Expert Systems With Applications 56 (2016) 349–359

Contents lists available at ScienceDirect

Expert Systems With Applications

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

DENDIS: A new density-based sampling for clustering algorithm

Frédéric Ros[a,*], Serge Guillaume[b]

[a] Laboratory PRISME, Orléans university, 8 Rue Léonard de Vinci - 45072 Orléans, France
[b] Irstea, UMR ITAP, 361, rue J.F. Breton B.P. 5095 34196, Montpellier Cedex 5, France

ARTICLE INFO

Article history:
Received 6 November 2015
Revised 6 February 2016
Accepted 3 March 2016
Available online 17 March 2016

Keywords:
Density
Distance
Space coverage
Clustering
Rand index

ABSTRACT

To deal with large datasets, sampling can be used as a preprocessing step for clustering. In this paper, an hybrid sampling algorithm is proposed. It is density-based while managing distance concepts to ensure space coverage and fit cluster shapes. At each step a new item is added to the sample: it is chosen as the furthest from the representative in the most important group. A constraint on the hyper volume induced by the samples avoids over sampling in high density areas. The inner structure allows for internal optimization: only a few distances have to be computed. The algorithm behavior is investigated using synthetic and real-world data sets and compared to alternative approaches, at conceptual and empirical levels. The numerical experiments proved it is more parsimonious, faster and more accurate, according to the Rand Index, with both k-means and hierarchical clustering algorithms.

1. Introduction

Summarizing information is a key task in information processing, either in data mining, knowledge induction or pattern recognition. Clustering (Ling, 1981) is one of the most popular techniques. It aims at grouping items in such a way that similar ones belong to the same cluster and are different from the ones which belong to other clusters. Many methods (Andreopoulos, An, Wang, & Schroeder, 2009) have been proposed to identify clusters according to various criteria. Some of them (Nagpal, Jatain, & Gaur, 2013) are based on an input space partition (k-means, spectral clustering, Clarans) or grid techniques (like Sting or Clique), others are density-based (Dbscan, Denclue, Clique). Some of these techniques benefit a tree implementation: Birch, Cure, Diana, Chameleon, Kd-tree.

Algorithms are becoming more and more sophisticated in order to be able to manage data with clusters of various shapes and densities. This leads to an increased computational cost which limits their practical use, especially when applications concern very large database like records of scientific and commercial applications, telephone calls, etc. Clearly, most of mature clustering techniques address small or medium databases (several hundreds of patterns) but fail to scale up well with large data sets due to an excessive computational time. Therefore, in addition to the usual performance requirements, response time is of major concern to most data clustering algorithms nowadays. Obviously, algorithms with quadratic or exponential complexity, such as hierarchical approaches, are strongly limited, but even algorithms like *k-means* are still slow in practice for large datasets.

While some approaches aim to optimize and speed up existing techniques (Chiang, Tsai, & Yang, 2011; Viswanath, Sarma, & Reddy, 2013), sampling appears as an interesting alternative to manage large data sets. In our case, sampling is a preprocessing step for clustering and clustering is assessed according to cluster homogeneity and group separability. This calls for two basic notions: density and distance. Clusters can be defined as dense input areas separated by low density transition zones. Sampling algorithms are based upon these two notions, one driving the process while the other is more or less induced.

Various techniques have been proposed in the abundant literature. Some algorithms estimate local density, using neighborhood or kernel functions (Kollios, Gunopulos, Koudas, & Berchtold, 2003), in order to bias the random sampling to make sure small clusters are represented in the sample (Ilango & Mohan, 2010; Palmer & Faloutsos, 2000). Others work at a global scale, like the popular *k-means* or evolutionary approaches (Naldi & Campello, 2015). In the former, the number of representatives is a priori set, each center induces an attraction basin. The third category includes incremental algorithms. They can be driven either by the attraction basin size (Yang & Wu, 2005) favoring the density search or by distance concepts (Rosenkrantz, Stearns, & Lewis, 1977; Sarma, Viswanath, & Reddy, 2013) promoting the coverage aspect. Incremental algorithms differ in the heuristics introduced to balance the

* Corresponding author. Tel.: +33 238642588.
E-mail addresses: frederic.ros@univ-orleans.fr, frederic.ros@free.fr (F. Ros), serge.guillaume@irstea.fr (S. Guillaume).

http://dx.doi.org/10.1016/j.eswa.2016.03.008
0957-4174/

density and distance concepts, and also in the parametrization. A comparison of algorithms for initializing *k-means* can be found in Celebi, Kingravi, and Vela (2013).

Fulfilling the two conflicting objectives of the sampling, ensuring small clusters coverage while favoring high local density areas, especially around the modes of the spatial distribution, with a small set of meaningful parameters, is still an open challenge.

The goal of this paper is to introduce a new incremental algorithm (http://frederic.rosresearch.free.fr) to meet these needs. DENDIS combines density and distance concepts in a really innovative way. Density-based, it is able to manage distance concepts to ensure space coverage and fit cluster shapes. At each step a new item is added to the sample: it is chosen as the furthest from the representative in the most important group. A constraint on the hyper volume induced by the samples avoids over sampling in high density areas. The attraction basins are not defined using a parameter but are induced by the sampling process. The inner structure allows for internal optimization. This makes the algorithm fast enough to deal with large data sets.

The paper is organized as follows. Section 2 reports the main sampling techniques. Then DENDIS is introduced in Section 3 and compared at a conceptual level to alternative approaches in Section 4. The optimization procedure is detailed in Section 5. Section 6 is dedicated to numerical experiments, using synthetic and real world data, to explore the algorithm behavior and to compare the proposal with concurrent approaches. Finally Section 7 summarizes the main conclusions and open perspectives.

2. Literature review

The simplest and most popular method to appear was uniform random sampling, well known to statisticians. The only parameter is the proportion of the data to be kept. Even if some work has been done to find the optimal size by determining appropriate bounds (Guha, Rastogi, & Shim, 1998), random sampling does not account for cluster shape or density. The results are interesting from a theoretical point of view (Chernoff, 1952), but they tend to overestimate the sample size in non worst-case situations.

Density methods (Menardi & Azzalini, 2014) assume clusters are more likely present around the modes of the spatial distribution. They can be grouped in two main families for density estimation: space partition (Ilango & Mohan, 2010; Palmer & Faloutsos, 2000) and local estimation, using neighborhood or kernel functions (Kollios et al., 2003).

The main idea of these methods is to add a bias according to space density, giving a higher probability for patterns located in less dense regions to be selected in order to ensure small cluster representation. The results are highly dependent upon the bias level and the density estimation method. The local estimation approaches (*kernel* or *k-nearest-neighbors*) require a high computational cost. Without additional optimization based on preprocessing, like the bucketing algorithm (Devroye, 1981), they are not scalable. However this new step also increases their complexity.

Distance concepts are widely used in clustering and sampling algorithms to measure similarity and proximity between patterns. The most popular representative of this family remains the *k-means* algorithm, and its robust version called *k-medoids*. It has been successfully used as a preprocessing step for sophisticated and expensive techniques such as hierarchical approaches or Support Vector Machine algorithms (SVM) (Xiao, Liu, Hao, & Cao, 2014). It is still the subject of many studies to improve its own efficiency and tractability (Khan & Ahmad, 2013; Lv et al., 2015; Zhong, Malinen, Miao, & Fränti, 2015). The proposals are based on preprocessing algorithms which are themselves related to sampling or condensation techniques (Arthur & Vassilvitskii, 2007; Zahra et al., 2015) including evolutionary algorithms (Hatamlou, Abdullah, & Nezamabadi-pour, 2012; Naldi & Campello, 2015). These algorithms are still computationally expensive (Tzortzis & Likas, 2014).

While the *k-means* is an iterative algorithm, whose convergence is guaranteed, some single data-scan distance based algorithms have also been proposed, such as *leader family* (Sarma et al., 2013; Viswanath et al., 2013) clustering or the furthest-first-traversal (fft) algorithm (Rosenkrantz et al., 1977). The pioneering versions of distance based methods are simple and fast, but they also are limited in the variety of shapes and densities they are able to manage. When improved, for instance by taking density into account, they become more relevant but their overall performance depends on the way both concepts are associated and, also, on the increase of the computational cost. The *mountain method* proposed by Yager and its modified versions (Yang & Wu, 2005) are good representatives of hybrid methodologies as well as the recent work proposed by Feldman, Faulkner, and Krause (2011). Density is managed by removing from the original set items already represented in the sample.

Strategies usually based on stratification processes have also been developed to improve and speed up the sampling process (Gutmann & Kersting, 2007). Reservoir algorithms (Al-Kateb & Lee, 2014) can be seen as a special case of stratification approaches. They have been proposed to deal with dynamic data sets, like the ones to be found in web processing applications. These method need an accurate setting to become really relevant.

Even if the context is rather different, Vector Quantization techniques (Chang & Hsieh, 2012), coming from the signal area especially for data compression, involve similar mechanisms. The objective is to provide a codebook representative of the original cover without distortion. The *LBG* algorithm and its variations (Bardekar & Tijare, 2011) appear to be the most popular. The methods are incremental, similar to the global k-means family approaches (Bagirov, Ugon, & Webb, 2011; Likas, Vlassis, & Verbeek, 2003), as at each step a new representative is added according to an appropriate criterion. Recent literature (Ma, Pan, Li, & Fang, 2015; Tzortzis & Likas, 2014) reports the difficulty to find the balance between length of codebook entries, its quality and time required for its formulation.

This short review shows that sampling for clustering techniques have been well investigated. Both concepts, density and distance, as well as the methods have reached a good level of maturity. Approaches that benefit from a kd-tree implementation (Nanopoulos, Manolopoulos, & Theodoridis, 2002; Wang, Wang, & Wilkes, 2009) seem to represent the best alternative, among the known methods, in terms of accuracy and tractability. However, they are highly sensitive to the parameter setting. The design of a method that would be accurate and scalable allowing to process various kinds of large data sets with a standard setting, remains an open challenge.

3. DENDIS: the proposed sampling algorithm

The objective of the algorithm is to select items from the whole set, T, to build the sample set, S. Each item in S is called a representative, each pattern in T is attached to its closest representative in S. The S set is expected to behave like the T one and to be as small as possible. *DENDIS* stands for DENsity and DIStance, meaning the proposal combines both aspects.

Overview of the algorithm It is an iterative algorithm that add a new representative at each step in order to reach two objectives. Firstly, ensure high density areas are represented in S, and, keeping in mind the small size goal, avoiding over representation. The second objective aims at homogeneous space covering to fit cluster shapes. To deal with the density requirement the new representative is chosen in the most populated set of attached patterns. For space covering purposes, the new representative is the

furthest from the existing one. Over representation is avoided by a dynamic control of both parameters that define density: volume and cardinality. The latter is defined according to the unique input parameter, *granularity*, noted g_r, and the initial set size, n. The product defines the W_t threshold: the minimum number of patterns attached to a given representative. The volume is estimated by the maximum distance between an attached pattern and the representative.

The two steps of the algorithm

The algorithm is made up of two steps. The first one, Algorithm 1, is based on space density while taking into account distance notions. The second one, Algorithm 2, can be seen as a post processing step which aims at not selecting outliers as representatives.

Algorithm 1 The density-based sampling algorithm.

1: Input: $T = \{x_i\}$, $i = 1 \ldots, n$, g_r
2: Output: $S = \{y_j\}$, Ty_j, $j = 1, \ldots, s$
3: Select an initial pattern $x_{init} \in T$
4: $S = \{y_1 = x_{init}\}$, $s = 1$
5: ADD=TRUE, $K = 0.2$, $W_t = n\ g_r$
6: **while** (ADD==TRUE) **do**
7: **for all** $x_l \in T \setminus S$ **do**
8: Find $d_{near}(x_l) = \min_{y_k \in S} d(x_l, y_k)$
9: $T_{y_k} = T_{y_k} \cup \{x_l\}$ {Set of patterns represented by y_k}
10: **end for**
11: **for all** $y_k \in S$ **do**
12: Find $d_{max}(y_k) = \max_{x_m \in T_{y_k}} d(x_m, y_k)$
13: Store $d_{max}(y_k)$, $x_{max}(y_k)$
{where $d_{max}(y_k) = d(x_{max}(y_k), y_k)$}
14: **end for**
15: ADD=FALSE
16: Sort $y_{(1)}, \ldots, y_{(s)}$ with $|T_{y_{(1)}}| \geq \cdots \geq |T_{y_{(s)}}|$
17: **for all** y_k in S **do**
18: **if** $(|T_{y_k}| < W_t)$ **then**
19: break
20: **end if**
21: $\alpha_k = max(\frac{W_t}{|T_{y_k}|}, K)$
22: **if** $(d_{max}(y_k) \geq \alpha_k \overline{d_{max}(y_k)}_{y_k \in S})$ **then**
23: $x_* = x_{max}(y_k)$
24: ADD=TRUE, break
25: **end if**
26: **end for**
27: **if** (ADD==TRUE) **then**
28: $S = S \cup \{x_*\}$, $s = s + 1$
29: **end if**
30: **end while**
31: Run the post processing algorithm {Algorithm 2}
32: **return** S, T_{y_k}, $k = 1, \ldots, s$

The unique input parameter, except the data to be sampled, is called *granularity*, and noted g_r. Data independent, it is combined with the whole set cardinality to define a threshold, W_t, on the number of patterns attached to a given representative (line 5). The *granularity* impacts the S size, the lower g_r the higher the number of representatives. However, the relation between both is not deterministic, like in Sample Random Sampling. The number of patterns attached to a representative also depends on a volume estimation as explained below.

The first sample is randomly chosen (line 3). Then the algorithm iterates to select the representatives (lines 6–30). In a preparation phase, each not selected pattern, $x \in T \backslash S$[1], is attached to the closest selected one in S (lines 7–10) and, for each set T_{y_k}, the algorithm searches for the furthest attached pattern, $x_{max}(y_k)$, located at distance $d_{max}(y_k) = d(x_{max}(y_k), y_k)$ (lines 11–14).

Then a new representative is selected (lines 15–26). The selected items are sorted according to the cardinality of the set of patterns they are the representative (line 16) and these sets, T_{y_k}, are analyzed in decreasing order of weight. Each of them is split when two conditions are met (lines 18 and 22). The first one deals with the number of attached patterns: it has to be higher than the threshold, $W_t = n\ g_r$. Without any additional constraint, the representatives would tend to have the same number of patters attached, close to W_t. This behavior would lead to an over size sample in high density areas. Therefore, the other condition is related to the density, controlled by the induced hyper volume. At the beginning of the process, the d_{max} values are quite high, as well as the cardinalities $|T_{y_k}|$. The fraction of minimum volume is then limited by an upper bound, K. This allows for the space to be covered in an homogeneous way, the d_{max} values tend to a lower mean with a lower deviation. In the last steps of the process, α_k dynamically promotes dense areas in order for the sample to reflect the original densities: the larger the cardinality the smaller α_k and thus the constraint on the induced volume (line 21). The constant value, $K = 0.2$, has been empirically defined from experimental simulations.

As previously explained, the new representative is chosen as the furthest attached pattern, $x_{max}(y_k)$, for space covering purposes (line 23).

The process is repeated until there is no more set to split (line 18–19).

When all the s representatives are selected, the post processing step, Algorithm 2, discards outliers as representatives. As the new selected item is chosen as the furthest from the ones which are already selected, the S set is likely to include some outliers. Two cases may occur.

In the first one (lines 2–3), when the representative is isolated, the number of attached patterns is lower or equal than the noise threshold $|T_{y_i}| \leq W_n$, inferred from the T_{y_k} distribution. Let $T' = \{T_{y_k} | |T_{y_k}| < \overline{|T_{y_k}|}\}$ be the reduced set of representatives with a number of attached pattern less than the average, and let m, σ and min, the mean, standard deviation and minimum of the $|T'|$. The noise threshold is defined as: $W_n = \max(m - 2\sigma, min)$. The choice is then to remove this representative labeled as noise.

Algorithm 2 The post processing algorithm.

1: **for all** y_i in S **do**
2: **if** $(|T_{y_i}| \leq W_n)$ **then**
3: $S = S - \{y_i\}$, $s = s - 1$
4: **end if**
5: **if** $(d_{max}(y_i) > \overline{d_{max}(y_i)}_{y_i \in S})$ **then**
6: $y_i = \arg\min_{x_l \in T_{y_i}} d(x_l, B)$ {B is the barycenter of T_{y_i}}
7: **end if**
8: $T_{y_i} = \{y_i\}$
9: **end for**
10: **for all** $x_l \in T \setminus S$ **do**
11: Find $d_{near}(x_l) = \min_{y_k \in S} d(x_l, y_k)$
12: $T_{y_k} = T_{y_k} \cup \{x_l\}$
13: **end for**

[1] '\' stands for the set difference operation.

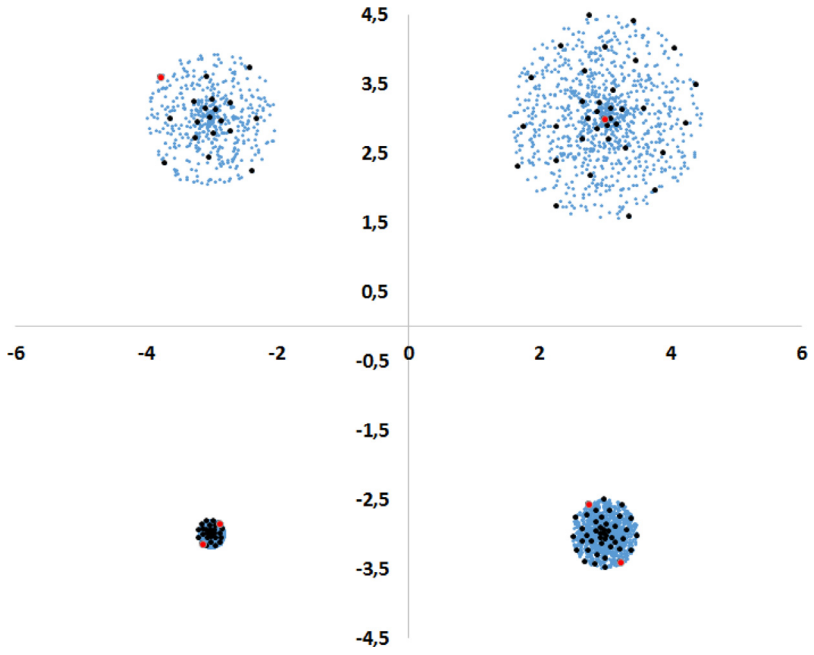

Fig. 1. Impact of the induced volume constraint. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

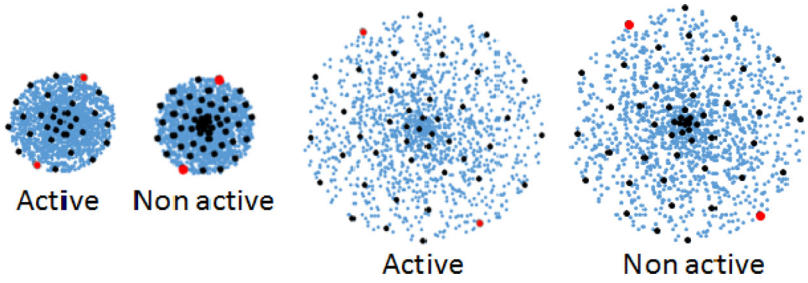

Fig. 2. Zoom of the two densest clusters. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the other case, the outlier detection is based upon the induced volume: the corresponding d_{max} is higher than average (line 5). In the post processing phase, the input space coverage is quite homogeneous: the mean can be used as a threshold. In this case, the new representative is chosen as the closest to the barycenter, B, of the set (line 6). This way of doing is similar to the usual practice: the representative is set at the center of the dense areas, like in kernel and neighboring approaches. By contrast, the proposal comes to select the representative at the border of the dense area. Once at least a representative has been changed or removed, an update of the attached patterns is needed (lines 10–13), and to do this the sets of attached patterns must be previously reset (line 8).

Fig. 1 illustrates the impact of the constraint on the induced volume (Algorithm 1, line 22). The data (blue) are well structured in four clusters of heterogeneous densities. The six first selected representatives are plotted in red, while the following ones appear in black. The small groups, in the bottom part of the figure, are denser than the others. The results for these two clusters are displayed in a zoom version in Fig. 2, with and without this constraint.

Without the mentioned constraint, the new representatives are located in the denser areas until the number of attached patterns become smaller than the W_t threshold. When the constraint is active the number of representatives in the dense area is limited by the induced volume. Density and distance are both useful to avoid an over representation in dense areas.

4. Conceptual comparison with alternative approaches

DENDIS shares some ideas with known algorithms but the way they are combined is really innovative. This can be highlighted by describing the main characteristics of the proposal.

It is density based. Many methods estimate the local density thanks to a parameter that defines the attraction basin either by counting items to induce a corresponding volume, like the popular K-nearest-neighbors algorithm, or by defining a volume, e.g. Parzen window, Mountain method (Yang & Wu, 2005), or static grids. In this case, the result is highly dependent on the setting. To fit the data structure, some methods propose an adaptive process. Attraction basin can also be induced thanks to a recursive partitioning, like in trees or dynamic grids, or by a probabilistic process. In the *k-means++* (Arthur & Vassilvitskii, 2007) a new seed is selected according to the probability computed as $\frac{d_{near}(x_i)^2}{\sum_{k=1}^{n} d_{near}(x_k)^2}$, where $d_{near}(x_i)$ is the distance from $x_i \in T$ to its closest representative in S. This probability mechanism tends to favor representatives located in dense areas. Outliers, even with a high individual probability are less likely to be selected.

In the method proposed by Feldman et al. (2011), the representatives are randomly selected, yielding a higher probability for dense areas, and inducing variable size basins. In DENDIS the

attraction basin is also induced: the average volume is estimated when the space coverage is homogeneous.

It ensures space coverage. The methods which include a neighborhood definition, e.g. leader family, grids, Mountain method, achieve a total coverage. Depending on the parameter setting they either require a large sample or may miss important details. The dynamic ones, Feldman or trees, are more powerful. In DENDIS, choosing the furthest item in the group from the representative ensures small clusters are represented. This idea is shared by the *fft* algorithm (Rosenkrantz et al., 1977). The main difference is that in *fft* the new sample item is chosen as the furthest from its representative, the maximum distance being computed over all the groups, while in DENDIS it is the furthest in the most populated group, balancing the density and coverage constraints.

It is little sensitive to noise. Looking after small clusters may result in selecting noise. Introducing a bias according to local density to favor sparse areas increases this risk. This also holds for other biased random methods like *k-means++* or Feldman. DENDIS is aware that noisy representatives are likely to be selected as they are chosen at the group border. A post-processing step is dedicated to noise management.

It is little sensitive to randomness. It is well known that random use highly impacts processes. This is true for *k-means* initialization, but also holds for other algorithms like Feldman. In *k-means++*, as denser areas have a higher probability to be selected, this impact is reduced. In DENDIS, only the first representative is randomly chosen. After a few iterations, the same border items are selected. DENDIS could be made fully deterministic by adding an extra iteration to select as the first representative the furthest from the minimum (or maximum) in each dimension.

It is data size independent. Density based methods aim to design groups with a similar density: the sample size increases with the data size. This is not the case for neighborhood based methods: in this case the sample size only depends on the neighborhood one. Thanks to the constraint on the induced volume, DENDIS adapts the sample size to the data structure not to the data size.

It is driven by one meaningful parameter. Most of sampling methods require several parameters which are more meaningful to the computer engineer than to the user and remain difficult to set. DENDIS, like *k-means++* or *fft*, needs only one. The *granularity* parameter is not dependent on the size, like the one of uniform random sampling, nor on the number of groups, like in the *k-means, k-means++* or *fft* algorithms. It is a dimensionless number whose meaning is very clear: it represents the minimal proportion of data a group has to include to be split. Combined with the data size, it is quite similar to the minimal size of a node in a tree.

Granularity is not directly linked to accuracy. As there are several internal parameters induced from data, even if accuracy is sensitive to granularity, the relationship between both is not modeled. An intermediate value, e.g. $g_r = 0.01$, generally provides good results with the risk to be not fine enough to catch small clusters. Choosing a small value, e.g. $g_r = 0.001$, ensures a good accuracy in most cases, whatever the data structure. The price to pay is a risk of over representation. This risk is however limited thanks to the distance constraint. Users may be interested in setting the algorithm according to the desired accuracy. This opens a stimulating perspective.

DENDIS presents similar ideas than popular algorithms that hybridize density and distance concepts and dynamically define attraction basins. The way these concepts are managed produces a really new algorithm.

5. Optimization

Distance-based algorithms have an usual complexity of $O(n^2)$. This is not the case for the proposal. Many distance computations can be avoided thanks to the algorithm structure itself and by embedding some optimization based on the triangular inequality.

The time complexity of Algorithm 1 is mainly due to the two first loops. For each of the s iterations, the first loop, lines 7–10, computes $(n-s)n$ distances while the second one, lines 11–14, calculates n more ones.

5.1. Reducing time complexity

These two loops can be combined in a single one , lines 8–18 in Algorithm 3. This allows for only computing $n-s$ distances to the new representative, y_*, at each of the s iterations.

Algorithm 3 The first two loops are combined into a single one.

```
1: while (ADD==TRUE) do
2:   for all x_l ∈ T \ S do
3:     Compute d = d(x_l, y_*)
4:     if (d < d_near(x_l)) then
5:       T_{y_*} = T_{y_*} ∪ {x_l}, T_{y(x_l)} = T_{y(x_l)} \ {x_l}
6:       d_near(x_l) = d, y(x_l) = y_*
7:     end if
8:     if (d > d_max(y_*)) then
9:       x_P = x(x_s), Y_P = y_*
10:      d_max(y_*) = d, x(y_*) = x_l
11:    end if
12:  end for
13:  Find a new representative y_* {Lines 16-26 of Algorithm 1}
14: end while
```

The complexity is then $O(ns)$, with $s \ll n$.

The number of distances to be computed is:

$$T = \sum_{l=s}^{n}(l-1) = \frac{n\ (n-1)}{2} - \frac{s\ (s-1)}{2} \tag{1}$$

The spatial complexity for this time optimization can be considered as reasonable: $n + 2s$ distances between the representatives are stored: n $d_{near}(x)$and s $d_{max}(y)$ as well as the corresponding elements, y for $d_{near}(x)$, and x for $d_{max}(y)$.

5.2. Using the triangle inequality

A given iteration only impacts a part of the input space, meaning the neighborhood of the new representative. Moreover as the process goes on, the corresponding induced volume decreases. This may save many distance calculations.

When a new representative in S has been selected, y_*, the question is: should a given initial pattern, x_i, be attached to y_* instead of remaining in T_{y_j}? The triangular inequality states: $d(y_j, y_*) \leq d(x_i, y_j) + d(x_i, y_*)$. And, $x_i \in T_{y_*} \iff d(x_i, y_*) < d(x_i, y_j)$. So, if $d(y_j, y_*) \geq 2\,d(x_i, y_j)$, x_i remains in T_{y_j}, no change needs to be made. Only two distances are needed to check the inequality, and discard any further calculations in the case of no change. In our algorithm, there is no need to check this inequality for all the initial patterns. For each representative, y_k, $d_{max}(y_k)$ is stored. If $d(y_k, y_*) \geq 2\,d_{max}(y_k)$, meaning the furthest initial pattern from y_k remains attached to y_k, this also holds $\forall x_i \in T_{y_k}$. Then, these representatives and their attached patterns are not concerned by the main loop of the algorithm (Algorithm 4, line 8–9). When this is not the case, the same triangle inequality provides a useful threshold. All $x_i \in T_{y_k}$ with $d_{near}(x_i) \leq 0.5\,d(y_j, y_*)$ remain attached to T_{y_k} (line 10).

Algorithm 4 The optimized density-based sampling algorithm.

1: Input: $T = \{x_i\}$, $i = 1 \ldots, n$, g_r
2: Output: $S = \{y_j\}$, $\{T_{y_j}\}$, $j = 1, \ldots, s$
3: ADD=TRUE, $W_t = n\ g_r$
4: Select an initial pattern $x_{init} \in T$
5: $S = \{y_1 = y_* = x_{init}\}$, $s = 1$
6: $d_{near}(x_i) = \infty$, $i = 1 \ldots, n$
7: **while** (ADD==TRUE) **do**
8: $F = \{T_{y_j} | d(y_j, y_*) \geq 2\ d_{max}(y_j)\}$
9: **for all** $x_l \in T \setminus \{S \cup F\}$ **do**
10: **if** $(d_{near}(x_l) > 0.5\ d(y(x_l), y_*))$ **then**
11: Compute $d = d(x_l, y_*)$
12: **if** $(d < d_{near}(x_l))$ **then**
13: $T_{y_*} = T_{y_*} \cup \{x_l\}$, $T_{y(x_l)} = T_{y(x_l)} \setminus \{x_l\}$
14: $d_{near}(x_l) = d$, $y(x_l) = y_*$
15: **end if**
16: **if** $(d > d_{max}(y_*))$ **then**
17: $x_P = x(x_s)$, $Y_P = y_*$
18: $d_{max}(y_*) = d$, $x(y_*) = x_l$
19: **end if**
20: **end if**
21: **end for**
22: Find a new representative y_* {Lines 16-26 of Algorithm 1}
23: **end while**
24: Run the post processing algorithm {Algorithm 2}
25: **return** S, $T_{y_k} \forall k \in S$

To take advantage of the triangular inequality properties, the number of distances between representatives to be stored is $s(s-1)/2$.

The optimized version of the sampling algorithm is shown in Algorithm 4.

5.3. Estimating the number of computed distances

The number of computed distances cannot be rigorously defined as it depends on the data, but it can be however roughly estimated under some weak hypothesis. Each iteration of this distance based algorithm impacts only the neighborhood of the new representative. Let k be the number of neighbors to consider. The number of distances to be calculated is $(n-1)$ at the first step, then the number of representatives to take into account is $min(k, s)$ and the number of patterns for which the distance to the representatives has to be computed is only a proportion, δ, of the set of the attached ones as the others are managed by the triangular inequality properties. A value of $\delta = 0.5$ seems to be reasonable. This means that a high proportion of representatives are concerned at the starting of the algorithm but the process then becomes more and more powerful when s increases compared to k. The real number of computed distances can be estimated as follows.

$$C = (n-1) + \sum_{i=s}^{n-1} \sum_{l=1}^{min(k,s-i)} \delta * |T_{y_l}(i)| \tag{2}$$

where $|T_{y_l}(i)|$ is the number of patterns attached to representative l when i representatives are selected.

To approximate C, one can consider that on average the representatives have a similar weight $\forall y, |T_{y_l}(i)| \approx n/i$. When the two cases, $i \leq k$ and $i > k$, are developed, the approximation becomes:

$$C = (n-1) + \delta \left(\sum_{i=2}^{k+1} (i-1)\ \frac{n}{i} + \sum_{i=s}^{n-k-2} k\ \frac{n}{i} \right)$$

As $\sum_{i=2}^{k+1}(i-1)\ \ (n/i) \leq \sum_{i=2}^{k+1}(i)\ \ (n/i)$ and $\sum_{i=s}^{n-k-2} k\ \ \frac{n}{i} \leq \sum_{i=s}^{n-k-2} k\ \frac{n}{s}$, an upper bound of C can be defined as follows:

$$C \leq (n-1) + \delta \left(n(k-1) + k\frac{n}{s}(n-k-2-s) \right) \tag{3}$$

As an illustration, using $n = 20000$, $s = 250$, $k = 10$ and $\delta = 0.6$ the decrease ratio, of the number of computed distances to the same number without optimization, as given in Eq. (1), is:

$$D = \frac{C}{T} \leq 5\%$$

This estimation is clearly confirmed by the experiments.

Under some reasonable assumptions, it can be estimated that most of distance calculations can be saved by judiciously using the triangle inequality. This optimization makes the algorithm very tractable.

6. Numerical experiments

The main objective of the sampling is to select a part that behaves like the whole. To assess the sample representativeness, the partitions built from the sample sets are compared to the ones designed from the whole sets using the same clustering algorithm. The Rand Index, *RI*, is used for partition comparison. Two representative clustering algorithms are tested, the popular *k-means* and one *hierarchical* algorithm. The resulting sample size as well as the computational cost are carefully studied as they have a strong impact on the practical use of the algorithm. In this paper we use a time ratio to characterize the CPU cost. It is computed as the sampling time added to the clustering on sample time and divided by the time required to cluster the whole data set.

Twenty databases are used, 12 synthetic, *S*#1 to *S*#12, and 8 real world data sets, *R*#1 to *R*#8. The synthetic ones are all in two dimensions and of various shapes, densities and sizes: {2200, 4000, 2200, 2000, 4000, 4500, 3500, 3500, 3000, 7500, 2500, 9500}. They are plotted in Fig. 3. The real world data are from the UCI public repository. They are of various sizes and space dimensions, with unknown data distribution. Their main characteristics are summarized in Table 1. All the variables are centered and normalized.

6.1. Sample size

Figs. 4 and 5 shows the reduction ratio of the size of the sample sets for each of the synthetic and real world data sets for different values of *granularity*.

The reduction ratio highly depends on the data, on their inner structure. The maximum ratio on Fig. 4 is 8% for *S*#1, which comes to $2200 \times 0.08 = 176$ representatives. As expected, the sample set size is higher when the granularity is lower. This evolution is monotonic but not proportional. This is explained by the restriction on the volume induced by the patterns attached to a representative (line 22 of Algorithm 1). When a dense area is covered, a lower *granularity* won't add new representatives.

6.2. Quality of representation

To assess the representativeness of the sample set, the same clustering algorithm, either *k-means* or the *hierarchical* one, is run with the whole set and the sample set. Then the resulting partitions are compared using the Rand Index. Dealing with the sample set, each non selected pattern is considered to belonging to the cluster of its representative.

Let's consider the *k-means* algorithm first. The number of clusters being unknown, it has been set to each of the possible values

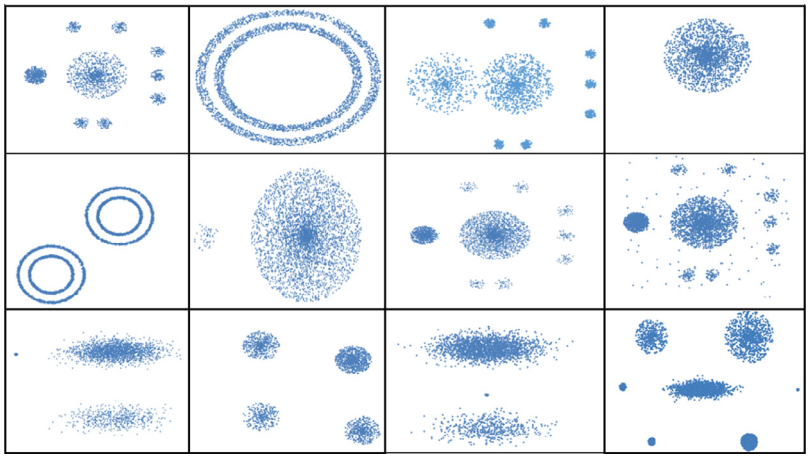

Fig. 3. The twelve synthetic data sets, *S*#1 to *S*#12.

Table 1
The eight real world data sets.

	Size	Dim	Name
R#1	434874	4	3D road network
R#2	45781	4	Eb.arff
R#3	5404	5	Phoneme
R#4	1025010	10	Poker hand
R#5	58000	9	Shuttle
R#6	245057	4	Skin segmentation
R#7	19020	10	Telescope
R#8	45730	10	CASP

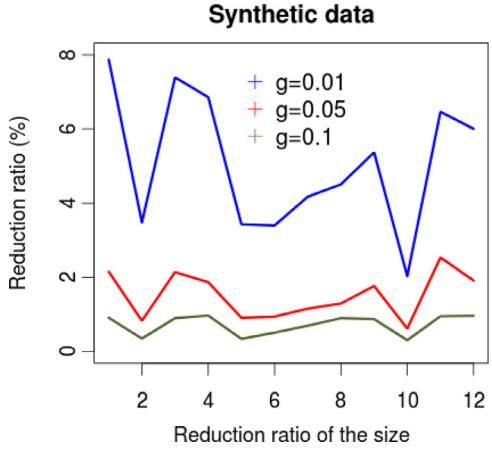

Fig. 4. Size reduction ratio for the synthetic data sets.

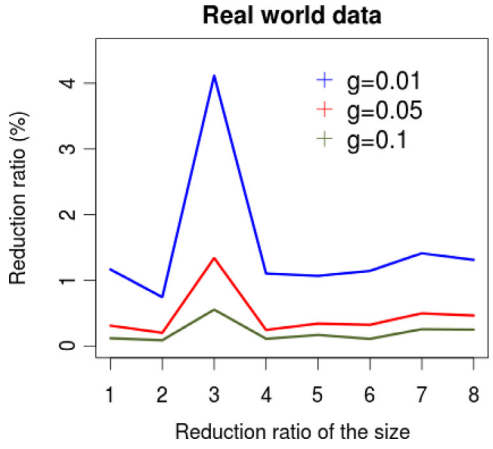

Fig. 5. Size reduction ratio for the real world data sets.

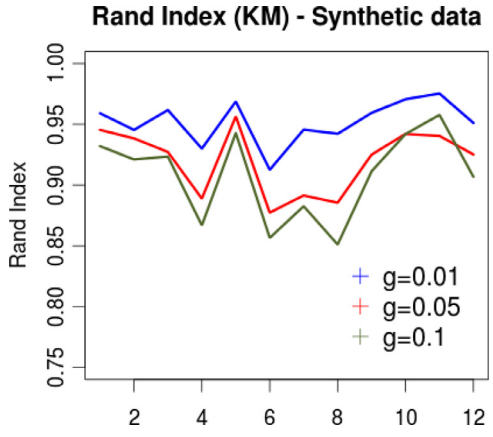

Fig. 6. The RI with the k-means algorithm for the synthetic data sets.

in the range 2 to 20. As the algorithm is sensitive to the initialization, a given number of trials, 10 in this paper, are run for a configuration.

For each data set, synthetic and real world, the resulting RI is averaged over all the experiments, meaning all the trials for all the configurations.

The results are shown in Figs. 6 and 7. The average *RI* is higher than 0.85 for all the data sets except for *R*#4, the *Poker Hand* data. These results can be considered as good. Is it worth reaching a perfect match with $RI = 1$? The cost increase may be high just to make sure all the items, including those located at the border of clusters, whose number varies from 2 to 20, are always in the same partition. is not required to consider the results as good.

It is expected that the bigger the sample set, the higher the *RI*, at least until the *RI* becomes high enough. This can be observed in the plots of Figs. 6 and 7. There is one exception, for *S*#11 and granularity of 0.05 and 0.01. This situation can be explained by the stochastic part of the test protocol and the data structure: two large clusters with different densities, and a very dense tiny one. In this case, with a fixed small number of clusters, different from the optimum, a random behavior can be observed as there are different solutions with similar costs.

Comparison with uniform random sampling (URS) is interesting to assess the relevance of the algorithm. Theoretical bounds

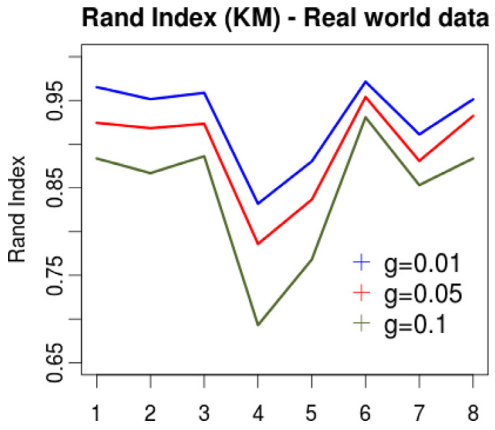

Fig. 7. The RI with the k-means algorithm for the real world data sets.

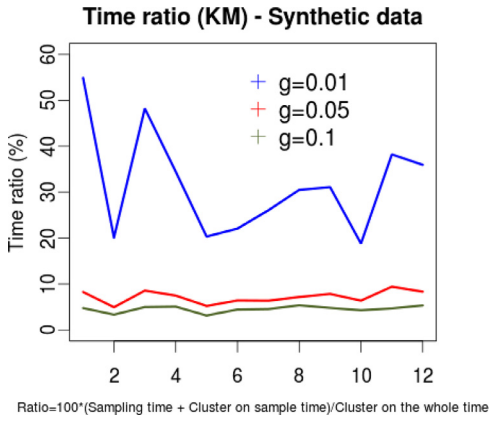

Fig. 8. The time ratio (%) with the k-means algorithm for the synthetic data sets.

Table 2
Comparison with uniform random sampling (URS) for the real world datasets.

	DENDIS			URS	
	$\|S\|$	*RI*	g_r	$\|S\|$	*RI*
R#1	702	0.96	0.0095	2014	0.93
R#2	471	0.96	0.0085	1996	0.94
R#3	271	0.96	0.015	270	0.96
R#4	750	0.85	0.015	2000	0.85
R#5	661	0.90	0.02	2006	0.94
R#6	662	0.98	0.015	2850	0.96
R#7	732	0.94	0.008	951	0.91
R#8	851	0.97	0.0095	1998	0.96

like the ones proposed in Guha et al. (1998) guarantee the URS representativeness in the worst case. As the data are usually structured, this leads to an oversized sample. Table 2 reports the results of some comparisons with URS size smaller than the theoretical bounds. The granularity parameter has been set to reach a similar Rand Index than the one yielded by URS. The results show that, for similar *RI*, the sample size is usually smaller when resulting from the proposal than the one given by URS. However, in some cases like *R*#3 and *R*#7, the results are comparable meaning that the underlying structure is well captured by URS.

In the case of the hierarchical approach, various dendrograms can be built according to the linkage function, e.g. *Ward criterion* or *single link*. To get a fair comparison the number of groups is chosen in S in the range [2, 20] and the cut in T is done to get a similar explained inertia. When the *Ward criterion* is used the number of groups in S and in T are quite similar while using the *single link* aggregation criterion, the generated partitions are generally of different sizes. The average and standard deviation of the Rand Index were computed for all the databases, reduced to 3000 patterns for tractability purposes, and different level of *granularity*. For *granularity* $= 0.04$, with the *Ward* criterion, the *RI* is $(\mu, \sigma) = (0.86, 0.03)$ for the synthetic databases and $(\mu, \sigma) = (0.87, 0.04)$ for the real ones. With the *single link* one, it is $(\mu, \sigma) = (0.87, 0.05)$ for the synthetic databases and $(\mu, \sigma) = (0.88, 0.08)$ for the real ones. In this case, the standard deviation is higher than the one corresponding to the *Ward* criterion. This can be due to the explained inertia which may be slightly different and more variable with the *single link* criterion due to its local behavior.

6.3. Computational cost

The sampling algorithm must be scalable to be used in real world problems. The index used to characterize the algorithm efficiency is computed as a ratio. The numerator is the sum of the

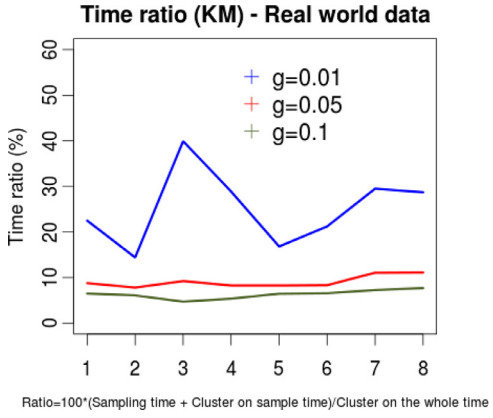

Fig. 9. The time ratio (%) with the k-means algorithm for the real world data sets.

Table 3
Time ratio with the *hierarchical algorithm*.

	Time r. (%)		Time r. (%)
S#1	0.026	*R*#1	0.031
S#2	0.021	*R*#2	0.023
S#3	0.029	*R*#3	0.043
S#4	0.021	*R*#4	0.040
S#5	0.020	*R*#5	0.026
S#6	0.022	*R*#6	0.028
S#7	0.019	*R*#7	0.045
S#8	0.020	*R*#8	0.011
S#9	0.024		
S#10	0.048		
S#11	0.021		
S#12	0.031		

sampling time and the time needed to cluster the sample set, while the denominator is the time for clustering the whole data.

The results for the *k-means* algorithm are shown in Figs. 8 and 9. The time ratio drops below 10% when the *granularity* is higher than 0.05. With the *hierarchical* algorithm the same ratio is significantly smaller.

The average time ratios (in percent) obtained with *granularity* $=$ 0.01 and for all the databases reduced to 3000 patterns are reported in Table 3. All of them fall between 0.02% and 0.048%, meaning the proposal is 2000 times faster.

6.4. Comparison with known algorithms

In order to compare *DENDIS* with concurrent algorithms, 12 sampling representative approaches were considered.

Table 4
The twelve concurrent approaches.

	Name	Param(s)	Range
A1	Uniform random sampling	$\|S\| = \frac{\|T\|}{\lambda^*}$	[10, 500]
A2	Leader (pioneer) (Ling, 1981)	$t = \frac{d_m}{\lambda^*}$	[2, 10]
A3	Leader (improved) (Viswanath et al., 2013)	$\|S\| = \frac{\|T\|}{\lambda^*}, c = \frac{\|T\|}{\mu^*}$	[10, 500], [2, 5]
A4	k-means sampling (Xiao et al., 2014)	$b, \|S\| = \frac{\|T\|}{\lambda^*}$	[10, 500]
A5	Kernel sampling (Kollios et al., 2003)	$b, \|S\| = \frac{\|T\|}{\lambda^*}$	[10, 500]
A6	Grid sampling (Palmer & Faloutsos, 2000)	$b, N^*_{cut}(axis)$	[2, 10]
A7	k-nearest-neighbors (Franco-Lopez, Ek, & Bauer, 2001)	$b, k = \lambda^*\sqrt{\|T\|}$	[0.2, 0.5]
A8	Tree sampling (Ros et al., 2003)	$b, min_{size} = \frac{\|T\|}{\lambda^*}, N^*_{cut}$	[50, 200], [1, 4]
A9	Bagged sampling (Dolnicar & Leisch, 2004)	$N^*_{strata}, N_r = \frac{\|T\|}{\lambda^*}$	[4, 20], [2, 100]
A10	k-means++ (Arthur & Vassilvitskii, 2007)	$\|S\| = \frac{\|T\|}{\lambda^*}$	[10, 500]
A11	fft (Rosenkrantz et al., 1977)	$\|S\| = \frac{\|T\|}{\lambda^*}$	[10, 500]
A12	Hybrid (Feldman et al., 2011)	$\beta^*, \|S\| = \frac{\|T\|}{\lambda^*}$	[10, 100], [10, 500]

Table 5
Concurrent approaches: Cumulative averaged time (s) over the 8 data sets.

A1	A2	A3	A4	A5	A6	A7
0.004	7.7	858	52	4951	**2.3**	9651
A8	A9	A10	A11	A12	DENDIS	
27	**2.4**	**4.3**	**6.0**	**2.7**	0.5	

Table 4 summarizes their input parameters[2]. The bias level, b, common to different approaches, ranges in $[-1, +1]$.

The protocol was the one described in the previous section for each algorithm. Only real data sets are considered. The number of partitions ranges from 4 to 10. The maximum sample size, set to $min(0.005\,n,\ 2000)$, has been used as an extra stop criterion. For each configuration, the different algorithms were run 10 times and the average considered. Both the *hierarchical* and *k-means* clustering were considered but the whole tests have been restricted to the *k-means* algorithm as some data are not tractable with the hierarchical one. The results of these extensive experiments are summarized to highlight the main trends.

The sampling time and the quality of representation are analyzed.

Sampling time. Table 5 summarizes the average (over all the experiments including the different data and partitions) sampling times in seconds. As expected the URS, *A*1, is the fastest algorithm, but *DENDIS* is quite swift too, faster than the grid ones (*A*6, *A*9) and other competitors (*A*10 to *A*12). These six algorithms are the only ones with a time ratio less than 1 for the *k-means* clustering.

This is obviously not the case when the clustering algorithm is *hierarchical*, all the algorithms are efficient even if *A*5 and *A*7 are still limited by a high sampling time.

Quality of representation. The results reported in Table 6 are the best *RI*, on average, over the 8 real data sets, of all the tested configurations. The maximum of the *RI* for each data set is highlighted in bold font. When several algorithms have reached the same accuracy, and DENDIS is among this group, the bold font is used in the DENDIS row.

[2] *A*8 is similar to Ros, Taboureau, Pintore, and Chretien (2003) except that the bins are not fuzzy, and are ordered via their weights until reaching a lower bound. For *A*12 the input parameters (β and λ) are directly linked to the original parameters δ, ε, k (Feldman et al., 2011).

Table 6
Best Rand Index on average over the real data for each algorithm.

	R#1	R#2	R#3	R#4	R#5	R#6	R#7	R#8	Mean
A1	0.92	0.92	0.91	0.85	0.93	0.92	0.90	0.99	0.92
A2	0.93	0.93	0.93	0.86	0.84	0.90	0.89	1.00	0.91
A3	0.90	0.91	0.88	0.83	0.87	0.86	0.88	1.00	0.89
A4	**0.96**	0.93	0.94	0.90	0.94	0.91	0.86	1.00	0.93
A5	0.94	0.93	0.94	0.88	0.93	0.91	0.89	1.00	0.93
A6	0.88	0.86	0.86	0.86	0.83	0.85	0.87	0.99	0.87
A7	0.96	0.93	0.94	**0.90**	0.94	0.91	0.86	1.00	0.93
A8	0.94	0.93	0.95	0.89	0.93	0.95	0.87	1.00	0.93
A9	0.94	0.91	0.91	0.86	0.94	0.94	0.89	1.00	0.92
A10	0.90	0.91	0.94	0.86	**0.99**	0.91	0.87	1.00	0.92
A11	0.90	0.94	0.88	0.86	0.83	0.93	0.83	1.00	0.90
A12	0.94	0.90	0.92	0.86	0.98	0.92	**0.92**	1.00	0.93
DENDIS	0.95	**0.94**	**0.96**	0.89	0.95	**0.95**	0.90	**1.00**	**0.94**

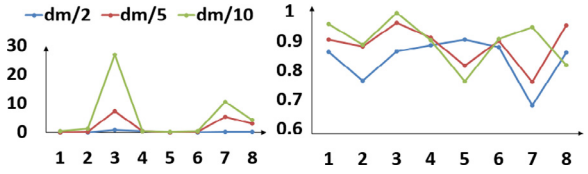

Fig. 10. Leader behavior according to λ^*.

In four of the eight cases, DENDIS is among the most accurate algorithms. This explains the higher accuracy on average.

The algorithms are quite accurate, few *RI* are below 0.85. The URS is a powerful algorithm, but it requires more representatives when the data are structured.

The algorithms' performances are highly dependent on the setting as they require several parameters that need to be combined. The concern is particularly checked for the leader methods: they are the most difficult to parameterize as illustrated in Fig. 10. The d_m parameter was estimated on a random sample, 10% of the whole set, and computed as: $d_m = max\ d(\mu, x_i)$, μ being the average of the set of vectors x_i. The results for different values of λ^* are shown in Fig. 10. The left part shows the sample size (% of the whole), the right part illustrates the *RI* variation according to λ^*: no monotonicity can be deduced; the best value depends on the data.

*A*6 is fast but yields the poorest results. It is difficult to tune especially to find 'generic' grid parameters.

The bias level, b, is quite influential: low negative values give the best average results. This is the case with approaches *A*5 to *A*9.

Table 7
Best concurrent approaches: Detailed comparisons with noise.

	Sr-(%)	time	RI	Sr-(%)	time	RI	Sr-(%)	time	RI
A9	0.5	318	0.956	0.6	314	0.945	0.83	376	0.948
A10	0.49	541	0.900	1	1663	0.932	5	30056	0.952
A12	0.33	1163	0.967	0.59	1225	0.971	1.09	1365	0.986
DENDIS	0.15	44	0.964	0.23	93	0.987	0.35	113	0.995

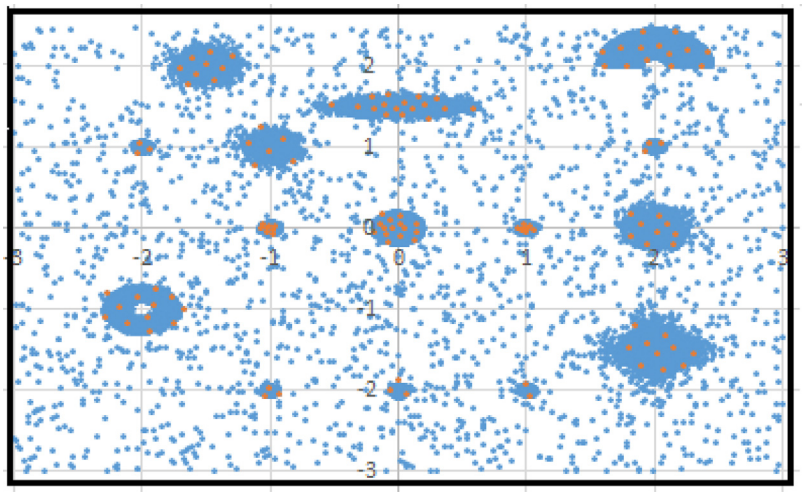

Fig. 11. DENDIS representatives for the noise data ($g_r = 0.01$).

*A*5 and *A*7 are too computational expensive. Without additional optimization like the bucketing algorithm, which is itself a preprocessing step, they are not scalable.

The *k-means* (*A*4) and *tree* (*A*8) algorithms are always faster than the former ones, and can yield high *RI*, but with a higher sample size. The competitors that meet both criteria of scalability and accuracy are *A*9, *A*10 and *A*12.

Complementary comparison of the best algorithms. The best algorithms (*A*9, *A*10, *A*12 and *DENDIS*) are now compared according to their behavior in presence of noise.

A synthetic dataset (40000 patterns in R^2) with natural clusters of different cardinalities (from 1000 to 5000 items) was generated. An important level of noise, 4%, has been added. Noise values are computed independently in each dimension, according to the whole range of the given feature: $noise_j = \min_j + U[0,+1] * (\max_j - \min_j)$. Each sampling algorithm is applied and the *k-means* algorithm is run with $k = 10$.

In order to limit the number of tests, only the input parameter that most influences (directly or indirectly) the amount of representatives was considered: λ for *A*9, *A*10, and *A*12 and g_r for *DENDIS*. The others were fixed at nominal values. These values were selected from the previous tests as the ones that lead to a good trade-off between accuracy and tractability in average. They respectively correspond to $N_r = 20$ for *A*9 and $\beta = 20$ for *A*12.

Different experiments have been carried out with the objective to reach the best *RIs* with the smallest sample sizes. They are summarized in Table 7 where three of them are reported. In the first one, columns 2–4 in Table 7, the corresponding input setting for these methods is: $\lambda \simeq 135$ (in the range [134, 137] for the four sets and $g_r = 0.1$. Then the input parameter has been set to get a higher sample size, up to 5% for *A*9, *A*10, *A*12. The *DENDIS* granularity was 0.03 and 0.008. The reported results in the two last trials are kept in a range where the *RI* is improved. Beyond this range, even with a 5% size sample, no improvement can be observed, only the running time is different. Using *A*9 the sample size which yields the best results is below 1%. Using *DENDIS*, one can note that the sample size increases with the granularity parameter, but not in a proportional way.

*A*12 and *DENDIS* appear to be the most robust to noise as they yield the best results for all the trials. This is not so surprising as both are based upon similar concepts. In the *A*12 approach, dense areas are first covered by a uniform random sampling, then the initial patterns represented in the sample are no more considered in the next iterations. The corresponding sample sizes are also comparable, even if *DENDIS* samples are always smaller. The main difference between both algorithms is the computational time: *A*12 is significantly slower than the proposal.

Fig. 11 shows that *DENDIS* is still able to identify and represent the data structure even with a high level of noise. The 129 representatives, plotted in orange, only belong to the clusters and ensure shape coverage.

7. Conclusion

A new sampling for clustering algorithm has been proposed in this paper. *DENDIS* is an hybrid algorithm that manages both density and distance concepts. Even if the basics of these concepts are known, their specific use produces a really new algorithm able to manage high density as well as sparse areas, by selecting representatives in all clusters, even the smaller ones.

It is density-based: at each iteration the new representative is chosen in the most populated group. It allows for catching small clusters: the new representative is the furthest, from the representative, of the attached patterns. There is no need to estimate local density, neither to define a neighborhood. The attraction basin is dynamically determined thanks to hidden parameters induced from the data.

DENDIS is driven by a unique, and meaningful, parameter called granularity: it is dimensionless and represents the minimal proportion of data a group has to include to be split. Combined with the data size, it is quite similar to the minimal size of a node in a tree.

Without any additional constraint, the representatives would tend to have a similar number of patterns. To manage different local densities, and to avoid over representation in high density areas, a volume restriction is added for group splitting. It is based upon the average induced volume estimated by the maximum within group distance. This makes the sample size independent on the data size, depending only on the data structure. Others parameters are used, but they are inferred from data. This makes the algorithm really easy to tune.

The inner structure of the algorithm, especially the selection of the furthest item in the group as a new representative, favors the use of the triangle inequality because a new representative only impact its neighborhood. Even if the exact number of avoided computations cannot be rigorously defined, an upper bound can be estimated under weak assumptions. It shows that only 5% of the total number of distances are really computed.

The algorithm behavior has been studied using 12 synthetic and 8 real world datasets. It has been compared to 12 concurrent approaches using the real world data sets according to three criteria: the sample size, the computational cost and the accuracy. The latter was assessed by the Rand Index, for two types of clustering algorithms, k-means or hierarchical: the partitions resulting from the clustering on the sample against the ones yielded by the same clustering method on the whole set.

These experiments show that *DENDIS* has some nice properties. It is parsimonious. The sample size is not an input parameter, it is an outcome of the sampling process. It is, as expected, smaller than the theoretical bound suggested in Guha et al. (1998) for uniform random sampling. *DENDIS* yields comparable accuracy to the most popular concurrent techniques with a similar number, or even fewer, representatives. It is fast. Thanks to an internal optimization, it has a very low computational cost. This scalability property allows for its use with very large data sets. It is robust to noise: a post-processing step remove representatives labeled as noise.

Future work will be mainly dedicated to improve the hybrid algorithm to become self-tuning, capable of finding by itself the suitable granularity to reach a given level of accuracy. Even if the dimensionless parameter is meaningful to the user and impacts accuracy, the relationship between both is not really modeled. From an empirical point of view, the challenge consists in finding the appropriate mechanisms without penalizing the running time which is a quite interesting feature of the proposal. The first iterations of the algorithm are the most computationally expensive. The starting steps can be improved.

From a more conceptual view, a similar approach than the one used to estimate the number of distances could be useful to investigate other research directions, for instance the relationship between the granularity, the sample size and the accuracy, based on a real time estimation of clustering cost of the whole data.

References

Al-Kateb, M., & Lee, B. (2014). Adaptive stratified reservoir sampling over heterogeneous data streams. *Information Systems, 39*(1), 199–216.

Andreopoulos, B., An, A., Wang, X., & Schroeder, M. (2009). A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics, 10*(3), 297–314.

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 1027–1035). Society for Industrial and Applied Mathematics.

Bagirov, A. M., Ugon, J., & Webb, D. (2011). Fast modified global k-means algorithm for incremental cluster construction. *Pattern Recognition, 44*(4), 866–876.

Bardekar, M. A. A., & Tijare, M. P. (2011). A review on LBG algorithm for image compression. *International Journal of Computer Science and Information Technologies, 2*(6), 2584–2589.

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications, 40*(1), 200–210.

Chang, C.-C., & Hsieh, Y.-P. (2012). A fast VQ codebook search with initialization and search order. *Information Sciences, 183*(1), 132–139.

Chernoff, H. (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics, 23*(4), 493–507.

Chiang, M.-C., Tsai, C.-W., & Yang, C.-S. (2011). A time-efficient pattern reduction algorithm for k-means clustering. *Information Sciences, 181*(4), 716–731.

Devroye, L. (1981). On the average complexity of some bucketing algorithms. *Computers & Mathematics with Applications, 7*(5), 407–412.

Dolnicar, S., & Leisch, F. (2004). Segmenting markets by bagged clustering. *Australasian Marketing Journal, 12*(1), 51–65.

Feldman, D., Faulkner, M., & Krause, A. (2011). Scalable training of mixture models via coresets. In *Advances in neural information processing systems* (pp. 2142–2150).

Franco-Lopez, H., Ek, A. R., & Bauer, M. E. (2001). Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method. *Remote sensing of environment, 77*(3), 251–274.

Guha, S., Rastogi, R., & Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. *SIGMOD Record, 27*(2), 73–84.

Gutmann, B., & Kersting, K. (2007). Stratified gradient boosting for fast training of conditional random fields. In *Proceedings of the 6th international workshop on multi-relational data mining* (pp. 56–68).

Hatamlou, A., Abdullah, S., & Nezamabadi-pour, H. (2012). A combined approach for clustering based on k-means and gravitational search algorithms. *Swarm and Evolutionary Computation, 6*, 47–52.

Ilango, M. R., & Mohan, V. (2010). A survey of grid based clustering algorithms. *International Journal of Engineering Science and Technology, 2*(8), 3441–3446.

Khan, S. S., & Ahmad, A. (2013). Cluster center initialization algorithm for k-modes clustering. *Expert Systems with Applications, 40*(18), 7444–7456.

Kollios, G., Gunopulos, D., Koudas, N., & Berchtold, S. (2003). Efficient biased sampling for approximate clustering and outlier detection in large data sets. *IEEE Transactions on Knowledge and Data Engineering, 15*(5), 1170–1187.

Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern recognition, 36*(2), 451–461.

Ling, R. F. (1981). Cluster analysis algorithms for data reduction and classification of objects. *Technometrics, 23*(4), 417–418.

Lv, Y., Ma, T., Tang, M., Cao, J., Tian, Y., Al-Dhelaan, A., & Al-Rodhaan, M. (2015). An efficient and scalable density-based clustering algorithm for datasets with complex structures. *Neurocomputing, In Press*.

Ma, X., Pan, Z., Li, Y., & Fang, J. (2015). High-quality initial codebook design method of vector quantisation using grouping strategy. *IET Image Processing, 9*, 986–992.

Menardi, G., & Azzalini, A. (2014). An advancement in clustering via nonparametric density estimation. *Statistics and Computing, 24*(5), 753–767.

Nagpal, A., Jatain, A., & Gaur, D. (2013). Review based on data clustering algorithms. In *Information & communication technologies (ICT), IEEE conference on* (pp. 298–303). IEEE.

Naldi, M., & Campello, R. (2015). Comparison of distributed evolutionary k-means clustering algorithms. *Neurocomputing, 163*, 78–93.

Nanopoulos, A., Manolopoulos, Y., & Theodoridis, Y. (2002). An efficient and effective algorithm for density biased sampling. In *Proceedings of the eleventh international conference on information and knowledge management* (pp. 398–404).

Palmer, C. R., & Faloutsos, C. (2000). Density biased sampling: An improved method for data mining and clustering. In *Proceedings of the ACM SIGMOD INTL conference on management of data* (pp. 82–92). Dallas.

Ros, F., Taboureau, O., Pintore, M., & Chretien, J. (2003). Development of predictive models by adaptive fuzzy partitioning. application to compounds active on the central nervous system. *Chemometrics and Intelligent Laboratory Systems, 67*(1), 29–50.

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M., II (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing, 6*(3), 563–581.

Sarma, T., Viswanath, P., & Reddy, B. (2013). Speeding-up the kernel k-means clustering method: A prototype based hybrid approach. *Pattern Recognition Letters, 34*(5), 564–573.

Tzortzis, G., & Likas, A. (2014). The minmax k-means clustering algorithm. *Pattern Recognition, 47*(7), 2505–2516.

Viswanath, P., Sarma, T., & Reddy, B. (2013). A hybrid approach to speed-up the k-means clustering method. *International Journal of Machine Learning and Cybernetics, 4*(2), 107–117.

Wang, X., Wang, X., & Wilkes, D. M. (2009). A divide-and-conquer approach for minimum spanning tree-based clustering. *IEEE Transactions on Knowledge and Data Engineering, 21*(7), 945–958.

Xiao, Y., Liu, B., Hao, Z., & Cao, L. (2014). A k-farthest-neighbor-based approach for support vector data description. *Applied Intelligence, 41*(1), 196–211.

Yang, M.-S., & Wu, K.-L. (2005). A modified mountain clustering algorithm. *Pattern Analysis and Applications, 8*(1–2), 125–138.

Zahra, S., Ghazanfar, M. A., Khalid, A., Azam, M. A., Naeem, U., & Prugel-Bennett, A. (2015). Novel centroid selection approaches for kmeans-clustering based recommender systems. *Information Sciences, 320*, 156–189.

Zhong, C., Malinen, M., Miao, D., & Fränti, P. (2015). A fast minimum spanning tree algorithm based on k-means. *Information Sciences, 295*, 1–17.