Quantization based clustering: an iterative approach

Thomas Laloë

Quantization based clustering: an iterative approach

Thomas Laloë

Abstract

In this paper we propose a simple new algorithm to perform clustering, based on the Alter algorithm proposed in [5] but lowering significantly the algorithmic complexity with respect to the number of clusters. An empirical study states the relevance of our iterative process and a confrontation on simulated multivariate and functional data shows the benefits of our algorithm.

1. Introduction

Clustering consists in partitioning a set of unlabeled objects into homogeneous groups (or clusters), so that the data in each subset share some common trait (see [4] for a thorough introduction to the subject) . Over the years, many methods have been proposed to deal with clustering : density based clustering [10], Hierarchical clustering [11] and partitioning clustering... We focus in this paper on the last one. More precisely we use a method coming from the signal compression theory : the quantization [7].

The proximity notion is crucial in the definition of what is a "good clustering". We propose here to rely on the method proposed in [5] which is based on a L_1 (or Manhattan) distance. The algorithm (called Alter) proposed to perform the clustering is proved to be consistent but suffers from a high complexity. A first alternative has been proposed in [6] to lower the complexity, adapting the X-means approach proposed in [8].

The purpose of this paper is to propose a new alternative to lower the complexity of the Alter algorithm (with respect to the number of clusters), best preserving its ability to converge to the global optimum.

The paper is organized as follows: the Alter algorithm and its theoretical properties are summarized in Section 2. Then our new algorithm is presented in Section 3. Finally, a comparative study on simulated data is provided in Section 4.

2. Quantization based clustering

Let us summarize the Alter algorithm. All the theoretical results presented in this section come from [5]. The method is based on quantization, which is a commonly used technique in signal compression [2, 7]. Consider $(\mathcal{H}, \|.\|)$ a normed space and let X be a $\mathcal{H}$-valued random variable with distribution μ such as $\mathbb{E}\|X\| < \infty$. Given a set $\mathcal{C}$ of points in $\mathcal{H}^k$, any Borel function $q : \mathcal{H} \to \mathcal{C}$ is called a quantizer. The set $\mathcal{C}$ is called a codebook, and the error made by replacing X by $q(X)$ is measured by the distortion:

$$D(\mu, q) = E\,\|X - q(X)\| = \int_{\mathcal{H}} \|x - q(x)\|\mu(dx).$$

Note that $D(\mu, q) < \infty$ since $\mathbb{E}\|X\| < \infty$. For a given k, the aim is to minimize $D(\mu, .)$ among the set $\mathcal{Q}_k$ of all possible k-quantizers. The optimal distortion is then defined by

$$D_k^*(\mu) = \inf_{q \in \mathcal{Q}_k} D(\mu, q).$$

When it exists, a quantizer q^* satisfying $D(\mu, q^*) = D_k^*(\mu)$ is said to be an optimal quantizer.

As detailed in [5], a quantizer is characterized by its codebook $\mathcal{C} = \{y_i\}_{i=1}^k$ and a partition of $\mathcal{H}$ in cells $S_i = \{x \in \mathcal{H} : q(x) = y_i\},\ i = 1, \ldots, k$ via the rule

$$q(x) = y_i \iff x \in S_i.$$

Moreover it is proved in [5] that for a given codebook an optimal partition is a nearest neighbor one. So we can consider only nearest neighbor quantizers, which means that a quantizer q will be characterized by its codebook $\mathcal{C} = \{y_i\}_{i=1}^k$ and the rule

$$q(x) = y_i \iff \forall 1 \leq j \leq k, j \neq i, \|x - y_i\| \leq \|x - y_j\|,$$

with ties arbitrary broken. Thus, a quantizer can be defined by its codebook only. Moreover the aim is to minimize the distortion among all possible nearest neighbor quantizers.

However, in practice, the distribution μ of the observations is unknown, and we only have at hand n independent observations $X_1, \ldots, X_n$ with the same distribution than X. The goal is then to minimize the empirical distortion:

$$\frac{1}{n} \sum_{i=1}^{n} \|X_i - q(X_i)\|.$$

Then, clustering is done by regrouping the observations that have the same image by q. More precisely, we define a cluster $\mathcal{C}$ by $\mathcal{C} = \{X_i : q(X_i) = \hat{x}_{\mathcal{C}}\}$, $\hat{x}_{\mathcal{C}}$ being the representative of cluster $\mathcal{C}$.

Unfortunately, the minimization of the empirical distortion is not possible in practice and that is why an alternative is proposed: the Alter algorithm. The idea is to select an optimal codebook among the data set. More precisely the outline of the algorithm is:

1. List all possible code books , i.e., all possible k-tuples of data;

2. Compute the empirical distortion associated to the first codebook. Each observation X_i is associated with its closed center;

3. For each successive codebook, compute the associated empirical distortion. Each time a codebook has an associated empirical distortion smaller than the previous smallest one, store the codebook;

4. Return the codebook that has the smallest distortion.

It is proved that the convergence rate is of the same order than the theoretical method described above (minimization of the empirical distortion over all possible quantizers). Moreover, this algorithm does not depend on initial conditions (unlike the K-Means or K-Medians algorithm) and it converges to the optimal distortion. Unfortunately its complexity is $O(n^{k+1})$ and it is impossible to use it for high values of n or k. Worse, even for small n, it is not possible to consider large number of clusters. That is why we wanted to propose the Iterative Alter algorithm.

3. Iterative Alter

Let us now present our alternative to lower the complexity of the algorithm. For the sake of simplicity, let us take the case where the data belong to $\mathbb{R}$, and where we try to cluster them into two groups. If we perform Alter, we have to compute the distortion with centers given by any pair of data. Figure 1 show the behavior of the distortion with respect to the values of the two centers and Figure 2 show the behavior of the distortion with respect to one center while the other is fixed.

Looking at this figures it seems possible to get the best pair of centers by successively optimizing each center (fixing the other). Thus the process could be :

Step 1 : Select (randomly) two data to be the initial pair of center;

Step 2 : Fix the first center and optimize the distortion on the second;

Step 3 : Fix the second center (with the value obtained at Step 1) and optimize the distortion on the first center;

Step 4 : Repeat Steps 2 and 3 until the centers no longer change.

The generalization to k clusters is then trivial :

Step 1 : Select (randomly) k data to be the first center and optimize the distortion on the second center;

Steps 2 : For i from 1 to k, fix all centers expect the ith one and optimize the distortion on this one;

Step 3 : Iterate until the centers no longer change.

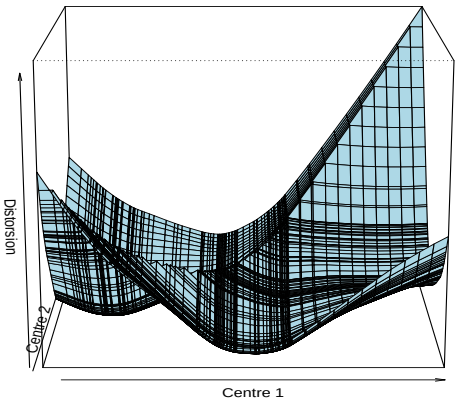

Figure 1: Behavior of the distortion with respect to two centers.

Of course we have to make the assumption that the data are continuously distributed. Otherwise specific counter example can be constructed. Before performing a comparative study, we empirically justify this procedure.

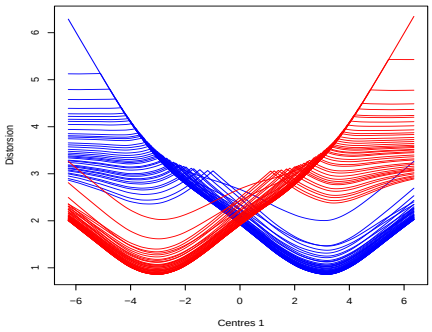

Figure 2: Behavior of the distortion with respect to one center. Each curve correspond to one value of the first center.

3.1. Empirical justification

In this section we propose to empirically justify our algorithm, showing that we indeed get the global minimum of the distortion. We begin by a multivariate case, before looking at a functional case.

3.1.1. Multivariate case

We simulate data sets (of size $n = 50, 100$ and 500) of six clusters in $\mathbb{R}^2$ (see Figure 3) and perform our algorithm with $k \in \{2, 4, 6\}$ clusters. The cluster are centered around $(-7, -5)$, $(-7, 0)$, $(-7, 5)$, $(7, -5)$, $(7, 0)$ and $(7, 5)$, and in each cluster the data are normally distributed around the center (with a standard deviation equal to 2). Moreover we add a "noise" cluster containing 10% of the data, centered on 0 and with standard deviation equal to 8.

For each configuration (i.e. couple (n, k), Figure 4 present the evolution of the averaged (over $M = 50$ repetition of the simulation process) distortion according to the number of cycles performed (a cycle is an iteration of Steps 2 and 3). In such a simple scenario we are also able to compute the real optimal distortion.

We see in Figure 4 that at the end of the first cycle we almost get the optimal distortion and that the

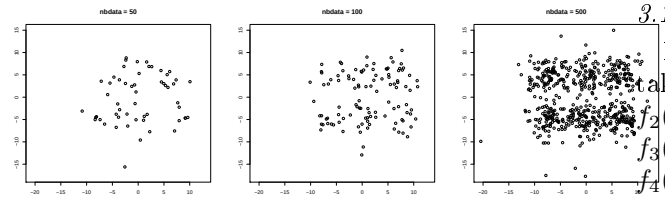

Figure 3: Example of multivariate data set for $n = 50, 100, 500$.

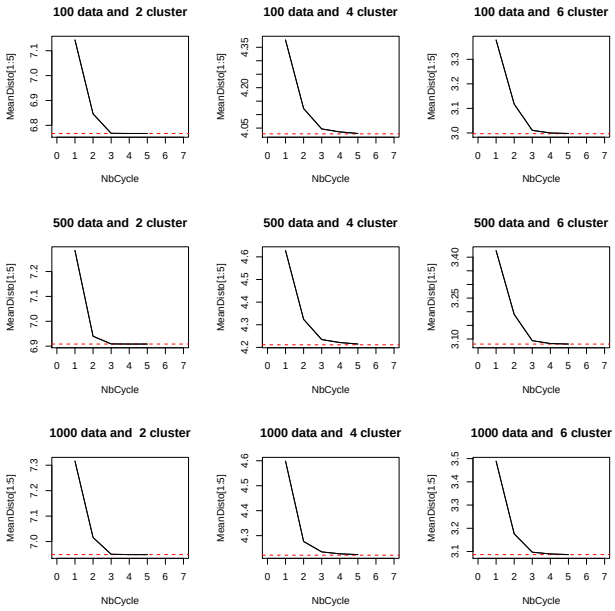

Figure 4: **Multivariate case.**Evolution of the distortion with respect to the number of Cycles. Dashed lines represent the optimal distortion.

number of cycles required to get the optimal distortion is really small. Since the complexity of our algorithm is of the order of (*number of cycles*) $\times$ (*number of data*), this small number of cycles ensure a reasonable complexity. Note that for each cycle, a parallelization of the optimization of each center is possible.

3.1.2. Functional case

Now we want to consider functional data. We take functions $f_1(x) = x^{0.1} + \cos(10x + \pi/2 - 10)/5$, $f_2(x) = x + \cos(10x + \pi/2 - 10)/5$, $f_3(x) = x^2 + \cos(10x + \pi/2 - 10)/5$ and $f_4(x) = x^{10} + \cos(10x + \pi/2 - 10)/5$ defined on $[0, 1]$ discretized 20 times. The term $\cos(10x + \pi/2 - 10)/5$ is added to disturb functions $x^{0.1}$, x, x^2 and x^{10}. Each data in $\mathbb{R}^{20}$ is noised with a vector composed by twenty Gaussian law $N(0, \sigma)$ where the value of σ is selected for each data using $\sigma \sim N(0.1, 0.02)$. The idea is to simulate two clusters (of sizes randomly selected between 15 and 30) around f_2 and f_3, and complicate the task by adding a small number (randomly selected between 1 and 5) of functions distributed around f_1 and f_4. Figure 5 shows examples of some of the functions that we want to classify.

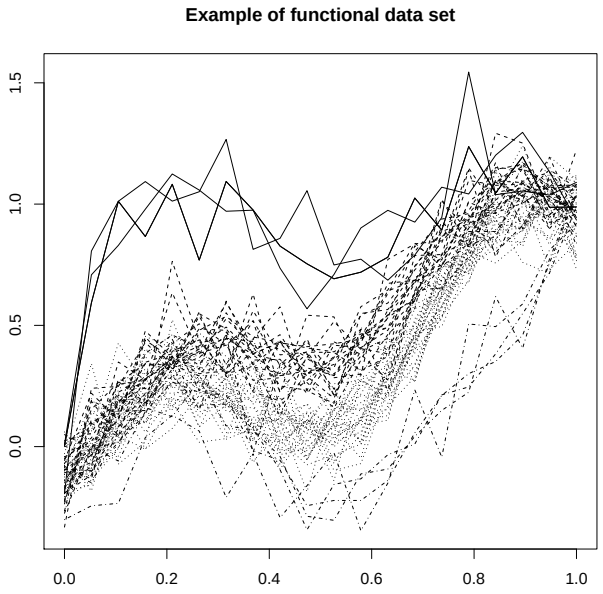

Figure 5: Example of multivariate data set.

We simulate 50 different data set to calculate

averaged distortions, and Figure 6 play the role of Figure 4: it presents the evolution of the averaged distortion according to the number of cycles performed (a cycle is an iteration of Steps 2 and 3). As in the multivariate case we see that the number of cycles required to get the optimal distortion is really small.

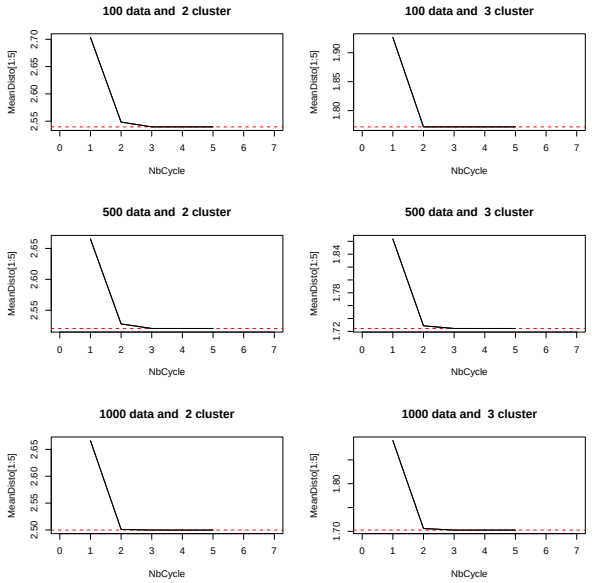

Figure 6: **Functional case.** Evolution of the distortion with respect to the number of Cycles. Dashed lines represent the optimal distortion.

Now that we have empirically stated the relevance of our iterative process, we can perform a comparative study both on simulated data.

4. Comparative study

We perform here an empirical study to show the relevance of our method. We confront our method to various simulated data sets, but also on classical real data sets. In order to evaluate the relevance we consider the Adjusted Rand Index (A.R.I.) [9, 3]. Moreover, we compare our method to a K-Medians algorithm proposed by [1].

4.1. Multivariate case

We perform here tests with a little more complicated data sets than in Section 3.1.1: each data set is composed of k cluster, and each cluster $(C_i,\ i = 1,\dots,k)$ contains n_i data normally distributed around m_i (in $\mathbb{R}^d$) and with standard deviation σ_i. All the parameters are randomly selected :

- k is uniformly selected in $\{2, 3, \dots, 8\}$;
- the n_i are uniformly selected between 5 and 25 ;
- each coordinate of each m_i are uniformly selected between -20 and 20 ;
- the σ_i are uniformly selected between 2 and 5 .

The K-Medians algorithm may strongly depend on the initial conditions. However it is possible to overcome this performing multiple intializations (we will call this R-K-Median, with R the number of initializations). Table 1 summaries the results averaged on 50 simulations (each run is done with a new set of randomly selected parameters), and for different values of the dimension of the data ($d = 2$, $d = 5$ and $d = 10$).

Table 1: Comparative study in the multivariate case.

Algorithm	Iter Alter	1-K-Medians	10-K-Medians	20-K-Medians
ARI dim 2	0.7	0.7	0.7	0.7
ARI dim 5	0.96	0.96	0.96	0.96
ARI dim 10	1	0.99	0.99	0.99

It seems here that our algorithm and the K-Medians have similar performance. However if we look more precisely things are a little different. Figures 7, 8, 9 show the distribution (over 300 repetitions of the previous process) of the averaged ARI for dimension $d = 2, 5$, and 10. With the increasing of the dimension, we clearly see the benefits of our algorithm. This is not surprising since the underlying method of our algorithm (see [5]) is thought for functional data. This benefit is even more significant if we look at the distribution of the minimal ARI (Figures 10, 11 and 12 below).

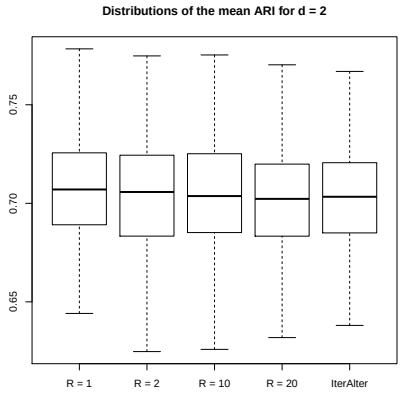

Figure 7: **Multivariate case** Distributions of the minimal ARI obtained with R-K-Medians and Alter ($d = 2$).

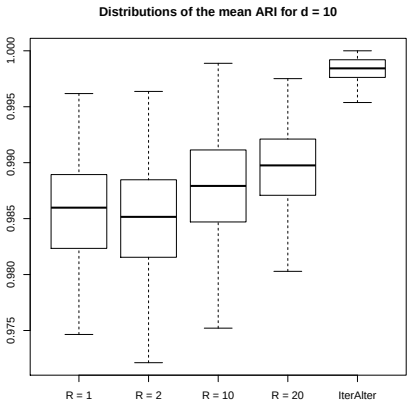

Figure 9: **Multivariate case** Distributions of the minimal ARI obtained with R-K-Medians and Alter ($d = 10$).

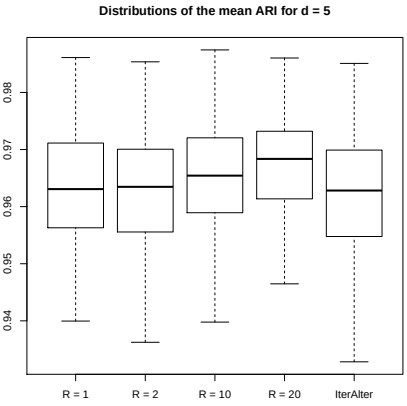

Figure 8: **Multivariate case** Distributions of the minimal ARI obtained with R-K-Medians and Alter ($d = 5$).

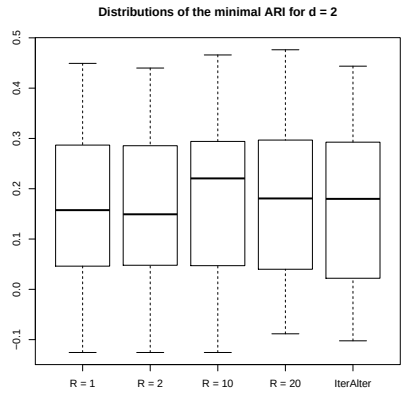

Figure 10: **Multivariate case ($d = 2$)** Distributions of the minimal ARI obtained with R-K-Medians and Alter.

In the next section we consider functional data and we will see that this benefit is even more pronounced.

4.2. Functional case

Now we want to consider functional data. We take the same configuration as in Section 3.1.2: We take functions $f_1(x) = x^{0.1} + \cos(10x + \pi/2 - 10)/5$, $f_2(x) = x + \cos(10x + \pi/2 - 10)/5$, $f_3(x) = x^2 + \cos(10x + \pi/2 - 10)/5$ and $f_4(x) = x^{10} + \cos(10x + \pi/2 - 10)/5$ defined on $[0, 1]$ discretized 20 times. Each data in $\mathbb{R}^{20}$ is noised with a vector composed by twenty Gaussian law $N(0, \sigma)$ where the value of σ is selected for each data using $\sigma \sim N(0.1, 0.02)$. We simulate two clusters (of sizes randomly selected between 15 and 30) around f_2 and f_3, and complicate the task by adding a small number (randomly selected between 1 and 5) of functions distributed

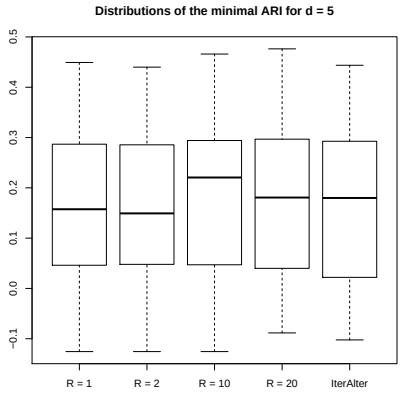

Figure 11: **Multivariate case** ($d = 5$) Distributions of the minimal ARI obtained with R-K-Medians and Alter.

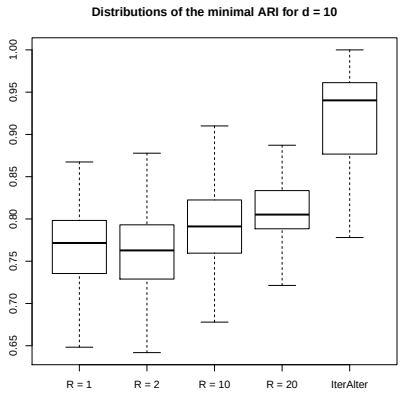

Figure 12: **Multivariate case** ($d = 10$) Distributions of the minimal ARI obtained with R-K-Medians and Alter.

around f_1 and f_4 (as a reminder one can look at Figure 5). The results presented in Table 4.2 are averaged over 50 simulated data sets.

As in the previous section we present in Figure 13 the boxplot of the minimum ARI obtained at each of the M repetitions. Once again, our method appears to be more reliable than the R-K-Medians, due to the consistency properties of the Alter algorithm.

Table 2: Comparative study in the functional case.

Algorithm	Iter Alter	1-K-Medians	10-K-Medians	20-K-Medians
ARI	0.99	0.3	0.35	0.37

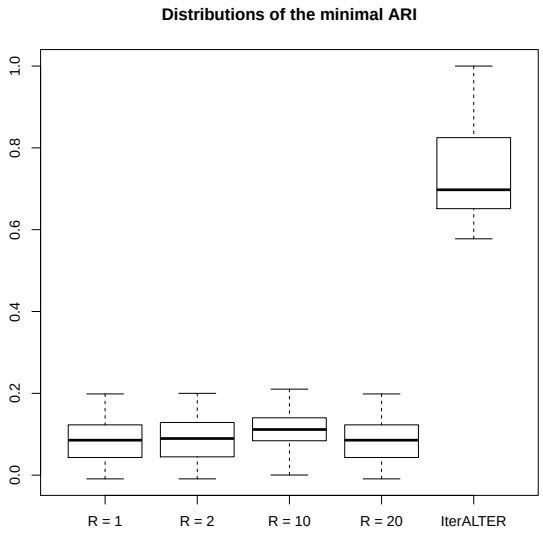

Figure 13: **Functional case** Distributions of the minimal ARI obtained with R-K-Medians and Alter.

Conclusion

We have presented a simple new algorithm to perform clustering, based on the Alter algorithm proposed in [5]. With this algorithm we lower significantly the algorithmic complexity. An empirical study stated the relevance of our iterative process and a confrontation on simulated data showed the benefits of our algorithm. However, theoretical guarantees remains to be proved and we did not address the problem of the selection of the numbers of cluster. This should be the subject of a future work.

References

[1] Cardot, H., Cénac, P., Monnez, J.M., 2012. A fast and recursive algorithm for clustering large datasets with k-medians. Computational Statistics & Data Analysis 56, 1434 – 1449. URL: https://doi.org/10.1016/j.csda.2011.11.019.

[2] Graf, S., Luschgy, H., 2000. Foundations of Quantization for Probability Distributions. volume 1730 of *Lecture Notes in Mathematics*. Springer-Verlag. URL: https://doi.org/10.1007/bfb0103945.

[3] Hubert, L., Arabie, P., 1985. Comparing partitions. Journal of Classification 2, 193–218. URL: https://doi.org/10.1007/bf01908075.

[4] Kaufman, L., Rousseeuw, P., 1990. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons. URL: https://doi.org/10.1002/9780470316801.

[5] Laloë, T., 2010. L_1 quantization and clustering in banach spaces. Mathematical Methods of Statistics 19, 136–150. URL: https://hal.archives-ouvertes.fr/hal-01292694.

[6] Laloë, T., Servien, R., 2013. The X-Alter algorithm : a parameter-free method to perform unsupervised clustering. Journal of Modern Applied Statistical Methods 12, 90–102. URL: https://hal.archives-ouvertes.fr/hal-00674407.

[7] Linder, T., 2002. Learning-theoretic methods in vector quantization, in: Principles of Nonparametric Learning (Udine, 2001). Springer, Vienna. volume 434 of *CISM Courses and Lectures*, pp. 163–210. URL: https://doi.org/10.1007/978-3-7091-2568-7_4.

[8] Pelleg, D., Moore, A., 2000. X-means: Extending k-means with efficient estimation of the number of clusters, in: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 727–734.

[9] Rand, W., 1971. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66, 846–850. URL: https://doi.org/10.2307/2284239.

[10] Sander, J., Ester, M., Kriegel, H.P., Xu, X., 1998. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data Mining and Knowledge Discovery 2, 169–194. URL: https://doi.org/10.1023/A:1009745219419, doi:10.1023/A:1009745219419.

[11] Zhao, Y., Karypis, G., Fayyad, U., 2005. Hierarchical clustering algorithms for document datasets. Data Mining and Knowledge Discovery 10, 141–168. URL: https://doi.org/10.1007/s10618-005-0361-3, doi:10.1007/s10618-005-0361-3.