A progressive sampling framework for clustering

Frédéric Ros, Serge Guillaume

A progressive sampling framework for clustering

Frédéric Ros[a,*], Serge Guillaume[b]

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

Abstract

Clustering algorithms become more and more sophisticated to cope with large data sets of increasing complexity. Sampling selection methods are likely to provide an interesting alternative as they can reduce memory requirements, and reduce execution time. Many sampling algorithms for clustering are efficient but they each have their own limitations with large data sets. In this paper, we introduce a sampling framework for clustering algorithms that inherits from both progressive sampling and stratification concepts. Driven by two parameters, the iterative process consists in managing representatives of independent strata that carry similar statistical information regarding the clustering objective. At each iteration, the candidate representatives of the incoming stratum are examined. The interesting feature of the framework stems from the idea of selecting new representatives of the incoming stratum only if they improve the representation quality of the already selected set of samples. The algorithm stops when new representatives are no longer needed, which is likely to happen without examining the whole data set. The tests conducted on synthetic and real world datasets proved that the progressive sampling framework yielded similar results to the sampling algorithm applied to the whole set in a low computational time. In comparison with progressive sampling techniques, using the proposed framework enables smaller sampling sets to be used without loss of accuracy.

Keywords: stratification, progressive selection, clustering, nearest neighbor

1. Introduction

Clustering [1, 2, 3] is probably the most powerful unsupervised tool to identify a structure in a collection of unlabeled data. It can be formally defined as the process of organizing the data into groups whose members are similar in some way and different from the members of other clusters. Many methods were developed in the past and the topic has been continuously investigated

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

[4, 5, 6]. One of the recurrent challenges is related to the search for solutions to manage cluster complexity (variation in size, shape, density, overlapping, noise...) The era of big data has raised additional issues [7] for clustering. Most of the popular clustering algorithms were developed at a time when datasets were small or medium. Large datasets are intractable using these algorithms due to memory constraints or runtime issues. Improving and accelerating existing algorithms can help but is only a partial and limited solution in terms of scalability [8]: a new class of scalable processes for clustering is required. The main trend is to split the dataset and to handle the subsamples separately. Graph based structures [9, 10] can be used: the BIRCH algorithm [11] is based on a clustering feature tree structure. Other approaches implement incremental algorithms [12, 13, 14, 15, 16] or divide-and-conquer [17, 18] strategies.

In this context sampling, also known as instance selection in the machine learning community, appeared as an interesting alternative. The goal is to select a sample that behaves like the whole, i.e. without losing any valuable information [19, 20]. Sampling techniques [21, 22] have been explored and studied in many works [23]. Beyond uniform random sampling, to deal with data variety and to keep the sample size small, sampling algorithms were designed to account for the data structure, e.g. the density distribution [24, 25]. Unfortunately, the price to pay for this efficiency is that the use of the algorithms is restricted to moderate size datasets.

Stratification and progressive sampling are two ways to combine data splitting and sampling [26, 27]. In the stratification case, the dataset is split into strata, then an instance selection algorithm is applied to each stratum and the resulting sample is the union of the samples selected from each stratum. The complexity, both in time and in space, is reduced but all the strata are considered. Progressive sampling [28] consists, from a starting sample, in using increasing random samples until the model accuracy no longer improves. The learning algorithm, of which clustering is a good example, is generally embedded in the incremental process [29], meaning that the process is designed for a specific learning algorithm. The sampling schedule to be used with progressive sampling techniques is still an ongoing issue of research. Available schemes may either yield oversized samples or take too long when the sample size grows too slowly. It is worth mentioning that the progressive sampling approach does not require all the dataset to be analyzed.

This work proposes a sampling for clustering that inherits from both progressive sampling and stratification concepts. The dataset is split into strata that are statistically representative of the whole. The stratum size is designed for clustering purposes. The progressive characteristics do not require all the strata to be analyzed: the process ends when a stopping criterion is reached. The framework is generic: neither the sampling nor the clustering algorithms are embedded in the proposal. Any sampling/clustering combination can be used. The progressive building of the sample accounts for the already selected items. At a given iteration the incoming stratum is first managed by the already selected set of samples, then new representatives from the incoming stratum are selected only when this is needed to improve the quality of representation. The stopping criterion is based on the gain in quality of representation: when it drops below a defined threshold the algorithm ends. The contribution of this work includes a generic algorithm based on independent strata, a theoretical justification of appropriate stratum sizes suitable for clustering goals, a selection of new items based on their representative ability, and the proposition of a stopping criterion independent of any sampling or clustering algorithm.

The rest of the paper is organized as follows. Section 2 summarizes the state of the art about progressive sampling and stratification approaches. The proposed algorithm is introduced in Section 3 and its behavior with respect to two sensitive parameters, the stratum size and the stopping criterion, is illustrated using benchmark synthetic datasets. Numerical experiments using synthetic and real world datasets are reported in Section 4. First, the proposed framework is compared to the sampling algorithm applied to the whole set and, second, to other progressive sampling schemes. Finally, the main conclusions and open perspectives are stated in Section 5.

2. Related work

Sampling, also called instance selection in the machine learning community, is applied to speed up processes, especially clustering algorithms. Samples often provide sufficient accuracy with far less computational cost than clustering the whole dataset. Working with a reduced but representative sample proved necessary as the algorithms became more sophisticated to handle complex data and as the data size grew with the big data era.

Three kinds of strategies have been recently developed: sampling algorithms for moderate size datasets, stratification strategies for large datasets and progressive sampling for very large, even unloadable, datasets.

The first method to appear was random sampling, driven by the sample size parameter. Its main drawback is that there is no relationship between size and data structure. To ensure a good representation of the data, random sampling has to overestimate the sample size. Many methods were developed since this basic algorithm, either in a supervised learning context (the reader is referred to [23] for a recent review), or to deal with unsupervised tasks, such as clustering. In this case sampling algorithms became more and more complex to handle arbitrary shapes or densities. A review can be found in [30]. Recent algorithms are useful for knowledge discovery and data structure identification. They are mainly based on density [31], distance [30] or combine the two concepts [32]. Even if the new developments include an optimization effort, they are slower than uniform random sampling and their use is restricted to moderate size datasets.

To deal with large datasets, stratification strategies (not to be confused with stratified random sampling [33]), were introduced. The idea is to divide the whole set into disjoint strata and to process each stratum separately. The final result is the aggregation of each stratum result. This scheme is not restricted to sampling but also applies to clustering for instance. The process is obviously speeded up: a $O(n^2)$ algorithm runs faster when applied to strata of size $s \ll n$. Stratification techniques have received growing interest from the machine learning community in the 21st century [34, 35].

The strata can be designed to ensure that each of them has a similar distribution to that of the whole set, but even in this case the stratum size highly impacts the result [36]. Extensions have been proposed to work with non disjoint partitions, using replication techniques [37]. These methods take density into consideration. Dense areas are obviously represented in the strata while regions corresponding to noisy data are likely to be diluted in the whole set of strata. This way, they have less opportunity to be represented in the final set.

The algorithm proposed in [38] is driven by the sample size. To get the target size, the stratification strategy is recursively applied on the sampling result. In a supervised framework, the sample size is reduced using ensemble classifier concepts [39]. As each stratum is independently managed, a parallel implementation is possible. The Map-Reduce paradigm [40] provides a robust framework to process huge datasets using clusters of machines.

Progressive sampling was developed in parallel with stratification strategies [41, 28]. While the two concepts share the division into strata, the main difference is that using progressive sampling there is no need to examine all the strata but a stopping criterion is required. A generic progressive sampling algorithm is shown in Algorithm 1.

Algorithm 1 Generic progressive sampling algorithm

```
1:  Input: $X$                                   {Whole dataset}
2:  Output: $S$                                  {The sample set}
3:  Select initial sample $S$
4:  Compute $Q(S)$                               {Quality of sample $S$}
5:  repeat
6:      Select additional sample $S_i$
7:      $S = S \cup S_i$
8:      Compute $Q(S)$
9:  until ($Q(S)$ satisfies the stopping criterion)
10: return $S$
```

The concept is still an open research avenue [42, 22] even if it has been successfully used for supervised [43] as well as unsupervised tasks [29].

The proposed approaches differ in two key points: the sampling schedule, lines 3 and 6 of Algorithm 1, and the stopping criterion, lines 4 and 8. The first proposal was to design strata, the initial as well as the additional ones, with the same size. Theoretical work helped to set this parameter: the bounds come either from Chernoff [44] or Hoeffding [45] inequalities. According to [46], the two main methods are arithmetic or geometric [47] series for the additional sample size. The latter tends to overestimate the final sample size. The stopping criterion is met when the sample set is considered similar to the whole data. The *Probably Close Enough criterion* (PCE) [41] formalizes this idea when the whole dataset can be processed:

$$Pr(Q(X) - Q(S) > \varepsilon) \leq \delta \tag{1}$$

$Q(X)$ (resp. $Q(S)$) refers to the quality assessed on the whole (resp. sample) set, ε defines the meaning of *close enough*, and δ is a parameter describing what *probably* means.

In the unsupervised case, this is achieved using a statistical test to compare the two distributions. The most common ones are the *Chi-square* or the *divergence* tests. The latter was used in [48], where it was concluded that: "Unfortunately, the divergence test appears to be very conservative in practice, and the sample size yielded by this scheme often turns out to be nearly 50% of the whole, which is absolutely impractical for real very large data." The criterion may be specific to a given clustering algorithm: the sampling cost of the *k-means* is part of the whole stratification process in [49]. For supervised tasks, the quality function is generally based upon the classification accuracy.

When the whole dataset is unloadable or not available, only a sequential analysis is possible [50]. Chernoff inequality is also useful in this particular case where the expected value of the random variable is unknown. A relative error about $E[X]$ is considered, instead of an absolute one [51]. These concepts have been used in applications dealing with query size estimation for relational databases [52] or to track clusters in evolving datasets [53].

In [54] the authors used the evolution of the quality criterion to make the decision about convergence:

$$|Q^{i+1}(S) - Q^{i}(S)| \leq \varepsilon \tag{2}$$

where i and $i + 1$ are two consecutive iterations.

The proposal aimed at determining the additional sample size needed at each iteration until Eq. (2) is satisfied. For the next iteration, s_{i+1}, to meet Eq. (1), it is computed as:

$$s_{i+1} \geq \frac{2}{Q^{i}(S)} \left[\frac{1}{\varepsilon^2} log \frac{1}{\delta}\right] \tag{3}$$

The quality criterion can be an internal index, e.g. the Silhouette index, in the unsupervised case, or the classification accuracy in supervised learning.

This short review shows that stratification and progressive sampling techniques have been investigated and proved useful in many application domains. These results are the basis of the present proposal.

3. The proposed algorithm

Our progressive sampling algorithm is designed for clustering purposes. The sample set has to be as accurate as the whole set to identify clusters of different shapes or densities. The proposal assumes that a cluster is characterized by the distribution of its neighboring patterns. This distribution is likely to be quite homogeneous in the core of the cluster, with a decreasing level of homogeneity when moving away from the core. This internal structure is expected to be different for distinct clusters: the difference between two clusters stems from their inner spatial arrangement and their proximity. The sample size cannot be a parameter as it depends on data structure.

3.1. Key ideas

The proposal is based on random strata that are statistically representative. The stratum size ensures that all the clusters are represented in the stratum but it does not guarantee that all the clusters are covered by one stratum due to cluster variability in shapes and densities. To avoid dealing with outliers each stratum is filtered and sampled using a sampling algorithm, the choice of which is left to the user: the condensed set is assumed to be an accurate representation of the stratum. Instead of merely adding the samples, the proposal aims to take advantage of their complementarity: the sampling is considered as completed when the already selected sample set is able to accurately represent a new condensed stratum. Three key ideas, to be detailed below, support the proposal. First, each random stratum is an independent representative of the whole set. The stratum size must be carefully chosen to ensure this assumption. Second, as a preprocessing step each stratum is sampled separately in order to filter the data and to work with a smaller representative set. The third key notion is that there is no need to study all the strata thanks to a stopping criterion: the algorithm ends when the new stratum is represented, with the desired approximation level, by the already identified sample set.

Stratum size. The algorithm is based upon an important assumption: strata are independent and representative, meaning that each stratum carries similar statistical information regarding the clustering objective. Chernoff bounds were used to determine the stratum size for the smallest cluster to be represented in the sample with a minimum size. The Chernoff inequality states that large sums of independent variables are very predictable in their behavior.

Theorem 1. *Let X be the sum of independant variables defined in the unit interval, for any $0 < \varepsilon \leq 1$:*

$$P[X \leq (1-\varepsilon)E(X)] < e^{\frac{-E(X)\varepsilon^2}{2}}$$

Proof The proof is detailed in [55].

In the clustering framework X_i is a random variable that is 1 if the i^{th} point in the sample belongs to cluster c and 0 otherwise. The $X_1, \ldots, X_s$ are assumed to be independent. Let $X = \sum_{i=1}^{s} X_i$ and $|c_{min}|$ be the size of the smallest cluster in the dataset. The objective is then to have a minimum representation of this group in the sample, meaning that the probability of the number of points belonging to this group in the sample falls below λ, defined as a fraction of $|c_{min}|$, is less than δ. This is stated by Eq. (4).

$$P[X < \lambda] < \delta \tag{4}$$

Combining these two constraints, the minimum size is given in Theorem 2.

Theorem 2. *Let k be the number of clusters, let $\rho \geq 1$ define the minimum cluster size with respect to the whole dataset size, n: $c_{min} = \frac{n}{k\rho}$. The probability that any group includes fewer items than λ is less than δ ($0 \leq \delta \leq 1$) when the stratum size satisfies:*

$$s \geq \lambda k\rho + k\rho \; ln\frac{1}{\delta} + k\rho\sqrt{2\lambda \; ln\frac{1}{\delta} + \left(ln\frac{1}{\delta}\right)^2}$$

Proof The theorem was published in [56].

As the number of clusters is usually unknown, a cautious attitude is to overestimate it. In the remainder of this study, the product $k\rho$ is set at 20. The probability δ is set at 0.01. Therefore, the stratum size depends on λ which represents the minimum number of points per cluster in the sample.

Stratum sampling. The second idea is not to deal with the strata but to process each stratum with a sampling algorithm. This algorithm does not need to be specified in this section as the framework is independent of it. The whole set X, size n, is partitioned into f strata of size $s = n/f$: $X = \bigcup_{i=1}^{f} W_i$. The sampling algorithm is applied to each stratum, yielding $T_i \subset W_i$. Each item, y_m, in the sample set, T_i, is characterized by the number of items in W_i it represents, w_m. This is the number of items in W_i whose nearest neighbor in T_i is y_m. It is computed according to Eq. (5).

$$w_m = |\{x \in W_i \mid d(x, y_k) = \min_j d(x, y_j)\}|, \; j = 1, \ldots, |T_i|. \tag{5}$$

Stopping criterion. For stratification approaches, the resulting sample set would be the union of the samples from each stratum. The present proposal uses a stopping criterion based on the ability for the selected samples to represent an unknown stratum. At each step, t, of the algorithm three sets are considered:

- S_{t-1}: the current sample set, built during all the previous iterations ;
- T_{t-1}: this stratum is used as a candidate set of samples jointly with S_{t-1} ;
- T_t: the current stratum is used to test the representativeness of the candidate sample set.

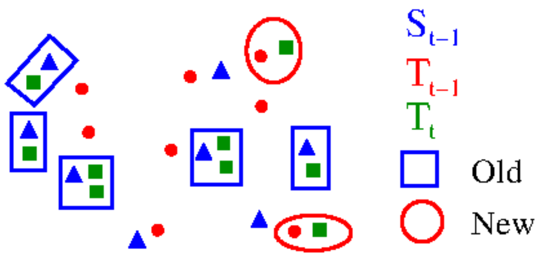

Figure 1: Algorithm behavior: illustration

The process is illustrated in Figure 1.

Each item from T_t (the green squares in Figure 1) is attached to the closest item in $P = S_{t-1} \cup T_{t-1}$. The distance between a set, A, and an item, $x \notin A$, is defined according to Eq. (6).

$$d_A(x) = \min_{y \in A} d(x, y) \tag{6}$$

The set of representatives of T_t is:

$$P_t = \{y \in P \mid \exists x \in T_t \ d_P(x) = d(x, y)\ \}$$

The items from P_t come from either S_{t-1} (the blue triangles in the figure) or T_{t-1} (the red disks). The first type, the already identified representatives, are plotted in a blue rectangle in the figure while the latter, the new representatives, are in a red circle. The stopping criterion involves the sampling cost. Let P_{t-1} be the set of representatives for T_t without considering the new set of candidates, T_{t-1}:

$$P_{t-1} = \{y \in S_{t-1} \mid \exists x \in T_t \ d_{S_{t-1}}(x) = d(x, y)\ \}$$

Let $C_{P_t}(T_t)$ and $C_{P_{t-1}}(T_t)$ be the corresponding sampling costs for T_t. The cost computation is an approximation as the weight is applied to the representative, x, instead of involving the distances between each item in W_t and x. It is computed for, e.g., P_t according to Eq. (7), where w is defined in Eq (5).

$$C_{P_t}(T_t) = \sum_{x \in T_t} w_x * d_{P_t}(x) \tag{7}$$

As $P_{t-1} \subset P_t$, it follows that $C_{P_t}(T_t) \leq C_{P_{t-1}}(T_t)$. When the ratio $\dfrac{C_{P_t}(T_t)}{C_{P_{t-1}}(T_t)}$ is above a given threshold, q_{th}, the new representatives make only a small contribution to the representation of the new stratum and the algorithm ends. This statement can be expressed in the coreset formalism [57].

Definition 1. *A subset Q of the input set R is called an ϵ-coreset of R with respect to an optimization problem, if solving the optimization problem on Q gives and ϵ-approximate solution of the problem on the whole input set, R.*

This definition can be specified when the optimization involves a cost function.

Definition 2. *$Q \subset R$ is an ϵ-coreset of R, if for any S:*

$$(1-\epsilon)Cost_Q(S) \leq Cost_R(S) \leq (1+\epsilon)Cost_Q(S)$$

where $Cost_X(S)$ denotes the cost of S when the problem is solved using X.

Theorem 3. *Given $P = S_{t-1} \cup T_{t-1}$ and the nearest neighbor problem for T_t in P, when q_{th} is close to 1, $P_{t-1} \subset S_{t-1}$ is an ϵ-coreset of P.*

Proof Let $q_{th} + \epsilon = 1$ be the relation between q_{th} and ϵ.
The stopping criterion states: $q_{th} \leq \dfrac{C_{P_t}(T_t)}{C_{P_{t-1}}(T_t)} \leq 1$.
Substituting q_{th} and multiplying by $C_{P_{t-1}}(T_t)$ yields:

$$(1-\epsilon)C_{P_{t-1}}(T_t) \leq C_{P_t}(T_t) \leq C_{P_{t-1}}(T_t)$$

And finally: $(1-\epsilon)C_{P_{t-1}}(T_t) \leq C_{P_t}(T_t) \leq (1+\epsilon)C_{P_{t-1}}(T_t)$.

When the stopping criterion is reached P_{t-1} is an *ϵ-coreset* of P for an unknown candidate stratum T_t. As $P_{t-1} \subset S_{t-1}$, S_{t-1} is an *ϵ-coreset* of P for T_t, and as this statement also holds for the previous steps, S_{t-1} is an *ϵ-coreset* of P for the union of the T_j strata, $j = 1, \ldots, t$.

3.2. Description of the algorithm

The process is summarized in Algorithm 2. The inputs of the algorithm are the dataset, the sampling algorithm and two parameters that drive the process: the stratum size and the threshold, q_{th}, that serves as a stopping criterion. These parameters are studied in the experimental section.

The partition into strata is achieved in line 4. As previously described, three sets of data are needed at a given iteration, t: the current stratum, T_t, is used as a validation set, the previous one, T_{t-1} is part of the candidate set of representatives jointly with the set of already selected representatives S_{t-1}. The algorithm starts at $t = 1$ and the set of already selected items is empty, $S_0 = \emptyset$ (line 3). The strata are sampled before being processed, lines 5 and 7.

The main *while* loop begins after this initialization stage (lines 6-26). The current stratum is sampled (line 7) and processed as illustrated in Figure 1. The candidate set of representatives, P, is defined in line 7. Then the two sets of representatives, the one including the candidates from T_{t-1}, called P_t, and the one based only on the already selected samples, called P_{t-1}, are computed in lines 9 and 10. Then, except for the first iteration, line 11, the cost ratio is computed, lines 12. The algorithm stops when it is higher than the threshold parameter, q_{th}, line 14. Otherwise, the new selected items are added to the sample set in line 17 and the next stratum is processed, line 18. If needed, a new random partition is generated in line 20. This may be needed when the number of strata is quite small, when the desired quality of representation is high and for small/medium databases.

Algorithm 2 *Progressive sampling*

```
1: Input: $X$ (dataset), $s$ (stratum size), $A$ (sampling algorithm), $q_{th}$
2: Output: $S$ (selected subset $S \subset X$)
3: $S_0 = \emptyset, f = n/s, q = 0, t = j = 1$, stop=**false**
4: $\{W_i\}$ = **Partition**(X,s)    $\{X = \bigcup_{i=0}^{f} W_i\}$
5: $T_0$ = **StratumSampling**($W_0$,$A$)
6: **while** (stop=**false**) **do**
7:   $T_t$ = **StratumSampling**($W_j$,$A$)
8:   $P = S_{t-1} \cup T_{t-1}$    {Candidate set of samples at step $t$.}
9:   $P_t = \{y \in P \mid \exists x \in T_t \ d_P(x) = d(x,y) \}$
10:  $P_{t-1} = \{y \in S_{t-1} \mid \exists x \in T_t \ d_{S_{t-1}}(x) = d(x,y) \}$
        {Set of representatives of $T_t$ in $P$ (line 9) and $S_{t-1}$}
11:  **if** ($S_{t-1} \neq \emptyset$) **then**
12:    $q = \frac{C_{P_t}(T_t)}{C_{P_{t-1}}(T_t)}$    {Costs computed according to Eq. (7)}
13:  **end if**
14:  **if** ($q \geq q_{th}$) **then**
15:    stop=**true**
16:  **else**
17:    $S_t = S_{t-1} \cup P_t$
18:    $t = t + 1$, $j \equiv (t + 1)[f]$
19:    **if** ($j = 0$) **then**
20:      $\{W_i\}$ = **Partition**(X,s)    {Partition regeneration}
21:    **end if**
22:  **end if**
23: **end while**
24: Return $S_t$
```

Optimizing the number of computed distances. The number of groups, g, is the size of the first set of representatives, $g = |T_0|$ (line 5 of Algorithm 2). Each of them is characterized by its size, w_i for group i, and a hyper radius, r_i. After the initialization, the size is one and is radius is zero for all groups. When a new representative is added to the sample it is attached to the nearest group, and the corresponding values, w_i and r_i, are updated. The algorithm also updates the distances between group centers. To find the nearest neighbor of a given item, x, only a small region of the space needs to be considered. Thanks to a smart use of the triangle inequality, see Figure 3 in [32], only a reduced number of distances has to be computed: $g + w_i + \epsilon \ll |S_{t-1}|$. This number includes three components: the number of initial groups, the number of points in the nearest group and a remaining term that depends on the data structure to account for the possibility that the nearest neighbor may not belong to the nearest group. This leads to a substantial time reduction and can be improved even further.

Algorithm complexity. The complexity of the proposal is $O(gs^2)$ instead of $O(n^2)$ In the case of small to medium database sizes, there is no obvious time-saving advantage in applying progressive sampling as g can be greater than s.

3.3. Algorithm behavior

The *stratum size* is an important parameter. To assess its influence on the process, tests with synthetic data are conducted. Five clusters following a Gaussian distribution are generated [58] in various dimensions and with a varying total number of records. The only parameter used in this work is the minimum number of points for any cluster, λ. The remaining parameters required for stratum size computation, Theorem (2), are: $k = 20$ and $\delta = 0.01$. The number of clusters is set at a high value as it is usually unknown. The sampling algorithm is *ProTraS* [32]. It is a recursive partitioning algorithm: at each step a new representative is added to the sample until the cost falls below a threshold. The new representative is chosen as the farthest from the one already selected in the group with the highest probability of cost reduction. This probability is computed according to the within group distance and to the representativeness of the sample item, assessed by the number of items in the whole set it represents. This algorithm is fully driven by a unique parameter: the sampling cost. The lower the cost, the more accurate the representation and the larger the sampling set.

The *ProTraS* parameter cost is 0.1 and the threshold is $q_{th} = 0.99$.

The accuracy is assessed by the difference in the center locations between the partitions given by the *k-means* algorithm applied to the sample, c_i, and to the whole set, c_j. This difference is weighted by the cluster sizes w_i and w_j. The value is divided by the space dimension, p, as shown in Eq. 8. The distance,

d_m, is the Manhattan distance, and n is the size of the dataset.

$$acc = \frac{\sum_{i=1}^{k} (w_i + w_j) \quad \underset{c_j}{min} \quad d_m(c_i, c_j)}{2np} \tag{8}$$

The randomly chosen initial values, one for each dimension, of the centers are the same for the two sets as suggested in [58]. The running time is the ratio of the time required by the proposal to the one corresponding to the sampling of the whole dataset. The sample size is another important characteristic of the process.

Process stability. The first experiment aims to check the stability of the process. Tests were conducted with a wide range of dimensions and dataset size. The results are illustrated using data of 200000 items in 15 dimensions. For each stratum size, 50 runs were carried out for a number of clusters from 2 to 7. The number of iterations achieved by the proposal was also analyzed. Table 1 shows the results. Each value is the average over the 300 trials (50 runs for each of the 6 numbers of clusters).

Table 1: The stability test results

	Acc		Sample size ratio		Time ratio	
size	mean	sd	mean	sd	mean	sd
258	0.106	0.150	0.003	0.001	0.009	0.020
504	0.100	0.138	0.004	0.001	0.012	0.063
1531	0.105	0.150	0.011	0.002	0.087	0.071
2706	0.086	0.120	0.019	0.000	0.261	0.091

The stratum sizes of 258, 504, 1531 and 2706 correspond to a minimum number of points for the smallest cluster, i.e. 2, 10, 50 and 100.

The results show a great stability with all the indices. The accuracy is good for all the configurations in average. The variations in the standard deviation are due to the *k-means* random seeding. The number of samples increases with the stratum size, but not in the same proportion: it is ten times bigger when the stratum size is multiplied by fifty. This also holds for the running time. Using the *k-means* algorithm and with this data configuration, the proposal is quicker, i.e. with a time ratio < 1, than the clustering applied to the whole set even if it becomes slower for large stratum sizes.

Stratum size impact according to data size and dimension. Experiments were conducted using data sizes up to 5 million items with an input space dimension up to 80. The main conclusions to be drawn from the results, given in Appendix A, are:

- The accuracy does not depend on the dataset size nor on the input space dimension. The variations are only due to the *k-means* random initialization.

- The ratio of the sample size to the whole size results clearly increases with the stratum size, it decreases with the dataset size and does not depend on the space dimension. For a given stratum size, the number of samples does not depend on the dataset size, it is related to the data structure.
- The time ratio increases with the stratum size while it decreases with the number of items in the dataset and, to a lesser extent, with the input space dimension.

Stopping criterion influence. The *threshold* q_{th} is the stopping criterion. To assess its influence on the process, extensive tests with a synthetic benchmark database [59] were conducted with different stratum sizes. As the conclusions were the same for all the stratum sizes, only one experiment, with a stratum size of 504, is reported. The dataset contains 6751 patterns in 10 dimensions. It is structured in 9 Gaussian clusters with different levels of separability. q_{th} ranges from 0.6 to 0.999 and the algorithm was evaluated for 2 to 11 clusters. Table 2 shows the Adjusted Rand Index (ARI) values for each configuration. The average ARI for all configurations with a given threshold is in the last column.

Table 2: Behavior of the algorithm according to the stopping criterion threshold

q_{th}	Sample size	2	3	4	5	6	7	8	9	10	mean
0.60	40	0.872	0.824	0.932	0.898	0.922	0.928	0.926	0.900	0.884	0.890
0.70	55	0.895	0.954	0.959	0.914	0.888	0.931	0.944	0.927	0.918	0.918
0.80	59	0.894	0.940	0.970	0.937	0.948	0.929	0.954	0.954	0.922	0.939
0.90	68	0.931	0.932	0.959	0.993	0.928	0.901	0.974	0.947	0.939	0.945
0.92	81	1.000	0.897	0.932	0.837	0.970	0.917	0.947	0.970	0.961	0.937
0.94	90	0.931	0.978	0.968	0.905	0.959	0.974	0.941	0.980	0.914	0.950
0.96	108	0.870	1.000	0.930	0.981	0.942	0.935	0.972	0.971	0.877	0.942
0.98	110	1.000	0.988	0.959	0.976	0.954	0.922	0.965	0.950	0.953	0.964
0.99	122	0.961	0,954	0.960	0.989	0.914	0.976	0.977	0.960	0.990	0.970
0.999	210	0.951	0.965	0.999	0.951	0.943	0.999	0.912	0.948	0.956	0.959

It is shown that the ARI are all high and differ slightly with the number of clusters. A finer analysis shows that sometimes there is a difference between the required number of clusters and their real number. Some clusters are empty or not representative (less than 1% of the data size). This may occur for both the sample and the whole sets. Even when limited to an easy scenario, as the clusters are well-separated, these results show that the higher q_{th} is, the better the efficiency tends to be (0.970, 0.958 for $q_{th} \geq 0.99$). When the threshold is low (0.6) the ARI can be less than 0.85 for some configurations with a small number of clusters. To summarize, when *natural* clusters exist, or when the groups are well separated, a large range of threshold values yields similar good results.

Hierarchical clustering algorithm. The experimental section is based on the use of *k-means* as it allows extensive tests in a reasonable time. It is however interesting to evaluate the time ratio when a more complex algorithm is used such as a hierarchical clustering one. The improved hierarchical algorithm from [60] was considered. This kind of algorithm is quite slow but efficient in finding

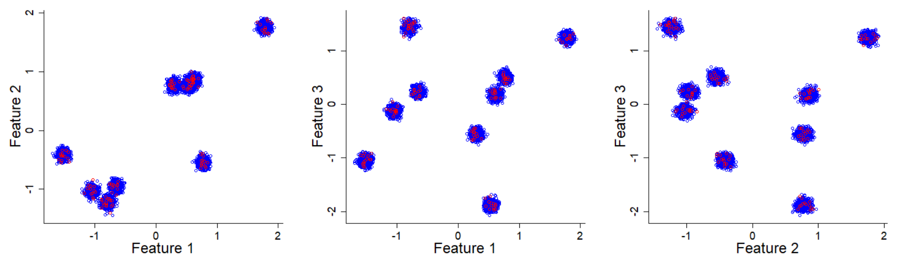

Figure 2: 9 Gaussian data and samples in feature spaces 1-2, 1-3 and 2-3 (data in blue and samples in red)

acceptable partitions even in the presence of noise. A systematic comparison can be done as it takes the number of clusters as an input parameter. The ARI is obtained by applying the protocol of [31]. Using a single computer, the hierarchical algorithm takes more than 30 minutes to process the whole data set (6751 patterns) as it takes less than 5 seconds to process the prototype data set (122 patterns) with an $ARI \simeq 1$.

4. Numerical experiments

The proposal is now tested using synthetic and real datasets. The experiments compare the sample size, the running time and the partitions that result from a clustering applied to the whole set to the ones yielded by the same clustering algorithm applied to the sample. The final sample is given either by the proposal, the stratum sampling algorithm run on the whole set or alternative progressive sampling strategies. Three are considered: in the first one the final sample is the union of the samples from all strata, in the two remaining ones the stratum size follows an arithmetic or a geometric progression. The accuracy is measured by the ARI, the Ratio and Time indices are the ones used in the previous section. The ARI returns a value of 1 when two partitions are in complete agreement, 0 when the partitions are expected by chance and a negative value when the partitions are in greater disagreement than would be expected from chance.

Despite its well known limitations, the *k-means* algorithm is mainly used in this study as it allows for extensive tests even on large datasets. Moreover the objective is not to find a good partition but to compare the partitions built from the whole set or samples given by alternative sampling methods. The sampling algorithm used in all these experiments was *ProTraS* [32]. The default value used in this paper is 0.1. The threshold used as a stopping criterion for the algorithm is $q_{th} = 0.99$.

Table 3: The synthetic datasets

	Size	Dim	Name	Origin
S1	3000	2	A.set 1	[61]
S2	5250	2	A.set 2	[61]
S3	7500	2	A.set 3	[61]
S4	100000	2	Birch-set 3	[62]
S5	5000	2	S.sets 1	[63]
S6	5000	2	S.sets 2	[63]
S7	5000	2	S.sets 3	[63]
S8	5000	2	S.sets 4	[63]
S9	1351	2	Dim sets 1	Footnote 1
S10	2701	4	Dim sets 2	Footnote 1
S11	4051	6	Dim sets 3	Footnote 1
S12	5401	8	Dim sets 4	Footnote 1
S13	6751	10	Dim sets 5	Footnote 1
S14	2000	2	Spirale set	[64]
S15	8000	2	t4.8k	[65]
S16	10000	9	cluto-t7.10k	Footnote 2
S17	5500	2	Homemade	Fig. 3
S18	3800	2	Homemade	Fig. 3
S19	12500	2	Homemade	Fig. 3
S20	40000	2	Homemade	Fig. 3

4.1. The synthetic and real world datasets

Some are from the data clustering repository of the computing school of Eastern Finland University[1], while others come from a benchmark data for clustering repository [2] or were proposed in the published literature. These data are quite interesting as their structure is known and, in most cases, they are easy to display. They are used to ensure that the proposal is able to handle complex data of varying shapes and densities. The data characteristics are summarized in Tables 3 and 4.

[1]https://cs.joensuu.fi/sipu/datasets/
[2]https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/

Table 4: The real world datasets

	Size	Dim	Name	Origin
R1	68040	9	Color moments	Footnote 1
R2	169308	3	Differential coordinates	[63]
R3	13467	2	User Location (Finland)	[63]
R4	857357	3	Transactions90k	[66]
R5	1837	3	House5	[63]
R6	34112	3	House8	[63]
R7	45781	3	Tamildanu	*UCI*
R8	434874	3	3D road network	*UCI*
R9	245057	3	Skin segmentation	*UCI*
R10	1044506	9	House power	*UCI*

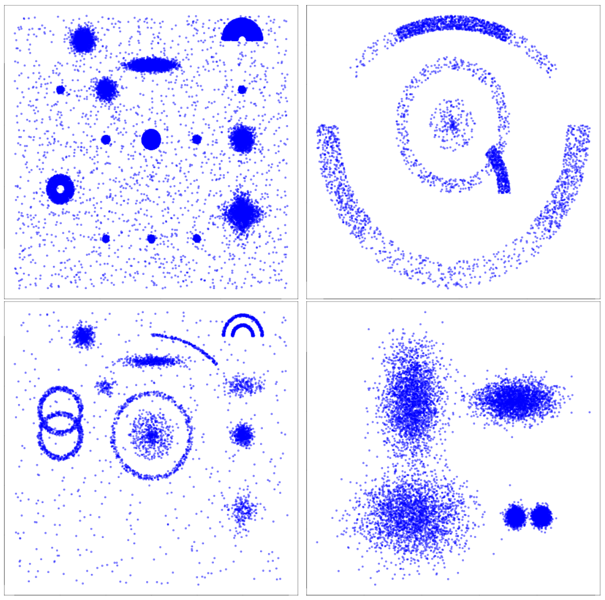

Figure 3: Datasets S17, S18, S19 and S20

4.2. Comparison with the clustering algorithm applied to the whole set

The results are given in Tables 5 for the synthetic datasets and 6 for the real ones. The two tables have a similar structure: each value is the average of ten runs for each configuration with a number of clusters between 2 and 10. The *mean* (resp. *sd*) is the mean of the *mean* (resp. *sd*) of the column. In these tables, the same four stratum sizes used in Table 1 are considered. To alleviate the notations the stratum size is referred to as the minimum number of points for the smallest cluster, i.e. 2, 10, 50 or 100.

Table 5: Comparison with the clustering algorithm applied to the whole set for the synthetic datasets

	ARI				Sample size ratio				Time ratio			
	a2	a10	a50	a100	r2	r10	r50	r100	t2	t10	t50	t100
S1	0.929	0.945	0.952	0.958	0.175	0.234	0.231	0.225	0.730	1.405	1.357	1.351
S2	0.922	0.942	0.944	0.927	0.119	0.155	0.189	0.184	0.488	0.819	2.464	1.901
S3	0.922	0.923	0.923	0.936	0.087	0.120	0.153	0.155	0.363	0.659	4.370	3.636
S4	0.907	0.927	0.930	0.953	0.006	0.009	0.014	0.017	0.027	0.052	0.317	1.348
S5	0.932	0.961	0.951	0.953	0.093	0.126	0.152	0.151	0.349	0.669	1.501	1.528
S6	0.939	0.940	0.958	0.962	0.121	0.177	0.211	0.206	0.452	0.954	2.045	2.052
S7	0.932	0.954	0.965	0.964	0.150	0.218	0.268	0.269	0.535	1.137	2.424	2.529
S8	0.895	0.911	0.924	0.923	0.156	0.222	0.276	0.275	0.549	1.142	2.478	2.733
S9	0.900	0.907	0.892	0.887	0.020	0.019	0.020	0.020	0.134	0.134	0.141	0.136
S10	0.955	0.931	0.950	0.939	0.012	0.010	0.009	0.009	0.087	0.132	0.135	0.127
S11	0.926	0.940	0.944	0.945	0.011	0.013	0.014	0.014	0.052	0.093	0.164	0.166
S12	0.906	0.927	0.950	0.956	0.011	0.015	0.018	0.018	0.045	0.090	0.231	0.216
S13	0.906	0.927	0.950	0.956	0.012	0.016	0.023	0.022	0.040	0.100	0.331	0.293
S14	0.789	0.818	0.837	0.825	0.270	0.283	0.286	0.284	1.086	1.303	1.377	1.324
S15	0.907	0.926	0.961	0.939	0.109	0.166	0.229	0.225	0.291	0.627	3.654	3.859
S16	0.944	0.959	0.967	0.967	0.108	0.152	0.205	0.215	0.283	0.553	2.841	3.576
S17	0.919	0.947	0.948	0.951	0.090	0.121	0.150	0.147	0.315	0.602	1.628	1.598
S18	0.951	0.944	0.953	0.946	0.116	0.139	0.146	0.147	0.387	0.651	0.964	1.039
S19	0.940	0.949	0.967	0.978	0.037	0.048	0.071	0.079	0.153	0.264	1.410	7.044
S20	0.942	0.935	0.960	0.961	0.006	0.009	0.015	0.018	0.028	0.067	0.332	1.891
mean	0.918	0.930	0.941	0.942	0.085	0.113	0.134	0.134	0.320	0.573	1.508	1.917
std	0.035	0.030	0.031	0.032	0.004	0.004	0.004	0.005	0.109	0.222	0.732	0.912

The main trends that can be analyzed from these tables are that the stratum size has a strong influence on the results: large sizes yield a better accuracy but a slower running time and a higher sample size ratio. Moreover the standard deviation decreases with the stratum size, reflecting a more stable process. These results were expected as a larger stratum size is likely to be more representative of the whole set but requires more computational time.

To analyze these results, one has to be aware that the accuracy is measured using the Adjusted Rand Index, not the classical one. Moreover, as each value is the average for a number of clusters between 2 and 10, the accuracy may be poorer when the number of requested clusters does not correspond to the data structure. This especially holds when the number of clusters is higher than the number of *natural* groups. In this case, several clusters correspond to a group and this is likely to generate more ambiguous partitions from one run to another, or from one set to another.

With the real world datasets the stratum size has a major impact on the accuracy. This is because these data are not as structured as the synthetic ones and a larger size is needed to make the strata representative.

The time ratio is less than 1, meaning that the proposal is faster than the alternative without any sampling, for the two smallest stratum sizes in average for the two kinds of datasets. This result should be carefully analyzed as the datasets have a highly variable number of records, from 1351 for $S9$ to more than one million for $R10$. The progressive sampling algorithm is useful only to manage large datasets, otherwise applying a sampling algorithm to the whole set is smarter. Nevertheless, all these datasets were used as classical benchmarks to ensure that the proposal is able to select a representative sample set. When

Table 6: Comparison with the clustering algorithm applied to the whole set for the real world datasets

	ARI				Sample size ratio				Time ratio			
	a2	a10	a50	a100	r2	r10	r50	r100	t2	t10	t50	t100
R1	0.643	0.724	0.758	0.861	0.023	0.059	0.154	0.239	0.167	0.701	4.110	5.437
R2	0.573	0.666	0.790	0.844	0.005	0.009	0.017	0.019	0.044	0.143	0.582	0.628
R3	0.969	0.980	0.981	0.967	0.014	0.019	0.021	0.023	0.112	0.251	0.821	1.894
R4	0.905	0.928	0.961	0.968	0.002	0.005	0.009	0.008	0.020	0.110	0.532	0.239
R5	0.760	0.799	0.800	0.854	0.370	0.340	0.439	0.392	1.451	1.481	1.533	1.463
R6	0.903	0.890	0.906	0.976	0.032	0.024	0.070	0.070	0.298	0.244	2.121	2.415
R7	0.851	0.913	0.949	0.965	0.027	0.062	0.129	0.120	0.141	0.620	2.550	2.716
R8	0.908	0.943	0.956	0.965	0.004	0.011	0.034	0.032	0.030	0.198	1.130	1.142
R9	0.941	0.918	0.958	0.979	0.002	0.004	0.006	0.007	0.020	0.005	0.270	0.265
R10	0.880	0.904	0.947	0.969	0.002	0.004	0.014	0.017	0.010	0.047	0.600	0.566
mean	0.833	0.866	0.906	0.935	0.048	0.054	0.089	0.131	0.228	0.380	1,419	1.676
sd	0.132	0.102	0.084	0.056	0.113	0.102	0.133	0.003	0.438	0.449	1.194	1.576

the data size is too small, not only are all the strata processed but, to reach a good accuracy more partitions are generated (line 22 of Algorithm 2). When dealing with large datasets the time ratio is always less than 1, even for large stratum sizes, e.g. $R2$, $R4$, $R9$, $R10$.

On the other hand, when the data are structured in groups the results are quite good even with small stratum sizes. This is the case for $S18$ and $S20$. The latter, processed using the smallest stratum size (258 points), yields an ARI of 0.942 with a sample size ratio of 0.006, meaning that the sample size is 240, and a time ratio of 0.028, meaning that the proposal is 35 times faster.

4.3. Comparison with other methods

The proposed progressive sampling algorithm is now compared to alternative methods. Four were considered:

1. *ProTraS* [32]: the sampling algorithm is applied to the whole set. The parameter is the cost sampling.
2. Full: each stratum is sampled and the resulting sample set is the union of the samples from all strata [67]. The parameters are the *ProTraS* cost and the stratum size.
3. Geom [46]: the stratum size follows a geometric progression. The parameter is the common ratio.
4. Arith [46]: the stratum size follows an arithmetic progression. The parameter is the common difference.

For each of the competitive algorithms various configurations were tested. The values of the parameters are given in parenthesis in the corresponding tables. With the last two methods, the initial value is $max(n/1000, 500)$ and the algorithm ends when the centers given by the *k-means* algorithm are stable, meaning that the average distance change is less than 0.001. The proposal is run with a *ProTraS* cost set at 0.1 and the parameter is the stratum size. For each dataset, the process was run 10 times for each number of clusters, between 2 and 10. The values are averaged over all the number of clusters and all the runs for each dataset. These results are then averaged for all the datasets, either

Table 7: Comparison with other methods: synthetic datasets

	Acc		Sample size ratio		Time ratio	
	mean	std	mean	std	mean	std
ProTraS (0.15)	0.745	0.151	0.015	0.000	0.568	0.303
ProTraS (0.1)	0.836	0.128	0.027	0.000	0.598	0.305
ProTraS (0.05)	0.926	0.087	0.082	0.001	0.723	0.347
ProTraS (0.02)	0.968	0.050	0.276	0.002	1.349	0.524
Full (0.1,50)	0.956	0.143	0.261	0.089	1.944	0.004
Full (0.2,2)	0.940	0.143	0.145	0.089	1.153	0.004
Full (0.2,10)	0.929	0.143	0.145	0.089	1.153	0.004
Full (0.3,50)	0.891	0.143	0.083	0.089	0.683	0.004
Geom (1.1)	0.874	0.116	0.148	0.027	1.909	0.950
Geom (1.2)	0.906	0.120	0.238	0.099	3.218	1.959
Geom (1.5)	0.934	0.100	0.466	0.166	4.284	1.254
Geom (1.8)	0.945	0.085	0.538	0.122	3.685	0.590
Arith (200)	0.911	0.110	0.222	0.061	2.820	1.176
Arith (500)	0.925	0.077	0.338	0.101	3.688	1.432
Arith (1000)	0.955	0.064	0.461	0.118	4.124	1.129
Arith (2000)	0.922	0.071	0.558	0.129	3.646	0.542
Proposal (2)	0.918	0.035	0.085	0.004	0.320	0.109
Proposal (10)	0.930	0.030	0.113	0.004	0.573	0.222
Proposal (50)	0.941	0.031	0.134	0.004	1.508	0.732
Proposal (100)	0.942	0.032	0.134	0.005	1.917	0.912

synthetic or real world, in Tables 7 for the synthetic data and 8 for the real world ones.

Compared to *ProTraS* applied to the whole set, the proposal yields comparable accurate results, especially with a 0.1 cost, which is the *ProTraS* parameter for the proposal. That result was expected as *ProTraS* is used by the proposal. That also means that the progressive sampling scheme does not harm the performance. The sample set given by *ProTraS* is smaller but for the lowest cost (0.02), which also corresponds to a higher accuracy. The computational time required by the proposal is smaller for the real world datasets. Moreover the use of *ProTraS* with the whole set is restricted due to memory constraints: the limitations come from the dataset size but also from the number of representatives. This is not the case for the proposal.

When the sample set is the union of all the representatives of the strata, *Full*, the proposal seems to be more efficient. Considering the real world data, the most accurate configuration for the *Full* scheme is with the parameters 0.1 for *ProTraS* and 50 for the stratum size. The accuracy is comparable to the proposal with the smaller stratum size. In this case, the sample set is 6 times smaller and the algorithm is twice as fast.

When the stratum size follows an arithmetic or geometric sequence, the sample size is significantly higher than the one yielded by the proposal and the required processing time is noticeably increased.

The poor performance of the competitive progressive sampling frameworks may be explained by the lack of quality control: they just add randomly chosen representatives without considering the ones already selected. The proposal thus gives smaller sample sets in less time to get a comparable, or even higher, accuracy.

The comparison is now illustrated using the $R10$ real world dataset. Figure 4

Table 8: Comparison with other methods: real world datasets

	Acc		Sample size ratio		Time ratio	
	mean	std	mean	std	mean	std
ProTraS (0.15)	0.818	0.159	0.033	0.001	0.672	0.299
ProTraS (0.10)	0.868	0.119	0.058	0.001	0.778	0.347
ProTraS (0.05)	0.911	0.088	0.113	0.001	1.492	1.449
ProTraS (0.02)	0.931	0.079	0.172	0.001	2.096	0.924
Full (0.1,50)	0.835	0.138	0.315	0.001	0.415	0.030
Full (0.2,10)	0.694	0.206	0.165	0.001	0,245	0.031
Full (0.3,100)	0.657	0.189	0.073	0.000	0.478	0.201
Full (0.1,10)	0.789	0.146	0.258	0.001	0.396	0.052
Geom (1.1)	0.841	0.132	0.075	0.041	1.419	1.258
Geom (1.2)	0.853	0.173	0.201	0.103	2.925	1.523
Geom (1.5)	0.882	0.125	0.280	0.093	2.387	0.695
Geom (1.8)	0.903	0.114	0.310	0.084	2.115	0.551
Arith (200)	0.865	0.135	0.132	0.033	2.086	0.693
Arith (500)	0.866	0.127	0.155	0.022	1.793	0.528
Arith (1000)	0.858	0.152	0.179	0.036	1.888	0.870
Arith (2000)	0.872	0.119	0.176	0.043	1.959	0.707
Proposal (2)	0.833	0.132	0.048	0.113	0.228	0.438
Proposal (10)	0.866	0.102	0.054	0.102	0.380	0.449
Proposal (50)	0.906	0.084	0.089	0.133	1.419	1.194
Proposal (100)	0.935	0.056	0.131	0.089	1.676	1.576

shows the evolution of the quality criterion, q, for the four stratum sizes until the threshold, $q_{th} = 0.99$ is reached. The colors for increasing sizes are: Green, Grey, Red and Blue. The abscissa is the sample set size. The smaller the stratum size, the smaller the sample set.

Figure 5 shows the evolution of the sampling cost when *ProTraS* is applied to the whole set $R10$. The ordinate is the *ProTraS* cost that is based on distances between the original items and their representatives [32]. The smaller the cost, the higher the number of representatives. The process was run with a cost parameter of 0.02 but, due to memory constraints, it stopped when the sample reached the size of 14000 items. The current cost was then 0.17.

Figure 6 shows the evolution of the sample set size when the stratum size follows an arithmetic sequence. The evolution is similar when the progression is geometric. The four colors correspond to increasing common differences.

As the evolutions plotted in the figures use internal indices, the comparison can only be based upon common criteria. It is important to clarify the relationship between the sampling cost used by *ProTraS* and the Adjusted Rand Index. The former is the sum of distances of the original items to their representatives. When it tends to zero, the ARI tends to one. But the opposite does not hold: the ARI may be very high with a large sampling cost when the groups are well separated. The ARI is the basis of the comparison, with the corresponding sample size and time ratio.

For the $R10$ data, with more than $1M$ of data patterns in 9 dimensions, *ProTraS* and the proposal reach comparable high ARI with a similar number of prototypes (from 2000 to 14000). For both of them the increase in ARI (from about 0.9 to 0.960) is very small compared to the increase in the sample size. The main difference is in the running time. *ProTraS* is comparatively slower: it selects a 14000 sample set in a time ratio of about 0.86 compared to about

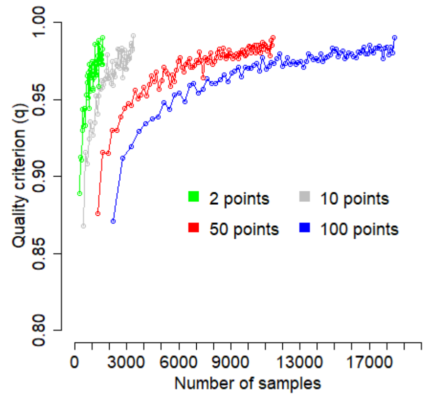

Figure 4: Illustration of the process using the proposed progressive sampling algorithm for $R10$ data

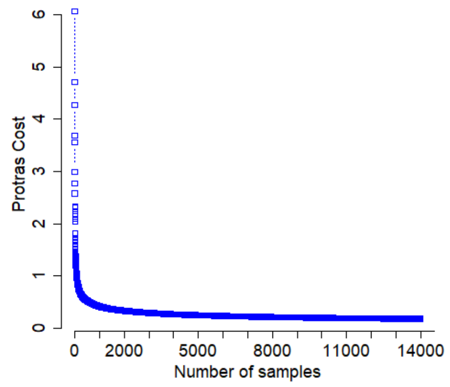

Figure 5: Evolution of the $ProTraS$ cost applied to the whole set for $R10$ data

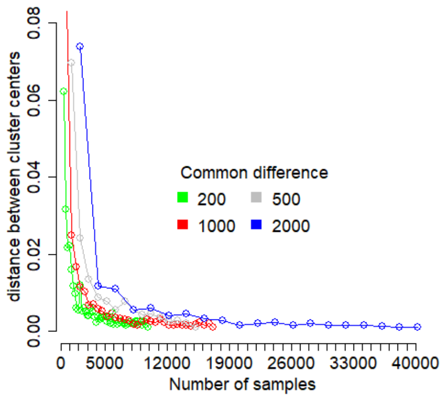

Figure 6: Arithmetic sequence for $R10$ data: evolution of the average distance between cluster centers with the sample set size

0.5 for the same sample set size using the proposal.

With progressive sampling schemes based on an arithmetic sequence, ARI values range from 0.887 to 0.960 with a sample size from 11000 to 40000. Figure 6 shows that the larger the common difference, the greater the sample size. It also shows that an increase of about 30000 in the sample size, which is only 11000, is needed to improve ARI in 0.07. To ensure that a good level of representation is reached, the sample size must be overestimated due to the lack of internal process information as the samples are selected in a random way: only their number is controlled, not their contribution to the representation. Similar ARI are achieved using the proposal for sample sizes 5 times smaller: from 2000 to 14000. The same remark holds for the time ratio: it is in the range [0.1,0.5] for the proposal instead of [0.81,0.86] for the arithmetic sequence.

5. Conclusion

Progressive sampling allows for managing large datasets while smart sampling algorithms are restricted to moderately-sized ones due to memory or time constraints. The proposed framework is independent of the sampling algorithm and of the clustering algorithm. This is an important characteristic as many studies are limited to the classical *k-means*. In this work, the sampling algorithm is *ProTraS* and tests were carried out with a hierarchical clustering algorithm and, more extensively, with the famous *k-means*.

The progressive sampling framework is easy to use. It requires two parameters. The first one is the stratum size. It is defined in a clustering context to yield statistically representative strata: the larger the better the representativeness. Experiments showed that small sizes also yield good results with structured data. The second parameter defines the stopping criterion: it is the quality representation of the already selected items for a new stratum. This is an important feature of the framework: the new items are not randomly added to the sample as is usually the case. They are chosen because they improve the quality of representation. A careful study showed that the behavior of the algorithm is stable for a wide range of values of this threshold.

The tests conducted on synthetic and real world datasets proved that the progressive sampling framework yielded similar results to the sampling algorithm applied to the whole set. This means that the sampling scheme does not decrease the performances despite the dataset splitting. The framework can be used with small datasets but, in this case, it is slower than the sampling algorithm applied to the whole set. With large datasets the advantage of using the framework becomes obvious as the runtime is a function of the stratum size instead of the whole set size. The progressive framework proved highly efficient with large datasets and/or clustering algorithms with a quadratic, or higher, time complexity. The framework is smarter than other classical progressive and stratification strategies as it includes some intelligent mechanisms that select only the necessary representatives and at low computational cost. The neighbor search was optimized but time improvement is still possible thanks to approximate search techniques, e.g. using a *k-d tree* based algorithm [68]. The current framework addresses only one aspect of big data issues: the number of records. Dealing with higher dimension spaces would require additional effort to cope with the curse of dimensionality [**?**]: feature selection, input space transform or the use of non euclidean distances could be considered.

An interesting perspective would be to adapt the framework to deal with data streams. The main issue to be addressed is the stratum size setting. In the current version it is defined according to the whole dataset size, as shown in Theorem 2, but the size concept is meaningless for data streams.

Appendix A. Stratum size impact according to data size and dimension

This experiment studied the impact of the stratum size on the result for various configurations, number of items and input dimensions. Four data sizes were considered, $ds1$ to $ds4$: 200000, 400000, 1000000, 5000000 items and seven input space dimensions, $dd1$ to $dd7$: 2, 5, 10, 15, 20, 50, 80. The two heaviest configurations, 1000000 and 5000000 with 80 dimensions, were not run as the data could not be loaded in the computer memory. In the following tables, *gmean* and *gsd* stand for the global mean and standard deviation computed for all the data sizes and input space dimensions, $dsim$ is the mean for all the space dimensions of all the datasets with the same size, with $i = \{1, 2, 3, 4\}$, and $ddjm$ the mean for all the configurations with the same input space dimension, $1 \leq j \leq 7$, computed for all the dataset sizes.

The accuracy does not depend on the dataset size nor on the input space dimension. The variations are only due to the *k-means* random initialization.

Table A.9: Accuracy results

	gmean	gsd	ds1m	ds2m	ds3m	ds4m	dd1m	dd2m	dd3m	dd4m	dd5m	dd6m	dd7m
258	0.114	0.153	0.107	0.144	0.115	0.112	0.095	0.099	0.098	0.103	0.127	0.131	0.132
504	0.116	0.146	0.130	0.135	0.109	0.091	0.0901	0.102	0.103	0.096	0.099	0.123	0.172
1531	0.087	0.139	0.099	0.077	0.084	0.086	0.099	0.098	0.102	0.086	0.098	0.091	0.087
2706	0.084	0.114	0.115	0.064	0.079	0.056	0.074	0.081	0.079	0.082	0.084	0.085	0.089

Table A.10: Ratio of the sample size to the whole size results

	gmean	gsd	ds1m	ds2m	ds3m	ds4m	dd1m	dd2m	dd3m	dd4m	dd5m	dd6m	dd7m
258	0.003	0.002	0.006	0.003	0.001	0.000	0.003	0.003	0.003	0.002	0.003	0.002	0.002
504	0.005	0.004	0.011	0.006	0.002	0.001	0.005	0.007	0.006	0.005	0.005	0.004	0.004
1531	0.015	0.012	0.032	0.016	0.007	0.002	0.014	0.02	0.018	0.016	0.018	0.013	0.011
2706	0.021	0.016	0.044	0.022	0.009	0.002	0.021	0.021	0.021	0.021	0.024	0.021	0.019

The ratio of the sample size to the whole size results clearly increases with the stratum size, it decreases with the dataset size and does not depend on the space dimension. For a given stratum size, the number of samples does not depend on the dataset size, it is related to the data structure. For a stratum size of 258, it tends to zero when the dataset has $5M$ of items.

Table A.11: Running time results

	gmean	gsd	ds1m	ds2m	ds3m	ds4m	dd1m	dd2m	dd3m	dd4m	dd5m	dd6m	dd7m
258	0.022	0.023	0.044	0.023	0.011	0.005	0.029	0.035	0.027	0.019	0.022	0.011	0.008
504	0.082	0.084	0.154	0.092	0.046	0.018	0.098	0.142	0.103	0.079	0.082	0.037	0.029
1531	0.781	0.772	1.467	0.862	0.476	0.132	0.790	1.225	1.002	0.870	0.890	0.352	0.259
2706	1.381	1.338	2.715	1.638	0.680	0.134	2.131	1.144	1.165	1.417	1.633	1.185	1.058

The time ratio increases with the stratum size while it decreases with the number of items in the dataset and, to a lesser extent, with the input space dimension.

References

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

[2] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651–666.

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

[4] F. Ros, S. Guillaume, Munec: A mutual neighbor-based clustering algorithm, Information Sciences 486 (2019) 148–170. doi:10.1016/j.ins.2019.02.051.

[5] F. Ros, S. Guillaume, M. El Hajji, R. Riad, Kdmutual: A novel clustering algorithm combining mutual neighboring and hierarchical approaches using a new selection criterion, Knowledge-Based Systems (2020) 106220.

[6] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

[7] D. Qiu, Wu, A survey of machine learning for big data processing, EURASIP J. Adv. Signal Process 67 (2016) 3688–3702.

[8] R. J. Hathaway, J. C. Bezdek, Extending fuzzy and probabilistic clustering to very large data sets, Computational Statistics & Data Analysis 51 (1) (2006) 215–234.

[9] L. Jing, K. Tian, J. Z. Huang, Stratified feature sampling method for ensemble clustering of high dimensional data, Pattern Recognition 48 (11) (2015) 3688–3702.

[10] Y. Wang, S.-T. Xia, A novel feature subspace selection method in random forests for high dimensional data, in: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 4383–4389.

[11] T. Zhang, R. Ramakrishnan, M. Livny, Birch: A new data clustering algorithm and its applications, Data Mining and Knowledge Discovery 1 (2) (1997) 141–182.

[12] J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Springer Science & Business Media, 2013.

[13] S. Chakraborty, N. Nagwani, Analysis and study of incremental k-means clustering algorithm, in: International Conference on High Performance Architecture and Grid Computing, Springer, 2011, pp. 338–341.

[14] S. Nassar, J. Sander, C. Cheng, Incremental and effective data summarization for dynamic hierarchical clustering, in: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004, pp. 467–478.

[15] D. H. Widyantoro, T. R. Ioerger, J. Yen, An incremental approach to building a cluster hierarchy, in: 2002 IEEE International Conference on Data Mining, 2002. Proceedings., IEEE, 2002, pp. 705–708.

[16] Y.-m. Cheung, Y. Zhang, Fast and accurate hierarchical clustering based on growing multilayer topology training, IEEE transactions on neural networks and learning systems 30 (3) (2018) 876–890.

[17] D. Cheng, R. Kannan, S. Vempala, G. Wang, A divide-and-merge methodology for clustering, ACM Transactions on Database Systems (TODS) 31 (4) (2006) 1499–1525.

[18] Z. Liang, P. Chen, Delta-density based clustering with a divide-and-conquer strategy: 3dc clustering, Pattern Recognition Letters 73 (2016) 52–59.

[19] J. A. R. Rojas, M. B. Kery, S. Rosenthal, A. Dey, Sampling techniques to improve big data exploration, in: 2017 IEEE 7th symposium on large data analysis and visualization (LDAV), IEEE, 2017, pp. 26–35.

[20] Y. Zhang, Y.-m. Cheung, Y. Liu, Quality preserved data summarization for fast hierarchical clustering, in: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 4139–4146.

[21] Y. Tillé, Sampling algorithms, Springer, 2006.

[22] O. F. Bachem, Sampling for large-scale clustering, Ph.D. thesis, ETH Zurich (2018).

[23] F. Ros, S. Guillaume, From supervised instance and feature selection algorithms to dual selection: A review, in: Sampling Techniques for Supervised or Unsupervised Tasks, Springer, 2020, pp. 83–128.

[24] C. R. Palmer, C. Faloutsos, Density biased sampling: An improved method for data mining and clustering, in: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 82–92.

[25] H. Liu, H. Motoda, Instance selection and construction for data mining, Vol. 608, Springer Science & Business Media, 2013.

[26] J. C. Bezdek, R. J. Hathaway, J. M. Huband, C. Leckie, R. Kotagiri, Approximate clustering in very large relational data, International journal of intelligent systems 21 (8) (2006) 817–841.

[27] A. Podgurski, C. Yang, Partition testing, stratified sampling, and cluster analysis, ACM SIGSOFT Software Engineering Notes 18 (5) (1993) 169–181.

[28] F. Provost, D. Jensen, T. Oates, Efficient progressive sampling, in: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 23–32.

[29] J. C. Bezdek, R. J. Hathaway, Progressive sampling schemes for approximate clustering in very large data sets, in: 2004 IEEE International Conference on Fuzzy Systems (IEEE Cat. No. 04CH37542), Vol. 1, IEEE, 2004, pp. 15–21.

[30] F. Ros, S. Guillaume, Dides: a fast and effective sampling for clustering algorithm, Knowledge and Information Systems 50 (2) (2017) 543–56. `doi:10.1007/s10115-016-0946-8`.

[31] F. Ros, S. Guillaume, Dendis: A new density-based sampling for clustering algorithm, Expert Systems with Applications 56 (2016) 349–359. `doi:10.1016/j.eswa.2016.03.008`.

[32] F. Ros, S. Guillaume, Protras: A probabilistic traversing sampling algorithm, Expert Systems with Applications 105 (2018) 65–76. `doi:10.1016/j.eswa.2018.03.052`.

[33] H. Aoyama, A study of stratified random sampling, Annals of the Institute of Statistical Mathematics 6 (1) (1954) 1–36.

[34] A. H. Dorfman, R. Valliant, Stratification by size revisited, Journal of official statistics 16 (2) (2000) 139.

[35] N. Diamantidis, D. Karlis, E. A. Giakoumakis, Unsupervised stratification of cross-validation for accuracy estimation, Artificial Intelligence 116 (1-2) (2000) 1–16.

[36] J. R. Cano, F. Herrera, M. Lozano, Stratification for scaling up evolutionary prototype selection, Pattern Recognition Letters 26 (7) (2005) 953–963.

[37] K. Machová, M. Puszta, F. Barčák, P. Bednár, A comparison of the bagging and the boosting methods using the decision trees classifiers, Computer Science and Information Systems 3 (2) (2006) 57–72.

[38] A. de Haro-García, N. García-Pedrajas, A divide-and-conquer recursive approach for scaling up instance selection algorithms, Data Mining and Knowledge Discovery 18 (3) (2009) 392–418.

[39] C. García-Osorio, A. de Haro-García, N. García-Pedrajas, Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts, Artificial Intelligence 174 (5-6) (2010) 410–441.

[40] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, Mrpr: a mapreduce solution for prototype reduction in big data classification, neurocomputing 150 (2015) 331–345.

[41] G. H. John, P. Langley, Static versus dynamic sampling for data mining., in: KDD, Vol. 96, 1996, pp. 367–370.

[42] A. El Rafey, J. Wojtusiak, A hybrid active learning and progressive sampling algorithm, International Journal of Machine Learning and Computing 8 (5).

[43] A. Satyanarayana, Performance modeling of cmos inverters using support vector machines (svm) and adaptive sampling, Microprocessors and Microsystems 46 (2016) 193–201.

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

[45] P. Domingos, G. Hulten, A general method for scaling up machine learning algorithms and its application to clustering, in: ICML, Vol. 1, 2001, pp. 106–113.

[46] A. El Rafey, J. Wojtusiak, Recent advances in scaling-down sampling methods in machine learning, Wiley Interdisciplinary Reviews: Computational Statistics 9 (6) (2017) e1414.

[47] A. Nanopoulos, Y. Theodoridis, Y. Manolopoulos, Indexed-based density biased sampling for clustering applications, Data & Knowledge Engineering 57 (1) (2006) 37–63.

[48] L. Wang, J. C. Bezdek, C. Leckie, R. Kotagiri, Selective sampling for approximate clustering of very large data sets, International Journal of Intelligent Systems 23 (3) (2008) 313–331.

[49] X. Zhao, J. Liang, C. Dang, A stratified sampling based clustering algorithm for large-scale data, Knowledge-Based Systems 163 (2019) 416–428.

[50] G. B. Wetherill, K. D. Glazebrook, Sequential methods in statistics, unknown, 1986.

[51] O. Watanabe, Simple sampling techniques for discovery science, IEICE Transactions on Information and Systems 83 (1) (2000) 19–26.

[52] R. J. Lipton, J. F. Naughton, D. A. Schneider, S. Seshadri, Efficient sampling strategies for relational database operations, Theoretical Computer Science 116 (1) (1993) 195–226.

[53] D. Barbará, P. Chen, Tracking clusters in evolving data sets., in: FLAIRS Conference, 2001, pp. 239–243.

[54] A. Satyanarayana, I. Davidson, A dynamic adaptive sampling algorithm (dasa) for real world applications: Finger print recognition and face recognition, in: International Symposium on Methodologies for Intelligent Systems, Springer, 2005, pp. 631–640.

[55] W. Mulzer, Five proofs of chernoff's bound with applications, arXiv preprint arXiv:1801.03365.

[56] S. Guha, R. Rastogi, K. Shim, Cure: an efficient clustering algorithm for large databases, Information Systems 26 (1) (2001) 35 – 58. `doi:https://doi.org/10.1016/S0306-4379(01)00008-4`.

[57] D. Feldman, Core-sets: Updated survey, in: Sampling Techniques for Supervised or Unsupervised Tasks, Springer, 2020, pp. 23–44.

[58] P. M. Domingos, G. Hulten, A general method for scaling up machine learning algorithms and its application to clustering, in: ICML, 2001, pp. 106–113.

[59] I. Kärkkäinen, P. Fränti, Gradual model generator for single-pass clustering, Pattern Recognition 40 (3) (2007) 784–795.

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

[61] I. Kärkkäinen, P. Fränti, Dynamic local search algorithm for the clustering problem, Tech. Rep. A-2002-6, Department of Computer Science, University of Joensuu, Joensuu, Finland (2002).

[62] P. Fränti, S. Sieranoja, K-means properties on six clustering benchmark datasets, Applied Intelligence 48 (12) (2018) 4743–4759.

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

[64] J. Alvarez-Sanchez, Injecting knowledge into the solution of the two-spiral problem, Neural Computing & Applications 8 (3) (1999) 265–272.

[65] G. Karypis, E.-H. Han, V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.

[66] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework., Journal of Multiple-Valued Logic & Soft Computing 17.

[67] I. Triguero, J. Derrac, F. Herrera, S. García, A study of the scaling up capabilities of stratified prototype generation, in: 2011 Third World Congress on Nature and Biologically Inspired Computing, IEEE, 2011, pp. 297–302.

[68] M. Muja, D. G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (11) (2014) 2227–2240.