KdMutual: A novel clustering algorithm combining mutual neighboring and hierarchical approaches using a new selection criterion

Frédéric Ros, Serge Guillaume, Mohamed El Hadjji, Rabia Riad

KdMutual: a novel clustering algorithm combining mutual neighboring and hierarchical approaches using a new selection criterion

Frédéric Ros[a,*], Serge Guillaume[b], Mohamed El Hajji[c], Rabia Riad[d]

[a] *Laboratory PRISME, Orléans university, France*
[b] *ITAP, Univ Montpellier, INRAE, Montpellier SupAgro, Montpellier, France*
[c] *IRF-SIC, Ibn Zohr university, Morocco*
[d] *ERMAM Team, Ibn Zohr university, Morocco*

Abstract

New clustering algorithms are expected to manage complex data, meaning various shapes and densities while being user friendly. This work addresses this challenge. A new clustering algorithm *KdMutual*[1] driven by the number of clusters is proposed. The idea behind the algorithm is based on the assumption that working with cluster cores rather than considering frontiers makes the clustering process easier. *KdMutual* is based on three steps: The first one aims at identifying the potential core clusters. It relies on mutual neighborhood and includes specific mechanisms to identify and preserve potential core clusters. The second step is based on a constrained hierarchical process that deals with noise. In the last step the potential clusters are selected using a specific ranking criterion and the final partition is built. *KdMutual* combines the best characteristics of density peaks and connectivity-based approaches. It is capable of detecting the non-presence of natural clusters. Tests were carried out to compare the proposal with 14 other clustering algorithms. Using 2-dimensional benchmark datasets of various shapes and densities they showed that *KdMutual* was highly effective

*Corresponding author
Email addresses: frederic.ros@univ-orleans.fr (Frédéric Ros), serge.guillaume@inrae.fr (Serge Guillaume), m.elhajji@uiz.ac.ma (Mohamed El Hajji), r.riad@uiz.ma.ac (Rabia Riad)

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

in matching a ground truth target. It also proved efficient in high dimensions when clusters are well separated. Moreover, it is able to identify clusters of various densities, partially overlapping and including a large amount of noise within spaces of moderate dimension.

Keywords: clustering, mutual neighbors, agglomerative, dissimilarity, density

1. Introduction

There are two kinds of clustering algorithms that are used in two different situations. When the purpose is knowledge discovery or data structure identification, the number of clusters is unknown and the algorithm is expected to propose acceptable partitions. In this case, the number of clusters cannot be a parameter of the algorithm. In contrast, there are also situations where the number of clusters is *a priori* defined. In image processing, the number of different objects may be known, e.g. roads, forests, crop fields, buildings in remote sensing. Similarly, when defining business strategies or market segmentation, the number of groups is given by the user. The present proposal deals with the second case: the number of groups is the main parameter of the algorithm.

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.

When the data are easy to cluster, meaning that the groups are well separated, most of the existing algorithms are likely to yield a good result. Their limitations are well-known in presence of complex data, groups with different sizes, shapes or densities, or the presence of noise. Recent developments have focused on more and more complex structures using new criteria and heuristics: non-linear distances with the kernel k-means, neural-networks, Bregman distances or graph-based algorithms, hierarchical representation with agglomer-

ative algorithms, based on density with *DBSCAN* [1], *Recon-DBSCAN* [2] and *Optics* [3], *Chameleon* [4], *DENCLUE* [5], the mean shift algorithm, *SCDOT* [6] or *Munec* [7].

Among recent algorithms the density peaks clustering algorithm, *DP*, was proposed in 2014 [8] and has become popular. It 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 and the distance, δ, is the minimum distance to any higher density point. 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. *DP* is well known for its ability to identify clusters even in complex situations (partial overlapping, non spherical shape, presence of noise). The pioneering algorithm however suffered from some drawbacks due to its simplistic partition strategy: once a high density point was mishandled, its lower-density neighbors were more likely to be misclassified. Improvements were recently proposed [9, 10, 11, 12, 13].

Globally, recent algorithms have proven to be more efficient but as they include additional heuristics, they produce more complex algorithms that are not really user-friendly. In addition, these heuristics are usually extensively tested using low dimensional data (2 or 3-D) and may exhibit an unstable behavior with increasing, even moderate, dimension spaces.

Practitioners prefer to use efficient clustering algorithms, based on simple ideas, and that are easy to tune even with complex data. Unfortunately, the most popular algorithms are still the classical ones, often in their pioneering version despite available improvements. This work addresses this challenge.

The proposal assumes that a cluster is characterized by the distribution of its neighboring patterns. This distribution is likely to be quite homogeneous in the core of the cluster, with a decreasing level of homogeneity when moving away from the core. This internal structure is expected to be different for distinct clusters: the difference between two clusters stems from their inner

spatial arrangement and their proximity.

The goal for the algorithm is to yield representative clusters that include a significant number of items. Classical algorithms such as *k-means* or *hierarchical clustering* cannot guarantee this result. The practitioner also expects robustness to noise and variability in shape or density, and the ability to handle spaces of moderate dimension ($\approx$10). As the algorithm is driven by the number of clusters, the remaining parameters should be limited in their number as well as in their impact, and easy to understand. Recent studies show that the available algorithms fail in at least one of these requirements [7].

The main ideas the proposed algorithm is based upon are the following ones. First, clusters are easier to distinguish using cores rather than frontiers: the spatial arrangement of patterns is likely to change when moving away from the core. Searching for density peaks is part of the answer as it does not work with frontiers. Unfortunately, there exist configurations where the density peak is not well marked or, in the opposite case, where the cluster includes several peaks. The second idea the proposal is based upon is that connectivity techniques are good at identifying complex-shaped clusters. But it is not easy to find a universal metric able to deal with the different kinds of frontiers. The proposal aims to combine the best of these techniques in an innovative scheme. Mutual neighborhood is useful as no threshold is needed, whether for distance or neighborhood definition. This enables varying density to be managed in the same input space.

The algorithm comprises three steps. The first one is the identification of potential core clusters based on mutual neighborhood in order to protect this structure by forbidding the merging of two cores. In the second step, the potential cores are allowed to grow under constraints to manage noise and scarcely populated area. This is done using a single linkage, which generalizes the mutual neighborhood concept to groups, filtered with noise. In the last step, the potential cores are ranked according to a new selection criterion to define the k final clusters. The criterion is based on three components: the cluster size (cardinality), its compactness assessed by the mean distance between mutual

neighbors and the partition separability, measured by the distance to the nearest cluster of higher size. To build the final partition, noise can be assigned a specific label. The proposal is called *KdMutual* where k indicates that it is driven by the number of clusters, d indicates the role of density in the cluster definition and *Mutual* as this is the central concept for merging.

The rest of the work is organized as follows. Section 2 reviews the most closely related work to *KdMutual* and section 3 is dedicated to the presentation of the algorithm itself. The main idea is illustrated using a simple example illustrating the global behavior and the different steps. The novel ranking criterion is presented in Section 3.4, the numerical experiments carried out are reported in Section 4 while the final remarks and open perspectives are stated in Section 5.

2. Related work

As the proposal includes two distinct contributions, this section deals with the two issues: clustering algorithms and merging criteria.

2.1. Neighborhood-based clustering algorithms

Several reviews of clustering algorithms are available [14, 15, 7]. This section is restricted to the approaches that are close to the proposal. Neighborhood is a transversal notion that can be used either in distance (volume) or density (number of points) based algorithms, or as the basis of the algorithm. A recent neighborhood-based clustering literature review can be found in [7]. The neighborhood definition usually involves 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.

The concept of mutual nearest neighbors was introduced in 1978 [16] in the same period as the pioneering method of Shared Nearest Neighbors [17]. The authors' motivation came "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" [16].

The mutual neighbor concept is useful for describing the inner structure as well as for characterizing the between group proximity. It can be extended to clusters.

Two clusters, c_l and c_m, are mutual nearest neighbor clusters, c_l **Mnnc** c_m, 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 mutual neighbors is the single-link distance between the mutual nearest neighbor clusters:

$$d_{l,m} = \min_{x \in c_l, y \in c_m} d(x, y) \tag{1}$$

A cluster, l, is characterized by [7]:

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

When two clusters, l and m, are merged, the internal descriptors, n and d, become:

$$n = n_l + n_m + 1, \qquad d = \frac{1}{n}(n_l d_l + n_m d_m + d_{l,m}) \tag{2}$$

A similarity index was proposed in [7]. It is based on three distances, without

accounting for the cluster cardinalities. It is computed as:

$$s = \sqrt{s_{l,m}\ s_{m,l}} \tag{3}$$

where $s_{l,m} = \dfrac{min(d_l, d_{l,m})}{max(d_l, d_{l,m})}$, and $s_{m,l}$ is defined in the same way with respect to d_m.

The closer the distances $d_l, d_m, d_{l,m}$, the higher the index and the more suitable the merging of the considered sub-clusters. Neighboring clusters are merged when the similarity index is higher than a threshold, e.g. $s_{th} = 0.2$, in order to avoid noise.

The properties of single linkage, which generalizes the concept of mutual neighborhood to clusters, were studied in [18]. This work also proposed two complementary improvements. First, the hierarchical algorithm forbids the merging of representative clusters, higher than a minimum size, once they have been identified. Second, the single linkage criterion takes into account the local density to make sure the distance involves core points of each group. In this work, the distance between the groups was a weighted average of all the distances between two core points. These two ideas are used in this work, with a simplified distance between groups as detailed in the next section.

2.2. Review of the main criteria

The goal of the criterion is to select the final core of clusters that will not be merged in the last step father algorithm and that are likely to form a *good* partition, i.e., with compact clusters well separated from each other. Two kinds of measures may be of interest: merging criteria and cluster validity indices.

Many operators were proposed and studied in the scientific literature. In this section, some of them are reviewed according to the kind of information they are based on.

Distance-based agglomerative criteria. This family includes single, complete or average linkage, centroid, median or ward criteria. They are all based solely on

proximity and a recent study [18] showed that the single linkage is powerful and able to handle various kinds of shape even if sensitive to noise.

Combining distance and neighborhood. The *Chameleon* approach [4] 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, p and q, are connected by an edge if:

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

where $N^k(p)$ is the set of the k nearest neighbors of p, defined as:
$N^k(p) = \{x_{(1)}, x_{(2)}, \ldots, x_{k)}\}$, with $||x_{(1)} - p|| \leq ||x_{(2)} - p|| \leq \ldots ||x_{(n-1)} - p||$.

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 [21]. 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 to weight 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 about how to choose this influential parameter. There is no clear insight into noise management.

Combining distance and density. A recent study [8] was 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 a r-hypersphere and the distance, δ, is the minimum distance to any higher density point.

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. Then the groups are ranked according to the $\rho\delta$ product in decreasing order.

The main parameter is the radius that defines the neighborhood. It is difficult to define. 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.

Combining distance and shared neighborhood. In [22, 23] the index used penalizes the connectivity, assessed by the number of shared neighbors, by the distance as follows:

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

The cardinality is only used to compute the mean of the distribution. The result is very sensitive to the number of neighbors which is difficult to set in the general case. Moreover, the computation of the neighbors is quite slow even for moderately sized datasets.

Silhouette. The Silhouette index [24] for a cluster, c_i, is computed as the average of a value, s, that characterizes each data point, p, of the cluster, c. The latter involves two components:

$$a(p) = \frac{1}{|c| - 1} \sum_{q \neq p \in c_i} d(p, q) \tag{7}$$

$$b(p) = \min_{j \neq i} \frac{1}{|c_j|} \sum_{q \in c_j} d(p, q) \tag{8}$$

$a(p)$ represents the mean distance between p and all the other points in the cluster while $b(p)$ is the smallest mean distance between p and all the points of another cluster. The s value is defined as follows:

$$s(p) = \frac{b(p) - a(p)}{\max\big(a(p), b(p)\big)} \; if \; |c| > 1, \; 0 \; otherwise \tag{9}$$

According to Eq. (9), $-1 \leq s(p) \leq 1$. This is also the range of the Silhouette index. The higher the value, the better.

Averaged over all the clusters it is used as a cluster validity measure to characterize the whole partition.

Eq. (7) assesses the compactness of the cluster and Eq. (8) its distance from the nearest one. It is worth pointing out that $a(p)$ involves all the internal distances, thus the value is likely to be shape dependent: it is smaller for a disk than for a line. In the next section, a cluster is characterized by the mean distance but between neighbors only. This makes a big difference.

3. The *KdMutual* algorithm

The clustering algorithm is driven by the number of desired clusters. It aims to yield representative clusters even with complex data, groups of varying shape and density, with a high level of noise that makes the clusters overlap. The whole approach includes three steps as shown in Algorithm 1.

Algorithm 1 Overview of the clustering algorithm

1: Input: X (n items), k, s_{th}, *noise*
2: Output: S a partition of X with an optional noise label.
3: $distinguish = TRUE$
4: $\alpha = 0.2$, prop $= 0.7$, $\lambda = min\left(5, \frac{100}{k}\right)$ {Internal parameters}
5: S = CoreClust(X, k, s_{th}, λ, *distinguish*)
{Core cluster building using mutual neighbors - Algorithm 2}
6: **if** (*distinguish*==FALSE) **then**
7: S = CoreClust(X, k, s_{th}, λ, *distinguish*)
{Run again the algorithm with $distinguish = FALSE$}
8: **else**
9: S = HierClust(S, k, *prop*, λ, α)
{Decrease number of cores using hierarchical merging - Algorithm 3}
10: **end if**
11: S = FinalPartition(S, k, *noise*) {Algorithm 5}
12: return S

The core cluster identification may fail when the groups are difficult to distinguish. First the algorithm is run with the corresponding flag set at $TRUE$ (Algorithm 1, line 3). If it is turned to $FALSE$, when the most populated cluster includes half of the items (Algorithm 2, line 34), then the algorithm is run again with this information and the merging process ends when one cluster reaches a size of $\frac{n}{\lambda k}$ (Algorithm 2, lines 3 and 7), and the hierarchical algorithm is skipped (Algorithm 1, line 9).

The first two stages deal only with core clusters ignoring frontiers. The specific goal of the first one is core identification. The number of potential cores is greater than the number of clusters. This step is based on mutual neighbor merging. This concept does not require any distance threshold and thus is able to manage different densities. Potential cores are identified in a parallel way. Once the groups have been designed, it is possible to characterize each of them, the mean of the distance between mutual neighbors is used in this work. Based

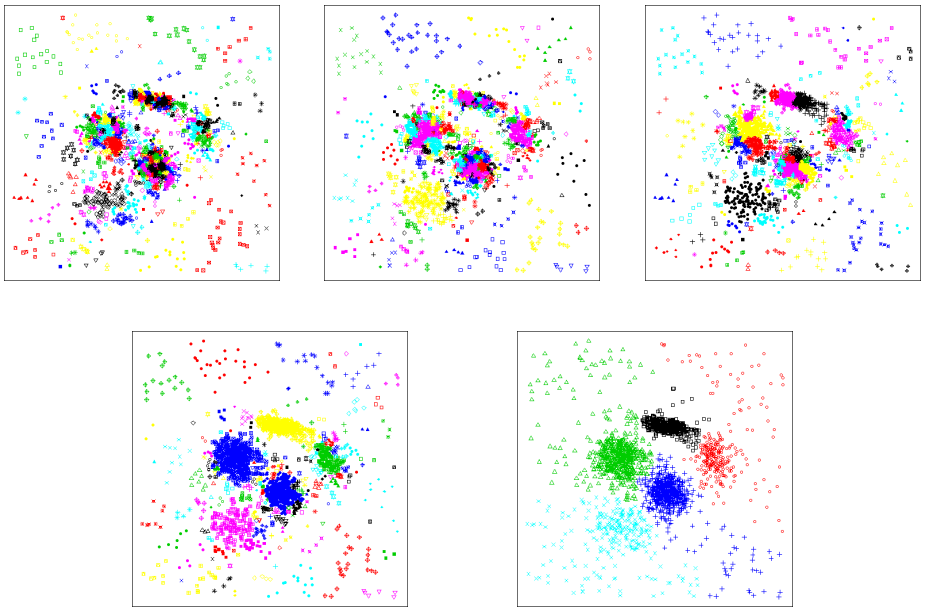

Figure 1: Illustration of the whole process. From left to right and from top to bottom: after 20 iterations, after the first stage of step 1 (Algorithm 2, line 16), at the end of step 1, at the end of step 2, (Algorithm 3), the final partition.

on the local density distribution, items can be labeled as noise and not be taken into account at step 2. The connectivity technique used in the hierarchical process in this step allows for the cores to grow while forbidding between-core merging. This is a sequential process: only one merging is done at each iteration. To achieve the last mergings in step 3, a new criterion is introduced to select the final cores and build the resulting partition. The whole process is illustrated in Figure 1: it can be seen that despite the presence of noise and partial overlapping several sub-clusters are quickly formed and grow without interfering with one another, which makes the identification and building of the final partition easier.

The three steps are now detailed while the criterion is described in Section 3.4.

3.1. Merging of mutual neighbors

The first step is summarized in Algorithm 2. Instead of absolute distance, proximity is based on mutual neighborhood. This makes it possible to manage various densities thanks to local distances between mutual neighbors.

Algorithm 2 CoreClust

Input: X, k, s_{th}, λ, *distinguish*.
Output: S, a partition of X, $MinSize$
$S = X$, $|S| = n$, $size = 0$, $MaxSize = \frac{n}{\lambda k}$
if $(distinguish == TRUE)$ **then**
  Condition=$(size \leq \frac{n}{2}$ AND $|S| \geq \lambda k)$
else
  Condition= $\underset{\forall c_i \subset S}{|c_i|} \leq MaxSize$
end if
while (Condition) **do**
  Compute s_{th} {median of all the mutual neighbor similarities}
  for all $(c_i, c_j \in S \times S)$ **do**
    if (c_i **Mnnc** c_j AND $s_{c_i,c_j} > s_{th}$) **then**
      $Merge(c_i, c_j)$, $|S| = |S| - 1$
    end if
  end for
  Sort $z_i \subset S$, $i \in [1, \lambda k]$, such as $|z_{(1)}| \geq |z_{(2)}| \geq \ldots |z_{(\lambda k)}|$
  $size = 0$
  for $(i \in [1, \lambda k])$ **do**
    $size = size + |z_i|$ {number of items in the λk most populated clusters}
  end for
end while
Tag the λk representative clusters that include at least $n/2$ items.
$MinSize = |z_k|$ {input of algo 3}
while $(|S| \geq \lambda k)$ **do** {Keep merging to get λk clusters}
  for all $(c_i, c_j \in S \times S)$ **do**
    if $\big(not(tag(c_i)\ AND\ tag(c_j)\big)$ **then**
      if (c_i **Mnnc** c_j AND $s_{c_i,c_j} > s_{th}$) **then**
        $Merge(c_i, c_j)$, $|S| = |S| - 1$
      end if
    end if
  end for
end while
if $(max_{c_i \subset S}(|c_i|) \geq n/2)$ **then**
  $distinguish = FALSE$
end if
return S, $MinSize$

The first while loop of Algorithm 2, lines 9-21, aims to identify the potential cores of the clusters. The number of cores is set at a multiple of the desired number of clusters, λk, defined as an internal parameter in line 4 of Algorithm 1. The assumption is that the k clusters can be identified from the λk potential cores defined using mutual neighborhood. The stopping condition depends on the *distinguish* parameter (lines 5 and 7). When the groups are distinguishable, the loop ends when at least λk clusters that represent half of the data size have been identified. The loop usually ends when the size condition is fulfilled. At this stage, the λk potential cores are tagged, line 22, and the size of the k^{th} biggest group is stored to be used in the next step, line 23. Otherwise, when the groups are not distinguishable, the process stops as soon as one group reaches the maximum authorized size defined as $\frac{n}{\lambda k}$ (line 3) to avoid a large coalescence.

To strengthen the robustness of the mutual neighbor concept, not all the mutual neighbors are merged. A threshold is defined on the similarity index, defined in Eq. (3), that involves the mean of the distances between neighbors in each group as well as the between group distance, line 10. It is computed at each iteration, as the median of all the similarity values between mutual neighbors. This way, only half of the pairs, the most similar ones, are merged, avoiding potential outliers.

Illustration 1. This part of the process is illustrated in Figure 2 with synthetic data inspired from [8]. The set is made up of 5000 $2D$-points organized in 5 clusters with different shapes, sizes, partial overlapping and a significant amount of noise. This kind of data cannot be managed by classical algorithms such as *k-means* or hierarchical ones. Even more recent algorithms, such as *DBSCAN*, would be difficult to tune in this case. The proposal is run with $k = 5$ and yields the same partition with $\lambda \in [3, 7]$. The plot corresponds to $\lambda = 3$, meaning that the process ends when the 15 potential cores include half of the data. The total number of groups is 502 and the cardinality of the 5^{th} biggest cluster is $MinSize = 184$.

Then, in a second while loop, lines 24-32, the algorithm continues to merge

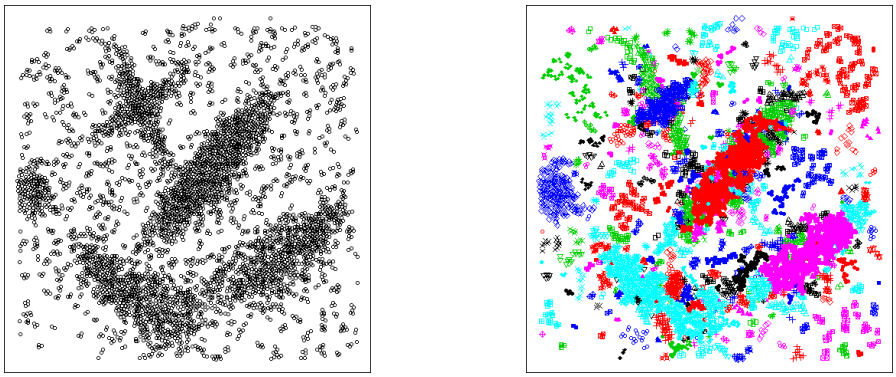

Figure 2: Data (left) and the result of the first while loop of Algorithm 2. The axis labels are the x and y coordinates.

until the number of clusters is λk with an important restriction: the merging between the identified cluster cores is forbidden. Thus tagged clusters may be merged with untagged ones and the groups that were not identified as potential cores can also grow. The merging is controlled by a threshold on the similarity index, e.g. $s_{th} \in [0.2, 0.5]$. The threshold value has a limited impact on the result. The higher its value the less numerous the mergings, meaning that more work is left to the next step.

Illustration 2. Figure 3 shows the impact of three threshold values on the data previously used. At the end of Algorithm 2 the number of groups is respectively 20, 163 and 478 for s_{th} values of 0.2, 0.5 and 0.8. Whatever the threshold, the structure is preserved.

3.2. Hierarchical merging

This step starts with λk cores representing more than 50% of the whole population and ends with a number of clusters $nClust$, such as $k \leq nClust \leq \lambda k$, that represents a higher percentage of the population, e.g. $prop = 70\%$. It carries the same meaning as the previous one but there are two main differences with the mutual merging stage. First, the number of groups is controlled as there is only one merging at each iteration. Second, noise is explicitly taken

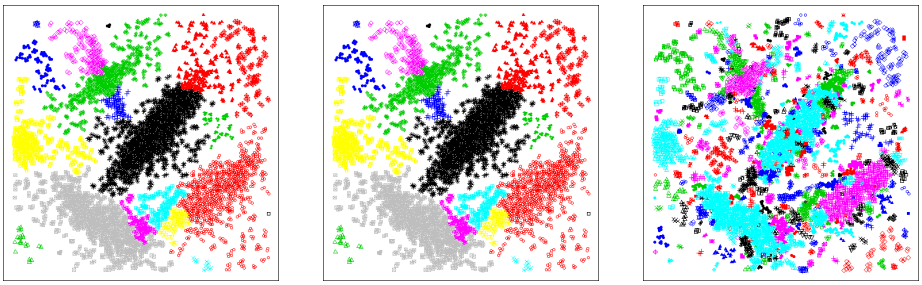

Figure 3: Clusters given by Algorithm 2 using three s_{th} values: 0.2, 0.5 and 0.8. The axis labels are the x and y coordinates.

into account. Mutual neighbor merging is based on the single linkage, whereas the hierarchical process uses a filtered single linkage (fsl): it computes the single linkage criterion but after trimming the density distributions.

$$fsl(c_i, c_j) = sl(c_i^*, c_j^*),\ c^* = c \setminus \{\cup x_i | x_i\ is\ noise\} \tag{10}$$

where sl is the single linkage criterion defined as: $sl(c_i, c_j) = \min_{x \in c_i, y \in c_j} d(x, y)$.

Thus fsl yields the distance between core clusters while the single linkage with noise, sln, proposed in a previous study [18], was a weighted average of the closest core items and the noise between the two groups. This precision was not required for the clustering algorithm and the proposed fsl is appreciably faster. The hierarchical process is described in Algorithm 3.

Algorithm 3 HierClust

1: Input: S, λ, $prop$, $dist$, $MinSize$
2: Output: S
3: $End = FALSE$
4: **for** $(c_i \in S)$ **do**
5: $NoiseLabeling(c_i)$ {Label noise points in cluster c_i, Algorithm 4}
6: **end for**
7: **while** $(|S| \geq k$ AND $End == FALSE)$ **do**
8: $min = \infty$
9: **for** $(c_i, c_j \in S \times S)$ **do**
10: **if** $(fsl(c_i, c_j) < min)$ **then**
11: $c_{w1} = c_i$, $c_{w2} = c_j, min = fsl(c_i, c_j)$ {Eq. (10)}
12: **end if**
13: **end for**
14: $Merge(c_{w1}, c_{w2})$, $|S| = |S| - 1$
15: $d_w = \frac{(|c_{w1}| - 1)d_{w1} + (|c_{w2}| - 1)d_{w2} + fsl(c_{w1}, c_{w2})}{|c_{w1}| + |c_{w2}| - 1}$ {Eq. (2)}
16: $NoiseLabeling(c_w)$ {Label noise points in the new cluster}
17: $size = 0$, $MinSize = max(MinSize, n/50)$
18: **for** $(c_i \in S)$ **do**
19: **if** $(|c_i| > MinSize)$ **then**
20: $size = size + |c_i|$
21: **end if**
22: **if** $(size \geq prop\ n)$ **then**
23: $End = TRUE$
24: **end if**
25: **end for**
26: **end while**
27: return S

The first step of the algorithm, line 5, is noise labeling which is detailed in Algorithm 4.

The local density for a given point in a cluster, $x \in c$, is thus the number of items of the cluster that fall within the hyper-volume centered on x having as radius $r = 2d_c$, d_c being the average between neighbors that characterizes the cluster and that is used to compute the similarity index, line 4.

Algorithm 4 NoiseLabeling

1: Input: c (a cluster), α
2: Output: $noise_c$, binary vector identifying noise points.
3: **for all** $(i \in c)$ **do**
4: $dens_c[i] = \sum_{j(\neq i) \in [1,|c|]} H(2d_c - d(i,j)) \frac{d_c}{max(d_c, d(i,j))}$
{Only the points inside the hypersphere centered in i with radius $2d_c$ are considered. H is the Heaviside function, d the euclidean distance.}
5: **end for**
6: Q = Quartiles($dens_c$)
7: **for all** $(i \in c)$ **do**
8: $noise_c[i] = FALSE$
9: **if** $(dens_c[i] \leq Q_1 - \alpha(Q_3 - Q_1))$ **then**
10: $noise_c[i] = TRUE$
11: **end if**
12: **end for**

The whole distribution is taken into account to identify noise items. The noise detection is based upon the interquartile range. A data item, x, is labeled as noise when $dens(x) < Q_1 - \alpha(Q_3 - Q_1)$, line 9. It has to be highlighted that this density is relative to each group and not to the whole dataset. This is essential to manage clusters with distinct densities. α is defined as an internal parameter of Algorithm 1 in line 3. To display the outliers in the boxplot, the value of $\alpha = 1.5$ was proposed by [19]. The objective in this work is not outlier identification, but to ensure that the points that are not labeled as noise are part of the cluster. A typical value of $\alpha = 0.2$ was used in this paper.

Once noise points have been labeled, mergings are carried out in the main loop of Algorithm 3, lines 7-26. The *for* loop, lines 9-13, identifies the two

clusters to be merged: the ones for which the single link distance, computed without taking noise points into account, is minimum. The characteristics of the new cluster are easily updated according to Eq. (2), line 15, and the noise labeling is done for the new cluster, line 16.

The stopping criterion of Algorithm 3 is *size*: proportion of the population in the representative clusters, line 22. To account for the data structure, the cardinality of the k^{th} largest core identified in the previous step serves as a threshold to define a representative cluster, line 19. However to avoid including small cores due to high variation in the local spatial arrangements, the whole size is also taken into account. The final value, $Minsize = max(|s_k|, n/50)$, line 17, ensures that the representativeness is computed from only well formed groups.

Illustration 3. Figure 4 shows that the result of Algorithm 3 is barely visible on the left part, $s_{th} = 0.2$, but highly noticeable on the right part, $s_{th} = 0.8$: the number of groups decreases from 478 to 174, while only two more mergings were done in the case of $s_{th} = 0.2$, and four with the intermediate value of 0.5.

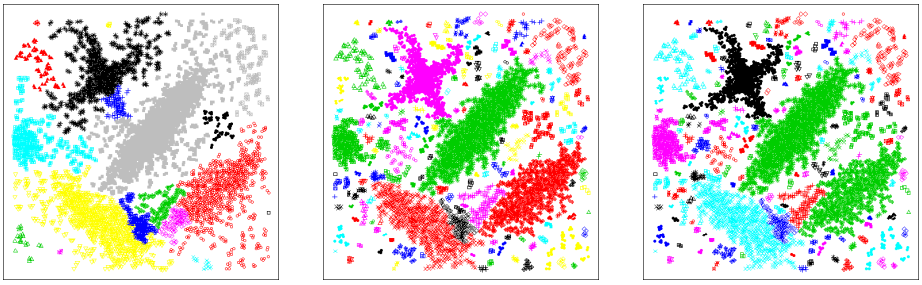

Figure 4: Clusters given by Algorithm 3 for the three values of s_{th}, 0.2, 0.5 and 0.8 from left to right, and $prop = 0.7$. The axis labels are the x and y coordinates.

The proportion parameter is a key feature of the algorithm. If too low, e.g. $prop < 0.6$, there is the risk of confusion in the presence of local peaks in the same cluster: they could be identified as two cores and the decision

in the following steps could be more difficult to make. This risk must not be overestimated as the previous step, Algorithm 2, yielded a well defined structure with λk potential core clusters representing at least half of the data. If too high, when the amount of noise is significant, partially overlapping clusters are likely to be merged.

Illustration 4. With $prop = 0.8$, on the left part of Figure 5, two clusters that partially overlap are merged (black) and, as a result, some noise points are included in a cluster (green). With $prop = 0.9$ the phenomenon is amplified. The four main clusters are merged. The algorithm fails when $prop$ is too high.

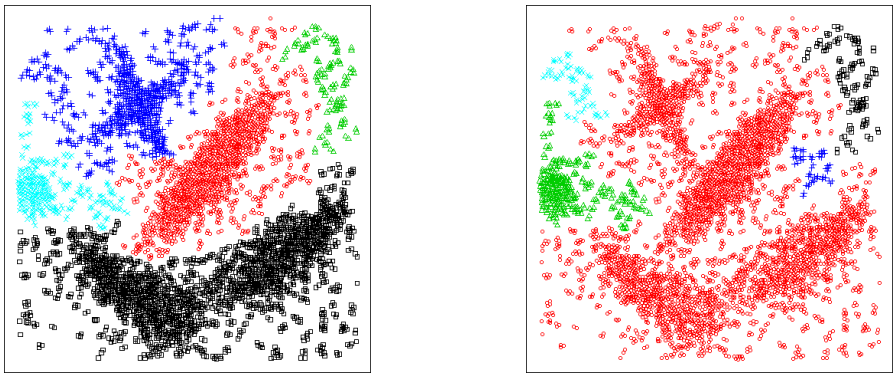

Figure 5: Final partitions with two different values, 0.8 (left) and 0.9 for the *prop* parameter in Algorithm 3 and $s_{limw=0.8}$. The axis labels are the x and y coordinates.

A value of $prop = 0.7$ is a good trade-off as it allows a large amount of noise (30%) to be handled.

3.3. The last merging steps

This is the most difficult part of the whole process. At the beginning of this step the number of core clusters is likely to be higher than k.

The goal in this stage is twofold: select the k final core clusters and carry out the last mergings to yield the final partition. A new criterion is proposed, and is detailed in Section 3.4, to select the final core clusters, line 7 of Algorithm 5.

Algorithm 5 *FinalPartition*

Input: S, k, $noise$
Output: S
$cardmin = min(\frac{n}{100}, 3)$
for all ($c_i \in S$) **do**
  $Crit[i] = 0$
  if ($|c_i| \geq cardmin$) **then**
    $Crit[i] = |c_i|\ d_{Near+}[i]/d_{c_i}$
  end if
end for
Sort z_i, $z_i \subset S$ such as $Crit[z_{(1)}] \geq Crit[z_{(2)}] \geq \ldots Crit[z_{(\lambda k)}]$
Tag the first k clusters
$Merging = TRUE$
while ($|S| \geq k$ AND $Merging == TRUE$) **do**
  $Merging = FALSE$
  for all ($c_i, c_j \in S \times S$) **do**
    if $\big(not(tag(c_i)$ AND $tag(c_j))$ AND c_i **Mnnc** $c_j\ \big)$ **then**
      if $\big($ not(noise) OR (noise AND $s_{c_i,c_j} > s_{th})\big)$ **then**
        $Merge(c_i, c_j)$, $|S| = |S| - 1$, $Merging = TRUE$
      end if
    end if
  end for
end while
return S

The criterion involves the mean distance between neighbors and it can only be computed for clusters with a minimum cardinality, line 3.

The clusters are sorted in decreasing order according to the criterion. The final core clusters are the first k ones, and they are tagged, line 11. Then the mergings are carried out, but they cannot involve two tagged clusters, line 16. Depending on the flag *noise* all items are clustered or the ones for which the similarity index is lower than the limit, $s_{th} = 0.2$, are assigned a noise label, line 17.

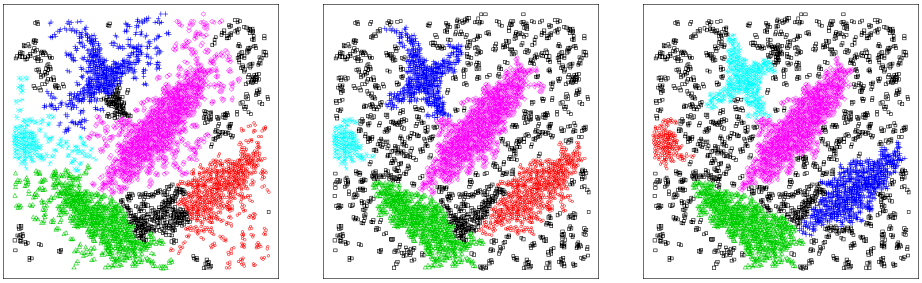

Figure 6: Selected final core clusters given by Algorithm 5 for the three values of s_{th}: 0.2, 0.5 and 0.8 from left to right. The axis labels are the x and y coordinates.

Illustration 5. For the three values of s_{th} the final core clusters are correctly identified as shown in Figure 6. Using $s_{th} = 0.2$, more items are already assigned a cluster while using higher values they are still isolated. The final partitions plotted in Figure 7 are similar. No significant difference can be highlighted. This is a strong asset of the whole algorithm: the progressive structure identification, with stronger control mechanisms, depends only slightly on this parameter.

Complexity analysis. The hierarchical algorithm (Algorithm 3) complexity is $O(n^3)$. The first step (Algorithm 2) is likely to speed up the process by achieving several mutual mergings at each iteration. However, as the number of mergings is data dependent, the whole complexity remains that of the hierarchical algorithm. In our implementation, the distance matrix is stored yielding a space

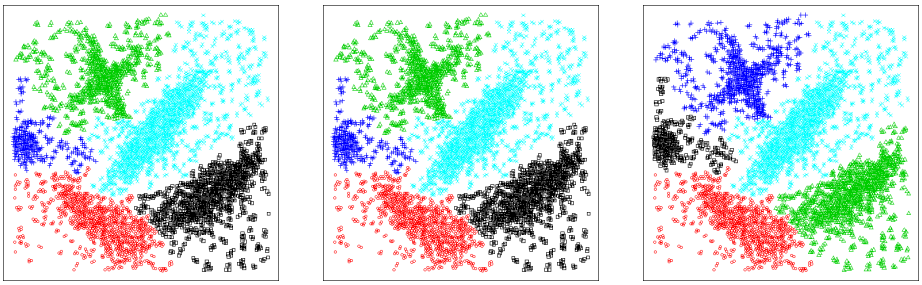

Figure 7: Final partitions given by Algorithm 5 for the three values of s_{th}: 0.2, 0.5 and 0.8 from left to right. The axis labels are the x and y coordinates.

complexity of $O(n^2)$. The algorithm can however be run without the storage of the distance matrix. This would allow for managing larger datasets but with an increase in the running time. For large datasets, it is recommended to preprocess the data with sampling techniques to improve tractability. Smart algorithms such as *ProTraS* [20] are available.

3.4. The proposed criterion

The new criterion combines density and distance. Density is computed as the ratio of the cardinality, $|c|$, to the mean distance between neighbors, d_c. The latter better characterizes a group than the average internal distances as it reflects the tightness of the connection between neighbors whatever the cluster shape.

Distance is considered, but only distance from larger groups:

$$d_{near+}(c) = \min_{|c_i|>|c|} fsl(c, c_i) \tag{11}$$

The main idea is that a potential core, built in the previous steps, is likely to be a core cluster if it is dense and far from another dense group.

First the most populated group is chosen, then the remaining ones are ranked according to decreasing values of the criterion given in Eq. (12).

$$Crit(c) = \frac{|c|\ d_{near+}(c)}{d_c} \tag{12}$$

The criterion is computed only among representative groups, $|c| > n/100$.

The three components of the criterion account for the cardinality of the cluster as representative groups have a significant size, for its compactness assessed using the mean distance between neighbors and the separability of the groups in the partition quantified by the minimum distance to a bigger group.

In order to assess the behavior of the proposed criterion, a comparison was carried out, using the data already used in the previous section, with known alternatives.

The first criterion is based on the single linkage. It selects the clusters for which the minimum filtered distance to the nearest cluster is maximum as shown in Eq. (13).

$$Crit_{sl}(c_i) = \max_{c_j \neq c_i \in S_R} \big(\min(sl(c_i, c_j)) \tag{13}$$

Only the set of representative clusters, S_R, is considered.

The second one is based on Eq. (6). The maximum value is used as a merging criterion. To select the cores the criterion is computed following Eq. (14).

$$Crit_{Lee}(c_i) = \max_{c_j \neq c_i \in S_R} \min \left(\frac{\displaystyle\sum_{i \in c_i} \sum_{j \in c_j} \frac{b_{ij} + b_{ji} + 1}{d(i,j)}}{|c_i| \; |c_j|} \right) \tag{14}$$

There is a difference with Eq. (6): a fixed value, 1, has been added to the number of shared neighbors to account for the pairs of groups that do not share any neighbors. These groups are not likely to be merged, hence the sum of shared neighbors in Eq. (6), but may be interesting to select, especially if the between-group distance is high.

Finally, the *Silhouette* index, Eq. (9), can also be used to rank the cores.

Illustration 6. Figure 8 shows that the three criteria fail to identify the suitable core clusters. The left plot shows that a single distance, even filtered with noise, is not appropriate. The criterion used in the center plot is more complex

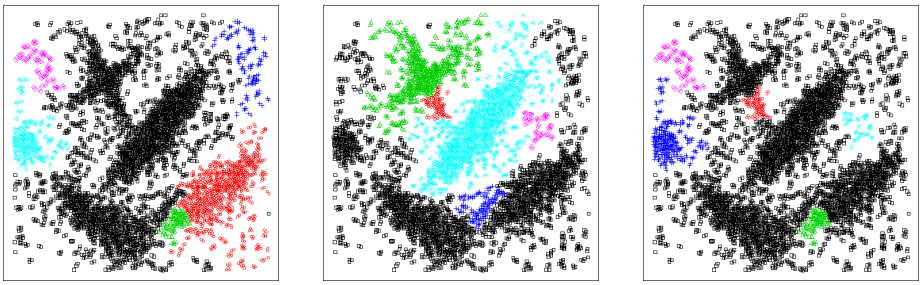

Figure 8: The selected clusters are plotted in color (not black). They are selected according to the single link based criterion, Eq. (13), (left), the combination of neighborhood ($k = 3$) and distance, Eq. (14), (center) and the *Silhouette* index, Eq. (9), (right). The axis labels are the x and y coordinates.

as it combines distance, shared neighborhood and cardinality. Various values of k were tested but none gave the expected result. The cores selected by the *Silhouette* index are the more spherical ones because the index accounts for all the pairwise distances and cardinality is not taken into account.

In contrast, the combination proposed with the new criterion works as expected as shown in Figure 6.

4. Numerical experiments

The *KdMutual* was then compared to rival algorithms using two types of data: $2D$-benchmark datasets known in the literature and data generated using the *genRandomClust* R package[2]. All the methods were evaluated until $d = 10$ and only the best ones for $d = 50$ and $d = 100$.

The clusters have different sizes, densities and separation levels. The data were standardized and when the data size was higher than 5000 a sampling using the *ProTraS* algorithm [20] was carried out in order to limit the runtime.

The objective of the quantitative comparison was to check whether each of

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

the competitive algorithms was able to reach the ground truth target. This choice of external validation was motivated by the lack of a generally accepted internal index validation. Three popular indices were used for partition comparison: the Rand Index [25], the Mutual Information Index [26] and the F-measure [27]. When noise is identified using a specific label in the ground truth, noisy points are not taken into account for index computation.

4.1. Competitive algorithms

The proposal was compared to fourteen selected algorithms, described in Table 1 with their user parameters. In this table, k stands for the number of clusters. This is the unique parameter for the proposal; the internal ones were set at their default values for all the experiments: $\alpha = 0.2$, $prop = 0.7$ and $\lambda = min(5, 100/k)$.

The first competitors are classical algorithms whose limitations are also well known: *k-means* generates spherical clusters, *Single-linkage* hierarchical clustering is sensitive to noise, the *Ward-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* [2]. 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* [17], as well as its variants [28], 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

Table 1: The competitive algorithms

Algorithm	Parameters	Range	Reference
$kmeans^{++}$	k		[14]
Single-linkage	k		[14]
Ward-linkage	k		[14]
DBSCAN	ϵ,Minpts	$[0.05, 0.25]$, $\sqrt{n} \cdot [0.05, 0.25]$	[1]
Recon-DBSCAN	ϵ,θ,τ	$\sqrt{n} \cdot [0.05, 0.25]$, $\epsilon \cdot [1, 5]$	[2]
SNN	nn, $Minpts$	$\sqrt{n} \cdot [0.05, 0.5]$, $nn \cdot [0.2, 0.8]$	[17, 28]
SNN-Radius	nn, $Minpts$, Radius	$\sqrt{n} \cdot [0.05, 0.5]$, $nn \cdot [0.2, 0.8]$,$[0.05, 0.5]$	[17, 28]
MutualClust	k		[16]
***D**ensity**P**eaks*	k		[8]
DP-DataField	k		[29]
Compar-DP	k		[11]
SCDOT	k		[6]
Munec	u	$[0.01, 0.10]$	[7]
HierOpt	k		[18]
KdMutual	k		

others, j, is less 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 *MutualClust* algorithm [16] is based on the mutual neighborhood strength which is quantified by the mutual neighborhood value:

$$mnv(x, y) = \begin{cases} e + f & if \quad x \ \mathbf{Mnn} \ y \\ \infty & otherwise \end{cases} \tag{15}$$

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

The mutual neighboring relationship, **Mnn**, between two items, x and y, is defined as:

$$x \ \mathbf{Mnn} \ y \iff x \in N^k(y) \ AND \ y \in N^k(x) \tag{16}$$

where $N^k(x)$ is the set of the k nearest neighbors of x.

The merging is done according to increasing values of mnv and, in the event of equality, increasing values of the distance between neighbors, 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.

Two improved versions of the Density Peaks algorithm [29, 11], *DP* introduced in Section 1, were considered. In [29], the threshold distance d_c is now automatically set using the potential entropy of the data field from the original dataset. Moreover, 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. In the *comparative density peaks* algorithm [11], the idea is to consider the parameter $\theta_i = \delta_i - \tau_i$ instead of δ_i, τ_i being the distance between the point i and its nearest neighbor of lower density. θ embodies the relative magnitude of δ by comparing with τ and thus helps to identify the potential cluster centers. As in the pioneering version, the automatic selection keeps points with the largest product distance by density.

In the($SCDOT$) [6] method 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 *DP*. A neighboring graph is constructed, as in *Chameleon* clustering [4], 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 [8]. 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.

The *Munec* algorithm [7] is based on a iterative process that merges mutual nearest neighbors. 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. The algorithm is not driven by the number of clusters, but by a single user parameter, $u \in [0.02, 0.1]$, that 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. Whatever its value, no unexpected merging is done.

The *HierOpt* algorithm was detailed in [18]. Its basis is a hierarchical clustering algorithm using the single linkage criterion. Two improvements were proposed to deal with noise. First, the single linkage criterion takes into account the local density to make sure that the distance involves core points of each group. Second, the hierarchical algorithm forbids the merging of representative clusters, higher than a minimum size, once identified. These ideas are still used in the proposal. The unique parameter of this algorithm is the number of clusters. The internal ones are set at their default values.

For each parameter, five values were tested in the range and the best result is stored. For the algorithms that are not driven by the number of clusters, e.g. *DBSCAN*, the result with the closest number of clusters to the required number is kept.

4.2. Comparison of criteria using twelve benchmark datasets

Twelve 2−dimensional datasets, representative of the diversity of situations a clustering algorithm has to cope with, were selected. They are plotted in Figure 9.

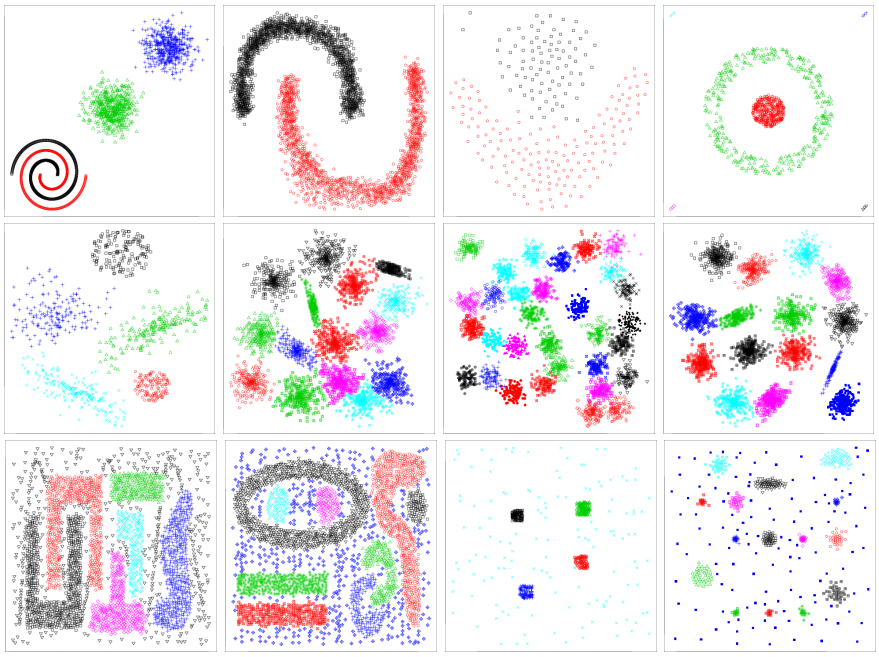

Figure 9: The twelve datasets. The axis labels are the x and y coordinates.

Some are from the data clustering repository of the computing school of Eastern Finland University[3], while others come from a benchmark data for clustering repository [4] or were proposed in the published literature. These datasets, detailed in Table 2, are usually considered for testing new clustering algorithms but they do not represent the diversity of cases a clustering algorithm has to tackle. One homemade dataset, with well separated clusters but of different sizes, was added to complete this diversity.

The selected data include some variability in cluster shape, size, density, amount of noise and degree of separation. The datasets plotted in the first row of Figure 9, from $D1$ to $D4$, show a diversity of shapes. In the second row, from $D5$ to $D8$, the shapes are quite simple, with different elongation and some

[3] https://cs.joensuu.fi/sipu/datasets/

[4] https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/

Table 2: The twelve datasets

	Name	Size	k	Origin
D1	2Sp2glob	2000	4	[30]
D2	BANANA	4811	2	Footnote 4
D3	FLAME	240	2	[31]
D4	TARGET	770	2	Footnote 4
D5	DS850	850	4	Footnote 4
D6	S3	5000	15	Footnote 3
D7	D31	3100	31	[32]
D8	S2	5000	15	Footnote 3
D9	Chameleon	8000	6	[4]
D10	cluto-t7.10k	10000	9	Footnote 4
D11	Zelnik4	622	4	Footnote 4
D12	Home	588	16	Homemade

overlap between groups. In the last row, the datasets include some noise with a diversity of shapes, $D9$ and $D10$, or well separated clusters, $D11$ and $D12$.

The Mutual Information index for the 12 datasets and the 15 algorithms is reported in Table 3.

Table 3: *Mutual Information Index* for the 12 datasets and the 15 algorithms

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
$kmeans^{++}$	0.756	0.325	0.398	0.298	0.805	0.924	0.844	0.932	0.670	0.612	0.869	0.961
Single-linkage	0.856	1.000	**0.024**	**0.063**	0.690	**0.016**	**0.607**	0.791	**0.000**	**0.000**	**0.000**	**0.302**
Ward-linkage	0.761	0.605	0.374	0.204	0.826	0.919	0.952	0.961	0.684	0.648	1.000	0.984
DBSCAN	1.000	1.000	1.000	1.000	1.000	0.952	0.939	0.994	1.000	0.998	1.000	1.000
Recon-DBSCAN	0.856	1.000	0.000	1.000	1.000	0.554	0.682	0.942	**0.000**	**0.000**	1.000	1.000
SNN	**0.713**	0.961	**0.016**	0.939	0.712	0.519	0.843	0.916	0.351	0.379	0.624	0.949
SNN-Radius	1.000	1.000	0.406	0.384	**0.643**	0.362	0.661	0.791	**0.000**	0.525	0.364	0.970
MutualClust	1.000	1.000	**0.000**	1.000	0.690	**0.000**	**0.611**	**0.720**	**0.000**	**0.000**	1.000	0.758
***D**ensity**P**eaks*	1.000	**0.032**	0.413	0.197	1.000	0.961	0.957	0.992	0.778	0.669	1.000	1.00
DP-DataField	1.000	**0.033**	1.000	0.264	1.000	0.959	0.959	0.992	0.728	0.681	1.000	1.000
Compar-DP	1.000	**0.032**	0.413	0.197	1.000	0.961	0.957	0.992	0.778	0.669	1.000	1.000
SCDOT	1.000	1.000	**0.000**	1.000	0.924	0.768	0.858	0.865	0.571	0.821	0.996	0.958
Munec	0.971	1.000	1.000	1.000	0.906	0.932	0.953	0.960	0.894	0.936	1.000	1.000
HierOpt	1.000	1.000	1.000	0.939	1.000	0.945	0.958	0.988	1.000	0.996	1.000	1.000
KdMutual	1.000	1.000	1.000	0.939	1.000	0.945	0.958	0.988	1.000	0.996	1.000	1.000

The diversity of shape, density and size is discriminating. For each dataset, the minimum values are plotted is bold font. The methods mainly based on distance are unable to manage non spherical shapes, and the peak density family also has some trouble when the clusters are of various shapes and close to each other, as illustrated with $D9$.

The means and standard deviations of three indices for this experiment are reported in Table 4.

Table 4: Summary of three indices for the 12 datasets and the 15 algorithms.

	Rand Index		Mutual Inf.		F-measure	
	Mean	Std	Mean	Std	Mean	Std
kmeans^{++}	0.856	0.124	0.699	0.241	0.770	0.136
Single-linkage	0.526	0.313	0.362	0.396	0.511	0.266
Ward-linkage	0.871	0.150	0.743	0.254	0.844	0.145
DBSCAN	**0.996**	0.010	**0.990**	0.021	**0.973**	0.050
Recon-DBSCAN	0.718	0.380	0.669	0.428	0.673	0.365
SNN	0.866	0.133	0.660	0.296	0.585	0.371
SNN-Radius	0.727	0.248	0.592	0.311	0.547	0.308
MutualClust	0.674	0.348	0.565	0.438	0.625	0.318
***D**ensity**P**eaks*	0.869	0.188	0.750	0.349	0.857	0.176
DP-DataField	0.900	0.173	0.801	0.328	0.872	0.181
Compar-DP	0.869	0.188	0.750	0.349	0.857	0.176
SCDOT	0.903	0.146	0.813	0.286	0.816	0.177
Munec	**0.976**	0.036	**0.963**	0.039	**0.949**	0.054
HierOpt	**0.997**	0.005	**0.986**	0.024	**0.992**	0.014
KdMutual	**0.997**	0.005	**0.986**	0.024	**0.992**	0.014

The mean for the Mutual Information Index ranges from 0.362 for *Single-linkage* to 0.990 for *DBSCAN*. Three other algorithms have a mean higher than 0.95: *Munec*, *HierOpt* and *KdMutual*. The three indices yield similar results: *Single-linkage* obtains the poorest value whatever the index while the best ones are achieved by the identified group, *DBSCAN*, *Munec*, *HierOpt* and *KdMutual*. The proposal is part of the group of the most accurate algorithms able to deal with such a diversity of situations.

These results show that *KdMutual* is highly effective when dealing with 2D-data bases containing clusters of various sizes, shapes, densities and in presence of noise.

4.3. High dimension data with different group separation levels

The *genRandomClust* R package[5] was used for partition generation. This is an implementation of the method proposed in [35]. The degree of separation between any cluster and its nearest neighboring cluster can be set at a specified value regarding the separation index proposed in [36]. The cluster covariance matrices can be arbitrary positive definite matrices. 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 to produce more variation in the elongated shapes. The range of variances in the covariance matrix was set at $rangeVar = [1, 30]$. This value is chosen greater than the default one, $[1, 10]$, in order to yield a higher variation in the cluster densities. 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.

Nine sets were generated from dimension 2 to 10 with 4 values of $SepVal$. The number of groups is 5. The size of each group is randomly generated by the R package. The $RangeSize$ increases with the space dimension as follows: $[50, 300]$ from dimension 2 to 5, $[300, 600]$ from dimension 6 to 8, $[300, 800]$ from dimension 9 to 10. To each dataset 20% of noise is added. The generated groups are spherical based, more or less elongated. The difficulty stems from density, level of separation and the large amount of noise. An example in dimension 2 is plotted in Figure 10.

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

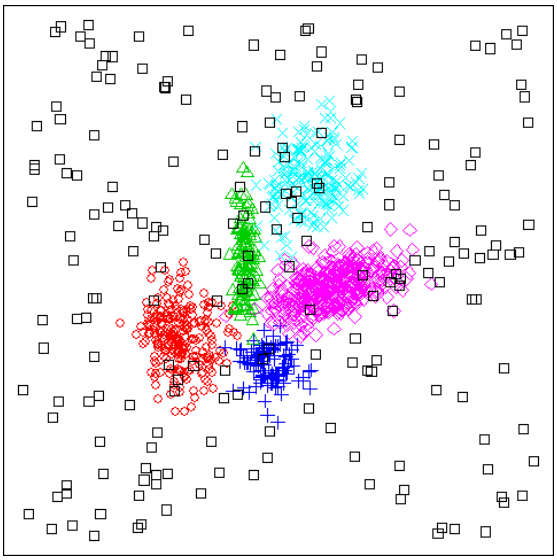

Figure 10: A 2-dimensional dataset generated using the *genRandomClust* R package.

For each configuration, the result is averaged over 10 runs. With $dim = 2$ and $SepVal = 0.1$, some algorithms have a Rand Index lower than 0.5 as shown in Table 5.

Table 5: Algorithms that fail with $dim = 2$ and $SepVal = 0.1$: mean and standard deviation for the Rand Index.

Name	Mean	Std
Single-linkage	0.220	0.02
Recon-DBSCAN	0.311	0.20
SNN-Radius	0.437	0.28
MutualClust	0.223	0.02
SCDOT	0.394	0.19

Some failures come from the intrinsic weakness of the algorithm. This is manifested by a low standard deviation for *Single-linkage* and *MutualClust*. This is also the case for *SCDOT* as the *SCDOT* algorithm based on neighboring is very sensitive to noise. The poor performances given by *Recon-DBSCAN* and *SNN-Radius* result from the difficulty of tuning the algorithm: 5 values for each

of the three parameters were tested, meaning 125 combinations. More trials would have been necessary to get a better performance. This drawback limits the practical use of these algorithms.

The remaining algorithms were tested with higher dimensions and yielded comparable results until $dim = 4$. Further differences appear for $dim = 5$ with $SepVal = 0.2$: two algorithms, based upon neighboring techniques, *DBSCAN* and *SNN*, become less efficient.

When the space dimension increases with a low level of separation $SepVal = 0.1$ or $SepVal = 0.2$ there is a loss of efficiency for all the algorithms. $kmeans^{++}$, *Ward-linkage*, the density peak family, *Munec* and *KdMutual* are however the most resistant. The results with $dim = 8$ and $SepVal = 0.2$ are summarized in Table 6.

Table 6: Results with $dim = 8$ and $SepVal = 0.2$ for the remaining competitive algorithms.

	Rand Index		Mutual Inf.		F-measure	
	Mean	Std	Mean	Std	Mean	Std
$kmeans^{++}$	**0.953**	0.054	0.897	0.081	**0.911**	0.1
Ward-linkage	**0.927**	0.06	0.876	0.073	0.878	0.084
***D**ensity**P**eaks*	0.875	0.059	0.794	0.093	0.815	0.093
DP-DataField	0.875	0.061	0.799	0.098	0.814	0.099
Compar-DP	0.878	0.071	0.839	0.102	0.864	0.110
Munec	**0.932**	0.028	**0.904**	0.03	**0.917**	0.04
HierOpt	0.837	0.118	0.752	0.131	0.789	0.091
KdMutual	**0.966**	0.043	**0.925**	0.053	**0.944**	0.069

From $dim = 9$ and $sepval = 0.1$, only two competitors ($kmeans^{++}$ and *Ward-linkage*) yield acceptable results: their means for $dim = 9$ and $dim = 10$ are respectively 0.898 and 0.864. *KdMutual* reaches a mean of 0.901. The scores of the remaining methods are seriously degraded: *Munec* drops below 0.3 as well as *HierOpt*, ***D**ensity**P**eaks* to 0.65.

For these three methods, the experiment was extended to $dim = 50$ and

$dim = 100$. In order to obtain representative results, 100 data sets were generated for each space dimension. The results are summarized in Table 7.

Table 7: *Mutual Information Index* averaged over 100 runs for $SepVal = 0.1$.

	$d = 50$		$d = 100$	
	Mean	Std	Mean	Std
$kmeans^{++}$	0.676	0.120	0.577	0.142
Ward-linkage	0.652	0.132	0.541	0.151
KdMutual	0.691	0.129	0.549	0.151

With increasing dimensions and the same separation level ($sepval = 0.1$) the mean decreases for all the methods and the standard deviation increases. The clusters are still identified in such configurations with a high dimension and low separation level but they cannot be fully distinguished.

In high dimension spaces the distance measure becomes less meaningful, this is known as the "curse of dimensionality" and was analyzed in various contexts [37]. The high dimensional space issue can be addressed using either subspace selection or space transformation techniques.

4.4. Statistical results: synthesis

To assess how significant the differences between *KdMutual* and the other studied algorithms are, a Wilcoxon signed-rank test was performed on the mean *Mutual Information* index values. The data sets for which results are not discriminant are not considered. This is the case with the R-data with $d > 8$ and low separation level: only three methods reach acceptable scores. The data sets generated from the R-package are split in two categories. In the first category, all the results using $sepval = 0.1$ and $sepval = 0.2$ from $d = 2$ to $d = 8$ were combined. To assess the degradation of the results for most methods for $d \geq 8$ the second category focused only on $sepval = 0.1$ from $d = 9$ to $d = 10$. The test was based upon the sign of the difference of the observed values, and the R

Project[6] implementation was used. The results are given in Table 8.

Table 8: statistical results: p_{value} with the alternative hypothesis KdMutual is "greater" than its competitors based on the paired option. Values higher than 0.05 are printed in bold font.

	2D data (2)	R data(2-8) sepval=$\{0.1; 0.2\}$	R data(9-10) sepval=0.1
kmeans^{++}	$4.88\ 10^{-4}$	**0.99**	**0.09**
Single-linkage	$3.83\ 10^{-3}$	$4.54\ 10^{-14}$	$4.78\ 10^{-7}$
Ward-linkage	$3.85\ 10^{-3}$	**1.00**	0.04
DBSCAN	**0.42**	$1.07\ 10^{-9}$	$4.11\ 10^{-7}$
Recon-DBSCAN	0.03	$1.6\ 10^{-12}$	$4.09\ 10^{-7}$
SNN	$3.85\ 10^{-3}$	$2.69\ 10^{-14}$	$2.32\ 10^{-10}$
SNN-Radius	$5.92\ 10^{-3}$	$2.8\ 10^{-14}$	$2.52\ 10^{-10}$
MutualClust	0.01	$2.69\ 10^{-14}$	$4.08\ 10^{-7}$
***D**ensity**P**eaks*	**0.08**	**0.96**	$4.06\ 10^{-7}$
DP-DataField	**0.20**	**0.97**	$4.18\ 10^{-7}$
Compar-DP	**0.08**	**0.99**	$4.21\ 10^{-7}$
SCDOT	0.01	$3.42\ 10^{-14}$	$2.03\ 10^{-8}$
Munec	**0.10**	0.04	$4.78\ 10^{-7}$
HierOpt	**1.00**	**0.07**	$4.08\ 10^{-7}$

These results show that *Kdmutual* is extremely competitive as it is better (at a confidence level $\alpha = 0.05$) than its competitors in many cases. The configurations for which the alternative hypothesis is rejected are in bold font in Table 8. For $2D$-data, the *Kdmutual* performances are not statistically different from those of*DBSCAN*, *DP-DataField*, *Munec* and *HierOpt*. Using the R-data, only *Ward-linkage*, and the density peak algorithms when $d \leq 8$, do not give significantly different results. The test does not show significant differences between the remaining algorithms for $d = 50$ and $d = 100$.

[6] https://r-project.org, wilcox.test function with the parameters paired=TRUE and alternative="greater"

4.5. Running time

The running time is an important characteristic of this kind of algorithm. The average time, in seconds, is reported in Table 9 for datasets of 2500 items with 5 input variables.

Table 9: Running time (s) for a $dim = 5$ dataset and the 15 algorithms.

kmeans^{++}	0.017	
***D**ensity**P**eaks*	0.54	++
DP-DataField	0.55	
Munec	3.2	
DBSCAN	7.2	
Recon-DBSCAN	7.2	
SNN	7.5	+
SNN-Radius	7.5	
KdMutual	7.7	
Compar-DP	8.1	
MutualClust	12	
Single-linkage	42	
SCDOT	43	−
HierOpt	69	
Ward-linkage	139	−−

For three algorithms this time is lower than one second, *kmeans*$^{++}$, ***D**ensity**P**eaks* and *DP-DataField*, while others are quite slow, e.g. *Single-linkage*, *SCDOT*, *HierOpt*, the slowest algorithm, 139 s, being *Ward-linkage*. The proposal is quite fast with a running time of less than 8 s.

5. Conclusion

A new clustering algorithm, driven by the number of clusters, was proposed. It includes several steps. The first mergings are based on mutual neighborhood in order to identify potential core clusters. In this way many cores are allowed

to grow at the same time whatever their local density as no distance parameter is required. The second step is a hierarchical merging using a filtered distance to deal with noise. The last step selects the core clusters and builds the final partition.

The potential clusters are ranked according to a new selection criterion based on three types of information: cluster size (cardinality), its compactness assessed by the mean of the between neighbors distances and its distance from the nearest larger neighbor. It proved more powerful than competitors.

The transition between steps is based on either the number of cores or the proportion of the whole population they include. These parameters are said to be internal or auxiliary. They were set at a default value in all the reported experiments. The number of initial cores was defined as 5 times the number of clusters, the proportion of the population was set at 50% and 70% for the two steps. The algorithms are quite simple: no specific heuristics is needed.

When the first step yields a unique cluster that includes 50% of the whole set that means that the clustering process failed due to a large overlapping. In this case, the algorithm is run again until one cluster reaches a significant size. Then the hierarchical step is skipped to rank and select the cores before building the final partition.

The proposal was compared to 14 alternative methods. The ones based on the detection of density peaks are unable to deal with complex shapes but are highly efficient, even in high dimensions, when clusters are defined by their density. Methods based on item connectivity, such as $DBSCAN$, can handle complex shapes but they are limited by their global setting: it may be difficult to find the set of parameters that can deal with the local density variations. The proposal combines the best characteristics of these two kinds of algorithms. *Kd-Mutual* can manage complex shapes while still being robust in higher dimension spaces. From the user point of view, it is easy to tune as the unique parameter is the number of clusters and it is among the fastest algorithms studied in these experiments.

One of the perspectives is to be able to use this kind of algorithm in large

dimension spaces. The standard procedure would require a preprocessing step to make an unsupervised feature selection [38]. In this case, *KdMutual* would be applied in different small spaces. An alternative interesting way would consist in investigating strategies of subspace clustering [39]. These new techniques seem to be promising for identifying clusters in different subsets of dimensions.

Another perspective would be to adapt the algorithm to design a version that does not require the number of clusters. The basis would be a high number of cores, for instance 100, and the criterion to rank them. The number of clusters would result from the distribution of the criterion values based on an Elbow method or it could be given by an evolutionary algorithm with a fitness function that combines the criterion values and the number of clusters.

References

[1] 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.

[2] Y. Zhu, K. M. Ting, M. J. Carman, Density-ratio based clustering for discovering clusters with varying densities, Pattern Recognition 60 (2016) 983–997.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, in: ACM SIGMOD international conference on Management of data, volume 28, ACM Press, 1999, pp. 49–60.

[4] G. Karypis, E.-H. Han, V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, Computer 32 (1999) 68–75.

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

[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] F. Ros, S. Guillaume, Munec: A mutual neighbor-based clustering algorithm, Information Sciences 486 (2019) 148–170.

[8] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492–1496.

[9] M. Du, S. Ding, H. Jia, Study on density peaks clustering based on k-nearest neighbors and principal component analysis, Knowledge-Based Systems 99 (2016) 135–145.

[10] X. Xu, S. Ding, Z. Shi, An improved density peaks clustering algorithm with fast finding cluster centers, Knowledge-Based Systems 158 (2018) 65–74.

[11] Z. Li, Y. Tang, Comparative density peaks clustering, Expert Systems with Applications 95 (2018) 236–247.

[12] M. Parmar, D. Wang, X. Zhang, A.-H. Tan, C. Miao, J. Jiang, Y. Zhou, Redpc: A residual error-based density peak clustering algorithm, Neurocomputing 348 (2019) 82–96.

[13] J. Jiang, Y. Chen, D. Hao, K. Li, Dpc-lg: Density peaks clustering based on logistic distribution and gravitation, Physica A: Statistical Mechanics and its Applications 514 (2019) 25–35.

[14] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (2010) 651–666.

[15] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, C.-T. Lin, A review of clustering techniques and developments, Neurocomputing 267 (2017) 664–681.

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

[17] R. A. Jarvis, E. A. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Transactions on computers 100 (1973) 1025–1034.

[18] F. Ros, S. Guillaume, A hierarchical clustering algorithm and an improvement of the single linkage criterion to deal with noise, Expert Systems with Applications 128 (2019) 96–108.

[19] J. W. Tukey, Exploratory Data Analysis, Behavioral Science: Quantitative Methods, Addison-Wesley, Reading, Mass., 1977.

[20] F. Ros, S. Guillaume, Protras: A probabilistic traversing sampling algorithm, Expert Systems with Applications 105 (2018) 65–76.

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

[22] J.-S. Lee, S. Olafsson, Data clustering by minimizing disconnectivity, Information Sciences 181 (2011) 732–746.

[23] 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.

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

[25] W. M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical association 66 (1971) 846–850.

[26] T. M. Cover, J. A. Thomas, Elements of information theory, 2nd ed., John Wiley & Sons, 2006.

[27] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel, et al., Performance measures for information extraction, in: Proceedings of DARPA Broadcast News Workshop, 1999, pp. 249–252.

[28] 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.

[29] 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 (2016) 397–402.

[30] J. Piantoni, K. Faceli, T. C. Sakata, J. C. Pereira, M. C. de Souto, Impact of base partitions on multi-objective and traditional ensemble clustering algorithms, in: International Conference on Neural Information Processing, Springer, 2015, pp. 696–704.

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

[32] C. J. Veenman, M. J. T. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Transactions on pattern analysis and machine intelligence 24 (2002) 1273–1280.

[33] I. Kärkkäinen, P. Fränti, Gradual model generator for single-pass clustering, Pattern Recognition 40 (2007) 784–795.

[34] P. Fränti, O. Virmajoki, V. Hautamäki, Fast agglomerative clustering using a k-nearest neighbor graph, IEEE Trans. on Pattern Analysis and Machine Intelligence 28 (2006) 1875–1881.

[35] W. Qiu, H. Joe, Generation of random clusters with specified degree of separation, Journal of Classification 23 (2006) 315–334.

[36] W. Qiu, H. Joe, Separation index and partial membership for clustering, Computational Statistics & Data Analysis 50 (2006) 585–603.

[37] M. Steinbach, L. Ertöz, V. Kumar, The challenges of clustering high dimensional data, in: New directions in statistical physics, Springer, 2004, pp. 273–309.

[38] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, X. Zhou, $l_{2,1}$-norm regularized discriminative feature selection for unsupervised, in: Twenty-Second International Joint Conference on Artificial Intelligence, 2011, p. paper 267.

[39] P. Agarwal, S. Mehta, Subspace clustering of high dimensional data using differential evolution, in: Nature-Inspired Algorithms for Big Data Frameworks, IGI Global, 2019, pp. 47–74.