{cases} 
1 - \frac{d(x, y)}{d_{\text{max}}} & \text{if } d(x, y) \leq \varepsilon_1, \\
0 & \text{otherwise.}
\end{cases}
\]

(7.1)

The above equation and the figure are the fuzzified versions of Equation (2.1) and Fig. 1. In order to obtain global values independent from the range of data, for parameters of \( \varepsilon \) and \( \text{MinPts} \) of DBSCAN, the data are normalized and an \( \varepsilon \) value between \([0, 1]\) is obtained to suppress the scale-dependency problem [18]:

\[
x_k^i = \frac{x_i - \min_x}{\max_x - \min_x}, \quad k = 1, ..., m.
\]

(7.2)

Instead of the \( \text{MinPts} \) of DBSCAN a new global parameter is defined as follows:

\[
\text{MinPts} = \frac{\varepsilon_2 \cdot w_{\text{max}}}{\max_{i \in \Omega} w_i}
\]

(7.3)

where \( w_{\text{max}} := \max_{i \in \Omega} w_i \), and \( w_i \) is the neighborhood set cardinality of the point \( x_i \) for a fixed \( \varepsilon_1 \), i.e.,

\[
w_i = |N(x_i, \varepsilon_1)|
\]

where \( |N(x_i, \varepsilon_1)| \) indicates the fuzzy...
cardinality of the set \( \mathcal{N}(x_i; \varepsilon_1) \). The function \( \mathcal{N}_i \) could be any fuzzy neighborhood function and the result of the process could be different according to its selection. Similarly, \( \varepsilon_2 \) is obtained by
\[
\varepsilon_2 = \xi = \min_{w_{\max}} \varepsilon_1. \tag{7.4}
\]
For a constant \( \varepsilon_1 \), the fuzzy neighborhood set of point \( x \in \mathcal{X} \) is formed by:
\[
\mathcal{F}\mathcal{N}(x; \varepsilon_1) = \{ y \mid y \in \mathcal{X}, \mathcal{N}_i(y) \geq \varepsilon_1 \}. \tag{7.5}
\]
A fuzzy core point is obtained by the following equation:
\[
\text{card } \mathcal{F}\mathcal{N}(x; \varepsilon_1, \varepsilon_2) = \sum_{y \in \mathcal{N}(x; \varepsilon_1)} \mathcal{N}_i(y) \geq \varepsilon_2. \tag{7.6}
\]
Since different neighborhood sensitivities could be trimmed by using different neighborhood membership functions, transforming DBSCAN algorithm to the FN-DBSCAN algorithm and using the benefit of fuzzy sets theory is plausible. Consequently, the FN-DBSCAN method could be more robust to the scale and density variations within the dataset [18, 20, 28, 29]. FN-DBSCAN algorithm is applied to various areas including bioinformatics, footprint generation for wild animals, crime regions detection, etc. [17, 21, 22, 25].

**FN-DBSCAN algorithm:**

1. Specify parameters \( \varepsilon_1, \varepsilon_2 \) and \( f \) (fuzzy neighborhood function).

   Step 1. Set \( i = 0 \). Set the cluster assignment for all points as unassigned.

   Step 2. For each point \( p \) in the dataset:

   1. Calculate cardinality of \( p \) by applying \( f \) to all points within distance \( \varepsilon_1 \).

   2. If cardinality \( p \geq \varepsilon_2 \):

      a. Assign \( p \) as a fuzzy core point.

      i. Start a new cluster \( C_i \) and assign \( p \) to \( C_i \).

      ii. Find all points within \( \varepsilon_1 \) distance of \( p \) and add to seed set \( S \).

   3. For each point \( q \) in the \( S \):

      a. Assign \( q \) as fuzzy core point.

      b. Find all points within \( \varepsilon_1 \) distance of \( q \) and add to seed set \( S \).

   4. Set \( i = i + 1 \).

   Step 3. Mark all the points which remain unassigned as noise.

The deficit of FN-DBSCAN and DBSCAN algorithms is that they are unable to catch overlapped clusters. For example, suppose the data given in Fig. 4 [6]. DBSCAN and FN-DBSCAN algorithms cannot cluster the dataset correctly. Because, if the
8. Scalable DBSCAN (SDBDC)

Density Based Distributed Clustering (DBDC) describes a strategy to make DBSCAN scalable, despite its distributed running principle [13, 14]. DBDC assumes the existence of multiple sites with local data and the goal is to cluster the global sum of all databases. This is analogous to making the processing of one large dataset scalable by breaking it into smaller subsets [23].

In DBDC algorithm, DBSCAN is locally run at each site and the detected subset of the core points, called specific core points, is stored. The criteria for specific core points are (i) none of the specific core points are within \( \varepsilon \) distance of each other and (ii) all of the non-specific core points are within \( \varepsilon \) distance of a specific core point.

Each local set of specific core points are combined to create a global model. Then DBSCAN is run again with \( \text{MinPts} \) set to 2. The idea is that only core points are present, so two core points that are reachable define a larger theoretical optimum setting. The parameter \( \varepsilon \) should be set larger for the global model to account for cases where parts of a cluster only exist in one local model. Specific core points for this cluster would be greater than \( \varepsilon \) apart and would not define a cluster. Setting \( \varepsilon \) high results in a model where two or more specific clusters are combined into one.

9. Scalable fuzzy neighborhood DBSCAN (SFN-DBSCAN)

Scalable FN-DBSCAN (SFN-DBSCAN) combines the ideas of Online FCM and Scalable Density Based Distributed Clustering (SDBDC) [11, 13, 14, 23]. In Online FCM, the fuzzy centroids stored by clustering each of \( \gamma \) subsets effectively represent the density of the data set up to that point of the processing. Storing the weighted centroids allows the algorithm to be zero on the centers of the data set. SDBDC’s strategy is also similar; it saves the covered points and covering radius of each stored point, and integrates them at the end of the process [22]. SDBDC algorithm saves points that are not within \( \varepsilon \) distance of each other in order to evenly sample the data subset just like FN-DBSCAN does [18, 29].

In SFN-DBSCAN algorithm, the data is partitioned into subsets and points sampling the density from each subset are stored with weights [22]. The stored points from each subset are combined and a weighted version of FN-DBSCAN is run on these points to create clusters. The last step is to assign points from the entire data set to the defined clusters.

**Scalable FN-DBSCAN algorithm:**

**Step 1.** Set parameters \( \varepsilon_1 \) and \( \varepsilon_2 \).

**Step 2.** Break the dataset into roughly equal sized subsets that contain randomly selected points.

**Step 3.** Calculate the maximum number of points to retain from each subset \((\text{MaxRetainedPts})\) by dividing the number of points in each subset by the number of subsets.

**Step 4.** For each subset:

1. Calculate the local density of each point by computing the sum of the fuzzy membership function for all points within \( \varepsilon \) distance.
2. Sort the points according to their local density.
3. Add the densest point to \( \text{RetainedPtList} \).
4. While \( \text{Size(} \text{RetainedPtList} \text{)} \) < \( \text{Size(} \text{MaxRetainedPts} \text{)} \) add densest point to \( \text{RetainedPtList} \) that is outside the \( \varepsilon_2 \) radius of all points currently in the \( \text{RetainedPtList} \).

**Step 5.** Combine the points from each subset’s \( \text{RetainedPtList} \) into one dataset.

**Step 6.** Reduce \( \varepsilon_2 \) w.r.t. the number of subsets.

**Step 7.** Run the weighted version of FN-DBSCAN on the combined dataset.
Step 8. Find core points from the weighted dataset.
Step 9. Distribute core points and calculate membership of the entire dataset w.r.t. the core points from the weighted dataset.

In step 3, the number of points retained per subset is a percentage based on the number of subsets. In step 5, the number of points retained is at most the number of subset. The reduction of the value of \( \epsilon \) reduces the density required to form a cluster when there is a reduced number of sample points. This step improves the concurrence of cluster assignments in comparison with FN-DBSCAN. This is a similar concept to increasing \( \epsilon \_1 \) and reducing \( \epsilon \_2 \) has the same effect of decreasing the density requirements for cluster construction [22].

10. Conclusion

DBSCAN-based algorithms are generally successful in detecting irregularly structured clusters. From a clustering perspective, DBSCAN has precise properties since it easily discovers clusters with irregular shapes. The information of noise points is also very valuable, since these points could be the objects with characteristic properties. One important property of DBSCAN is its low computational complexity and efficiency in large spatial data sets. For all of the aforementioned reasons, DBSCAN has a growing popularity among researchers. A variety of different versions for these algorithms have yet been proposed which consider density variations within the clustering structure, or various approaches for constructing the neighborhood strategy.

In our study, we investigated the advantages and disadvantages of the current algorithms. Conceptually, fuzzy and crisp neighborhood based various clustering algorithms were examined and their characteristics are manifested. Bifurcated FJP algorithm was handled in order to alleviate the low-speed disadvantage of fuzzy neighborhood-based clustering algorithms. Scalable DBSCAN and Scalable FN-DBSCAN algorithms, which are able to run under various density of clusters were also mentioned.

In conclusion, we believe that this study will provide a compact tool to researchers by investigating the properties of the widely used crisp and fuzzy neighborhood based clustering methods in detail.

Acknowledgments

This work is supported by TUBITAK Grant No.111T273. The authors are grateful to Prof. Duan for providing dataset LDBSCAN.

References


