A new density-based sampling algorithm

Frédéric Ros[1] **Serge Guillaume**[2]

[1]Laboratory PRISME, Orléans university, France
[2]Irstea, UMR ITAP, Montpellier, France

Abstract

To face the big data challenge, sampling can be used as a preprocessing step for clustering. In this paper, an hybrid algorithm is proposed. It is density-based while managing distance concepts. The algorithm behavior is investigated using synthetic and real-world data sets. The first experiments proved it can be accurate, according to the Rand Index, with both *k-means* and *hierarchical* clustering algorithms.

Keywords: Density, distance, space coverage, clustering, Rand index.

1. Introduction

Summarizing information is a key task in information processing, either in data mining, knowledge induction or pattern recognition. Clustering [1] 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 [2] have been proposed to identify clusters according to various criteria. Some of them [3] 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, Chamaleon, Kd-tree.

To cope with data complexity, algorithms are becoming increasingly 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, the majority of mature clustering techniques rather address small or medium databases (several hundreds of patterns) and fail to scale up well with large data sets, with an excessive computational time. Therefore, in addition to the usual performance requirements, response time is of major concern to most data clustering algorithms nowadays. As an example, algorithms with quadratic or exponential complexity, such as hierarchical approaches, are strongly limited. Their computational cost makes them inefficient with large data sets. Even algorithms like *k-means* are still slow in practice for large datasets.

While some approaches aims at optimizing and speeding up existing techniques [4, 5], sampling appears as an interesting alternative to manage large data sets.

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 [6], random sampling does not account for cluster shape or density. The results are interesting from a theoretical point of view [7], but they tend to overestimate the sample size in non worst-case situations.

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.

All density methods [8] aim at tracking the local density under the assumption that clusters are more likely present around the modes of the spatial distribution. They can be grouped in two main families: space partition [9, 10] and local density estimation, neighborhood or kernel estimation [11]. 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 so as to ensure the representation of small clusters. 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 (sampling, bucketing algorithm...), which also increases their complexity, they are not scalable.

Distance concepts are widely used in clustering and sampling algorithms as distance is used to measure similarity and proximity between patterns. The most popular algorithm representative of this family remains the *k-means*, and its robust version called *k-medoids*. 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* [4, 12] clustering. 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 on the increase of the computational cost. The *mountain method* proposed by Yager and its modified versions [13] are good representatives of hybrid methodologies.

Strategies usually based on stratification processes have also been developed to improve and speed up the sampling process [14]. Reservoir algorithms [15] 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. If interesting, these method need an accurate setting to become really relevant.

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 [16, 17] 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, scalable and self-adaptive, allowing to process various kinds of large data sets with a standard setting, remains an open challenge.

The goal of this paper is to introduce a new algorithm that fulfills these requirements. Based on space density, it is also able to manage distance concepts. The paper is organized as follows. Section 2 introduces the hybrid algorithm. The proposal is evaluated using synthetic and real world data in Section 3. Finally Section 4 summarizes the main conclusions and opened perspectives.

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

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.

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, $W_t = n\ g_r$,
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 $d_{max}(y_k) = \max_{x_m \in T \setminus S} d(x_m, y_k)$
13:        Store $d_{max}(y_k)$, $x_{max}(y_k)$, $|T_{y_k}|$
           {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 $|T_{y_{(1)}}| \geq \ldots \geq |T_{y_{(s)}}|$
17:    for all $y_k$ in $S$ do
18:        $\alpha_k = 1 + \frac{|T_{y_k}|}{W_t}$
19:        if $(d_{max}(y_k) \geq \alpha_k \overline{d_{max}(y_k)}_{y_k \in S}$
           and $|T_{y_k}| \geq W_t)$ then
20:            $x_* = x_{max}(y_k)$
21:            ADD=TRUE, break
22:        end if
23:    end for
24:    if ADD==TRUE then
25:        $S = S \cup \{x_*\}$, $s = s + 1$
26:    end if
27: end while
28: for all $y_i$ in $S$ do
29:     if $|T_{y_i}| < \beta\ W_t$ then
30:         $S = S - \{y_i\}$, $s = s - 1$
31:     end if
32:     if $d_{max}(y_i) \geq \gamma \overline{d_{max}(y_i)}_{y_i \in S}$ then
33:         $y_i = x_l \mid \min_{x_l \in T_{y_i}} d(x_l, B)$
34:     end if
35:     $T_{y_i} = \{y_i\}$
36: end for
37: for all $x_l \in T \setminus S$ do
38:     Find $d_{near}(x_l) = \min_{y_k \in S} d(x_l, y_k)$
39:     $T_{y_k} = T_{y_k} \cup \{x_l\}$
40: end for
41: return $S$, $T_{y_k} \forall k \in S$
```

The first sample is randomly chosen (line 3). Then the algorithm iterates to select the representatives (lines 6-27). In a preparation phase, each not selected pattern, $x \in T \setminus S$[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 15-23). The selected items are sorted according to the cardinality of the set of patterns they are the representative (line 16). Then these sets T_{y_k} are analyzed in decreasing order of weight. Each of them is split when two conditions are met (lines 19-20). The first one deals with the number of attached patterns: it has to be higher than the threshold, $W_t = n\ g_r$. The

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

other one is related to the induced hyper volume: the furthest attached pattern must be located at a distance higher than average. The corresponding induced volume depends on the cardinality of the set, taken into account with the α_k coefficient. This coefficient is guaranteed to be higher than one. Both notions, cardinality and induced volume, define the density. The d_{max} condition prevents an unsuitable over representation of dense areas. The new representative is chosen as the furthest attached pattern, $x_{max}(y_k)$, for space covering purposes.

When all the s representatives are selected, the post processing step (lines 28-40) 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 29-31), when the representative is isolated, the number of attached patterns is very low, $|T_{y_i}| < \beta \; W_t$, with $\beta < 1$, e.g $\beta = 0.1$. The choice is then to remove this representative. In the other case, the outlier detection is based upon the induced volume: the corresponding d_{max} is higher than average (line 32), $d_{max}(y_i) \geq \gamma \underset{y_i \in S}{\overline{d_{max}(y_i)}}$, with $\gamma > 1$, e.g $\gamma = 1.5$. In this case, the new representative is chosen as the closest to the barycenter, B, of the set (line 33). 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, an update of the attached patterns is needed (lines 37-40), and to do this the sets of attached patterns must be previously reset (line 35).

Figure 1 aims at illustrating the impact of the constraint on the induced volume (first part of line 19). 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 Figure 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.

Many distance computations in Algorithm 1 can be avoided thanks to the algorithm structure itself as only the neighborhood of the new representative is impacted at a given iteration. It means that the number of distances to be calculated drops with the number of iterations, as the induced volume decreases. This number cannot be rigorously defined as it depends on the data. Under some reasonable assumptions, it can be estimated that about 95% of distance calculations can be saved by judiciously using the triangle inequality.

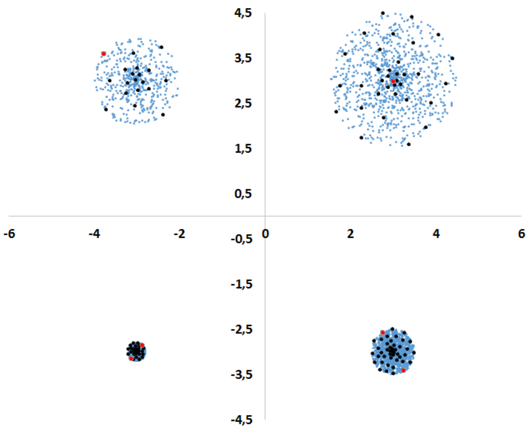

Figure 1: Impact of the induced volume constraint

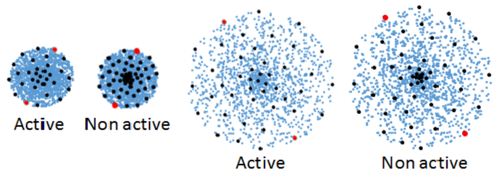

Figure 2: Zoom of the two densest clusters

3. Results and Discussion

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.

In order to assess the algorithm, 20 databases are selected, 12 synthetic and 8 real world data sets. The synthetic ones are all in two dimensions and of various sizes: {2200, 4000, 2200, 2000, 4000, 4500, 3500, 3500, 3000, 7500, 2500, 9500}. They are plotted in Figure 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.

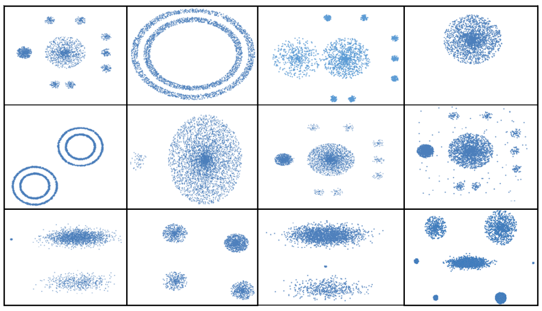

Figure 3: The twelve synthetic data sets

Table 1: The eight real world data sets

D	Size	Dim	Name
1	434874	4	3D Road Network
2	45781	4	Eb.arff
3	5404	5	Phoneme
4	1025010	11	Poker Hand
5	58000	9	Shuttle
6	245057	4	Skin Segmentation
7	19020	10	Telescope
8	45730	10	CASP

Sample size. Even if the algorithm includes some parameters, some of them can be considered as internal in the sense that the result, i.e. the sample set, is not really sensitive to their setting, when this parameter is chosen in a range defined as ±30% of their nominal value.The algorithm is thus driven by a unique, and meaningful, parameter called *granularity*.

Figures 4 and 5 shows the reduction ratio of the size of the sample sets for each of the synthetic and real world data sets.

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 19 of the algorithm). When a dense area is covered, a lower *granularity* won't add new representatives. For equal density data, the number of representatives would be the inverse of the *granularity*. Thanks to the density management, the sample set size is always smaller with structured data: the maximum ratio on Figure 4 is 8% for synthetic data set number 1, which comes to $2200 \times 0.08 = 176$ representatives.

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

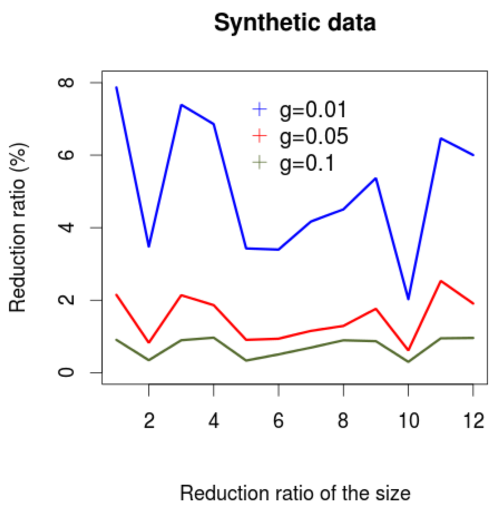

Figure 4: Size reduction ratio for the synthetic data sets

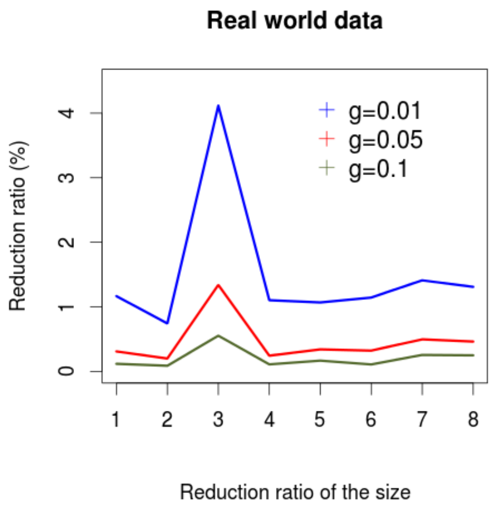

Figure 5: Size reduction ratio for the real world data sets

given configuration. The number of clusters being unknown, it has been set to each of the possible values in the range 2 to 20.

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 Figures 6 and 7. The average *RI* is higher than 0.85 for all the data sets except for the *Poker Hand* data. It is worth noting that a perfect match is not required to consider the results as good. Indeed, $RI = 1$ would mean that all the items, including those located at the border of clusters, whose number varies from 2 to 20, are always in the same partition.

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 Fig 6 and 7. There is one exception, for synthetic data set

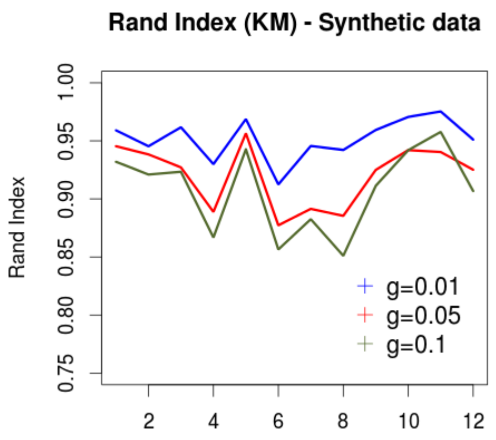

Figure 6: The RI with the k-means algorithm for the synthetic data sets

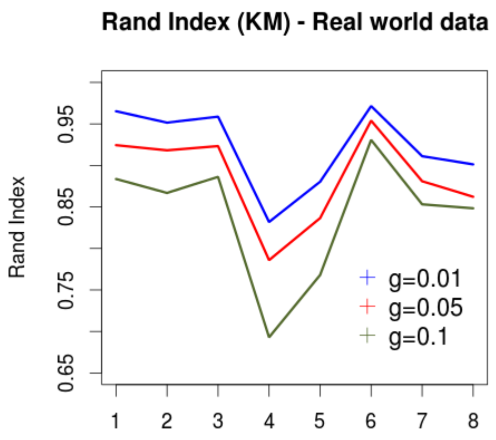

Figure 7: The RI with the k-means algorithm for the real world data sets

number 11 and granularity of 0.05 and 0.01. This situation can be explained by the stochastic part of both the test protocol and the algorithm initialization. The data are well structured in 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 like the ones proposed in [6] guarantee the URS representativeness in the worst case. As the data are usually structured, this leads to an oversized sample. The 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

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

	$S_{(alg)}$	RI	$S_{(URS)}$	RI
1	702	0.96	2014	0.925
2	471	0.963	1996	0.939
3	271	0.957	270	0.955
4	750	0.85	2000	0.849
5	661	0.9	2006	0.94
6	662	0.98	2850	0.96
7	732	0.94	951	0.91
8	851	0.973	1998	0.96

Index than the one yielded by URS. The granularity values are not reported in the table. 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 Data sets #3 and #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 $(\mu, \sigma) = (0.86, 0.029)$ for the synthetic databases and $(\mu, \sigma) = (0.87, 0.036)$ 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 difference between the explained inertia in both sets: even if they are close one to the other, they are more likely to be different with the *single link* criterion.

Computational cost. The sampling algorithm must be scalable to be used in real world problems. This work also includes time optimization which cannot be detailed in the present paper. The index used to characterize the algorithm efficiency is computed as a ratio. The numerator is the sum of the 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 Figures 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

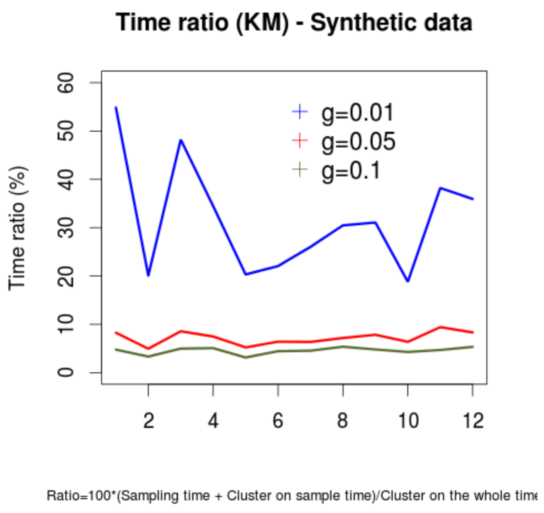

Figure 8: The time ratio (%) with the k-means algorithm for the synthetic data sets

smaller.

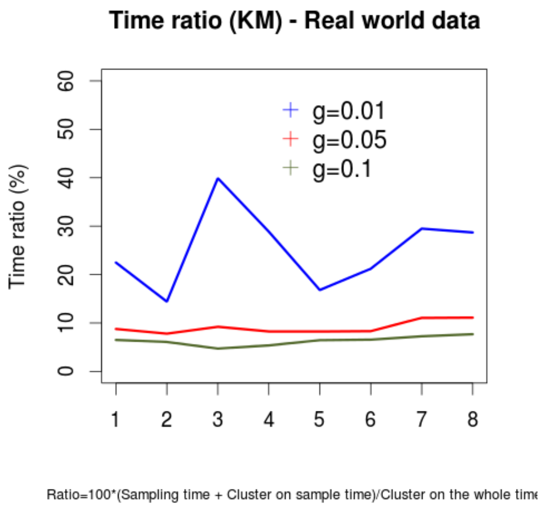

Figure 9: The time ratio (%) with the k-means algorithm for the real world data sets

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%.

Sampling as the first steps of clustering. The former experiments show that the sample behaves like the whole according to the Rand Index, meaning that the same clustering algorithm run with the two sets yields similar partitions. This conclusion suggests that the sampling can be considered as the probability density function of the original data set. To validate this hypothesis, the result of the sampling is now seen as a partition of $s = |S|$ groups, and this partition is compared to the one of the same size given by the clustering algorithm run on

Table 3: Time ratio with the *hierarchical algorithm*

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

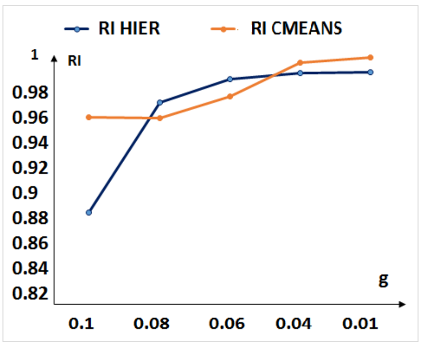

Figure 10: Rand Index for database 12

the whole set of data, T. The *k-means* and the *hierarchical* algorithm with the *Ward criterion* are used. The experiments were conducted with all the data sets, with five different granularities.

The Rand Index, for decreasing values of *granularity*, are plotted in Figure 10 for synthetic data set 12. The complementary data of this experiment are given in Table 4 for all the databases.

The first row is the index used as the abscissa in Fig. 10, the corresponding *granularity* is in the second row. Next row reports the ratio of the sample size to the whole set in percent. The two last rows give the time ratio in percent for both algorithms, meaning the sampling time divided by the clustering algorithm time needed to build the same size partition.

As expected, the evolution for all these values is monotonic: the sample size and the time ratio increase with decreasing values of *granularity*. The time ratio is, this was also expected, really low for the *hierarchical* clustering, but it is also significantly low for the *k-means* algorithm.

The Rand Index curves are also monotonic, the

Table 4: Averaged results for all the data sets

Index Fig. 10	1	2	3	4	5
Granularity	0.1	0.08	0.06	0.04	0.01
$100\ \lvert S\rvert/\lvert T\rvert$	0.80	2.27	2.95	7.68	9.98
TimeR. KM (%)	1.94	2.44	2.63	3.45	10.3
TimeR. Hier. (%)	0.001	0.002	0.005	0.006	0.01

smaller the granularity, the higher the RI. A quite good match is achieved as well as the granularity is below 6 % for both algorithms.

4. Conclusion

A new sampling for clustering algorithm has been proposed in this paper. It 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.

The first experiments show that the proposal has some nice properties.

It is parsimonious: the sample size is smaller than the theoretical bound suggested in [6].

It is accurate, according to the Rand Index, for the two types of clustering algorithms, *k-means* or *hierarchical*: the partitions resulting from the clustering on the sample are similar to the ones obtained by the same algorithm from the whole set of data. If the sampling algorithm can be used to speed up the clustering, this is because the sampling can be seen as the result of the first steps of the clustering: when the result of the sampling, a $|S|$-size partition, is compared to a same size partition resulting from the first steps of the clustering, the Rand Index is very high as soon as the granularity is low enough.

It is fast. Thanks to an internal optimization, the running time is close to the one of the uniform sampling. This scalability property allows its use with very large data sets.

It is driven by a unique, and meaningful, parameter called granularity. The lower granularity, the better the representativeness of the sample until all the clusters are represented in the sample set.

Future work will be dedicated to improve the hybrid algorithm to become self-tuning, capable of finding by itself the appropriate granularity to reach a given level of accuracy.

References

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

[2] Bill Andreopoulos, Aijun An, Xiaogang Wang, and Michael Schroeder. A roadmap of clustering algorithms: finding a match for a biomedical application. *Briefings in Bioinformatics*, 10(3):297–314, 2009.

[3] Arpita Nagpal, Aman Jatain, and Deepti Gaur. Review based on data clustering algorithms. In *Information & Communication Technologies (ICT), 2013 IEEE Conference on*, pages 298–303. IEEE, 2013.

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

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

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

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

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

[9] Christopher R. Palmer and Christos Faloutsos. Density biased sampling: An improved method for data mining and clustering. In *ACM SIGMOD Intl. Conference on Management of Data*, pages 82–92, Dallas, 2000.

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

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

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

[13] Miin-Shen Yang and Kuo-Lung Wu. A modified mountain clustering algorithm. *Pattern analysis and applications*, 8(1-2):125–138, 2005.

[14] Bernd Gutmann and Kristian Kersting. Stratified gradient boosting for fast training of conditional random fields. In *Proceedings of the 6th International Workshop on Multi-Relational Data Mining*, pages 56–68, 2007.

[15] M. Al-Kateb and B.S. Lee. Adaptive stratified reservoir sampling over heterogeneous data streams. *Information Systems*, 39(1):199–216, 2014.

[16] Alexandros Nanopoulos, Yannis Manolopoulos, and Yannis Theodoridis. An efficient and effective algorithm for density biased sampling. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 398–404, 2002.

[17] Xiaochun Wang, Xiali Wang, and D Mitch Wilkes. A divide-and-conquer approach for minimum spanning tree-based clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 21(7):945–958, 2009.