Information Sciences

Information Sciences 00 (2019) 1–37

Munec: A mutual neighbor-based clustering algorithm

Frédéric Ros[a,*], Serge Guillaume[b]

[a]*Laboratory PRISME, Orléans university, France*
[b]*ITAP, Irstea, Montpellier SupAgro, Univ Montpellier, Montpellier, France*

Abstract

It is expected for new clustering algorithms to find the appropriate number of clusters when dealing with complex data, meaning various shapes and densities. They also have to be self-tuning and adaptive for the input parameters to differentiate only between acceptable solutions. This work addresses this challenge. At the beginning mutual nearest neighbors are merged without any constraint until the number of groups including at least two items reaches a maximum. Subsequent mergings are only possible for mutual neighbor groups with a similar distance between neighbors. Finally, to manage more nuanced situations, heuristics that combine local density and distance are defined. The whole strategy aims to progressively consolidate the data representation structures. *Munec* requires some parameters. Most of them were integrated as constants and a single user parameter controls the process: the higher its value, the more constraints there are on the merging and the higher the number of clusters. Tests carried out using 2-dimensional datasets showed that *Munec* proved to be highly effective in matching a ground truth target. Moreover, with the same input configuration it can identify clusters of various densities, arbitrary shape and including a large amount of noise. These results hold for spaces of moderate dimension.

Keywords: mutual neighbors, single link, density, distance, clustering

1. Introduction

Data reduction plays an important role in data mining area and clustering is a popular way to achieve this goal. Clustering is probably the most extensively studied process in the pattern recognition community. It is considered as the reference tool for unsupervised classification as it is useful to summarize and understand the data. An operational definition of clustering can be stated as follows: Given a representation of objects, find the groups based on a measure of similarity such that the similarities between objects in the same group are high while the similarities between objects in different groups are low. After more than 40 years of research, several popular clustering approaches now exist. They differ in

*Corresponding author
Email addresses: frederic.ros@univ-orleans.fr (Frédéric Ros), serge.guillaume@irstea.fr (Serge Guillaume)

the definition of a cluster and the processes used to partition the data. Clustering remains a challenging issue for the pattern recognition community [24] as no universal algorithm yet exists. Several proposals aim to improve the popular approaches by solving specific cases to address well-identified drawbacks. The most representative example is the *k-means* algorithm: a large number of methods have been proposed to reduce its sensitivity to initialization or to make it faster. As datasets grow in size and dimension, resource management, runtime and memory space become important.

Three basic notions of what a cluster is lead to three main types of algorithms. If a cluster is defined by its center and a basin of attraction then distance is the central concept. It is also possible to define a cluster as a dense area separated from another cluster by a sparsely populated zone; in this case, density is the key idea. Finally, a third definition is based on a set of connected points, in which case neighborhood is of prime concern.

The best-known representative of distance-based clustering algorithm is the *k-means* one [19]. In the partitioning around medoids (PAM) version [29] the median item substitutes the centroid as the group representative. A graph implementation is called CLARANS [34]. The main drawback of these algorithms is that they are limited to spherically shaped clusters. Density-based algorithms, such as *DBSCAN* [13] and its derivatives [38, 2], or CLIQUE [10], are able to manage arbitrarily shaped clusters. They mainly differ in the density estimation method: grid [35, 1], kernel [30] or neighborhood [11, 5]. When the parameter for density estimation is static, these algorithms are unable to deal with density variations. Density and distance methods are hybridized [18, 40], especially through agglomerative approaches such as Chameleon [28], BIRCH [49] or DENCLUE [21], or split and merge frameworks [45]. The results are quite sensitive to the setting as only a few split and merge criteria can adapt to the data. The trend is to design increasingly complex algorithms restricted to data of moderate size, with the noticeable exception of [40].

Neighborhood is a transversal notion that can be used either in distance (volume) or density (number of points) based algorithms [9, 47], or as the basis of the algorithm [25, 17, 32]. Its definition is usually a highly sensitive parameter. To eliminate this parameter, the mutual nearest neighbors can be used. With the restriction to the first mutual nearest neighbor no threshold, whether on distance or on the number of neighbors, is required.

This concept may be useful if one considers a more nuanced cluster definition, based on internal structure, which can be called texture. In the field of image analysis [16], texture is not formally defined even if widely used. It can be based on structural patterns or statistical metrics. This concept is used here in a similar, intuitive, way. Neighboring and distance are of major importance for cluster separation. The human eye and brain are capable of combining the two in order to identify *natural* clusters. When the between group distance is high with respect to the within group organization, there is no ambiguity: distance is of prime concern. But when the between group distance decreases, the internal structure becomes the main criterion: two groups are well identified if they have a different internal structure or texture, even if they are close to each other. The internal structure is characterized by density, or distance between neighbors. This is obviously a matter of degree, and sometimes various configurations are acceptable.

Clustering algorithms have to deal with complex data, meaning different types of at-

tributes, various shapes and densities, and must include outlier and noise management. The expectations for new proposals can be grouped in two categories. The first one concerns the dimension and volume of 21^{st} century databases. It involves the curse of dimensionality and algorithm scalability. The second one is to identify data structure without a priori information. Clearly, approaches driven by the number of clusters are nowadays out of date as this number is usually unknown. More generally, knowing that there exists a set of parameters able to handle a particular situation does not help when the objective is knowledge discovery. The challenge for new algorithms is to be sufficiently self tuning and adaptive for the result to be acceptable whatever the input parameters, or put differently, for the same set of parameters to be able to manage various datasets. In that sense, the input parameters only differentiate between acceptable solutions and allow the user to select a more or less detailed representation of her data.

This work addresses the second type of challenge. The main ambition of the proposal concerns the discovery of clusters with different shapes and densities without laborious tuning procedures.

The proposal assumes that a cluster is characterized by the distribution of its neighboring patterns: there is no sharp change between neighboring patterns within the same cluster and the difference between two clusters stems from their inner spatial arrangement and their proximity. The mutual neighbor concept is useful for the inner structure description as well as for characterization of the between group proximity.

The proposed algorithm[1] is based on a iterative process that merges mutual nearest neighbors. The hierarchical process has to be stopped before all the items are grouped in a unique cluster. As no index is meaningful at the beginning, the first merging steps are only controlled by the number of sub-clusters, to yield a skeleton of the data structure. Then, two distinct stages are proposed. The first one involves the similarity of distances between neighbors, in each group and between groups. In a second phase, three heuristic conditions are introduced in order to discriminate between more nuanced situations. They are based on a combination of several notions such as distances between mutual neighbors, nearest neighbor group of higher size and local neighborhood density. They aim to provide the algorithm with self-control abilities to become as generic as possible with a reduced number of parameters.

In the following, the main neighborhood-based approaches are studied in Section 2 and their limits are highlighted using examples. The definitions required by the key ideas of the proposal are introduced in Section 3 and the process is illustrated using a toy example. Then the whole algorithm is detailed in Section 4. The heuristics are motivated and illustrated with data already used in the literature. In Section 5 the proposal is compared to alternative approaches using several datasets that illustrate the diversity of situations a clustering algorithm has to cope with. Finally the main conclusions are summarized in Section 6.

[1] A sample code is available at: `http://frederic.rosresearch.free.fr/mydata/homepage/`

2. Neighborhood-based clustering: literature review

Although developed independently the proposal draws on several concepts and ideas, such as neighborhood, peak density or single-link distance, that have proven relevant to cluster data for which classical techniques fail. This section presents a selected review of previous work.

The neighborhood concept has been used in many clustering techniques. There are two main ways to define a neighborhood. The first one counts items that fall within a space, usually a hypersphere of radius r centered on the point. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of items. The neighbors of x are the items located at a distance less than r:

$$N_r(x) = \{x_i \mid ||x_{(i)} - x|| < r\} \tag{1}$$

The second one is the set of nearest neighbors. Let $\{x_{(1)}, x_{(2)}, \ldots, x_{(n-1)}\}$ be the permutation of the elements of $X \setminus \{x\}$, such that:

$$||x_{(1)} - x|| \leq ||x_{(2)} - x|| \leq \ldots ||x_{(n-1)} - x|| \tag{2}$$

The k-nearest neighbors of the x item are the set defined as:

$$N^k(x) = \{x_{(1)}, x_{(2)}, \ldots, x_{(k)}\} \tag{3}$$

The *Chameleon* approach [28] was the pioneer and is still the basis of, or a source of inspiration for, recent developments. The algorithm uses a neighborhood sparse-graph for item representation. Two vertices, x and y, are connected by an edge if:

$$x \ \mathbf{Cham} \ y \iff x \in N^k(y) \ OR \ y \in N^k(x) \tag{4}$$

The edges are valued by the similarity between the considered items. In this way a connected sub-graph corresponds to a cluster. The algorithm includes two steps. First the graph is partitioned into many sub-clusters according to a min-cut criterion [8]. In the second step, sub-clusters are iteratively merged based on their similarity, defined as a combination of relative interconnectivity (RI) and relative closeness (RC). Using relative values instead of absolute ones enables an adaptive modeling. The similarity between sub-clusters c_i and c_j is defined as:

$$Sim(c_i, c_j) = RI(c_i, c_j) \cdot RC(c_i, c_j)^\alpha \tag{5}$$

$\alpha > 0$ is used for weighting the relative closeness with respect to the relative interconnectivity.

The relative interconnectivity is the absolute interconnectivity, the sum of the weights of the edges in the two clusters, normalized by their internal connectivity. The relative closeness is defined in a similar way with the average weight. The relative closeness discourages the merging of small sparse clusters into large dense ones, and the resulting cluster has a uniform degree of closeness among its items. The two parameters of the algorithm are α (in the original paper $\alpha = 2$) and the number of neighbors (10). No clue is given to choose this influential parameter. There is no clear insight about noise management.

2.1. The DensityPeaks algorithm

A recent study [40], referred to as *DensityPeaks* hereafter, is based on the idea that cluster centers are characterized by a higher density than their neighbors and by a large distance from items with a higher density. The local density, ρ, is estimated by the sum of distances of neighbors included in the r-hypersphere (Eq. 1) and the distance, δ, is the minimum distance to any higher density point. The distance to the highest density points is set at the maximum distance between two points.

The 2D-plot ρ-δ allows for the identification of centers and outliers. The former have high values on the two axes, while the latter are characterized by a low density and a high distance. The number of clusters can be found using the Elbow method [42] applied to the values of the $\rho\delta$ product plotted in decreasing order.

The main parameter is the radius that defines the neighborhood. As a rule of thumb, this paper proposes to choose the value that yields an average number of neighbors between 1 and 2 % of the data. This algorithm is already popular. It has been recently applied to real world problems [43] and has undergone various improvements. Recent studies aim to better select the neighborhood parameter [33, 44] that has a strong influence on the results, or to improve the decision making process thanks to graph properties [45, 9], beyond the original rule based on the $\rho\delta$ product.

This algorithm proved to be relevant for various datasets including popular difficult benchmark problems[2]. It has, however, some fundamental limitations when the local densities are rather homogeneous and not easily defined as peaks. In such cases, where the neighborhood is large, different peaks can be identified in a single cluster which is then split into different sub-clusters. This is the case for spirals with uniform densities.

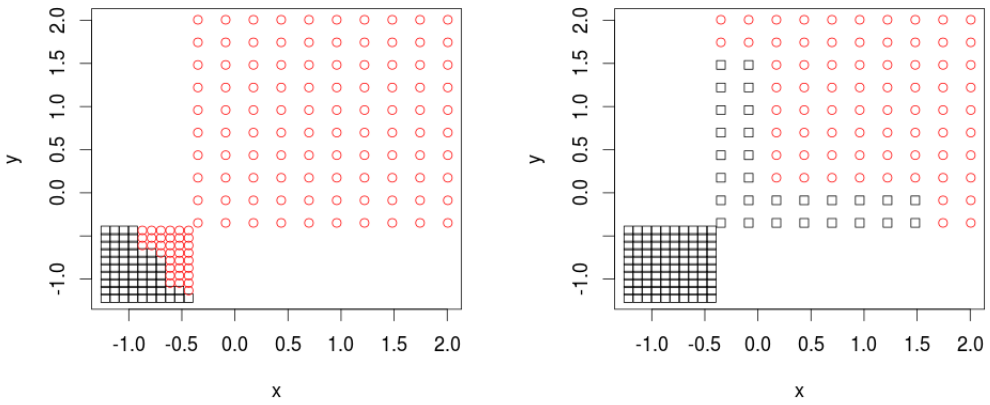

Figure 1. Illustrative example with density peak clustering [40]: the two clusters are not correctly identified.

[2]https://cs.joensuu.fi/sipu/datasets/, [15]

The limitations of the algorithm are illustrated in Figure 1. The data are organized in two clusters of the same size, 100 points, and different densities. Even if the local density is estimated using a Gaussian kernel the two clusters cannot be correctly separated whatever the radius parameter. The resulting clusters are shown in different colors in the figure. For small-medium values (average number of neighbors less than 3 % of the data), the first two dominant peaks are located in the densest cluster (left part of the figure). Using a larger radius, the first two dominant peaks may be located in the two clusters. But, in this case, some points of the less dense cluster have a higher density than expected and are attracted by their denser neighboring cluster, as shown in the right plot of Figure 1.

2.2. The SCDOT algorithm

The Spatial Clustering with Density-Ordered Tree (*SCDOT*) [6] method is based on the two previously described methods. Cluster centers are also assumed to be density peaks that have a relatively large distance from higher density peaks. Local density and distance are estimated in the same way as in [40]. The first difference is that there is only one highest density point. If there are several maxima, a small value, randomly chosen, is added to one of them. The other difference is in the local density estimation: in *DensityPeaks* a radius parameter is used whereas in *SCDOT* the k nearest neighbors are used. A graph is constructed, as in *Chameleon* clustering [28], but with an additional constraint to yield a tree. A node is connected to only one other node, i.e. its nearest neighbor of higher density. The edge valuation is the same as in [40]. Cluster centers are recognized as points for which the edge value is larger than the typical nearest neighbor distance. They are detected in the distribution using the box-plot parameters, based on the interquartile range (IQR). The outliers are labeled as *mild* (value higher than $1.5 \times IQR + Q3$, the third quantile) or *extreme* (value higher than $3 \times IQR + Q3$).

The algorithm has a unique parameter, the number of clusters.

The tree is partitioned according to the edge values. The extreme outlier-edges are first removed, yielding a number of sub-trees. If this number is below the desired number of clusters, then mild outlier-edges are removed. If the number of sub-trees remains less than the number of clusters then the edge with the largest value is removed until the desired configuration is reached. Sub-trees composed of a single node are considered as noise. The last step of the algorithm consists in merging the sub-clusters in a way similar to the *Chameleon* method. Instead of maximizing the similarity, the authors minimize the disconnectivity, according to the formula first published in [31], where $|c_i|$ stands for cluster i cardinality:

$$dis(c_i, c_j) = \frac{\sum_{i \in c_i} \sum_{j \in c_j} \frac{b_{ij} + b_{ji}}{d(i,j)}}{|c_i| \; |c_j|}, \; b_{ij} = \begin{cases} 1 & if \quad j \in N^k(i) \\ 0 & otherwise \end{cases} \tag{6}$$

The disconnectivity of two clusters is defined as the inverse of the sum of distances in the merged cluster. A penalty is added when one point does not belong to the set of the nearest neighbors of the other. The disconnectivity concept is then extended to the whole partition and the goal is to minimize its value by merging the less disconnected clusters

until the desired number of clusters is reached. The algorithm starts with a small number of neighbors, $k = 4$. The disconnectivity of two clusters may be zero with a small size neighborhood, i.e. when $b_{ij} = b_{ji} = 0$, and no merging is possible. To get the desired number of clusters, the split and merge loop is repeated with increasing values of k.

The algorithm is driven only by the number of clusters, c. As this number is usually unknown, this is a strong limitation of the algorithm: it does not contribute to structure identification. Even when this number is known, *SCDOT* may fail to identify the relevant structure, as illustrated in Figure 2.

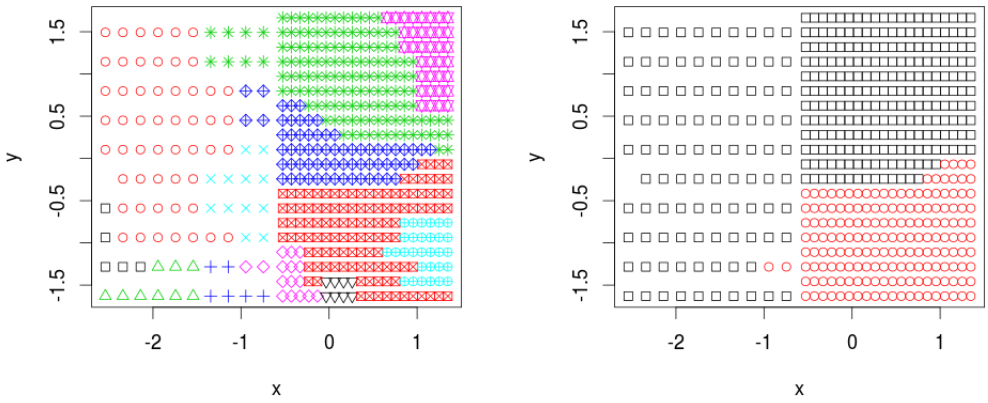

Figure 2. Illustrative example with the *SCDOT* algorithm [6]: two sub-clusters include points from the two clusters (left) giving an unexpected final partition (right).

The goal is to identify the two clusters with different densities: the less dense cluster is the set of points with an abscissa less than -0.6. The algorithm was run with different values of c, the number of desired clusters. For $c = 2$ (resp. 3), the number of sub-clusters after the split step is 17 (14). Using these values the merging is not efficient and yields 14 (15) final groups, meaning that the algorithm cannot yield the desired number of clusters. Figure 2 shows the sub-clusters (left plot) and the final partition after merging (right) when the algorithm is run with $c = 4$. The left plot shows that two sub-clusters include points belonging to the two different clusters. The 'natural' border at the abscissa of -0.6 is not respected in the upper part of the plot, $y \approx 1.1$ with the green star symbol, and in the lower part of the left plot, $y = -1.2$ with the pink diamond symbol. Then the merging step yields the unexpected final partition shown in the right part of Figure 2.

2.3. The MutualClust algorithm

The concept of mutual nearest neighbors was introduced in 1978 [17] in the same period as the pioneering method of Shared Nearest Neighbors [25]. The authors' motivation "comes from real life observations. Two persons A and B group together as close friends if they

mutually feel that the other is his closest friend. If A feels that B is not such a close friend to him, then even though B may feel that A is his closest friend, the bond of friendship between them is comparatively weak. If each feels that the other is not his friend, then the two do not group together as friends. In other words, the strength of the bond of friendship between two persons is a function of mutual feelings rather than one-way feeling. Similarly two samples form a cluster if they are mutually near neighbors rather than simply near neighbors" [17].

The mutual neighborhood concept was investigated for the development of recent clustering algorithms [4] or cluster validation indices [31, 32]. In [22], the algorithm is similar to the *Mountain Method* proposed by Yager [46] but the local density is assessed using mutual neighboring. The concept is used to find points with a local maximal density that are later merged to form the clusters [5]. Several existing algorithms referred to as shared nearest neighbor techniques are also based on this concept.

The objective of the pioneering work was to find mutual homogeneous clusters, with ideas that are still inspiring. This algorithm is referred to as *MutualClust*. Two items, x and y, are mutual nearest neighbors if:

$$x \ \mathbf{Mnn} \ y \iff x \in N^k(y) \ AND \ y \in N^k(x) \tag{7}$$

The mutual neighborhood strength is quantified by the mutual neighborhood value:

$$mnv(x,y) = \begin{cases} e+f & if \quad x \ \mathbf{Mnn} \ y \\ \infty & otherwise \end{cases} \tag{8}$$

where x is the e^{th} nearest neighbor of y, and y is the f^{th} nearest neighbor of x, $1 \leq e \ (f) \leq k$.

Mutual homogeneous clusters are based on the mutual neighborhood values. Let $(x_p, x_q) \in c_i$ be a merged pair of items,

$$M = \max_{(x_p,x_q)\in c_i} mnv(x_p,x_q), \qquad D = \max_{(x_p,x_q)=M} d(x_p,x_q) \tag{9}$$

c_i is said to be mutually homogeneous if:

$$\forall (x_p,x_q) \in c_i, \quad \forall y \notin c_i, \quad mnv(x_p,y) \geq M, \quad mnv(x_q,y) \geq M, \quad d(x_p,y) \geq D \quad and \quad d(x_q,y) \geq D \tag{10}$$

If this property stands for all the clusters in the partition, the partition is said to be mutually homogeneous.

The merging is done according to increasing values of mnv and, in the event of equality, increasing values of d, until a desired number of clusters is reached. No specific procedure is proposed for noise or outlier management and their presence is a potential source of failure. The algorithm also fails in situations without noise as illustrated in Figure 3 with three clusters of different density. The one with intermediate density is in the upper part of the plots, points with an ordinate higher than 0.5. The space covered by the three

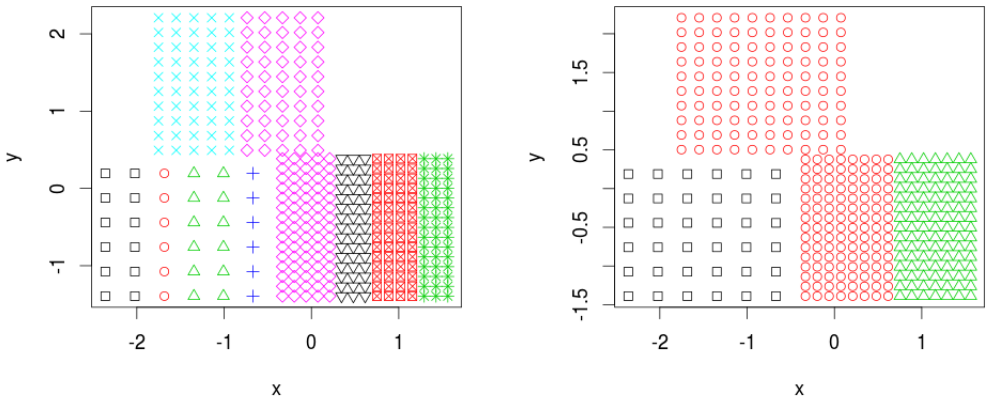

Figure 3. Illustrative example with the mutual neighborhood approach [17]: clusters are mixed in some sub-clusters at an intermediate stage (left) yielding an unexpected final partition (right).

clusters, from the sparsest to the densest, are the following intervals for the (x, y) coordinates: $([-2.2, -0.5], [-1.5, 0.2])$, $([-0.5, 1.7], [-1.5, 0.5])$ and $([-1.6, 0.1], [0.5, 2.2])$.

The algorithm *MutualClust* was run to obtain $c = 3$ clusters. The neighborhood size, k, was increased to get the desired number of clusters. The final number of groups for $k = 2$ to 5, was respectively 232, 87, 19 and 3. The configuration with $k = 5$ is plotted in Figure 3. The left part of the figure shows an intermediate step of the process: the number of sub-clusters is 9, and all are valid as all of them include only points belonging to the same cluster. At this stage, different candidates are possible for the merging, with no difference according to the mutual neighborhood value criterion. The algorithm therefore gives the preference to the minimum distance which yields an unexpected merging: the red-circle sub-cluster, points located at $(-1.7, [-1.5, 0.2])$, is merged with the cyan-cross one, points located at $([-1.8, -0.9], [0.5, 2.2])$, instead of being merged with the twelve black-square points at $([-2.3, -2], [-1.5, 0.2])$ or with the twelve green-triangle points at $([-1.3, -1], [-1.5, 0.2])$. This finally gives the unexpected configuration plotted in the right part of Figure 3.

This short review shows that even if some progress has been made, improvements are still needed. Not only do the studied algorithms fail in some situations, but they can also be difficult to tune in order to get the expected result.

3. *Munec*: key ideas

The algorithm is based on the mutual nearest neighbor notion and the key idea is quite simple: it is an iterative process that merges mutual nearest neighbors. This concept is extended here to the groups to be merged: they must be mutual nearest sub-clusters. This merging process can be carried out until all the items are grouped in a unique cluster. Let

us denote $x, y, \ldots$ the elements of a data set and let us denote $c_1, c_2, \ldots$ the clusters. Initially we have n clusters, each containing a single element, i.e. $c_i = \{x_i\}$, $i = 1, ..., n$.

At the beginning of the process, the mutual nearest neighbors are merged until the number of groups including at least two items reaches a maximum. The result is a skeleton of the data structure. Then, a similarity index is computed from the distances between neighbors in each group and the between group distance. These three distances must be similar for the groups to be merged.

In a second phase, heuristics are proposed in order to discriminate between more nuanced situations and to stop the algorithm. The first one is based on the homogeneity index. It is derived from the similarity index but the merging is oriented: a sub-cluster can only be merged with its mutual neighbor if it is of higher density. This idea,i.e the nearest group of higher density, is inspired from the *DensityPeaks* algorithm. Only two distances are used; the inner distance of the less populated group is not taken into account. A threshold on this index is progressively decreased in order to consolidate the existing structures, avoiding the setting of a sensitive parameter. The second one combines distance and local mutual neighborhood. A threshold is defined on the number of mutual neighbors between the two groups to be merged weighted by the homogeneity index: the merging is canceled when the number of mutual neighbors in the neighborhood of the connection is high with respect to the homogeneity index. A high value of this product is likely to reveal a discontinuity. The threshold on the product is not a parameter; the constant value is part of the algorithm. The last heuristics aims at texture discrimination: the difference in density between the two groups is weighted by the homogeneity degree. The unique parameter of the algorithm is the threshold defined on this heuristics.

The control mechanisms are detailed in Section 4. In this section, the mutual neighbor concept, defined for items, is extended to groups. Two cluster characteristics are proposed to summarize the information and they are easily updated after a given merging. The merging process is illustrated using a toy example. It is shown that none of these characteristics can be used as a stopping criterion.

3.1. Mutual nearest neighbor clusters

Two clusters, c_l and c_m, are mutual nearest neighbors if there exist $x \in c_l$ and $y \in c_m$ where x and y are mutual nearest neighbors when the neighbors in their respective groups are not considered.

The distance between the two neighbors is the single-link distance between the mutual nearest neighbor clusters:

$$d_{l,m} = \min_{x \in c_l, y \in c_m} d(x, y) \tag{11}$$

A cluster is characterized by:

- n: the number of distances between two neighbors. The total number of points is $n + 1$;
- d: the mean distance between two neighbors.

This distance is also called the inner distance of the cluster, in contrast with the between group distance. It is noted d_i for cluster i.

3.1.1. Hierarchical merging

At each step the process merges all the mutual nearest neighbor clusters. At the beginning, there are only isolated points, for which $n = 0$ and $d = 0$. First only mutual nearest neighbor pairs, (x_i, x_j), are connected: $n = 1$ and $d = d(x_i, x_j)$. There are two kinds of clusters, isolated points and pairs. At the next step, pairs can be connected together or a point may be connected to a pair.

Dealing with mutual nearest neighbors avoids setting the number of nearest neighbors to be considered in this kind of algorithm. This is important as neighbor-based algorithms are known to be sensitive to this value [48].

It is worth mentioning that several mergings can be done at each step. This is an important difference with respect to algorithms that only merge the pair that best satisfies the merging criterion, in this case the two clusters for which the single link distance is minimum. This way, the data structure is progressively captured whatever the distance between mutual neighbors in the different parts of the input space.

Merging update:. When two clusters, 1 and 2, are merged, the internal descriptors, n and d, become:

$$n = n_1 + n_2 + 1, \qquad d = \frac{1}{n}(n_1 d_1 + n_2 d_2 + d_{1,2}) \tag{12}$$

3.1.2. Merging process illustration

The whole merging process is illustrated with the synthetic data shown in Figure 4. The pairwise distance matrix, based on the Euclidean distance between points, is given in Table 1. Connections link mutual neighbors, either points or sub-clusters. A different color is used for the different merging steps.

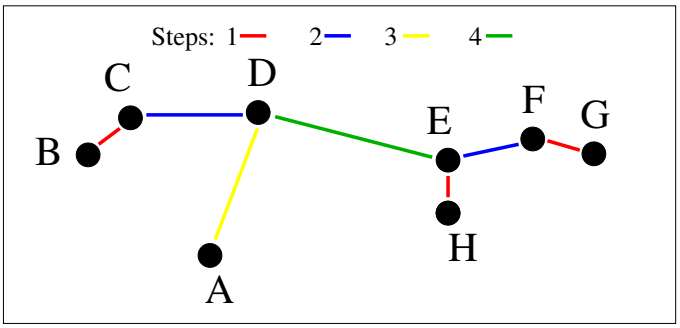

Figure 4. Mutual neighbor merging: illustrative example

The clustering process is as follows. At each step the new clusters, with their characteristics n and d, and the current partition, P, are given:

Table 1. Pairwise distance matrix for the data plotted in Figure 4

	A	B	C	D	E	F	G
B	24						
C	25	5					
D	22	23	19				
E	42	61	54	31			
F	58	77	70	46	11		
G	68	88	81	58	22	6	
H	40	67	57	35	4	16	24

1. Three pairs of mutual nearest neighbors
   BC (n=1, d=5)
   EH (1, 4)
   FG (1, 6)
   P={BC, EH, FG, A, D}
2. EH and FG are mutual nearest neighbors. E and F are mutual nearest neighbors considering the groups they are not part of, d(E,F)=11.
   BC and D are mutual nearest neighbors (D and C), d(C,D)=19.
   BCD (2, $d = \frac{1}{2}(5 + 0 + 19) = 12$)
   EFGH (3, $d = \frac{1}{3}(4 + 6 + 11) = 7$)
   P={EFGH, BCD, A}
3. BCD and A are mutual nearest neighbors (D and A), d(A,D)=22.
   ABCD (3, $d = \frac{1}{3}(2 \times 12 + 0 + 22) = 15.33$)
   P={ABCD, EFGH}
4. The last two groups are merged according to the distance between D and E, d(D,E)=31.
   ABCDEFGH (7, $d = \frac{1}{7}(3 \times 15.33 + 3 \times 7 + 31) = 14$)

This is a hierarchical process and, as illustrated in the example, it can be carried out till all the items are grouped into a single cluster, unless a meaningful stopping criterion is found.

3.1.3. The cluster characteristics cannot be used as a stopping criterion

Let $\overline{d}$ be the average of d over all the clusters of the partition. To be used as a stopping criterion this index would have to show a distinctive breakpoint in its evolution curve and the latter would have to be meaningful with respect to stopping the process. Unfortunately, this is not the case.

Property: $\overline{d}$ is monotonically increasing.

The proof is as follows.

A given merging modifies only two clusters, e.g. 1 and 2. If $\overline{d}$ could decrease, it would mean:

$$\frac{n_1 d_1 + n_2 d_2 + d_{1,2}}{n_1 + n_2 + 1} < \frac{n_1 d_1 + n_2 d_2}{n_1 + n_2} \tag{13}$$

The condition for $\overline{d}$ to decrease is:

$$d_{1,2} < \frac{n_1 d_1 + n_2 d_2}{n_1 + n_2} \tag{14}$$

This is not possible at the second step of the process. If some point, C, were closer to one point in the pair (A, B), e.g. A, then the mutual nearest neighbors would have been the pair (A, C) instead of (A, B). This is not possible between two pairs of mutual nearest clusters, such as (E, H) and (F, G) in Figure 4, for the same reason.

As the computation of the mean distance in the cluster, d, involves only neighbors, and not all the pair distances, if the distance between the two clusters were lower than the mean distance between neighbors the merging would have been done before.

None of the cluster characteristics can serve as a stopping criterion: the average distance, $\overline{d}$, and the number of distances between two neighbors, n, are monotonically increasing. To avoid unexpected mergings in more complex data, these basic ideas have to be complemented by adaptive mechanisms. They are introduced in the following section.

4. *Munec*: the proposed algorithm

The algorithm is made up of two distinct steps. The first one structures the data in subgroups that are small and numerous enough to ensure that their elements belong to the same cluster and the second one uses new heuristics to yield the final partitions.

4.1. Mutual nearest neighbor controlled step

At the beginning of the algorithm, only mutual neighborhood (without any restriction) is taken into account. This preliminary step, without any other control, is required to start the process with isolated points or pairs of points, for which any index would be meaningless.

In such an agglomerative process, the number of clusters larger than a threshold higher than 1 starts from zero and ends at one when all the items are grouped in a single cluster. This number grows monotonically until a maximum is reached and then decreasing untill the end.

This preliminary step is carried out until the maximum is reached for the number of clusters including at least two items. The smallest value, 2, is chosen as the objective is to build a skeleton of the data structure.

Once this skeleton has been defined, a distance-based group similarity index is introduced. The similarity index is only based on the three distances, without accounting for the cluster cardinalities. It is computed as:

$$s = \sqrt{s_{i,j}\ s_{j,i}} \tag{15}$$

where $s_{i,j} = \dfrac{min(d_i, d_{i,j})}{max(d_i, d_{i,j})}$, and $s_{j,i}$ is defined in the same way with respect to d_j.

The closer the distances, $d_i, d_j, d_{i,j}$, the higher the index and the more suitable the merging of the considered sub-clusters. In this preprocessing step, a high threshold value,

$SecureThres = 0.95$ is used. This value can be interpreted as follows: At this stage of the process the between group distance is larger than the two within group ones. To consider an extreme situation, let the between group distance be comparable to one of the inner distances, e.g. $d_{i,j} = d_i$. The merging condition, $s < 0.95$, requires $s_{j,i} < 0.90$, meaning that $d_{i,j} < 1.1\ d_j$. Only groups for which the distance between the mutual neighbors is less than 10% larger than the average of the distance between neighbors in each group are allowed to be merged. This is a restrictive constraint.

The first stage of the algorithm is described in Algorithm 1.

Algorithm 1 *Munec*: the preprocessing part of the algorithm

```
Input: X, dataset
Output: S, set of sub-clusters
S = X, |S| = n
while (The number of sub-clusters of size at least 2 is growing) do
   for all (i, j) ∈ S × S do
      if (i Mnn j) then
         Merge i and j. |S| = |S| − 1
      end if
   end for
end while
SecureThres = 0.95, Merge=true
while (Merge==true) do
   Merge=false
   for all (i, j) ∈ S × S do
      if (i Mnn j AND s_{i,j} > SecureThres) then
         Merge=true. Merge i and j. |S| = |S| − 1
      end if
   end for
end while
return S
```

Figure 5 illustrates the results of the process with data structured in clusters of different shapes including a large amount of noise. The original dataset includes 8000 2D items widely used by the community [28]. A sample of 754 items, yielded by the *DENDIS* algorithm [41] with a 0.002 granularity, is used.

The left part shows the number of sub-clusters (red line with circle) and the number of sub-clusters larger than $n/100 = 7$ according to the value of the similarity index, s. The first curve is monotonically decreasing while a maximum can be observed in the second one. Tests on various datasets proved that automatic procedures based on the location of this maximum are not robust enough, as they still yield unexpected mergings.

Until $s = 0.7$, the result is as expected as shown in the central part of Figure 5. But when the threshold is slightly decreased, $s = 0.66$, two sub-clusters are merged in the same group (the light blue points in the right part of the figure). It seems difficult to deduce the

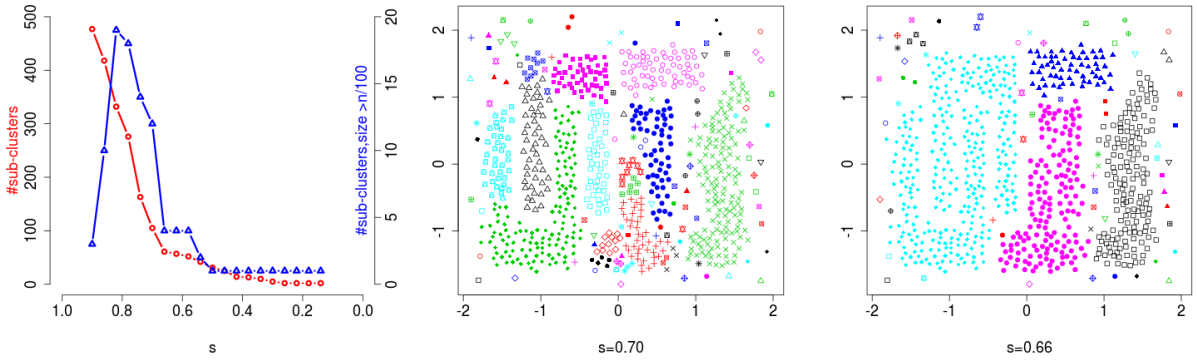

Figure 5. After the mutual neighborhood connection phase: the left plot shows the number of sub-clusters (red) and the number of sub-clusters with at least $n/100$ points (blue) for different values of s in two y axes. The central and right plots show the data partition for two values of s. The axes are the x and y coordinates.

suitable threshold value from the evolution curve plotted in the left part of Figure 5.

In this particular case, with nested shapes and noise, 0.7 would be an acceptable value for the threshold. This reinforces the choice of $SecureThres = 0.95$.

The first step of the algorithm aims to connect clusters based on their mutual neighborhood as introduced in Section 3. This step is easy to understand, and is controlled only by a cautious threshold for the similarity index. It yields a skeleton of the data structure which is the starting point of a smarter algorithm.

4.2. A second step based on new heuristics

While the first step only aims to connect mutual neighboring groups with similar between neighbor distances, density is now taken into account. The similarity index, see Eq. (15), is adapted as the homogeneity degree. There are three main differences between these two indices. First, for a pair of sub-clusters, (i, j), the homogeneity degree is positive if and only if j is the nearest neighbor of i, and if it is of higher size. Second, it is no longer required for i to be the nearest neighbor of j. Third, the distance between neighbors of the less populated group is not taken into account. The homogeneity merging degree is computed as:

$$h_{i,j} = \begin{cases} \dfrac{min(d_j, d_{i,j})}{max(d_j, d_{i,j})} & if \quad j = N^1(i) \quad and \quad n_j > n_i \\ 0 & otherwise \end{cases} \tag{16}$$

As in the *DensityPeaks* approach, when the homogeneity degree is high enough the two groups are merged to yield a bigger sub-cluster. The main differences with the *DensityPeaks* algorithm are the gradual consolidation of the final partition, thanks to an evolving merging threshold, and the restriction to the nearest neighbor, as shown in Eq. (16), making the algorithm free of the distance parameter.

Moreover three complementary tests to cope with the diversity of situations are introduced to avoid unsuitable mergings. The overall algorithm is described in Algorithm 2.

Algorithm 2 *Munec*: the heuristic part of the algorithm

1: Input: S, set of sub-clusters, u
2: Output: S, new set of sub-clusters
3: $SecureThres = 0.95$, $LowThres = 0.15$, $h_{th} = 0.8$
4: **while** $(h_{th} > LowThres)$ **do**
5: Merge = **true**
6: **while** Merge == **true do**
7: Merge = **false**
8: **for all** $(i, j) \in S \times S$ **do**
9: **if** $(h_{i,j} > SecureThres)$ OR $(C1(h_{th})$ AND $C2$ AND $C3(u))$ **then**
10: Merge=**true**. Merge i and j. Update Neighbors.
11: **end if**
12: **end for**
13: **end while**
14: $h_{th} = h_{th} - 0.05$
15: **end while**
16: return S

The three conditions, to be detailed and motivated in the sequel, are:

- C1: $h_{i,j} > h_{th}$ where h_{th} is a threshold value
- C2: $(1 - h_{i,j})\ Mnn(x, y) < \max(0.5 \cdot \min(v_i, v_j), 2)$,
  with $V_i = \{x_i \in i \mid d(x, x_i) < 2 \cdot d_{i,j}\}$, $v_i = |V_i|$ and $Mnn(x, y) = \sum_{x_i \in V_i, y_j \in V_j} 1_{(x_i \ \mathbf{Mnn}\ y_j)}$
- C3: $(1 - h_{i,j})\ \dfrac{min(v_i, v_j)}{max(v_i, v_j)} > u$, e.g. $u = 0.06$,

The items $x \in i$ and $y \in j$, used in conditional tests $C2$ and $C3$, are the mutual nearest neighbors of the two groups, the ones such that $d_{i,j} = d(x, y)$.

A *SecureThres* is still used in this part of the algorithm, meaning that when the homogeneity between a sub-cluster and its nearest neighbor, if it is bigger in size, is very high, then no further test is required for the merging (line 9). This is likely to occur at any stage of the process, whatever the current h_{th} value.

Condition 1. The first condition is the most important one. The restrictions it imposes on the merging, based on the homogeneity merging degree, define the originality of the proposal.

The mutual nearest neighbor connection process can be safely carried out until $s = 0.7$ as illustrated in the central plot of Figure 5. Thanks to $C1$ the first unexpected merging is delayed until $h = 0.5$ as illustrated on the left part of Figure 6 with the red group.

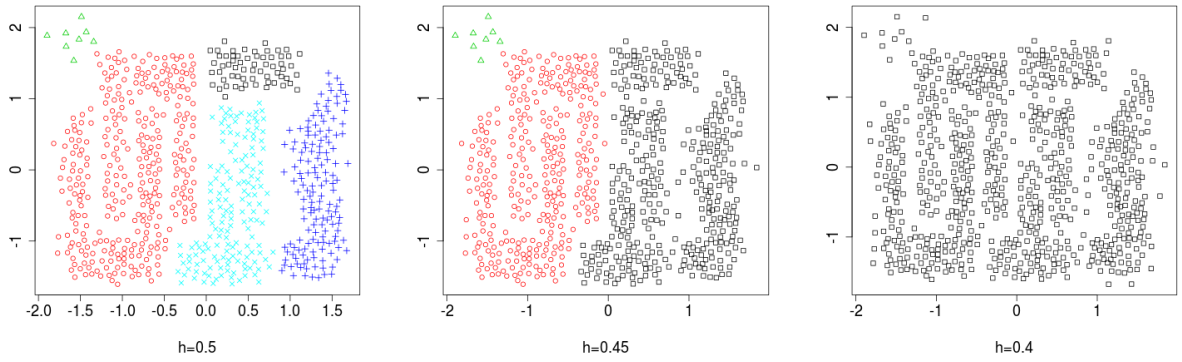

Figure 6. C1: Resulting partitions for three different thresholds on the homogeneity merging degree, h. The axes are the x and y coordinates.

As the threshold on the homogeneity merging index decreases, more unsuitable mergings are done, until all the points, either belonging to different clusters or noise, are gathered into a unique group as shown on the right part of Figure 6, for $h = 0.4$.

The lower limit of the threshold is set at 0.15 (line 4). As only two distances are considered, see Eq. (16), this value means that one distance is 6.66 times the other!

Condition 2. Some unsuitable mergings involve structured groups.To identify these groups, the neighborhood around the connecting points, x and y, is considered. The volume is defined by the radius in each dimension set to twice the single link distance $d_{i,j} = d(x, y)$. The set of neighbors in each cluster, V_i and V_j, is computed as well us the number of mutual neighbors between them. The condition weights the number of mutual nearest neighbors between the two groups, $Mnn(x, y)$, by the homogeneity merging degree: the lower the degree, the larger the number of mutual nearest neighbors. The result is compared to a threshold function of the smaller cardinality of the sets of neighbors.

This test is useful for the separation of parallel shapes.

The results are illustrated in Figure 7. The nested U-shaped groups are no longer merged despite their proximity and whatever the homogeneity degree threshold.

In the central plot, $h = 0.45$, the two clusters, in green and black, would have been merged using the $d_{i,j}$ criterion because of the data points located on the abscissa about 1. Even with a more tolerant threshold, $h = 0.25$ (right plot), they remain distinct.

However, some noise is still included in the groups, especially between the two subclusters with positive abscissae, in black and green points with $h = 0.25$. In all the plots, noise is not displayed as only data points belonging to a cluster larger than $n/100$ (7 in this case) are shown.

Condition 3. The main purpose of this test is to manage density difference, including noise filtering. The heterogeneity between groups also involves their cardinalities. It is measured by the ratio of the minimum to the maximum of the two cardinalities of the set of neighbors,

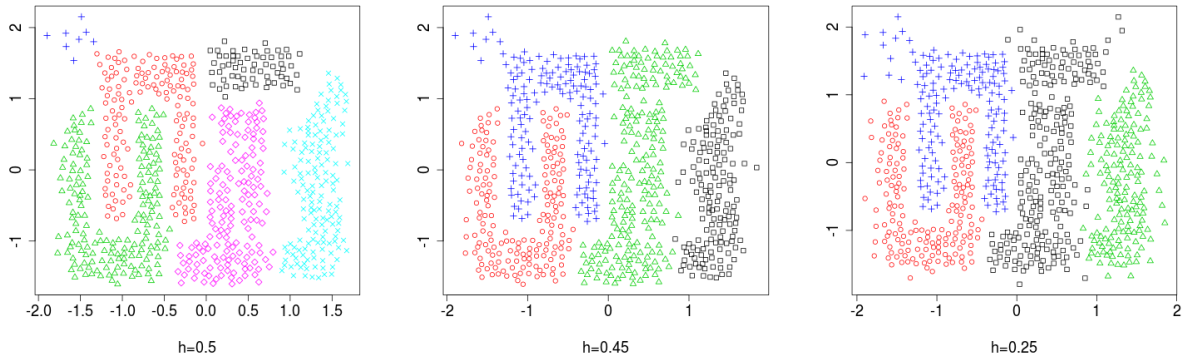

Figure 7. C1 and C2: Resulting partitions for three different thresholds on the homogeneity merging degree, h, when C2 is also applied. The axes are the x and y coordinates.

V_i and V_j, defined above. As in the previous test, the ratio is weighted by the homogeneity merging degree. The threshold value, e.g. $u = 0.06$ is discussed in the next section.

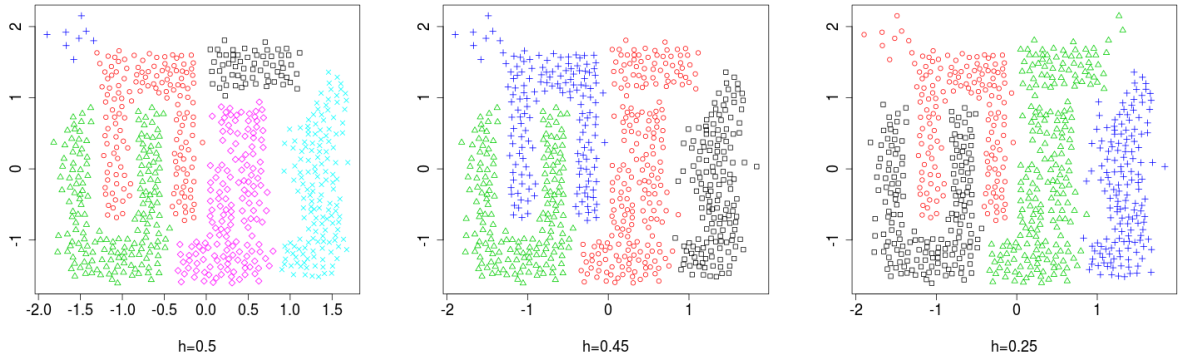

Figure 8. C1, C2 and C3: Resulting partitions for three different thresholds on the homogeneity merging degree, h, when C2 is also applied and $u = 0.06$. The axes are the x and y coordinates.

The three plots of Figure 8 show the impact of the last condition: compared to Figure 7, fewer noisy data points are gathered in clusters as the test does not allow for groups with different densities to merge. The process is still controlled by the homogeneity degree: the lower the threshold the higher the amount of noise included in clusters.

The condition behavior is illustrated using the data shown in Figure 9. This toy example includes two groups with distinct between neighbor distances, $d_{left} = 0.22$ and $d_{right} = 0.5$. The number of mutual neighbors is 3. To illustrate this behavior, the evolution of the conditional indices with respect to a decreasing distance between the two groups is shown in the central graph. As the left group is larger, the homogeneity merging degree is computed as $h_{r,l}$. In this case, the two groups are merged for $d_{r,l} < 0.63$. The right part of Figure 9

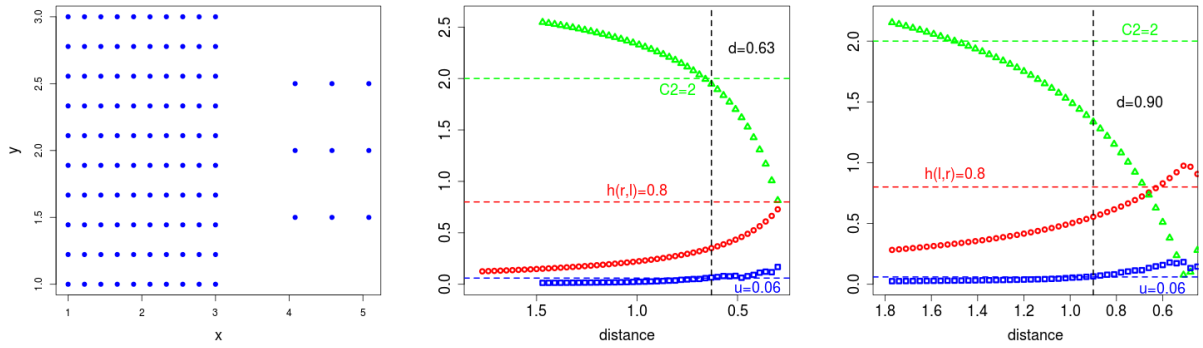

Figure 9. The global behavior is illustrated using the data shown in the left plot: the central plot shows the evolution of the three conditions C1 (red), C2 (green) and C3 (blue) according to the distance between the two groups of points. In the right plot, the same evolution is plotted when the more populated group is the one with $x > 4$. The ordinate is a common scale for the three conditions.

considers the other configuration: the right group is now the most important one, and the homogeneity is then computed as $h_{l,r}$. The behavior is noticeably different. The merging condition is reached for $d_{r,l} < 0.9$ and moreover the *SecureThres*, $h > 0.8$, is valid for $d_{r,l} < 0.63$. Within this distance range, $C2$ and $C3$ are not used.

The final results for the data in Figures 5-8 are shown in Figure 10. First, the number of clusters at the different steps of the algorithm is plotted for the three values of u: $0.02, 0.06, 0.10$. The higher the value, the more constrained the merging and, consequently, the higher the number of groups in the partition.

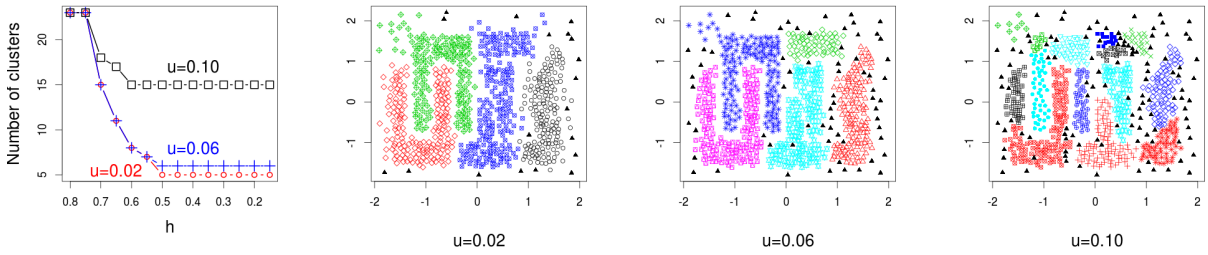

Figure 10. Sensitivity to the u parameter: The left plot shows the evolution of the number of clusters for the data used in Figures 5-8 and three values of u. The other three plots show the final partition for each of the u values. The axes are the x and y coordinates.

In the three partitions, thanks to the three complementary conditions, no unsuitable merging is done and only a small amount of noise is included in the clusters, even with a relatively low threshold value on the homogeneity degree.

4.3. Overview remarks

The dataset used in this section is representative of the main challenges for clustering algorithms: nested shapes, distinct densities, fuzzy boundaries due to a large amount of noise.

Despite its apparent complexity, *Munec* is extremely easy to use: the first step is fully automatic and the second one is driven by only one parameter, the threshold for the $C3$ test, u. The threshold for $C1$, h, is decreased from 0.8 to 0.15, and the one for $C2$, t, is set at 2 in all the experiments. The iterative evolution for h plays an important role: the structure is consolidated step by step and this helps the other two complementary conditions, $C2$ and $C3$, to work well. Moreover, thanks to its embedded controls, the algorithm yields acceptable results for a wide range of values.

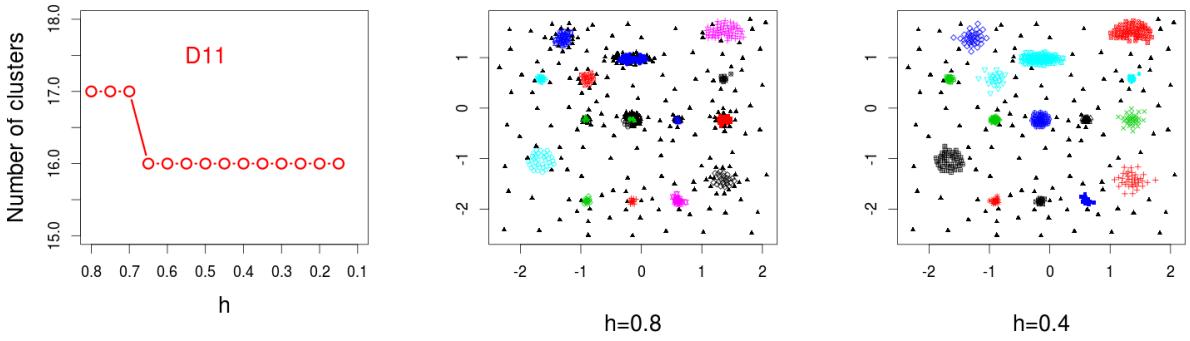

Figure 11. Illustration of the whole process with $D11$ data: the left plot shows the number of clusters versus the u parameter. The two possible final partitions for $u = 0.8$ and $u = 0.4$ are shown in the central and right plots. The axes are the x and y coordinates. These results stand for three values of u: $0.02, 0.06$ *and* 0.10.

The whole process is illustrated with two datasets, described in the next section, Table 3 and Figure 13, $D11$ and $D14$. Figures 11 and 12 show the same information for the two sets. The left part shows the evolution of the number of clusters versus the Condition 1 threshold on $h_{i,j}$. The center plot corresponds to the $h = 0.8$ configuration, meaning that only the first step of the algorithm, based on mutual neighboring was completed. In the right part of the two figures the threshold is set to $h = 0.4$.

The $D11$ data are correctly managed by the first step of the algorithm: the number of clusters is only decreased by one in the remaining part. This is not the case for the $D14$ data, the spirals. The final number of clusters, 2, is reached for threshold values lower than 0.4. These results are identical for three values of u, 0.02, 0.06 and 0.10.

More extensive tests are needed to validate the method. This is the goal of the next section.

Algorithm complexity. The number of distances to compute at each iteration in order to find the mutual neighbors is n^2. Each iteration of the two algorithms has the same complexity, in $O(n^2)$, even if in the second one more operations are performed to handle the heuristics.

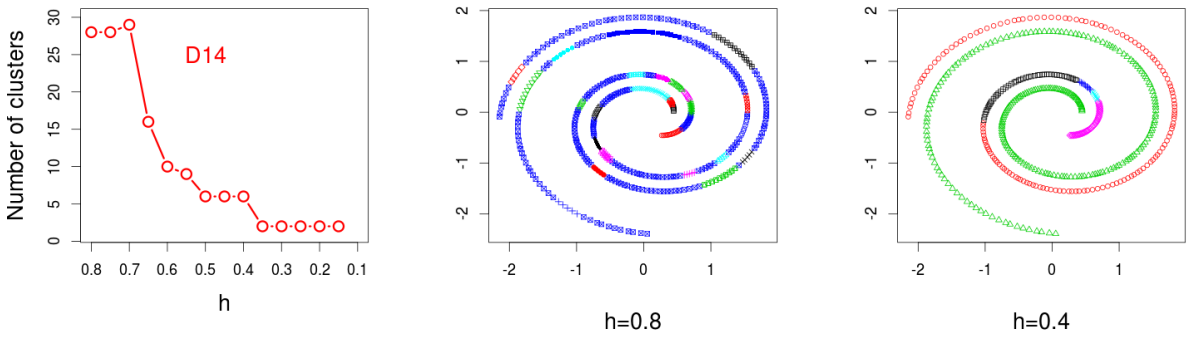

Figure 12. Illustration of the whole process with $D14$ data: the left plot shows the number of clusters versus the threshold on the homogeneity degree, h. Two partitions for $h = 0.8$ and $h = 0.4$ are shown in the central and right plots. The axes are the x and y coordinates. These results stand for three values of u: $0.02, 0.06$ *and* 0.10.

In the worst case, there are n-1 iterations. The algorithm complexity is thus $O(n^3)$. In practice the number of iterations is much lower than n: several mergings are allowed at each iteration and there are many pairs of mutual neighbors at the beginning of the process. The processing time can be decreased by increasing the space complexity: when the $n \times n$ distance matrix is stored and maintained, there is no need to compute the distances at each step.

5. Numerical experiments

Two kinds of experiments are proposed in this section. The first subsections deal with 2-dimensional datasets as they allow for a human assessment of what partitions are acceptable. To begin with, several algorithms are compared with respect to their ability to reach the partition target. In this case, external validation indices can be used to compare the expected partition with the one yielded by the algorithm. However, in real life the reference is unknown and clustering algorithms are used to analyze the data. In the case where there is no reference, in order to characterize the partition, only internal validation indices can be used. *Munec* was run with the same setting to check whether it is able to identify the data structure of various datasets. Then, a more qualitative comparison with the best alternative was carried out using a dataset for which no ground truth stands out. The last subsection illustrates the behavior of the proposal in various cases of higher dimension.

5.1. Datasets

A wide range of datasets (16) is used as benchmarks in this section. The data may include some variations in the clusters. The main sources of variation with their associated code are given in Table 2. They are the shape and the size of the clusters, their level of separation, the variation of density either between or within clusters and the amount of noise.

Table 2. The main sources of variation and their corresponding code

	1	2	3
Shape	Rather spherical or square	Long or thin	Ring, arbitrary, irregular
Size	Similar	Small variation	Large variation
Separation	Well separated	Separated	Overlap
Density	No variation	Inter clusters	Inter and intra clusters
Noise	None	Small amount	Large amount

Some are from the data clustering repository of the computing school of Eastern Finland University[3], while others come from the UCI machine learning repository[4] or were proposed in the published literature. These datasets are usually considered for testing new clustering algorithms but they do not represent the diversity of cases a clustering algorism has to tackle. Homemade data have been added to complete this diversity. They represent additional configurations where clusters are different in size, shape, density, amount of noise and degree of separation.

The preprocessing includes a $\{\mu, \sigma\}$ standardization step and, for the datasets with more than one thousand items, a sampling [41] in order to store the distance matrix in memory and to complete the tests in a reasonable amount of time.

Table 3. The sixteen datasets

	Size	Size-S	Partitions	Name	Origin
D1	3000	758	3	A.set 1	[26]
D2	5250	822	1	A.set 2	[26]
D3	7500	841	2	A.set 3	[26]
D4	240	240	1	FLAME	[15]
D5	373	373	1	JAIN	[23]
D6	5000	721	1	S.sets 1	[14]
D7	5000	694	3	S.sets 1	[14]
D8	5401	814	1	Dim sets 1	Footnote 3
D9	6751	754	2	Dim sets 2	Footnote 3
D10	8000	761	2	Chameleon	[28]
D11	40000	588	1	H1	Homemade
D12	3800	786	2	H2	Homemade
D13	2200	604	1	H3	Homemade
D14	2000	489	1	H4 (Spiral)	Homemade
D15	5500	717	1	H5	Homemade
D16	12500	603	1	H6	Homemade

Their main characteristics are summarized in Table 3 and the acceptable partitions are displayed in Figure 13. In Table 3, *Size-S* stands for the sample size and *Partitions* for

[3]https://cs.joensuu.fi/sipu/datasets/
[4]https://archive.ics.uci.edu/ml/

the number of acceptable partitions according to three human experts or the published literature.

The datasets considered in this study are now classified according to the main sources of variations, as shown in Table 4. The sum reported in the last column reflects the level of difficulty for a clustering algorithm to process the corresponding data.

Table 4. The sixteen datasets classified according to their irregularities

	Shape	Size	Separation	Density	Noise	Sum
D1	2	1	2	1	1	7
D2	1	1	3	1	1	7
D3	1	2	3	1	1	8
D4	3	2	3	1	2	11
D5	3	2	2	3	1	11
D6	2	1	2	1	1	7
D7	2	1	3	1	1	8
D8	1	1	2	1	1	6
D9	1	2	2	1	1	7
D10	3	3	3	1	3	13
D11	3	3	1	1	2	10
D12	3	3	1	3	1	11
D13	1	2	1	2	1	7
D14	3	1	2	1	1	8
D15	3	1	2	3	2	11
D16	1	2	3	2	1	9

5.2. Competitive algorithms

The proposal was compared to eleven selected algorithms, described in Table 5 with their free parameters. The first ones are classical algorithms whose limitations are also well known: *k-means* generates spherical clusters, *single-linkage* hierarchical clustering is sensitive to noise, the *complete-linkage* one tends to find compact clusters with equal diameters and *DBSCAN* does not cope with varying density clusters. This drawback is likely to be overcome by a recent improvement called *Recon-DBSCAN* [50]. While *DBSCAN* defines reachable points using two parameters, the radius ϵ and the minimum number of points in the corresponding volume, $Minpts$, *Recon-DBSCAN* considers two radii, ϵ and θ with $\theta \geq \epsilon$. The reachability is based on the density ratio $N_{pts}(\epsilon)/N_{pts}(\theta)$ compared to the τ threshold.

The Shared Nearest Neighbor algorithm, *SNN* [25], as well as its variants [12], is a density based clustering algorithm working similarly to *DBSCAN*. The main difference is that the volume is not defined by the radius but is induced by the nearest neighbors. The volume can be optionally limited by a radius. The algorithm is thus driven by two main parameters: the number of nearest neighbors to be considered and the minimum number of points that define the reachability. A less important parameter allows for noise management. When the sum of shared nearest neighbors for a given item, i, with all the remaining others, j, is less

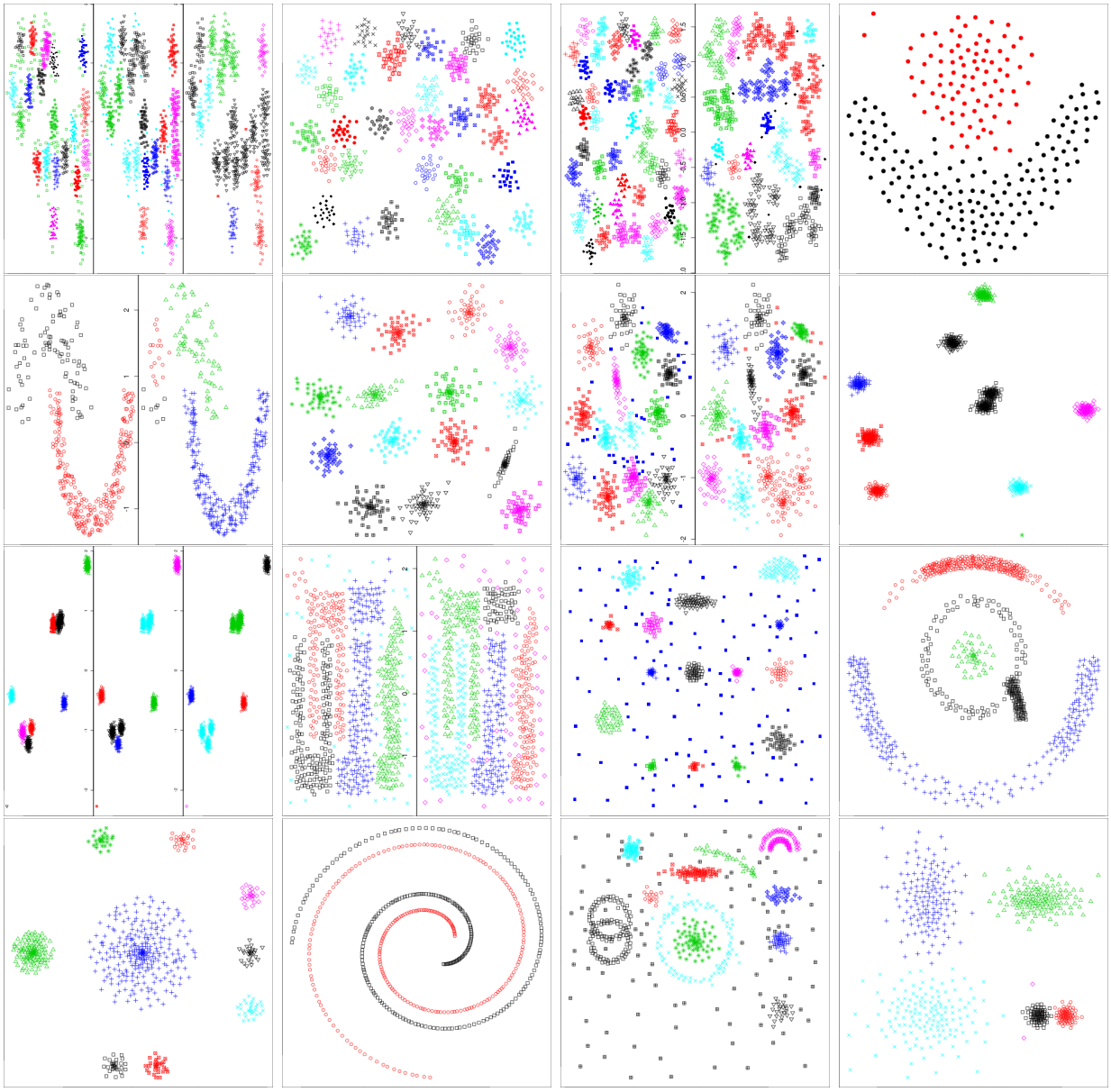

Figure 13. Ground truth for the 16 datasets (D1, top left - D16, bottom right). The axes are the x and y coordinates.

Table 5. The competitive algorithms

Algorithm	Acronym	Param	Range	Ref
k-means	A1	c	$[1, 50]$	[24]
Single-linkage	A2	c	$[1, 50]$	[24]
Complete-linkage	A3	c	$[1, 50]$	[24]
DBSCAN	A4	ϵ,Minpts	$[0.05, 0.25]$, $[1, 10]$	[13]
Recon-DBSCAN	A5	ϵ,θ,τ	$[0.05, 0.25]$, $[1.1, 1.5]$, $[0.25, 0.6]$	[50]
DensityPeaks (pioneer)	A6	c,d_c	$[1, 50]$, $[0.018, 0.05]$	[40]
DensityPeaks Data Field	A7	c	$[1, 50]$	[44]
SNN	A8	$MinPts$, k	$[2, 10]$, $\sqrt{n} \cdot [0.05, 0.5]$	[25, 12]
SNN (with Radius)	A9	$MinPts$, k, Radius	$[2, 10]$, $\sqrt{n} \cdot [0.05, 0.5]$, $[0.05, 0.3]$	[25, 12]
MutualClust	A10	c	$[1, 50]$	[17]
SCDOT	A11	c	$[1, 50]$	[6]
Munec	A12	u	$\{0.02, 0.06, 0.10\}$	

than a threshold value, i is labeled as noise. Tests on the sixteen datasets showed that the best configuration always involved the same smallest value: 1.

The other algorithms are detailed in Section 2 of this paper. The *DensityPeaks* algorithm has been recently improved [44]. The threshold distance d_c is now automatically set using a potential entropy of data field approach. The local density is now estimated using a Gaussian function instead of the classical nearest neighbor count. This important change was also implemented in the pioneering version of the algorithm for this study.

5.3. Quantitative comparison: protocol

All these algorithms were implemented in *C* and the experiments were carried out on a unique machine. Thirty runs were performed for *k-means* as it includes random aspects.

The objective of the quantitative comparison was to check whether each of the competitive algorithms is able to reach the ground truth target. This choice of external validation was motivated by the lack of a generally accepted internal index validation. Some of the selected data can be quite naturally clustered; in this case the three human experts agree on the target partition. Otherwise, the target is not unique but may include two or three partitions, as shown in Figure 13. In this case, the best fit is taken into account. Two popular indices were used for partition comparison: the Mutual Information Index [7] and the Rand Index [39].

Two strategies are possible for noise management as some algorithms, like *DBSCAN* or *Munec*, include a noise category. The usual one is to consider all the data points; this strategy penalizes the methods that explicitly define noise elements. The other one is to include a noise category in the ground truth. The mutual information index was performed by skipping noise points for the two partitions (while the amount of noise is kept below 20%). The Rand Index was computed from all the data, including noise.

For each algorithm considered, the input parameters were tuned in order to maximize the result for each dataset. The ranges are given in Table 5.

When the parameter was an integer, such as the number of clusters or the minimum number of points, all the values were used, whereas when it was a floating point number,

10 equally separated values were taken in the range. Some of the parameters were defined to limit the number of tests. This is the case for the cutoff distance of *DensityPeaks*, d_c for $A7$. It was calculated so that the average number of neighbors was between 1% and 2.5% of the total number of points in the dataset. For *SNN (with Radius)* only three values were selected for *MinPts* and *NoisePts* in their ranges. For *MutualClust* and *SCDOT* the maximum number of neighbors was set to 80.

The number of runs for each dataset is highly dependent on the algorithm. It ranges from 3 in the case of *Munec* to 1000 for *SNN (with Radius)* or *Recon-DNSCAN*.

5.4. Quantitative comparison: results

The results for the mutual information index are summarized in Table 6. A row corresponds to a dataset and reports the mutual index when noise is considered as a category for each of the algorithms. The values below 0.8 are in bold font. In the last two rows the mean and standard deviation are computed for each algorithm over the 16 datasets. The values of mean higher than 0.9 are in bold font.

For *k-means*, the standard deviation of the mutual index corresponding to the best configuration for each dataset is reported in Table 7.

Table 6. Mutual Index when noise is considered as a separate category for the 16 datasets and the 12 algorithms

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
D1	0.940	0.950	0.928	0.964	0.9950	1.000	0.992	0.838	0.870	0.965	0.944	0.989
D2	0.971	0.863	0.951	0.938	0.997	0.975	0.986	0.966	0.922	0.885	0.924	0.980
D3	0.937	0.968	0.978	0.960	0.997	0.947	0.987	0.981	0.975	0.939	0.932	0.981
D4	**0.560**	0.808	**0.364**	**0.402**	**0.480**	**0.706**	1.000	**0.541**	**0.623**	**0.420**	**0.625**	1.000
D5	**0.520**	0.991	**0.664**	0.862	0.898	**0.505**	**0.520**	1.000	0.964	1.000	0.977	1.000
D6	0.980	0.967	0.976	1.000	1.000	0.995	0.995	0.985	0.974	0.923	0.965	0.991
D7	0.977	0.802	0.945	1.000	1.000	0.981	0.981	0.931	0.908	**0.792**	0.936	1.000
D8	0.943	1.000	1.000	1.000	1.000	1.000	0.998	0.998	1.000	1.000	0.980	1.000
D9	0.960	1.000	1.000	1.000	1.000	1.000	0.998	0.902	0.918	1.000	0.987	1.000
D10	**0.668**	**0.655**	**0.652**	0.919	0.952	**0.780**	**0.780**	**0.796**	0.861	0.870	**0.786**	0.945
D11	0.982	0.931	0.987	1.000	1.000	1.000	1.000	1.000	1.000	0.996	0.977	1.000
D12	**0.690**	1.000	**0.596**	0.915	1.000	**0.664**	**0.673**	1.000	1.000	1.000	0.985	1.000
D13	0.921	1.000	0.9401	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.997	1.000
D14	**0.193**	1.000	**0.560**	1.000	1.000	1.000	**0.331**	1.000	1.000	1.000	**0.415**	1.000
D15	0.839	**0.554**	0.846	0.983	0.983	0.897	0.897	0.975	0.990	0.929	0.934	1.000
D16	0.916	0.863	0.900	0.930	0.902	0.994	0.994	0.981	0.916	0.918	0.963	0.982
Mean	0.812	0.897	0.799	**0.930**	**0.950**	**0.903**	0.883	0.930	**0.932**	**0.915**	0.896	**0.990**
σ	0.227	0.134	0.270	0.147	0.130	0.154	0.205	0.121	0.093	0.145	0.159	0.016

Table 7. Standard deviation (regarding the best configuration) of the Mutual index for the 16 datasets for *k-means* (A1)

	D1	D2	D3	D4	D5	D6	D7	D8
σ	0.026	0.011	0.038	0.007	0.003	0.031	0.029	0.057
	D9	D10	D11	D12	D13	D14	D15	D16
σ	0.069	0.035	0.030	0.000	0.053	0.053	0.029	0.035

The poorest results are for $D4$, $D5$ and $D10$. Except *Munec*, *SNN* is the only one able to find the 2-cluster partition for $D4$, but this method performs poorly with $D5$ and $D10$.

For datasets $D1$, $D2$, $D3$, $D6$, $D8$, $D9$, $D11$ and $D13$ all the algorithms yield a result higher than 0.8. Even if they differ in the amount of noise, these data share some characteristics likely to explain these results: clusters are represented by density peaks and are more or less spherical.

The other datasets illustrate the limitations of some competitors. In $D7$ some groups are not spherical-shaped. This is also the case in $D12$ and $D14$. As these datasets do not contain density peaks, the methods based upon this concept, $A6$ and $A7$, fail to reach the correct partition. With $D16$, *SNN* is the least efficient algorithm, showing that sharing neighbors is not enough to obtain the expected partition.

k-means includes a random aspect giving variable results for the same cluster number (Table 7). σ for the mutual index varies from 0.0003 ($D12$) to 0.069($D9$) for the 16 datasets. $D4$, $D5$ and $D12$ produce the smallest ones. These datasets present the common difficulty of having clusters with irregular or non-convex shapes that cannot be detected by the *k-means* algorithm independently of the random aspect. $D8$, $D9$, $D13$ and $D14$ are related to the highest σ. $D8$ and $D9$ share the difficulty of having one or two cluster subsets with overlapping leading to different clusters. For $D13$, the instability is more due to the difference in size. For $D14$, the reason is different. *k-means* is not suitable for spiral data. It has a tendency to split the clusters into different groups that change from one run to another then mitigating the performance and producing instability.

Munec is the only one of the studied methods able to identify one of the targets for all the datasets. This is highlighted by the last two rows of the table. *Recon-DBSCAN* and *Munec* values are above 0.95.

The same two algorithms, *Munec* and *Recon-DBSCAN*, give the best results when the Rand Index is used for all the data points, Table 9, despite the fact that the two methods are penalized as the noise labeled points, which are handled as a unique category, are scattered throughout the whole space.

The runtime must be considered as it is likely to restrict the practical use of slow algorithms to moderate size datasets. The average time for a single run on a dataset is given in Table 10. The last row gives the standard deviation over all the datasets.

k-means and *DensityPeaks* are the fastest algorithms while the runtime of *Munec* is intermediate, comparable to $A4$, $A5$, $A8$ and $A9$.

The two external indices used in Tables 6 and 9 handle the two types of errors in the same way. A non symmetric index, the *F-measure* was also computed from the results given by *Munec* (weighted combination). When noise is considered as a separate category the mean and standard deviation are 0.944 and 0.056. These values become 0.985 and 0.030 when noise points are not taken into account.

5.5. Working without ground truth

The former section showed that the studied algorithms are able to find, but not always, some input configuration to reach a predefined target. But clustering algorithms are used to identify unknown data structures without any a priori information. This is especially true

Table 8. The configuration that yields the best result in Table 6. When the parameter is a discrete value, this value is reported in the table. For a continuous range the code corresponds to the fraction of the range, 0 for the minimum and 9 for the maximum.

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
D1	31	31	50	20	5,4,5	19,0	20	4,6	0,4,2	38	21	0.10
D2	39	50	48	20	4,7,0	40,0	35	8,8	0,2,6	47	45	0.10
D3	24	18	21	60	4,0,8	16,1	27	0,4	0,4,4	20	23	0.10
D4	3	2	3	0	9,2,3	3,0	2	2,7	0,7,7	23	5	0.06
D5	8	4	8	80	9,2,3	12,0	10	0,7	0,7,9	3	4	0.02
D6	14	15	13	30	4,6,3	14,1	15	8,9	2,4,9	35	21	0.06
D7	16	12	16	80	9,9,6	14,1	14	4,6	0,4,4	50	16	0.06
D8	9	8	8	0	1,5,4	8,0	8	0,3	0,3,3	1	1	0.02
D9	6	4	39	20	1,2,6	5,0	5	7,8	0,3,0	1	5	0.06
D10	9	7	50	40	6,4,0	7,1	8	8,9	0,3,4	35	8	0.06
D11	21	47	50	40	3,2,3	14,0	15	3,6	0,4,3	37	21	0.06
D12	6	5	50	80	5,1,5	8,0	9	0,3	0,3,6	1	9	0.02
D13	7	9	49	50	7,0,0	11,0	11	0,4	0,4,5	1	4	0.02
D14	48	2	50	0	4,0,0	1,0	39	0,3	0,3,3	2	2	0.06
D15	14	47	50	50	4,2,0	8,0	9	4,6	2,4,3	46	21	0.06
D16	5	4	21	90	9,6,0	4,1	5	7,9	0,3,8	23	5	0.06

Table 9. Rand Index when all the data points, including noise, are considered: mean and standard deviation over the 16 datasets for the 12 algorithms

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
Mean	0.902	0.933	0.901	0.951	0.959	0.941	0.924	0.928	0.935	0.949	0.948	0.969
σ	0.127	0.115	0.127	0.024	0.018	0.077	0.129	0.077	0.071	0.032	0.082	0.023

when dealing with high dimensional data, that are not easy to visualize. Algorithms driven by the number of clusters, a category that includes some recent proposals such as *SCDOT* or *DensityPeaks*, are discarded. The only way to use them is to rank the resulting partitions according to a validation index and none of them reaches an agreement in the scientific community. Smarter algorithms, such as *DBSCAN*, are extremely sensitive to tuning. This is not the case for *Munec*.

Table 11 depicts the results obtained with *Munec* for three distinct values of the parameter, $u = \{0.02, 0.06, 0.10\}$. The last two rows report the mean and standard deviation for each configuration over the 16 datasets. The indices are similar to those in Table 6,

Table 10. Runtime (ms) for a run on a dataset for the 12 algorithms

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
Mean	20	2842	3283	210	209	18	941	228	229	476	838	197
σ	13	297	320	29	27	8	97	35	32	70	627	19

the Mutual index is computed and noise is not taken into account. The standard deviation becomes smaller with high mean values.

Table 11. *Munec* with three values of u: Mutual information index for the 16 datasets

u	0.02	0.06	0.10
D1	0.436	0.915	0.989
D2	0.326	0.949	0.980
D3	0.452	0.938	0.981
D4	0.000	1.000	1.000
D5	1.000	0.529	0.416
D6	0.914	0.991	0.947
D7	0.637	1.000	0.949
D8	1.000	0.964	0.962
D9	1.000	1.000	0.953
D10	0.852	0.945	0.768
D11	1.000	1.000	0.967
D12	1.000	0.791	0.633
D13	1.000	0.891	0.840
D14	1.000	1.000	1.000
D15	0.941	1.000	0.953
D16	0.882	0.982	0.944
Mean	0.778	0.931	0.892
σ	0.312	0.121	0.160

The variation of the index with respect to the parameter u depends on the data: it is monotonically increasing for $D2$ and monotonically decreasing for $D5$, and not monotone for $D7$. This is expected as all these indices are computed from a ground truth target.

Two values of u give good average results for the 16 datasets, higher than 0.9. The mean for $u = 0.02$ is penalized by the zero for $D4$ data. This value is easily explained: there is no mutual information when the algorithm yields a $1-$group partition. When the zero is removed, the mean becomes 0.911.

The best mean, 0.938 for $u = 0.06$ is close to the mean of *Recon-DBSCAN* in Table 6. The main difference is that the setting of the latter was tuned for each dataset whereas it is the result of a single parameter here.

5.6. Qualitative comparison

The most two effective methods, *Recon-DBSCAN* and *Munec* are now compared using the dataset introduced in [14] and called *S.sets 4*. It is made up of 4051 $2D$-points, which are processed to yield a sample of size 756. No ground truth stands out for these data as the groups of various densities are not clearly separated.

The two algorithms were run to generate three partitions. *Recon-DBSCAN* was first run to identify the parameters that yield stable results, $\varepsilon = 0.15$ and $\theta = 1.3\varepsilon$. Then the three

values of the threshold were used: $\tau = \{0.3, 0.5, 0.7\}$. The settings for *Munec* were chosen to yield a similar number of clusters to *Recon-DBSCAN*: $u = \{0.03, 0.08, 0.14\}$.

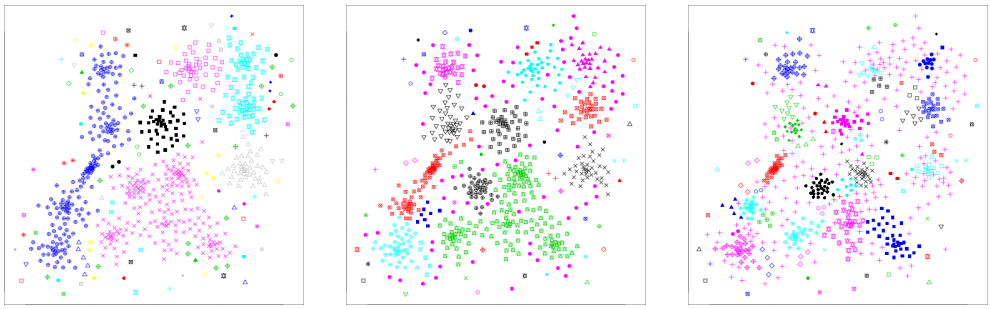

Figure 14. *Recon-DBSCAN*: Three partitions corresponding to $\tau = \{0.3, 0.5, 0.7\}$. The axes are the x and y coordinates.

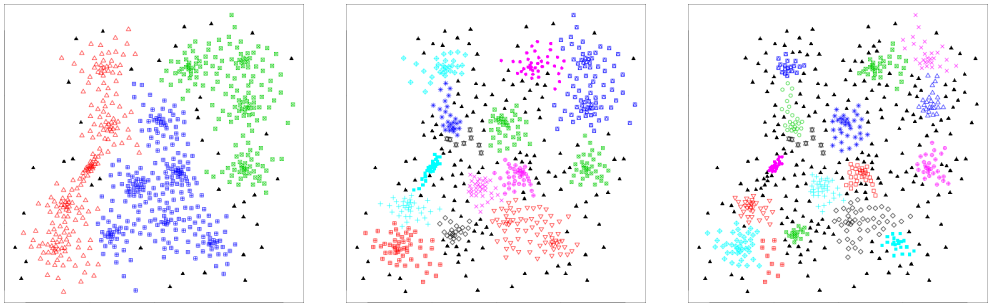

Figure 15. *Munec*: Three partitions corresponding to $u = \{0.03, 0.08, 0.14\}$. The axes are the x and y coordinates.

The resulting partitions are shown in Figures 14 and 15. The studied algorithms behave in the same way: the higher the constraint, τ or u, the higher the number of clusters as shown in Table 12. Several indices were proposed to characterize a partition, 42 are implemented in the *clusterCrit* R[5] package. All the indices aim to characterize both internal cluster cohesion and between cluster separation. They could be clustered in families according to their similarities and differences. The available scientific reviews do not conclude about when one of them should be preferred to another. In this work, as well as in many other publications, the Silhouette and Dunn indices are used [3]. They can be seen as complementary in the information they use: in Silhouette cohesion is based on the distance between all the points while Dunn restricts to the nearest neighbor; separation is evaluated using the nearest neighbor distance in Silhouette and the maximum cluster diameter in Dunn. Their values are given in the last two columns: they are similar for the two algorithms.

It is worth noting that the two algorithms are able to identify patterns based either on density peaks or texture differences. This result is important as the groups partially overlap.

[5]https://www.r-project.org/

Table 12. Silhouette and Dunn indices for three configurations of *Recon-DBSCAN* and *Munec* that yield a similar number of clusters

	Config	nb clusters	nb noise	Silhouette	Dunn
Recon-DBSCAN	30	7	99	0.192	0.028
	50	13	61	0.321	0.014
	70	18	111	0.472	0.005
Munec	0.03	3	40	0.159	0.026
	0.08	14	143	0.420	0.012
	0.14	17	231	0.477	0.009

5.7. Complementary experiments with datasets of higher dimension

Two kinds of experiments were carried out to assess the behavior of the algorithm in higher-dimensionnal spaces.

In the first one the goal is to identify 9 distinguishable clusters with Gaussian distributions in dimension 2 to 15. The data were proposed in [27]. The number of samples increases with the dimension. The sample size, n is proportional to the dimension, d: $n = 675d + 1$. *Munec* was run with the following values $u = \{0.02, 0.06, 0.1\}$. For all the dimensions, at least one value of u yielded the right number of clusters and a Rand Index of 1 with respect to the true partition. When $u = 0.1$ the constraints are so strong that the final number of clusters is usually higher than the expected one.

The second experiment is based on the *genRandomClust* R package. This is an implementation of the method proposed in [36]. The degree of separation between any cluster and its nearest neighboring cluster can be set to a specified value regarding the separation index proposed in [37]. The cluster covariance matrices can be arbitrary positive definite matrices. They correspond to different shapes, diameters and orientations. The *eigen* method is used in the experiment. It first randomly generates eigenvalues $(\lambda_1, \ldots, \lambda_p)$ for the covariance matrix then uses columns of a randomly generated orthogonal matrix, $Q = (\alpha_1, ..., \alpha_p)$, as eigenvectors. The covariance matrix is then built as $Q \cdot diag(\lambda_1, ..., \lambda_p) \cdot Q^T$. The package uses the basic parameters for cluster generation such as the number of clusters, the space dimension and their respective sizes but also allows for variability management. A ratio between the upper and the lower bound of the eigenvalues can be specified. The default value is 10, but 30 was used in all the experiments. The range of variances in the covariance matrix was set to the default value, $rangeVar = [1, 10]$. The only parameter used in this experiment is the value of the separation index between two neighboring clusters, $SepVal$. It ranges from -1 to 1. The closer to 1 the value, the more separated the clusters. The behavior of these two parameters is illustrated with 2-dimensional clusters on Figure 16 through 4 typical configurations.

The package was used to generate data structured in 5 clusters in spaces from 2 to 10 dimensions with a number of samples per cluster increasing linearly with the dimension from 300 to 1500. For each replicate, the pair of values was randomly chosen in each interval. The tests were carried out with 4 values of the separation degree: $SepVal = \{0.1, 0.2, 0.3, 0.4\}$. The RI was computed with respect to the true partition and averaged over the 10 replicates

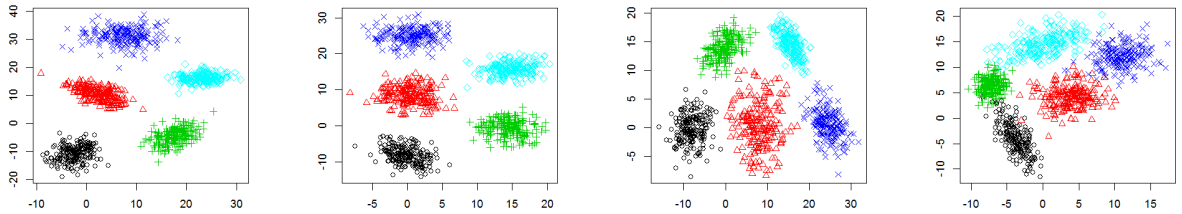

Figure 16. Four configurations of $SepVal$ for random cluster generation, from left to right: 0.4, 0.3, 0.2 and 0.1. The axes are the x and y coordinates.

for each configuration. Three values of u were studied, $u = \{0.02, 0.06, 0.1\}$, and the best RI was kept.

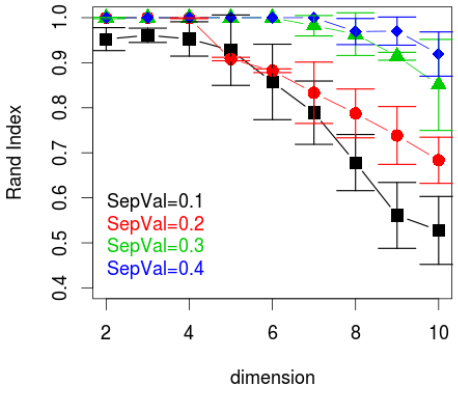

Figure 17. Random cluster generation: Rand Index versus the dimension for the 4 $SepVal$ configurations

The results are summarized in Figure 17. *Munec* is able to find the expected partition, whatever the dimension, when the clusters do not overlap too much. This ability decreases with the degree of cluster separation. When $SepVal = 0.1$, making the cluster poorly distinguishable, the correct structure is identified when the space dimension is at most 5.

The results for this configuration, $SepVal = 0.1$, are detailed in Table 13. For each dimension, the average and standard deviation of the RI and the final number of clusters are reported. It is worth mentioning that even with $u = 0.1$ the number of clusters found is slightly below the expected number, when the dimension is higher than 5. The average of the *Silhouette* and *Dunn* indices confirm that the cluster separation is not clear: the two indices drop to 0 as soon as the dimension is higher than 2. This is explained by the high level of noise and shape variation generated by the selected setting. The averaged *Silhouette*

Table 13. *Munec* results with random cluster generation and $SepVal = 0.1$

dim	RI		NbClust		Silhouette		Dunn	
	μ	σ	μ	σ	μ	σ	μ	σ
2	0.946	0.036	5	0	0.321	$5.1e^{-4}$	0.131	$2e^{-3}$
3	0.937	0.046	5	0	0.033	$8.4e^{-5}$	$1.2e^{-2}$	$1e^{-7}$
4	0.952	0.058	5	0	-0.074	$7.8e^{-4}$	$5.7e^{-3}$	$1e^{-7}$
5	0.913	0.099	4.8	0.4	-0.085	$4.5e^{-4}$	$4.7e^{-3}$	$1e^{-7}$
6	0.837	0.093	4.7	0.8	-0,113	$3.1e^{-4}$	$6.7e^{-4}$	$1e^{-7}$
7	0.768	0.083	4.9	0.75	-0,143	$2.3e^{-5}$	$6.0e^{-4}$	$1e^{-7}$
8	0.649	0.072	4.7	0.85	-0,169	$5.1e^{-6}$	$6.4e^{-4}$	$1e^{-7}$
9	0.497	0.081	4.5	1.5	-0.215	$1.0e^{-6}$	$6.3e^{-4}$	$1e^{-7}$
10	0.408	0.094	4.3	1.7	-0.237	$7.4e^{-6}$	$6.3e^{-4}$	$1e^{-7}$

value for $SepVal = 0.4$ and dimension 10 is -0.093. This difficult case highlights the good performance of the proposal. The performances obtained when running *Recon-DBSCAN* by tuning each parameter 8 times (8^3 trials) are comparable for the $2D$-data ($RI = 0.948$, $\sigma = 0.028$) with $SepVal = 0.1$ but are degraded for dimension 3 data, $RI = 0.576$. The same phenomenon is observed for higher values of $SepVal$ such as 0.3. In this case, the results are ($RI = 0.998$, $\sigma = 0.005$) for dimension 2, the RI is 0.834 for dimension 3 but it drops to 0.470 for dimension 4. These highly variable configurations are poorly managed by *Recon-DBSCAN*.

6. Conclusion

A new algorithm is proposed in this paper based on the hierarchical merging of mutual nearest neighbors. The single link distance generalizes the mutual neighboring to groups. A sub-cluster is characterized by the mean distance between neighbors. This value averaged over all the sub-clusters characterizes the partition. Unfortunately, it cannot serve as a stopping criterion because it is monotonically increasing.

The algorithm comprises two main steps. The first one is driven by a similarity index based on the three distances involved in the merging: the two inner ones and the single link distance between the two groups. It is self-controlled: a high threshold is used in order to yield a skeleton of the structure without doing any unexpected merging. In the second step some heuristics are proposed to go further. Besides neighborhood, density is taken into account: the merging index is no longer symmetrical but higher size group-oriented.

Three conditions are required for a merging. $C1$ is a threshold on the distance homogeneity, $C2$ is also based on the mutual neighborhood and $C3$ on the local neighborhood for density differentiation including noise filtering. The threshold used in $C1$ is iteratively decreased during the process. It is not a user parameter. $C2$ is computed from the homogeneity index and the number of mutual neighbors within a local neighborhood. The threshold is not a user parameter: it is dynamically set according to the total number of neighbors. Finally the unique user parameter is the threshold for $C3$.

Many of these concepts are shared by other algorithms. Nevertheless, *Munec* differs from them in several ways. The density concept is the basis of many algorithms, some old ones such as the *k-means*, or other recently published ones such as *DensityPeaks*. Three main differences are worth noting. *Munec* does not start the search for density peaks from points but from sub-clusters identified in the first step of the algorithm by mutual neighboring. The second difference stems from the iterative decrease of the homogeneity index threshold: this allows the progressive consolidation of the structure. The last difference is the use of additional constraints, $C2$ and $C3$, to avoid merging when the peak is not unique or not clearly defined.

Single link distance is known to be relevant but very sensitive to noise. In the proposal, it is used for selecting candidates for merging. Then the final decision is made after a careful inspection of the local context using $C2$ and $C3$. These conditions combine distance and neighborhood. They are based on known methods such as *Chameleon* or *SCDOT* but *Munec* uses a dynamic definition of the local neighborhood, which does not require any parameter, namely a hypersphere with a radius twice the single link distance.

Several parameters are required for *Munec* to prevent unexpected merging. Most of them were empirically defined and integrated as constants to make the algorithm user-friendly. Finally, a single parameter u is left to the non-expert user to drive the algorithm. Whatever its value, no unexpected merging is done. This parameter defines the partition granularity by controlling the level of density differentiation: the higher its value the stronger the constraints on the merging and the higher the number of clusters.

The study shows that the algorithm yields stable results with only a few distinct partitions to be analyzed or characterized, for the useful range of u. Three typical values were used in this work: $\{0.02, 0.06, 0.1\}$, 0.06 being the default value. Their impact is especially noticeable when the separation degree between clusters is low because of noise or partial overlap. Compared to alternative methods *Munec* proved to be extremely effective to match a ground truth target. Moreover, with the same input configuration, u, datasets of various densities, arbitrary shape and including a large amount of noise can be managed. This generic setting, validated using a wide range of experiments, is an interesting asset of the proposal. The next advance could be based on a post-processing stage that would complement the local view proposed in this paper by a partition-scale coherence analysis.

Another perspective is to deal with very large data bases. The first step is to extend the current algorithm to manage a huge number of observations. Various techniques can be envisaged. Some of them optimize the algorithm by using a kd-tree implementation or a neighborhood graph. Others divide the data via strata [20], run the algorithm on each subset and combine the results. *Munec* is currently part of a datamining package used by different startup companies hosted by the LAB'O, a business incubator located in Orléans (France). As an example, PSASS provides a distance expertise to analyze records for sleep apnoea diagnosis. This platform allows practitioners to save 80% of their time. The remote analysis is done by a human expert. To make the service scalable an automatic system is under design. The inputs are several attributes selected from spolysomnographic records using signal processing and feature selection techniques. *Munec* was used to identify the underlying structure in records. Natural clusters were detected as different values of u

provided the same number of groups. The partition is the basis of an interpretable and efficient decision tree that is currently in test under the expert control.

References

[1] A. Amini, T. Y. Wah, H. Saboohi, On density-based data streams clustering algorithms: a survey, Journal of Computer Science and Technology 29 (1) (2014) 116.

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, in: ACM SIGMOD Record, vol. 28, ACM, 1999, pp. 49–60.

[3] N. Bolshakova, F. Azuaje, Cluster validation techniques for genome expression data, Signal processing 83 (4) (2003) 825–833.

[4] M. Brito, E. Chavez, A. Quiroz, J. Yukich, Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection, Statistics & Probability Letters 35 (1) (1997) 33–42.

[5] D. Cheng, Q. Zhu, J. Huang, L. Yang, Q. Wu, Natural neighbor-based clustering algorithm with local representatives, Knowledge-Based Systems 123 (Supplement C) (2017) 238 – 253.

[6] Q. Cheng, X. Lu, Z. Liu, J. Huang, G. Cheng, Spatial clustering with density-ordered tree, Physica A: Statistical Mechanics and its Applications 460 (2016) 188 – 200.

[7] T. M. Cover, J. A. Thomas, Elements of information theory, John Wiley & Sons, 2012.

[8] D. Dantzig, G.B.and Fulkerson, On the max-flow min-cut theorem of networks, RAND corporation, 1964.

[9] M. Du, S. Ding, Y. Xue, A robust density peaks clustering algorithm using fuzzy neighborhood, International Journal of Machine Learning and Cybernetics 9 (7) (2018) 1131–1140.

[10] D. Duan, Y. Li, R. Li, Z. Lu, Incremental k-clique clustering in dynamic social networks, Artificial Intelligence Review 38 (2) (2012) 129–147.

[11] L. Ertöz, M. Steinbach, V. Kumar, A new shared nearest neighbor clustering algorithm and its applications, in: Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining, 2002, pp. 105–115.

[12] L. Ertöz, M. Steinbach, V. Kumar, Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data, in: Proceedings of the 2003 SIAM International Conference on Data Mining, SIAM, 2003, pp. 47–58.

[13] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., 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, 1996, pp. 226–231.

[14] P. Fränti, O. Virmajoki, Iterative shrinking method for clustering problems, Pattern Recognition 39 (5) (2006) 761–765.

[15] L. Fu, E. Medico, Flame, a novel fuzzy clustering method for the analysis of dna microarray data, BMC Bioinformatics 8 (1) (2007) 3.

[16] R. Gonzalez, R. Woods, Digital Image Processing, 4th ed., Prentice Hall, 2002.

[17] K. C. Gowda, G. Krishna, Agglomerative clustering using the concept of mutual nearest neighbourhood, Pattern Recognition 10 (2) (1978) 105 – 112.

[18] E. Güngör, A. Özmen, Distance and density based clustering algorithm using gaussian kernel, Expert Systems with Applications 69 (2017) 10–20.

[19] J. A. Hartigan, M. Wong, A k-means clustering algorithm, Applied Statistics 28 (1979) 100–108.

[20] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, J. Fan, Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce, in: Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on, IEEE, 2011, pp. 473–480.

[21] A. Hinneburg, D. A. Keim, A general approach to clustering in large databases with noise, Knowledge and Information Systems 5 (4) (2003) 387–415.

[22] Z. Hu, R. Bhatnagar, Clustering algorithm based on mutual k-nearest neighbor relationships, Statistical Analysis and Data Mining 5 (2) (2012) 100–113.

[23] A. Jain, M. Law, Data Clustering: A User's Dilemma, in: Proceedings of the First international conference on Pattern Recognition and Machine Intelligence, 2005, pp. 1–10.
[24] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651–666.
[25] R. A. Jarvis, E. A. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Transactions on computers 100 (11) (1973) 1025–1034.
[26] I. Kärkkäinen, P. Fränti, Dynamic local search algorithm for the clustering problem, Tech. Rep. A-2002-6, Department of Computer Science, University of Joensuu, Joensuu, Finland (2002).
[27] I. Kärkkäinen, P. Fränti, Gradual model generator for single-pass clustering, Pattern Recognition 40 (3) (2007) 784–795.
[28] G. Karypis, E.-H. Han, V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.
[29] L. Kaufman, P. Rousseeuw, Clustering by means of medoids, Statistical Data Analysis Based on the L1-Norm and Related Methods (1987) North–Holland.
[30] R. Kuo, Y. Huang, C.-C. Lin, Y.-H. Wu, F. E. Zulvia, Automatic kernel clustering with bee colony optimization algorithm, Information Sciences 283 (2014) 107–122.
[31] J.-S. Lee, S. Olafsson, Data clustering by minimizing disconnectivity, Information Sciences 181 (4) (2011) 732–746.
[32] J.-S. Lee, S. Olafsson, A meta-learning approach for determining the number of clusters with consideration of nearest neighbors, Information Sciences 232 (2013) 208–224.
[33] W. Liu, J. Hou, Study on a density peak based clustering algorithm, in: Intelligent Control and Information Processing (ICICIP), 2016 Seventh International Conference on, IEEE, 2016, pp. 60–67.
[34] R. T. Ng, J. Han, Clarans: A method for clustering objects for spatial data mining, IEEE Trans. on Knowl. and Data Eng. 14 (5) (2002) 1003–1016.
[35] A. H. Pilevar, M. Sukumar, Gchl: A grid-clustering algorithm for high-dimensional very large spatial data bases, Pattern Recognition Letters 26 (7) (2005) 999–1010.
[36] W. Qiu, H. Joe, Generation of random clusters with specified degree of separation, Journal of Classification 23 (2) (2006) 315–334.
[37] W. Qiu, H. Joe, Separation index and partial membership for clustering, Computational Statistics & Data Analysis 50 (3) (2006) 585–603.
[38] A. Ram, A. Sharma, A. S. Jalal, A. Agrawal, R. Singh, An enhanced density based spatial clustering of applications with noise, in: 2009 IEEE International Advance Computing Conference, 2009, pp. 1475–1478.
[39] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical association 66 (336) (1971) 846–850.
[40] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[41] F. Ros, S. Guillaume, Dendis: A new density-based sampling for clustering algorithm, Expert Systems with Applications 56 (2016) 349–359.
[42] R. L. Thorndike, Who belongs in the family?, Psychometrika 18 (4) (1953) 267–276.
[43] M. Wang, W. Zuo, Y. Wang, An improved density peaks-based clustering method for social circle discovery in social networks, Neurocomputing 179 (2016) 219–227.
[44] S. Wang, D. Wang, C. Li, Y. Li, G. Ding, Clustering by fast search and find of density peaks with data field, Chinese Journal of Electronics 25 (3) (2016) 397–402.
[45] J. Xu, G. Wang, W. Deng, Denpehc: Density peak based efficient hierarchical clustering, Information Sciences 373 (2016) 200–218.
[46] R. R. Yager, D. P. Filev, Generation of fuzzy rules by mountain clustering, Journal of Intelligent and Fuzzy Systems 2 (1994) 209–219.
[47] H. Ye, H. Lv, Q. Sun, An improved clustering algorithm based on density and shared nearest neighbor, in: 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, 2016, pp. 37–40.

[48] Z. Yu, H. Chen, J. Liu, J. You, H. Leung, G. Han, Hybrid k-nearest neighbor classifier, IEEE Transactions on Cybernetics 46 (6) (2016) 1263–1275.
[49] T. Zhang, R. Ramakrishnan, M. Livny, Birch: An efficient data clustering method for very large databases, in: ACM SIGMOD Record, vol. 25, 1996, pp. 103–114.
[50] Y. Zhu, K. M. Ting, M. J. Carman, Density-ratio based clustering for discovering clusters with varying densities, Pattern Recognition 60 (2016) 983–997.