Knowledge-Based Systems 117 (2017) 56–69

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

An efficient approximation to the *K*-means clustering for massive data

Marco Capó[a,*], Aritz Pérez[a], Jose A. Lozano[a,b]

[a] *Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain*
[b] *Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country UPV/EHU, 20018 Donostia-San Sebastián, Spain*

ARTICLE INFO

Article history:
Received 29 March 2016
Revised 13 June 2016
Accepted 27 June 2016
Available online 28 June 2016

Keywords:
K-means
Clustering
K-means++
Minibatch *K*-means

ABSTRACT

Due to the progressive growth of the amount of data available in a wide variety of scientific fields, it has become more difficult to manipulate and analyze such information. In spite of its dependency on the initial settings and the large number of distance computations that it can require to converge, the *K*-means algorithm remains as one of the most popular clustering methods for massive datasets. In this work, we propose an efficient approximation to the *K*-means problem intended for massive data. Our approach recursively partitions the entire dataset into a small number of subsets, each of which is characterized by its representative (center of mass) and weight (cardinality), afterwards a weighted version of the *K*-means algorithm is applied over such local representation, which can drastically reduce the number of distances computed. In addition to some theoretical properties, experimental results indicate that our method outperforms well-known approaches, such as the *K-means++* and the *minibatch K-means*, in terms of the relation between number of distance computations and the quality of the approximation.

1. Introduction

The exponential increase of the data volumes that scientists, from different backgrounds, face on a daily basis implies the development of simple yet scalable tools that eases the analysis and characterization of such information [12]. One of the most relevant analysis is data clustering. This process consists of grouping a given dataset into a predetermined amount of disjoint sets, called clusters. This is done in such a way that intra-cluster similarity is high and the inter-cluster similarity is low. Furthermore, clustering is a basic task of many areas, such as artificial intelligence, machine learning and pattern recognition [14,16].

Even when there exists a wide variety of clustering methods, the *K*-means algorithm remains as one of the most popular [17]. In fact, it has been identified as one of the top 10 algorithms in data mining [26].

1.1. K-means

Given a set of n data points (instances) $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ in $\mathbb{R}^d$ and an integer K, the ***K*-means problem** is to determine a set of K centroids $C = \{\mathbf{c}_1, \dots, \mathbf{c}_K\}$ in $\mathbb{R}^d$ so as to minimize the following **error function**:

$$E(C) = \sum_{\mathbf{x} \in D} \min_{k=1,\dots,K} \|\mathbf{x} - \mathbf{c}_k\|^2 \tag{1}$$

This is a combinatorial optimization problem since it is equivalent to finding the partition of the n instances in K groups whose associated set of centers of mass minimizes Eq. (1). In this case, the number of possible partitions is a Stirling number of the second kind [19].

Since finding the globally optimal partition is known to be **NP-hard**, even for instances in the plane [1], and exhaustive search methods are not useful in practice, iterative refinement based algorithms are commonly used to approximate the solution of the *K*-means and similar problems [18,20,21]. These algorithms iteratively relocate the data points between clusters until a locally optimal partition is attained. Among these methods the most popular is the ***K*-means algorithm** [17,20]

The *K*-means algorithm has two stages: **Initialization**, in which we set the starting set of centroids and, an iterative stage, called **Lloyd's algorithm** [20]. Lloyd's algorithm consists of two steps: A first step in which each instance is assigned to its closest centroid (**assignment step**), then the set of centroids is updated (**update step**). Finally, a stopping criterion is verified. The most common criterion implies the computation of the error function (Eq. (1)): if the error does not decrease significantly, with respect to the previous iteration, the algorithm stops. The time required for the assignment step is $O(nKd)$, while the update step of the set of centroids

* Corresponding author.
E-mail addresses: mcapo@bcamath.org (M. Capó), aperez@bcamath.org (A. Pérez), ja.lozano@ehu.eus (J.A. Lozano).

http://dx.doi.org/10.1016/j.knosys.2016.06.031
0950-7051/

requires $O(nd)$ computations and the stopping criterion based on the computation of the error function has a $O(nd)$ time complexity. Hence, the assignment step is the most computationally intensive due to the distance computations. For this reason, the main objective of our proposal consists of defining a variant of the K-means algorithm that controls the trade-off between the number of distances computed and the quality of the obtained solution.

Conveniently, every step of the K-means algorithm can be easily parallelized [27], which is a major key to meet the scalability of the problem [26].

1.1.1. K-means initialization

It is widely reported in the literature that the performance of the Lloyd's algorithm highly depends upon the initialization stage [[22]]: One might need several re-initializations before achieving a solution of acceptable quality. This is especially adverse when dealing with massive data applications since the number of distance computations is proportional to the number of instances, n. In addition, a poor initialization could lead to an exponential running time in the worst case scenario [25]. All these features are major downsides and show the importance of defining an appropriate initialization strategy.

There exist several approaches to initialize the K-means algorithm. One of the earliest, and most popular initialization strategies, was proposed by Forgy in 1965 [11]. It consists of defining the initial centroids set as K randomly selected instances from the dataset. The intuition behind this approach is that, by choosing the prototypes randomly, we are more likely to choose a point near an optimal cluster center, since such points tend to be where the highest density points are located [23]. The main disadvantage of this approach is that there is no guarantee that two, or more, of the selected seeds will not be near the center of the same cluster [23].

Moreover, there also exist well known initialization procedures that are based on simple probabilistic seeding techniques. In particular, the *K-means++* method, proposed by Arthur and Vassilvitskii in [2], consists of randomly selecting only the first centroid from the dataset. Each subsequent initial centroid is chosen with a probability proportional to the distance with respect to the previously selected set of centroids. The key idea of this cluster initialization technique is to preserve the diversity of seeds while being robust to outliers. The K-means++ algorithm leads to an $O(\log K)$ approximation of the optimal error after the initialization [2]. The drawback of this approach refers to its sequential nature, which hinders its parallelization, as well as to the fact that it requires K scans of the entire dataset, therefore it has a complexity of $O(nKd)$.

1.2. An alternative to Lloyd's algorithm

Apart from Lloyd's algorithm there exist several low computational cost alternatives that, by using statistical techniques, attempt to approximate a suboptimal solution to the K-means problem without processing the information of every instance in the dataset [5,7,24]. Among these algorithms we have the *minibatch K-means* proposed by Sculley in [24]. This algorithm seeks to reduce the computational cost by not using all the dataset at each iteration but small random batches of examples of a fixed size until convergence. This strategy reduces the number of distance computations per iteration at the cost of lower cluster quality. Empirical results, in a range of large web based applications, corroborate that a substantial saving of computational time can be obtained at the expense of some loss of cluster quality [24].

1.3. Contribution

In this work, we propose an approximation algorithm for the K-means problem based on a recursive data partitioning process that reduces the number of distance calculations and data scans, while generating competitive approximations. The algorithm considers a sequence of partitions of the dataset, in such a way that the partition at iteration i is thinner than the partition at iteration $i-1$. At each step, the mass center of each subset of the partition (set of representatives) is calculated and a weighted Lloyd's algorithm is applied using the current set of representatives as the dataset. Among other benefits, this approach reduces the number of distance computations over the whole dataset, which is the most computationally demanding stage of the K-means algorithm.

The rest of this article is organized as follows: In Section 2, we describe the idea behind our algorithm and introduce notation that we use, in Section 3, to state some theoretical guarantees of our approach. The proofs of such statements can be found in Appendix A. In Section 4, we present a set of experiments in which we analyze the effect of different factors, such as the size of the dataset and the dimension of the instances over the performance of our algorithm. Additionally we compare these results with the ones obtained by the K-means++ and the minibatch K-means methods. Finally, in Section 5, we define the next steps and possible improvements to our current work.

2. Recursive partition based K-means

We propose a novel, iterative approximation strategy for the K-means problem that is based on a sequence of recursive partitions of the dataset, being each partition thinner than the previous one. We call this approach **recursive partition based K-means** (**RPKM**). The idea behind the algorithm is to approximate the K-means problem for the full dataset by recursively applying a weighted version of the K-means algorithm over a growing, yet small, number of subsets of the dataset.

In the first step of the RPKM, the dataset is partitioned into a number of subsets each of which is characterized by a representative (center of mass) and its corresponding weight (cardinality). Finally, a weighted version of Lloyd's algorithm (see Section 2.2 for further details) is applied over the set of representatives. From one iteration to the next, a more refined partition is constructed and the process is repeated using the optimal set of centroids obtained at the previous iteration as initialization. This iterative procedure is repeated until a certain stopping criterion is met.

In the next section, we describe in detail the recursive partition process and characterize some of its properties. The notation introduced in this section will be used later for a formal description of the RPKM algorithm.

2.1. Recursive partitions

As previously mentioned, the recursive partition process is the first stage of the RPKM algorithm. This step consists of generating a thinner partition than the previous one at each iteration.

From now on we will use the following definition of partition of a dataset.

Definition 1 (Partition of a dataset D)**.** $\mathcal{P} = \{S_1, \ldots, S_t\}$ is a partition of the dataset D if $\bigcup_{i=1}^{t} S_i = D$ and if the subsets of $\mathcal{P}$ are (pairwise) disjoint and nonempty. Moreover, given a subset $S \in \mathcal{P}$, we define its weight as its cardinality, $|S|$, and its representative as its center of mass, $\overline{S} = \frac{\sum_{\mathbf{x}\in S} \mathbf{x}}{|S|}$.

The partition of the dataset allows us to describe it with a reduced number of representatives, which ultimately implies the reduction of the number of distance computations with respect to the Lloyd's algorithm for the full dataset. In the RP*K*M algorithm, we will use the set of representatives and weights, rather than the partition itself.

Definition 2 (Partition thinner than $\mathcal{P}$)**.** Given two partitions of the dataset D, $\mathcal{P}$ and $\mathcal{P}'$, we say that $\mathcal{P}'$ is a partition thinner than $\mathcal{P}$ ($\mathcal{P} \succ \mathcal{P}'$) if, for all $S \in \mathcal{P}$, $S = \bigcup_{R \in \mathcal{P}'[S]} R$, where $\mathcal{P}'[S] = \{R \in \mathcal{P}' : R \subseteq S\}$.

In other words, $\mathcal{P}'$ is a partition thinner than $\mathcal{P}$ if every subset of $\mathcal{P}$ can be written as the union of subsets of $\mathcal{P}'$.

The partition process generates a **sequence of thinner partitions** $\mathcal{P}_1, \ldots, \mathcal{P}_m$, such that $\mathcal{P}_{i-1} \succ \mathcal{P}_i$ for all $i \in \{2, \ldots, m\}$. Evidently, the number of representatives tends to increase as we generate a thinner partition. In the extreme case $\mathcal{P}_m = \{\{\mathbf{x}\} : \mathbf{x} \in D\}$, however, in practice, in order to reduce the computational complexity of the RP*K*M, we control the number of representatives so that $|\mathcal{P}_m| \ll n$.

Note that the weight and the representative of $S \in \mathcal{P}_i$ can be easily computed from $\mathcal{P}_{i+1}[S]$ as follows: $|S| = \sum_{R \in \mathcal{P}_{i+1}[S]} |R|$, $\overline{S} = \frac{\sum_{R \in \mathcal{P}_{i+1}[S]} |R| \cdot \overline{R}}{|S|}$. As we noted before, we are interested in the computation of the set of representatives and weights, thus, we will use $\mathcal{P}_m$ to generate the set of representatives and weights of the entire sequence of thinner partitions backward, from $\mathcal{P}_{m-1}$ to $\mathcal{P}_1$. Hence, the construction of $\mathcal{P}_i$ has a $O(|\mathcal{P}_{i+1}|d)$ time cost for $i < m$. Moreover, if the assignment criterion of each instance of D into its corresponding subset in $\mathcal{P}_m$ is of order $O(d)$, as it is in the case of the grid based RP*K*M (see Section 2.4), then the construction of $\mathcal{P}_m$ is $O(nd)$ and, therefore, the cost of the entire partition process is $O(d(n + \sum_{i=2}^{m} |\mathcal{P}_i|))$.

2.2. Weighted K-means problem

In this section, we introduce a generalization of the *K*-means problem defined over a set of weighted points, e.g., the set of representatives and their respective weights associated to a partition. As a first step, we define a clustering for a partition.

Definition 3 (Clustering for a partition $\mathcal{P}$)**.** We say that a partition of the dataset D, $\mathcal{G}$, is a clustering of the dataset for a partition $\mathcal{P}$, when $|\mathcal{G}| = K$ and $\mathcal{G} \succ \mathcal{P}$.

In other words, a cluster for a partition is a set of K subsets of points of D, such that all the points of any $S \in \mathcal{P}$ are assigned to the same cluster.

We call $\mathcal{G} = \{G_1, ..., G_K\}$ a **clustering induced by a set of centroids** $C = \{\mathbf{c}_1, ..., \mathbf{c}_k\}$, when $G_k = \bigcup_{S \in M_k} S$ for $k = 1, ..., K$, where $M_k = \{S \in \mathcal{P} : k = \arg\min_{j=1,...,K} \|\overline{S} - \mathbf{c}_j\|^2\}$. In other words, a clustering induced by a set of centroids is a partition of the dataset in which all the data points that have the same closest centroid from C are grouped in the same cluster. We denote that the clustering $\mathcal{G}$ is induced by a set of centroid C by $\mathcal{G} \leftarrow C$. Similarly, we call $C = \{\mathbf{c}_1, ..., \mathbf{c}_k\}$ a **centroids set induced by a clustering** $\mathcal{G} = \{G_1, ..., G_K\}$, when $c_i = \overline{G}_i$ for $i = 1, ..., K$. In other words, the set of centroids induced by a clustering $\mathcal{G}$ is the set of centers of mass associated to each cluster in $\mathcal{G}$. We denote that the set of centroids C is induced by a clustering $\mathcal{G}$ by $C \leftarrow \mathcal{G}$.

Given a partition of the dataset D, $\mathcal{P}$, the **weighted *K*-means problem** seeks to determine a set of k centroids $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$ in $\mathbb{R}^d$, so as to minimize the **centroid error** associated to a partition $\mathcal{P}$, which is defined as follows:

$$E_{\mathcal{P}}(C) = \sum_{S \in \mathcal{P}} |S| \min_{k=1,...,K} \|\overline{S} - \mathbf{c}_k\|^2 = \sum_{k=1}^{K} \sum_{S \in \mathcal{P}: S \subseteq G_k} |S| \cdot ||\overline{S} - \mathbf{c}_k||^2 \quad (2)$$

where the clustering $\mathcal{G}$ is induced by the set of centroids C. This error measures the weighted error between the representative of each subset with respect to its closest centroid.

In order to approximate the solution of the weighted *K*-means problem, we use a generalization of Lloyd's algorithm called the **weighted Lloyd's algorithm** (**WL**, see Algorithm 1). In the assignment stage of WL (Step 0 and Step 2), the clustering $\mathcal{G}_r$ is induced by the set of centroids C_r. Furthermore, in the update step (Step 1), the set of centroids C_r is induced by the clustering $\mathcal{G}_{r-1}$. Similarly to Lloyd's algorithm, an execution of WL with l iterations produces a sequence of sets of centroids and clusterings that can be represented as follows:

$$C_0 \to \mathcal{G}_0 \to C_1 \to \mathcal{G}_1 \to \ldots \to C_{l-1} \to \mathcal{G}_{l-1} \to C_l \to \mathcal{G}_l$$

where C_0 is the set of centroids used for initialization and C_l is the returned set of centroids.

Algorithm 1: Weighted Lloyd (WL)

Input: Set of representatives $\{\overline{S}\}_{S \in \mathcal{P}}$ and weights $\{|S|\}_{S \in \mathcal{P}}$, for the partition $\mathcal{P}$. Number of clusters K and initial set of centroids C_0.
Output: Set of centroids C_r and corresponding clustering pattern $\mathcal{G}_r$.
Step 0 (**Initial Assignment**):
$\mathcal{G}_0 \longleftarrow C_0$; $r = 0$.
while *not StoppingCriterion* **do**
  $r = r + 1$.
  Step 1 (**Update Step**): $C_r \longleftarrow \mathcal{G}_{r-1}$ — **Clustering error** $E_{\mathcal{P}}(\mathcal{G}_{r-1})$ (Eq. 4)
  Step 2 (**Assignment Step**): $\mathcal{G}_r \longleftarrow C_r$ — **Centroid error** $E_{\mathcal{P}}(C_r)$ (Eq. 2)
end
Return C_r and $\mathcal{G}_r$.

The assignment step requires $O(|\mathcal{P}|kd)$ computations, since we just need to compute the distance between the set of centroids and the set of representatives, while for the update step of the set of centroids and the computation of its error (centroid error) $O(|\mathcal{P}|d)$ computations are needed. Remember that the most common stopping criterion of the *K*-means algorithm consists of verifying that the difference of the set of centroids error, in two consecutive iterations, is smaller than a certain threshold. Moreover, observe that the set of weights is only used when updating the set of centroids. Since the number of representatives usually satisfies $|\mathcal{P}| \ll n$, when dealing with massive data problems, we can have a relevant reduction in the complexity with respect to the *K*-means algorithm for the full dataset.

2.3. RPKM algorithm

In this section, we formally present the RP*K*M algorithm. This algorithm mainly consists of constructing a sequence of thinner partitions $\mathcal{P}_1, \ldots, \mathcal{P}_m$ and then applying WL over the set of representatives of each partition in the sequence. From one iteration to the next, the preceding found solution is used as initialization. As we will show later, this initialization assignment allows us to reduce the maximum number of WL iterations at every

RP*K*M run. The pseudo-code of the RP*K*M algorithm can be seen in Algorithm 1.

In the first step of the RP*K*M algorithm, we obtain backwards (see Section 2.1) the set of representatives and weights associated to the sequence of thinner partitions $\mathcal{P}_1, \ldots, \mathcal{P}_m$. Observe that we are assuming, without loss of generality, that $|\mathcal{P}_1| > K$. In Step 2, we update the centroids approximation by applying WL using the representatives and weights set determined at the previous step, we take as initialization the approximation for the previous iteration, C_{i-1}. In the first RP*K*M iteration, we set C_{i-1} as K random representatives of $\{\bar{S}\}_{S\in\mathcal{P}_i}$ (Forgy's type initialization). The algorithm iterates until $i = m$ or until a stopping criterion is met. We recommend the computation of a centroid's set displacement measure as stopping criterion: $\delta(C_{i-1}, C_i) = \max_{j=1,\ldots,K} \|\mathbf{c}_j^i - \mathbf{c}_j^{i-1}\|^2$. If this value is smaller than a certain threshold, the algorithm stops, since the approximation did not improve significantly after the last RP*K*M iteration.

In relation to the complexity of Algorithm 2 , we know, from Section 2.1, that Step 1 has an $O(d(n + \sum_{i=2}^{m} |\mathcal{P}_i|))$ time cost. Moreover, at the ith RP*K*M iteration, the time required for WL (Step 2) is $O(|\mathcal{P}_i|Kd)$. Finally, the recommended stopping criterion just performs $O(Kd)$ computations. Hence, the overall complexity of the RP*K*M algorithm, in the worst case, is $O(\max\{d(n + \sum_{i=2}^{m} |\mathcal{P}_i|), |\mathcal{P}_m|Kd\})$.

Algorithm 2: RP*K*M algorithm

Input: Dataset D, number of clusters K, maximum number of iterations m.
Output: Set of centroids approximation C_i.
Step 1 Compute the set of weights and representatives of the sequence of thinner partitions, $\mathcal{P}_1, \ldots, \mathcal{P}_m$, backwards.
Set $i = 1$.
while *not Stopping Criterion* **do**
  Step 2 Update the centroid's set approximation,
  $C_i = \{\mathbf{c}_j^i\}_{j=1}^K$: $C_i = WL(\{\bar{S}\}_{S\in\mathcal{P}_i}, \{|S|\}_{S\in\mathcal{P}_i}, K, C_{i-1})$
  $i = i + 1$
end
Return C_i

2.4. RP*K*M implementation based on grid partitions

Later on, we will verify that the theoretical advantages of the RP*K*M algorithm hold independently of the geometry that we use to generate the partition. Nonetheless, one way to guarantee the generation of a sequence of thinner partitions of the dataset consists of partitioning the space in a recursive manner. To do so, one possibility is to use a generalization of the *quadtrees* for higher dimensions [9]. The quadtree data structure has been used in several areas such as dimension reduction problems, spatial indexing, storing sparse data, computer graphics: computational fluid dynamics, etc [8].

The d-dimensional generalization of a quadtree is a tree data structure that generates partitions of the space into d-dimensional hypercubes and, subsequently, of the dataset in subsets in the following way: each internal node of the tree is exactly divided in 2^d children, i.e., each subset of the ith partition is divided into, at most, 2^d sets of the $(i+1)$th partition (see Fig. 1). This property allows us to generate, in a simple manner, a sequence of thinner partitions at each iteration satisfying $|\mathcal{P}_i| \leq \min\{n, 2^{id}\}$.

In the following example, we consider a set of 10000 points generated from a mixture of three 2D Gaussians. We compute, as a reference, the solution for $K = 3$ using the K-means++ method. After ten runs, we obtained, on average, an error of 11393.45 with a standard deviation of 4.69. The number of distance computations was, on average, 642, 000. In Fig. 1, we show the evolution of the RP*K*M algorithm, for $m = 6$, the red circles represent the initial set of centroids, the yellow diamonds the final set of centroids and the blue points the set of representatives for each iteration.

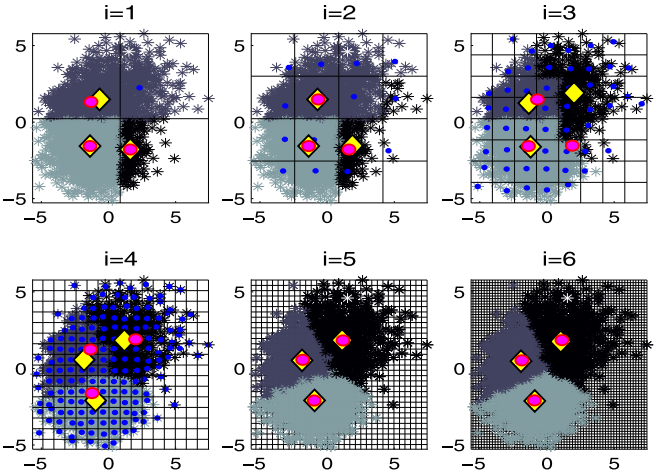

Fig. 1. Best clustering obtained at each RP*K*M iteration. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
RP*K*M iteration results.

i	Dis Com	$\lvert\mathcal{P}_i\rvert$	$E(C_i)$	i	Dis Com	$\lvert\mathcal{P}_i\rvert$	$E(C_i)$
1	24	4	14050.06	4	5697	173	11424.24
2	114	15	14024.38	5	10449	528	11408.40
3	1545	53	12350.41	6	26781	1361	11389.54

From Table 1, we can observe that, even at the fourth grid based RP*K*M iteration, which in this case implies 173 representatives (1.73% of the dataset), we have a fairly good approximation of the average best solution found by the K-means++ algorithm for the entire 10000 points. On average, the RP*K*M algorithm computed 0.887% and 4.17% of the total number of distance computations of the K-means++ algorithm, at the fourth and final iteration respectively.

As we consider higher iterations of the RP*K*M, the associated cost function converges to the best solution obtained by the the K-means++. The intuition behind this method is to transform a random initial set of centroids into a competitive approximation by using small groups of representatives, instead of the entire dataset. Next, we consider higher values of i to refine such an approximation.

3. Theoretical analysis of the RP*K*M algorithm

In this section, we perform a theoretical analysis of RP*K*M. In Section 3.1, we analyze the evolution of the clustering error at different steps of RP*K*M. Then, in Section 3.2, we investigate the repetitions of the clusterings obtained during the execution of RP*K*M and we bound the maximum number of WL iterations for different steps of RP*K*M.

Before starting with the theoretical results, we summarize an execution of RP*K*M with m steps given in terms of sequences of centroids and clusterings:

$$\mathcal{P}_1: \quad C_0^1 \to \mathcal{G}_0^1 \to C_1^1 \to \mathcal{G}_1^1 \to \ldots \to \mathcal{G}_{l_1-1}^1 \to C_{l_1}^1 \to \mathcal{G}_{l_1}^1$$
$$\mathcal{P}_2: \quad C_0^2 \to \mathcal{G}_0^2 \to C_1^2 \to \mathcal{G}_1^2 \to \ldots \to \mathcal{G}_{l_2-1}^2 \to C_{l_2}^2 \to \mathcal{G}_{l_2}^2$$
$$\ldots$$

$$\begin{aligned}
&\mathcal{P}_i: \quad C_0^i \to \mathcal{G}_0^i \to C_1^i \to \mathcal{G}_1^i \to \ldots \to \mathcal{G}_{l_i-1}^i \to C_{l_i}^i \to \mathcal{G}_{l_i}^i \\
&\ldots \\
&\mathcal{P}_m: \quad C_0^m \to \mathcal{G}_0^m \to C_1^m \to \mathcal{G}_1^m \to \ldots \to \mathcal{G}_{l_m-1}^m \to C_{l_m}^m \to \mathcal{G}_{l_m}^m
\end{aligned} \tag{3}$$

where l_i corresponds to the number of iterations of WL at step i of RPKM, the set of centroids C_{r+1}^i is induced by the clustering $\mathcal{G}_r^i$, and $\mathcal{G}_r^i$ is induced by C_r^i for $r = 1, \ldots, l_i - 1$ and $i = 1, .., m$. Each line corresponds to an execution of WL for a given partition $\mathcal{P}_i$ for $i = 1, \ldots, m$. It should be noted that, in step i of RPKM, the set of centroids C_0^i corresponds to the set of centroids obtained at the end of its previous step, that is $C_0^i = C_{l_i-1}^{i-1}$ for $i = 1, \ldots, m$. However, the clustering induced by $C_0^i = C_{l_i-1}^{i-1}$ for partition $\mathcal{P}_i$ does not have to correspond to the clustering induced for the previous partition $\mathcal{P}_{i-1}$. This fact is one of the main difficulties in guaranteeing a monotone decrement of the error function (see Eq. (1)) during an execution of RPKM.

In order to analyze the behavior of RPKM, we define the **clustering error** associated to a partition $\mathcal{P}$ as follows:

$$E_{\mathcal{P}}(\mathcal{G}) = \sum_{k=1}^{K} \sum_{S \in \mathcal{P}: S \subseteq G_k} |S| \cdot ||\overline{S} - \mathbf{c}_k||^2 \tag{4}$$

where the set of centroids C is induced by the clustering $\mathcal{G}$. This function measures the weighted error between each representative of a partition $\mathcal{P}$ and the center of mass of its corresponding cluster. Note that the only difference between the centroid error and clustering error is that, the centroid error is given in terms of a set of centroids and its induced clustering, while the clustering error is given by a clustering and its induced set of centroids. The importance of the clustering error is that it represents an intermediate value between the centroid errors obtained at two consecutive iterations of the algorithm, that is

$$E_{\mathcal{P}_i}(C_r^i) \geq E_{\mathcal{P}_i}(\mathcal{G}_r^i) \geq E_{\mathcal{P}_i}(C_{r+1}^i) \tag{5}$$

for $r = 0, \ldots, l_i - 1$ (see Eq. (3)). In the following subsections we will analyze the relation between the centroid error for different partitions of the data based on the inequality provided in Eq. (5).

3.1. Evolution of the centroids error

In this section we analyze the evolution of the centroid error for RPKM. The obtained results will be the basis for bounding the number of iterations of WL at each step of the RPKM. The next result will be used in order to analyze the relation between the clustering error given two partitions of the dataset (one thinner than the other).

Lemma 1. *Given a set of points D in $\mathbb{R}^d$ and a partition of it, $\mathcal{P}$. Then the function $f(\mathbf{c}) = |D| \cdot \|\overline{D} - \mathbf{c}\|^2 - \sum_{R \in \mathcal{P}} |R| \cdot \|\overline{R} - \mathbf{c}\|^2$ is constant.*

This result implies that the difference of the set of representatives with respect to a centroid, for two partitions of the dataset (one thinner than the other), is constant. The fact that such a difference is constant allows us to state, in the following lemma, the invariance of the clustering error for two different partitions of the dataset. Observe that Lemma 1 allows us to change the clustering, for both partitions, without changing the difference of the error associated to them.

Lemma 2. *[Invariance of the clustering error difference] Let $\mathcal{P}$ and $\mathcal{P}'$ be two partitions of the dataset D, with $\mathcal{P} \succ \mathcal{P}'$, and let $\mathcal{G}$ and $\mathcal{G}'$ be two clusterings of $\mathcal{P}$. Then, the difference between both clustering errors is constant with respect to the partitions $\mathcal{P}$ and $\mathcal{P}'$:*

$$E_{\mathcal{P}}(\mathcal{G}) - E_{\mathcal{P}}(\mathcal{G}') = E_{\mathcal{P}'}(\mathcal{G}) - E_{\mathcal{P}'}(\mathcal{G}')$$

In other words, the difference between two clustering errors is independent of the partition. For example, in Fig. 2, clustering $\mathcal{G}$ restricts the subsets with center of mass to the left (right) of the middle point of the bounding box to belong to the same cluster. The pink diamonds represent the centers of mass of each group of $\mathcal{G}$, evidently such centers of mass are invariant with respect to the partition that we use to represent $\mathcal{G}$. Furthermore, Lemma 2 states that the difference of the clustering error difference between the upper figures is equivalent to the difference of the error associated to the lower figures in Fig. 2.

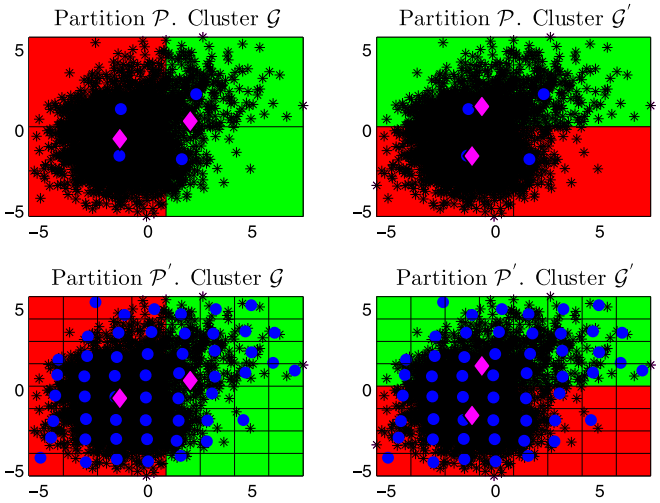

Fig. 2. Illustration of Lemma 2, for two clusterings $\mathcal{G}$ and $\mathcal{G}'$ defined on a partition $\mathcal{P}$.

In general, we can not guarantee a monotone descent of the error function given in Eq. (1), which corresponds to the centroid error for the thinnest partition, i.e., $\mathcal{D} = \{\{x\} : x \in D\}$. However, in the next result, under mild conditions related to the difference of the centroid error, we prove a monotone descent of the clustering error for two partitions, one thinner than the other. In consequence, if the conditions stated for the difference of the clustering error hold for all the steps of RPKM, a monotone descent of the error function in Eq. (1) is guaranteed.

Corollary 1. *[Condition for a monotone descent of the centroid error] Let C_i and C_{i-1} represent the set of centroids obtained at the ith and $(i-1)$th RPKM step, respectively. Then $E(C_i) \leq E(C_{i-1})$, if and only if $E(\mathcal{G}^{i-1}_{l_{i-1}-1}) - E(C_{i-1}) \leq \xi_i + (E(\mathcal{G}^i_{l_i-1}) - E(C_i))$, where $\xi_i = E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) - E_{\mathcal{P}_i}(\mathcal{G}^i_{l_i-1})$*

That is, if after the assignment step for both sets of centroids, C_i and C_{i-1}, with respect to the full dataset, the condition in Theorem 1 is satisfied at every RPKM iteration, then we can guarantee the monotone descent of the error over the entire dataset. In particular, if there are no reassignments for any of the two cases, with respect to their associated cluster membership, we can guarantee the monotone descent of the overall error. Clearly, as the difference of the local error of the initial and final cluster at the ith RPKM step is larger, then it is more likely to satisfy such a condition.

Even when the monotone descent over the entire dataset of the RPKM approximation, at every step, is not proved in general, we will see in the experiments summarized in Section 4 that, for real and artificial datasets, this property has been observed.

3.2. Bounding the iterations of the weighted Lloyd's algorithm

In this section, using the properties of the clustering error (Lemma 2), we can analyze the construction of the set of cluster-

ings at different RP*K*M steps, for example we can verify the implications of repeating a clustering from a previous RP*K*M step.

In Lemma 3, we state that the unique clustering that can be repeated in a step of RP*K*M, is the previous clustering of the sequence of clusterings generated by the RP*K*M. If the repeated clustering is obtained at the first iteration of WL, then the previous clustering corresponds to the one obtained at the last iteration of WL (at the previous step of RP*K*M). On the other hand, if the repeated clustering is not obtained at the first iteration of WL, then the previous clustering corresponds to the one obtained at the previous iteration of WL (in the same RP*K*M step).

Lemma 3. *At the ith step of the RPKM, if* $\mathcal{G}_r^i = \mathcal{G}_s^j$, *with* $j < i$, *for some* $r \in \{1, \ldots, l_i - 1\}$ *and* $s \in \{1, \ldots, l_j - 1\}$, *then* $l_{j+1} = \ldots = l_i = 1$. *Moreover, in that case,* $s = l_j - 1$.

In the following theorem, we use Lemma 3 to bound the number of WL iterations at each RP*K*M step. Lemma 3 indicates that the only cluster that can be repeated, from a previous RP*K*M step, is the last one generated by the corresponding WL execution. Therefore, we can eliminate, from the total number of possible clusterings, the ones that were generated at previous RP*K*M iterations (except the last one). In particular, if we have more than one Lloyd iteration at a certain RP*K*M step, then we automatically discard every single cluster that was generated at a previous RP*K*M step.

Theorem 1. *[Upper bound to the number of local WL iterations] An upper bound to the number of Lloyd iterations at the ith RPKM step is given by* $l_i \leq \left\{ {|\mathcal{P}_i| \atop K} \right\} - \sum_{j=1}^{i-1} (l_j - 1)$, *where* $\left\{ {|\mathcal{P}_i| \atop K} \right\}$ *is a Stirling number of the second kind.*

Following the same reasoning as in Theorem 1, observe that, if, at the $(i-1)$th RP*K*M step, WL converges to the associated global optima, then $l_i \leq \left\{ {|\mathcal{P}_i| \atop K} \right\} - \left\{ {|\mathcal{P}_{i-1}| \atop K} \right\} + 1$. Moreover, observe that all the clusters with an error greater than $E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1})$ will not be generated in the current or at any further RP*K*M iteration, however the amount of clusterings satisfying this condition can not be counted at the moment, one hypothesis is that the number of such clusterings is of order $O(\left\{ {|\mathcal{P}_i| \atop K} \right\})$.

For this reason, selecting the local initialization of WL in this manner may help reducing the number of Lloyd's iterations, while discarding, at each step, all the generated clusters (except one) and probably others of similar form. Not only that, but the discarding of such clusters occurs while analyzing a small number of representatives with regard to the full dataset, which implies, as we will see in the experimental section, a drastic reduction in the number of distance computations.

4. Experimental section

In this section, we perform a set of experiments so as to analyze the relation between the number of distance computations and the quality of the approximation for the RP*K*M algorithm proposed in Section 2. In particular, the experiments focus on the implementation of RP*K*M based on grid partitions proposed in Section 2.4. As the number of representatives induced by the grid based RP*K*M does not scale well with respect to the dimensionality of the problem, we consider a reduced number of dimensions, $d < 10$. In Section 5, we will elaborate on future steps to make this strategy more scalable with respect to this factor. In addition, we analyze the effect on the algorithm performance of varying the different parameters of the clustering problem: size of the dataset, n, dimension of the instances, d, and number of clusters, K. For the purposes of the experimental analysis, we compare the performance of the grid based RP*K*M algorithm against the ***K*-means++** [1] (***K*M++**) and the **minibatch** ***K*-means** (**MB**) on artificial and real datasets.

The grid based RP*K*M was implemented in Python, while we used the *K*M++ and MB implementations that are available in the open source machine learning library *scikit-learn* of Python. As stopping criterion for the RP*K*M, we just set a maximum number of iterations, m, since we want to analyze the behavior of the error function, at each step, as the number of representatives approaches the number of instances. For the analyzed datasets, we observe that, with $m \leq 6$, this occurs. Evidently, as we increase the dimension, this property will be seen inmediately, since the number of representatives increases exponentially with respect to this parameter.

In this section, we refer to the result obtained after the mth step of the grid based RP*K*M by **RP*K*M *m***, and to the solution obtained using MB with a batch size $b \in \{100, 500, 1000\}$ by **MB *b***. In equivalent experimentations similar batch sizes were used [24].

4.1. Artificial datasets results

The artificial datasets are generated as a d-dimensional mixture of K Gaussians. In particular, we set $K \in \{3, 9\}$, $d \in \{2, 4, 8\}$ and $n \in \{1000, 10000, 100000, 1000000\}$. For each setting, we generate 50 replicates of the dataset. Additionally, we consider a component overlapping lower than 5%.

4.1.1. Distance computations

In this section, we compare the behavior of RP*K*M, *K*M++ and MB in terms of the computed distances. As we commented in Sections 1.1 and 2.2, the most time consuming phase of the Lloyd's algorithm, and its weighted version, refers to the computation of distances. Especially at the initial steps, RP*K*M considers a number of representatives which is a small fraction of the size of the dataset. Thus, we would expect a greater reduction in the number of distance computations, with respect to the other methods, as we consider larger datasets.

In Fig. 3, we present the relation between the number of distance computations and the dataset size for the different settings.

At first glance, we observe that RP*K*M, in general, executes a much smaller number of distance computations than both *K*M++ and MB. Such a relation seems to change for the latter steps of RP*K*M when we consider larger dimensions. However, in that case, *K*M++ still requires a similar order of computations in comparison to the latter steps of the RP*K*M. Analogously, MB, for the different batches, is not able to match the number of distance computations of RP*K*M at its first steps, for any of the analyzed settings. In addition, we observe that, for some RP*K*M steps, the number of distance computations does not increase as we consider a higher number of instances, as happens with the other algorithms.

In particular, we notice that the number of distance computations, at the first steps of the RP*K*M, i.e., RP*K*M 1 and RP*K*M 2, does not necessarily increase with respect to the dataset size. This is plausible since, in this case, the number of representatives is of the same order, independent of the number of instances (see Fig. 4). Evidently, as we consider thinner partitions ($m \geq 3$), the number of representatives will increase and so will the number of distance computations.

Since, at the earlier stages of the RP*K*M, the number of representatives does not necessarily increase with respect to the dataset size, then the ratio between the number of distance computations of RP*K*M and *K*M++ (or MB) decreases with respect to the size of the dataset. In particular, for the larger number of instances, the number of distances computed by RP*K*M with respect to *K*M++ is

[1] Lloyd's algorithm initialized via *K*-means++ [2].

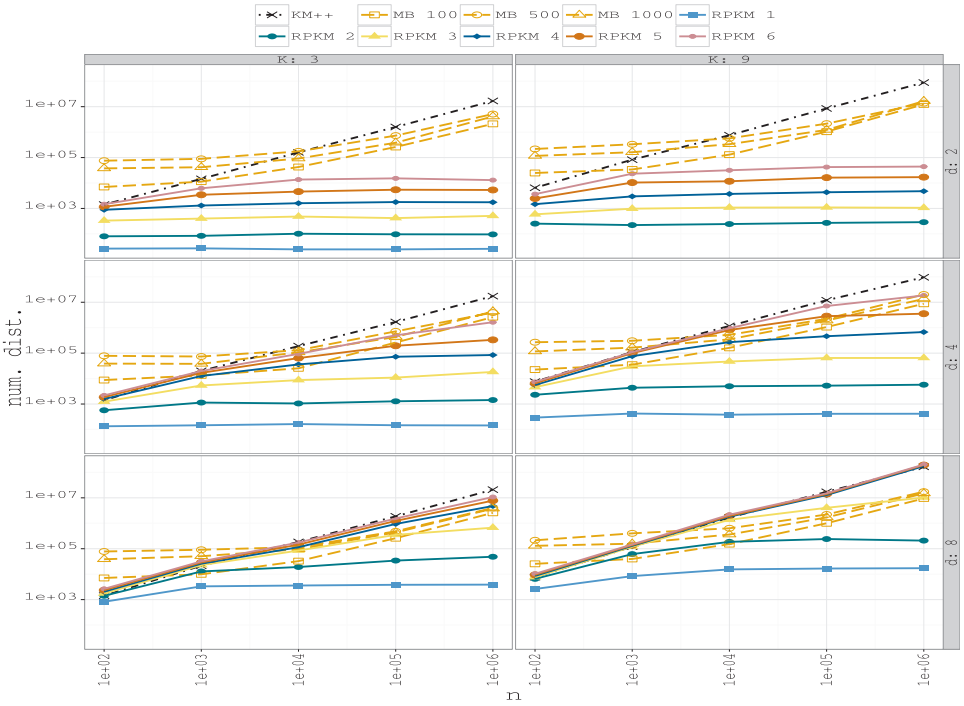

Fig. 3. This figure shows the number of distance computations with respect to the size of the dataset (n), for different numbers of dimensions (d) and numbers of clusters (K).

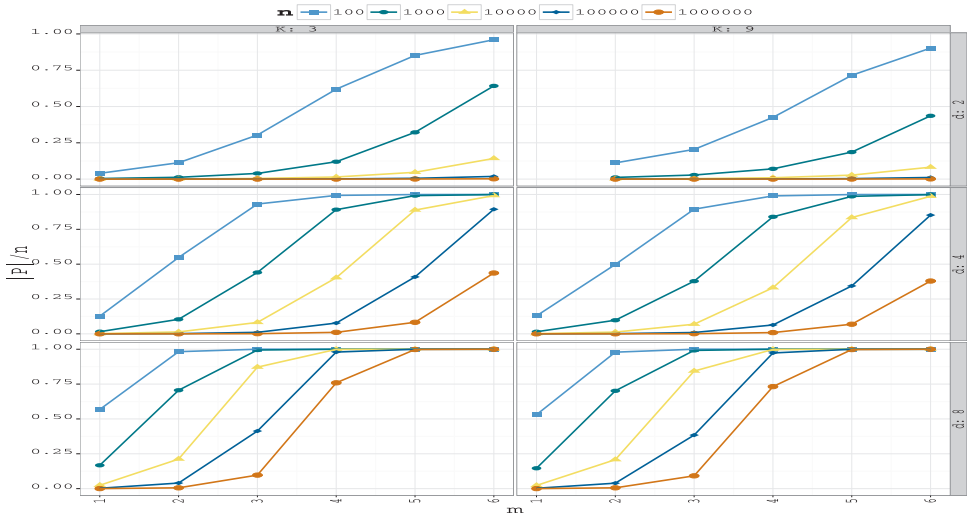

Fig. 4. This figure shows the ratio between the size of the partition P_m and the size of the data set n, for $m = 1, ..., 6$. The size of the partition $\mathcal{P}_m$ corresponds to the number of representatives used by RPKM at its mth step.

3 orders of magnitude lower for $K = 3$ and $d = 8$ and 6 orders of magnitude lower for $K = 3$ and $d = 2$. In comparison with MB, the number of distances computed by RPKM is 2 orders of magnitude lower for $K = 3$ and $d = 8$ and 5 orders of magnitude lower for $K = 3$ and $d = 2$. As we can see, the dimensionality of the problem, d, has a great impact on the number of distances computed by RPKM as m increases. The reason is that the number of representatives used by RPKM can increase exponentially with respect to both m and d. In addition, we can see that the number of distance computations, for all the algorithms, as expected, increases linearly with the number of clusters K.

4.1.2. Quality of the approximation

In the previous section, we observed that RPKM entails a drastic reduction in the amount of distance computations with respect to the other approaches (especially when we consider large dataset sizes). However, in this section, we would like to analyze the quality of the approximations obtained by means of RPKM.

In Fig. 5, we show the evolution of the standarized error (std.error) for the full dataset for the set of centroids obtained at the end of the mth step of the RPKM. The std.error is defined as $\rho(m) = \frac{E_m^* - E_m}{E_m^*}$, where E_m is the error for RPKM at the mth step, and E_m^* is the error obtained by K-means algorithm over the full dataset D, taking as initialization the centroids obtained by RPKM at the mth step. Observe that $\rho(m) \leq 0$ and it measures the percentage of error with respect to the K-means over the entire dataset.

In most of the cases, we observe a monotone descent of the centroid error with respect to the full dataset until convergence to

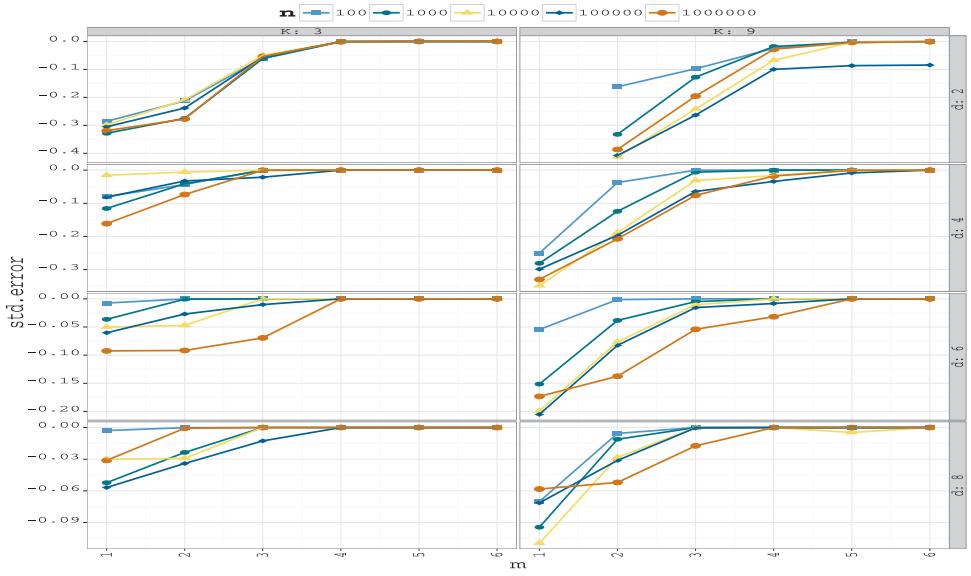

Fig. 5. Quality of the approximation (std.error) with respect to the RPKM step.

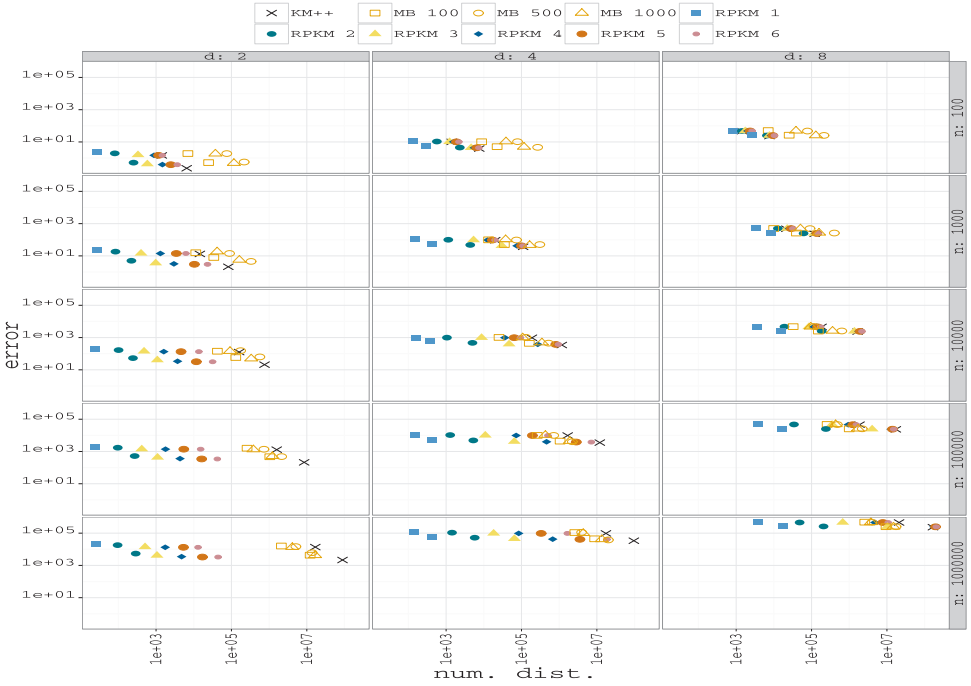

Fig. 6. Quality of the approximation vs number of distance computations.

the error associated to a solution of the K-means algorithm. This is remarkable since the approximation is constructed over a reduced number of representatives with respect to the total number of instances (see Fig. 4). In particular, at the third RPKM step, the error with respect to the K-means solution is under 10%. Evidently, as we increase the dimension, this percentage decreases until it achieves an error percentage smaller than 10% and 5% for the first and second RPKM step, respectively. Moreover, for $d = 2$ and $K = 9$, we observe no result for RPKM 1, this is due to the fact that, in this case, the number of representatives is smaller than the number of clusters, i.e., $|\mathcal{P}_1| < K$.

4.1.3. Relation distance computations - quality of the approximation

In this section, we fuse the results obtained at the previous sections and analyze, for the different algorithms, the trade-off between the number of distances computed and the quality of the obtained solutions.

In Fig. 6, we show the relation between the number of distance computations and the error of the obtained solutions for RPKM, KM++ and MB.

Besides the dimensionality of the problem, as we increase the number of instances, the cloud of points associated to KM++ and MB separates from the ones associated to the RPKM. This means that, as we increase the number of instances, KM++ and MB require a much larger number of distance computations in order to achieve a solution of similar quality to those obtained by the RPKM. In the best case scenario ($n = 1000000$, $d = 2$), RPKM reduces, at least, 6 and 4 orders of magnitude with respect to the K-means++ and the minibatch K-means. Evidently, for larger dimensions, the clouds associated to the RPKM, for the latter stages, can

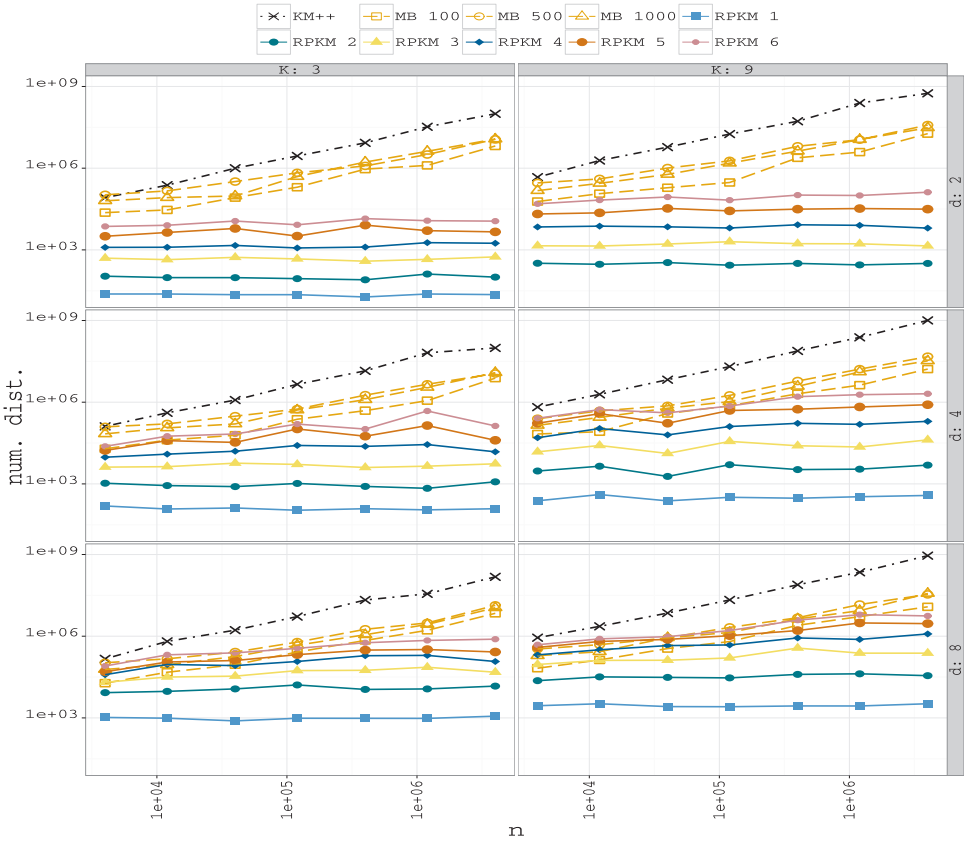

Fig. 7. Number of distance computations with respect to dataset size, n.

overlap those of *K*M++ and MB. In this case, even for the largest number of instances, we did not need to execute all the RP*K*M steps: at the fifth RP*K*M step, we have already generated, approximately, as many representatives as instances.

In particular, consider the extreme cases: $d = 2,\ n = 1000000$ (case 1) and $d = 8,\ n = 100$ (case 2). In the first case, after the third RP*K*M step the standard error is already under 5% and, after fourth step, the error is practically null. Such approximations are obtained after computing under 10^{-3}% (10^{-2}%) and slightly over 10^{-3}% (10^{-2}%) of the distances calculated by *K*M++ (MB) for RP*K*M 3 and RP*K*M 4, respectively.

In the second case, already after the first RP*K*M step, the standard error is under 8% and, after the second step, the error is fairly close to zero. Such approximations are obtained after computing under 10^{-1}% (1%)and under 1% (10%) of the distances calculated by *K*M++ (MB) for RP*K*M 1 and RP*K*M 2, respectively.

In the case of lower dimensions and greater number of instances, we require more RP*K*M steps to achieve an approximation with a similar standard error with respect to the case with greater dimensions and lower number of instances. As previously mentioned, this is due to the exponential growth of the number of representatives with respect to the dimension of the problem. Moreover, in the second case, we have a lower number of instances, hence, we need fewer RP*K*M steps in order to generate as many representatives as instances (in this example, this occurs for $m = 3$). However, having a lower proportion of representatives with respect to the number of instances also implies a greater reduction in the number of distance computations with respect to the full dataset for the first case, while obtaining a similar standard error in comparison to the second case.

4.2. Real datasets

In addition to the previous experimentation, we evaluate the performance of the grid based RP*K*M algorithm, *K*M++ and MB on a real-world dataset: the gas sensor array under dynamic gas mixtures dataset, which contains the acquired time series from 16 chemical sensors exposed to gas mixtures at varying concentration levels [10]. The dataset consists of 4178504 instances and 19 attributes and is available in the UC-Irvine Machine Learning Repository. The same experiment was performed over different datasets from the UC-Irvine Machine Learning Repository, achieving similar conclusions. For the sake of brevity, the corresponding graphics are not included in this work.

Using this real-world dataset, we generate different subsamplings that we use to analyze the features of the algorithms. In particular, we take $d \in \{2, 4, 8\}$ random attributes and $n \in \{4000, 12000, 40000, 120000, 400000, 1200000, 4000000\}$ random instances. The number of clusters is $K \in \{3, 9\}$. For each setting, we generate 10 replicates of the dataset.

4.2.1. Distance computations

For the real dataset experimentation, we perform the same analysis as that carried out for the artificial datasets case.

In Fig. 7, as in Fig. 3, we present the relation between the number of distance computations and the dataset size, this time for the real dataset. In general, we observe a similar behavior with respect to the artificial datasets case: RP*K*M reduces, in many orders of magnitude, the number of distance computations with respect to *K*M++ and MB. However, in this case, even at the last step of RP*K*M, the number of distances does not always increase with respect to the number of instances.

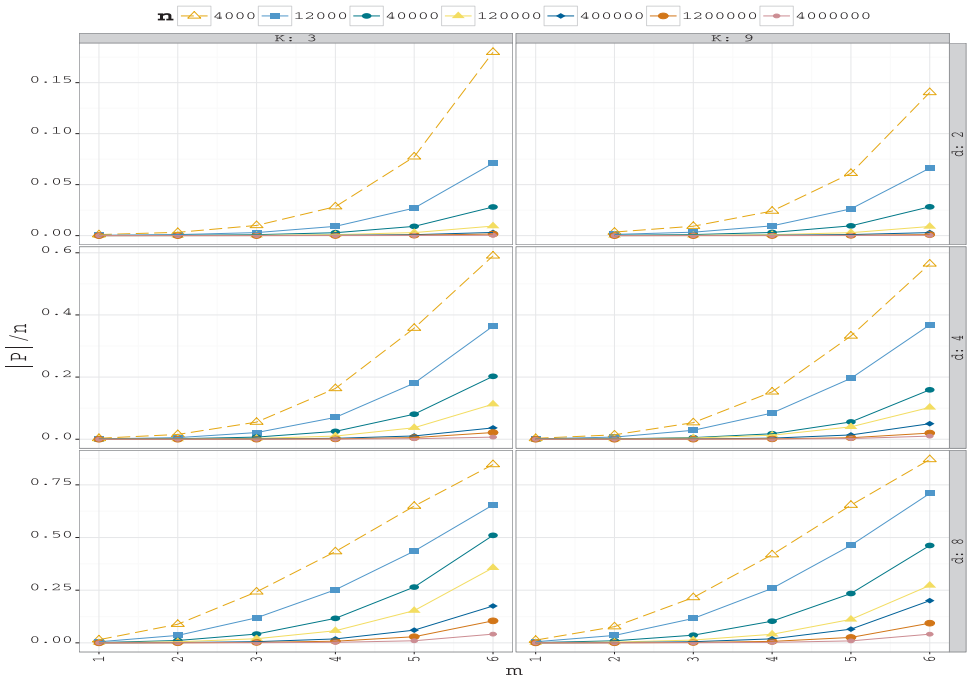

Fig. 8. Percentage of subsets with respect to RPKM step.

In particular, as can be verified in Fig. 8, the number of representatives does not necessarily increase as we consider higher dataset sizes. This is due to the fact that the data points, in this case, are grouped into more condensed clouds than in the case of the artificial Gaussian dataset case. Hence, it is plausible to observe that the number of distance computations, even at the latter stages of the RPKM, does not necessarily increase with respect to the dataset size. Furthermore, for low dimensions, this fact implies that, even at the last RPKM step, we have a lower number of distance computations than any version of MB and KM++.

In more detail, we notice that for the largest number of instances, the last step of the RPKM executes less than 1% and 0.1% of the distances computed by MB and KM++, respectively. For the largest dimension considered ($d = 8$), the latter steps RPKM execute a similar order of distance computations with respect to MB. However, intermediate RPKM steps (RPKM 3) still computes less than 1% and 0.1% of the distances computed with respect to MB and KM++. It is important to remember that the number of distance computations for KM++ and MB is independent of the dimensionality of the problem, which is not usually true for the RPKM algorithm.

For the analyzed dataset, we observe that the number of representatives does not grow as rapidly as in the artificial sets case. In particular, for the lowest dimension, the number of representatives barely achieves 20% of the dataset size at the last grid based RPKM step. As we increase the dimensionality, and therefore generate more representatives, this number grows to over 75% for the shortest dataset size case.

4.2.2. Quality of the approximation

In Fig. 9, we observe the evolution of the standarized error, for the full dataset, with respect to the set of centroids obtained at the mth step of the RPKM. As commented in the artificial datasets case, in most of the cases, there is a monotone descent of the centroid error with respect to the full dataset until convergence to the error associated to a solution of the K-means algorithm over the full dataset. Commonly, at the third RPKM step, the such standarized error is under 10%. This is remarkable since, as can be seen in Fig. 7, the number of distances computed by RPKM 3, compared to MB and KM++, is under 10^{-3} % and 10^{-4} %, respectively.

As we increase the dimension the standarized error, in most of the cases, is under 10%, even for RPKM 2 on the largest dataset size case.

4.2.3. Relation distance computations - quality of the approximation

In order to have a better understanding of the relation distance computations against overall error, we consider Fig. 10. As we observed in the artificial dataset case, when we increase the number of instances, the cloud of points associated to KM++ and MB separates from the ones associated to the RPKM. As we increase the number of instances, KM++ and MB require a much larger number of distance computations in order to achieve a solution of similar quality than those obtained by the RPKM. For the lowest dimensional case, even for an intermediate RPKM step, it reduces over 4 orders ($d = 2, n = 4000000$) and 5 orders of magnitude ($d = 2, n = 4000000$) with respect to MB and KM++, respectively. Evidently, as we increase the dimensionality of the problem and, therefore, generate a larger amount of representatives and compute more distances, we still observe a reduction of 4 and 5 orders of magnitude, but for the first RPKM step. In addition, we can see that the minibatch K-means with the smallest batch, MB 100, has, in some cases, a similar amount of distance computations with respect to the second step of RPKM.

Such a reduction in the number of distance computations is achieved while still obtaining competitive approximations. For instance, even in the case of greater dimension and lower number of instances ($d = 8,\ n = 4000$), after the second RPKM step the standard error is already under 5% and, after the third step, the error is practically null. These approximations are obtained after computing under 7×10^{-4}% (2×10^{-3}%)and 10^{-3}% (5×10^{-3}%) of the distances calculated by KM++ (MB), for RPKM 3 and RPKM 4, respectively.

In general, we observe that the grid based RPKM algorithm is able to generate competitive approximations with a significant re-

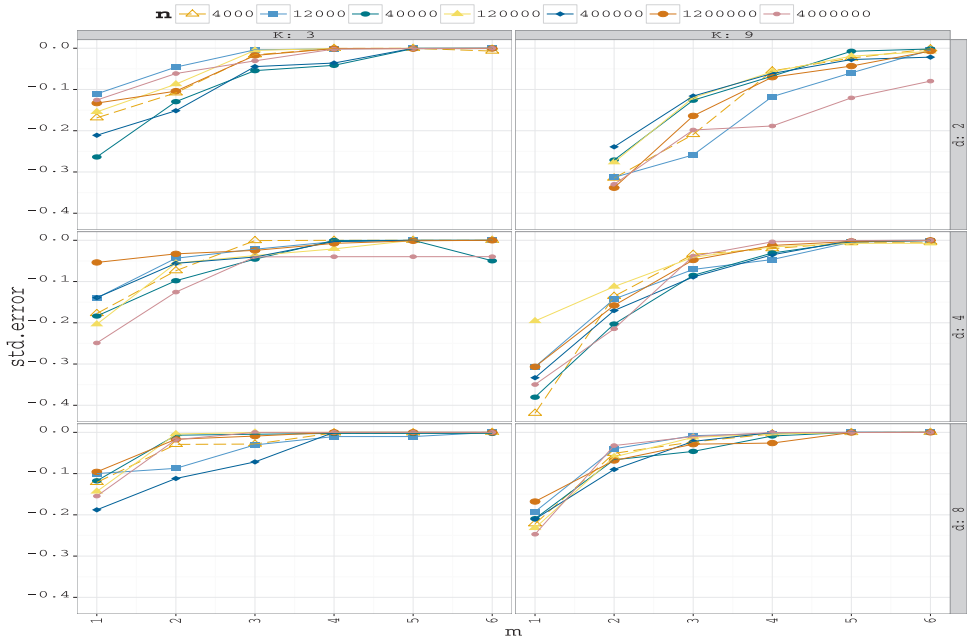

Fig. 9. Quality of the approximation with respect to RPKM step.

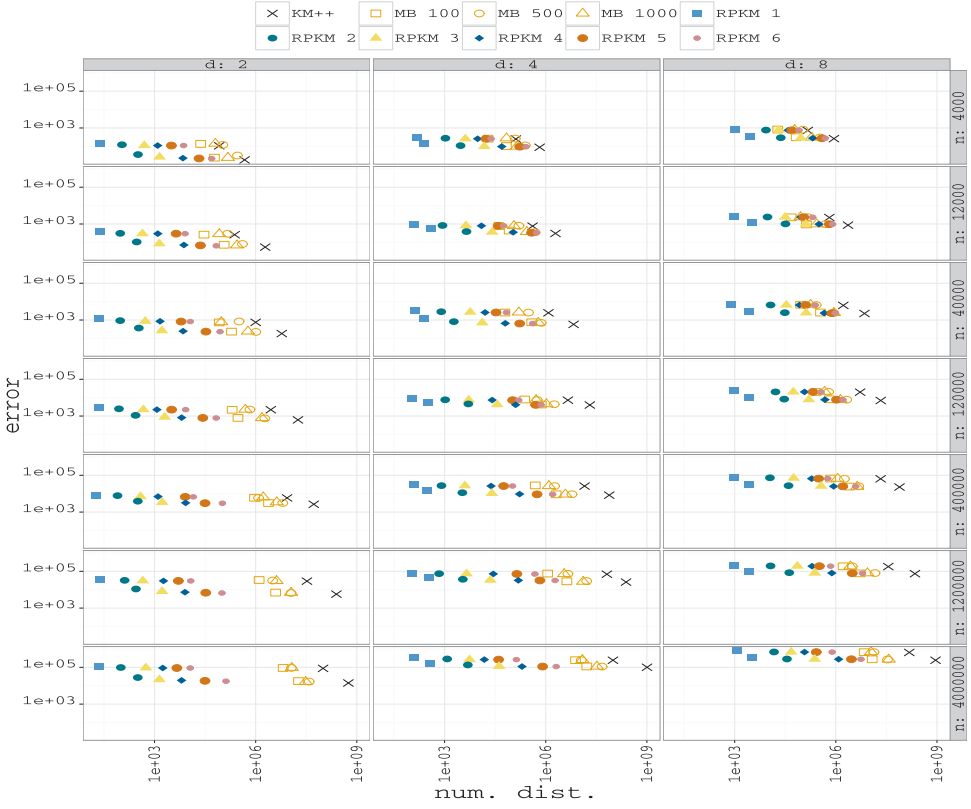

Fig. 10. Quality of the approximation vs number of distance computations.

duction in the number of distance computations[2]. As the dataset size increases, we observed a more drastic reduction in the number of distance computations as can be seen in Figs. 6 and 10. We observe the same behavior as we increase the dimension of the instances, however, the order of reduction with respect to MB and KM++ decreases, as expected. This is due to the fact that the number of RPKM representatives, in this case, grows exponentially with respect to the dimension of the dataset.

5. Final comments and future work

In this work, we present an alternative to the *K*-means algorithm applicable to massive data problems called recursive parti-

[2] Tables with a more detailed summary of the results, for the different datasets, can be found at https://bitbucket.org/mcapo/rpkm/src/.

tion based K-means (RPKM). This approach recursively partitions the entire dataset into a small number of subsets, each of which is characterized by its representative (center of mass) and weight (cardinality), after that a weighted version of the Lloyd's algorithm is applied over this local representation. The objective is to describe the full dataset by this representation, which ultimately leads to a reduction of distance computations. Indeed, in the experimental section, we observe that the RPKM algorithm generates competitive approximations, even at its earlier iterations, while reducing several orders of magnitude of distance computations.

In Section 3, we have derived some theoretical properties of the RPKM algorithm. Among other results, we can guarantee the non repetition of the clusterings generated at each RPKM iteration (except the last one), which ultimately implies the reduction of the total amount of Lloyd iterations, as well as leading, in most of the cases, to a monotone decrease of the overall error function.

In the experimental section, the grid based RPKM was compared with two well-known approaches: the K-means++ and mini-batch K-means. In this analysis, we observed a dramatic reduction in the number of distance computations with respect to both of them, as well as a consistent monotone decrease of the error function. Since the RPKM algorithm seeks to reduce the number of representatives used per iteration, we observed a larger reduction in the number of distance computations as we enlarged the number of instances of the dataset. Furthermore, at the earlier stages of the RPKM, the size of the dataset did not have a relevant impact on the number of iterations or distance computations for the associated weighted K-means problem. Thus, the number of computations, especially for massive data applications, can be greatly reduced.

On the other hand, it is important to remark that the number of subpartitions generated at each iteration of the grid based RPKM grows exponentially with respect to the dimension, d, of the dataset. As we can see in Fig. 1, some of these subpartitions might have a small probability of changing their current cluster affiliation (with respect to the assignment given by the previous RPKM iteration). In this sense, it might be more valuable to partition the areas that are more likely to have subpartitions associated with different clusters.

One possible approach consists of characterizing the subsets that lie on a cluster boundary, i.e., subsets that are close to two or more clusters [15]. In this approach, the number of representatives does not grow exponentially with respect to the dimension of the dataset. For this reason, as a future step, we plan to define a low computational cost algorithm to determine the cluster boundary at each iteration. The subsets in this area will have a greater priority when selecting the regions that we want to partition in the next RPKM iteration. Moreover, there exist different dimensionality reduction strategies for the Lloyd's algorithm that can be used to alleviate the dependency of the grid based partition with respect to this factor, such as random projections [4,6] and SVD-based feature selection [3].

One last, but still important, advantage of the RPKM algorithm is the fact that its parallelization is direct. The RPKM algorithm mainly depends on two steps: The data partition process and the application of the weighted version of Lloyd's algorithm. In the first step, each point can independently decide which subset it belongs to, hence the construction of the set of representatives and weights can be done in a parallel manner. Analogously, for the second phase, given a set of prototypes, each data point can separately decide which cluster it belongs to and the update of the centroid can be simply computed by averaging the points [13,27]. For this reason, we also plan to implement the RPKM algorithm on a parallel framework such as *Apache Spark*.

Acknowledgments

Marco Capó and Aritz Pérez are partially supported by the Basque Government, Elkartek and by the Spanish Ministry of Economy and Competitiveness MINECO: BCAM Severo Ochoa excelence accreditation SVP-2014-068574 and SEV-2013-0323. José A. Lozano is partially supported by the Basque Government (IT609-13), Elkartek and the Spanish Ministry of Economy and Competitiveness MINECO (TIN2013-41272P).

Appendix A. RPKM properties

Lemma 1. *Given a set of points D in $\mathbb{R}^d$ and a partition of it, $\mathcal{P}$. Then the function $f(\mathbf{c}) = |D| \cdot \|\overline{D} - \mathbf{c}\|^2 - \sum_{R\in\mathcal{P}} |R| \cdot \|\overline{R} - \mathbf{c}\|^2$ is constant.*

Proof. First of all observe that $|D| = \sum_{R\in\mathcal{P}} |R|$ and $\overline{D} = \frac{\sum_{R\in\mathcal{P}} |R|\cdot\overline{R}}{|D|}$. Given any pair $\mathbf{c}, \mathbf{c}' \in \mathbb{R}^d$, we have that

$$\begin{aligned} f(\mathbf{c}) &= |D| \cdot \|\overline{D} - \mathbf{c}\|^2 - \sum_{R\in\mathcal{P}} |R| \cdot \|\overline{R} - \mathbf{c}\|^2 \\ &= |D| \cdot (\|\overline{D} - \mathbf{c}'\|^2 + 2(\mathbf{c}' - \mathbf{c})^t(\overline{D} - \mathbf{c}') + \|\mathbf{c}' - \mathbf{c}\|^2) \\ &\quad - \sum_{R\in\mathcal{P}} |R| \cdot (\|\overline{R} - \mathbf{c}'\|^2 + 2(\mathbf{c}' - \mathbf{c})^t(\overline{R} - \mathbf{c}') + \|\mathbf{c}' - \mathbf{c}\|^2) \\ &= f(\mathbf{c}') + 2|D|(\mathbf{c}' - \mathbf{c})^t(\overline{D} - \mathbf{c}') - 2\sum_{R\in\mathcal{P}} |R|(\mathbf{c}' - \mathbf{c})^t(\overline{R} - \mathbf{c}') \\ &= f(\mathbf{c}') \end{aligned}$$

□

Lemma 2. *[Invariance of the clustering error difference] Let $\mathcal{P}$ and $\mathcal{P}'$ be two partitions of the dataset D, with $\mathcal{P} \succ \mathcal{P}'$, and let $\mathcal{G}$ and $\mathcal{G}'$ be two clusterings of $\mathcal{P}$. Then, the difference between both clustering errors is constant with respect to the partitions $\mathcal{P}$ and $\mathcal{P}'$:*

$$E_{\mathcal{P}}(\mathcal{G}) - E_{\mathcal{P}}(\mathcal{G}') = E_{\mathcal{P}'}(\mathcal{G}) - E_{\mathcal{P}'}(\mathcal{G}')$$

Proof. Let the index of $S \in \mathcal{P}$ on the cluster $\mathcal{G} = \{G_k\}_{k=1}^K$, $i(S, \mathcal{G})$, be defined as $i(S, \mathcal{G}) = \{k \mid S \subseteq G_k\}$, then

$$\begin{aligned} &E_{\mathcal{P}}(\mathcal{G}) - E_{\mathcal{P}'}(\mathcal{G}) \\ &= \sum_{S\in\mathcal{P}} |S| \cdot \|\overline{S} - \overline{G_{i(S,\mathcal{G})}}\|^2 - \sum_{S'\in\mathcal{P}'} |S'| \cdot \|\overline{S'} - \overline{G_{i(S',\mathcal{G})}}\|^2 \\ &= \sum_{S\in\mathcal{P}} (|S| \cdot \|\overline{S} - \overline{G_{i(S,\mathcal{G})}}\|^2 - \sum_{R\in\mathcal{P}'[S]} |R| \cdot \|\overline{R} - \overline{G_{i(R,\mathcal{G})}}\|^2) \\ &= \sum_{S\in\mathcal{P}} (|S| \cdot \|\overline{S} - \overline{G_{i(S,\mathcal{G})}}\|^2 - \sum_{R\in\mathcal{P}'[S]} |R| \cdot \|\overline{R} - \overline{G_{i(S,\mathcal{G})}}\|^2) \\ &= \sum_{S\in\mathcal{P}} (|S| \cdot \|\overline{S} - \overline{G'_{i(S,\mathcal{G}')}}\|^2 - \sum_{R\in\mathcal{P}'[S]} |R| \cdot \|\overline{R} - \overline{G'_{i(S,\mathcal{G}')}}\|^2) \\ &= \sum_{S\in\mathcal{P}} (|S| \cdot \|\overline{S} - \overline{G'_{i(S,\mathcal{G}')}}\|^2 - \sum_{R\in\mathcal{P}'[S]} |R| \cdot \|\overline{R} - \overline{G'_{i(R,\mathcal{G}')}}\|^2) \\ &= \sum_{S\in\mathcal{P}} |S| \cdot \|\overline{S} - \overline{G'_{i(S,\mathcal{G}')}}\|^2 - \sum_{S'\in\mathcal{P}'} |S'| \cdot \|\overline{S'} - \overline{G'_{i(S',\mathcal{G}')}}\|^2 \\ &= E_{\mathcal{P}}(\mathcal{G}') - E_{\mathcal{P}'}(\mathcal{G}') \end{aligned}$$

□

Before prooving Corollary 1 and Lemma 3, we analyze the error of a cluster with respect to a weighted K-means iteration. We observe that following inequality is satisfied at the rth weighted Lloyd's algorithm iteration:

$$E_{\mathcal{P}}(C_r) \geq E_{\mathcal{P}}(\mathcal{G}_r) \geq E_{\mathcal{P}}(C_{r+1}) \tag{A.1}$$

Furthermore, observe that, if $E_{\mathcal{P}}(C_r) = E_{\mathcal{P}}(\mathcal{G}_r)$, then, after the update step of the weighted Lloyd's algorithm, we obtain $C_r = C_{r+1}$ and the algorithm stops at the $(r+1)$th iteration. On the

other hand, if $E_{\mathcal{P}}(C_r) > E_{\mathcal{P}}(\mathcal{G}_r) = E_{\mathcal{P}}(C_{r+1})$, then, in the following weighted Lloyd's algorithm iteration, we obtain $E_{\mathcal{P}}(C_{r+1}) = E_{\mathcal{P}}(\mathcal{G}_{r+1}) = E_{\mathcal{P}}(C_{r+2})$ and $C_{r+1} = C_{r+2}$, hence the algorithm stops, at most, at the $(r+2)$th iteration.

Hence, we have the following chain of inequalities for any weighted Lloyd's algorithm run

$$E_{\mathcal{P}}(C_0) > E_{\mathcal{P}}(\mathcal{G}_0) > E_{\mathcal{P}}(C_1) > E_{\mathcal{P}}(\mathcal{G}_1) > E_{\mathcal{P}}(C_2) > \ldots > E_{\mathcal{P}}(\mathcal{G}_{l-2}) \geq E_{\mathcal{P}}(C_{l-1}) \geq E_{\mathcal{P}}(\mathcal{G}_{l-1}) \geq E_{\mathcal{P}}(C_l), \quad (A.2)$$

where l is the total number of weighted Lloyd iterations.

Corollary 1. *[Condition for a monotone descent of the centroid error] Let C_i and C_{i-1} represent the set of centroids obtained at the ith and $(i-1)$th RPKM step, respectively. Then $E(C_i) \leq E(C_{i-1})$, if and only if $E(\mathcal{G}^{i-1}_{l_{i-1}-1}) - E(C_{i-1}) \leq \xi_i + (E(\mathcal{G}^i_{l_i-1}) - E(C_i))$, where $\xi_i = E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) - E_{\mathcal{P}_i}(\mathcal{G}^i_{l_{i-1}})$*

Proof. From Lemma 2 and Eq. (A.2), we have the following inequalities:

$$E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) \geq E_{\mathcal{P}_i}(C_{i-1}) \geq E_{\mathcal{P}_i}(\mathcal{G}^i_{l_i-1}) \geq E_{\mathcal{P}_i}(C_i) \quad (A.3)$$

$$E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) - E_{\mathcal{P}_i}(\mathcal{G}^i_{l_i-1}) = E_{\mathcal{P}_j}(\mathcal{G}^{i-1}_{l_{i-1}-1}) - E_{\mathcal{P}_j}(\mathcal{G}^i_{l_i-1}) = \xi_i \geq 0 \quad (A.4)$$

Eq. (A.3) holds for $i \in \{1, \ldots, m\}$ and Eq. (A.4) for any $j > i$. From Eq. (A.4), we can see that the difference between both clusterings remain constant for any thinner partition $\mathcal{P}_j$. A consequence of this is that, if we take $\mathcal{P}_j$ as a partition thin enough such that there is only one instance per subset, then adding the difference clustering error (associated to both centroids) for the partitions $\mathcal{P}_j$ and $\mathcal{P}_i$, we have the following relation over the error for the entire dataset (observe that the following relation holds in general for any partition thinner than $\mathcal{P}_i$):

$$E(C_i) \leq E(C_{i-1}) \Longleftrightarrow E(\mathcal{G}^{i-1}_{l_{i-1}-1}) - E(C_{i-1}) \leq \xi_i + (E(\mathcal{G}^i_{l_i-1}) - E(C_i))$$

□

Lemma 3. *At the ith step of the RPKM, if $\mathcal{G}^i_r = \mathcal{G}^j_s$, with $j < i$, for some $r \in \{1, \ldots, l_i - 1\}$ and $s \in \{1, \ldots, l_j - 1\}$, then $l_{j+1} = \ldots = l_i = 1$. Moreover, in that case, $s = l_j - 1$.*

Proof. Using the chain of inequalities (A.2), we observe that the following inequalities hold at the first iteration of the RPKM:

$$E_{\mathcal{P}_1}(C^1_0 = C_0) > E_{\mathcal{P}_1}(\mathcal{G}^1_0) > E_{\mathcal{P}_1}(C^1_1) > E_{\mathcal{P}_1}(\mathcal{G}^1_1) > E_{\mathcal{P}_1}(C^1_2) > \ldots > E_{\mathcal{P}_1}(\mathcal{G}^1_{a_1-2}) \geq E_{\mathcal{P}_1}(C^1_{l_1-1}) \geq E_{\mathcal{P}_1}(\mathcal{G}^1_{l_1-1}) \geq E_{\mathcal{P}_1}(C_1 = C^1_{l_1}) \quad (A.5)$$

Analogously, for the subsequent ith iteration of the RPKM algorithm, we obtain the following set of inequalities

$$E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{a_{i-1}-1}) > E_{\mathcal{P}_i}(C^i_0 = C_{i-1}) > E_{\mathcal{P}_i}(\mathcal{G}^i_0) > E_{\mathcal{P}_i}(C^i_1) > E_{\mathcal{P}_i}(\mathcal{G}^i_1) > E_{\mathcal{P}_i}(C^i_2) > \ldots > E_{\mathcal{P}_i}(\mathcal{G}^i_{l_i-2}) \geq E_{\mathcal{P}_i}(C^i_{l_i-1}) \geq E_{\mathcal{P}_i}(\mathcal{G}^i_{l_i-1}) \geq E_{\mathcal{P}_i}(C_i = C^i_{l_i}), \ i \in \{2, \ldots, m\} \quad (A.6)$$

First of all, observe that the error associated to all the clusters generated at the jth RPKM iteration keep the same ordering for the error associated to a thinner partition $\mathcal{P}_i$. In particular, we have $E_{\mathcal{P}_i}(\mathcal{G}^j_{l_j-1}) \leq E_{\mathcal{P}_i}(\mathcal{G}^j_s)$ for $s < l_j - 1$. To verify this we make use of Lemma 2, from which we know that $E_{\mathcal{P}_i}(\mathcal{G}^j_{l_j-1}) - E_{\mathcal{P}_i}(\mathcal{G}^j_s) = E_{\mathcal{P}_j}(\mathcal{G}^j_{l_j-1}) - E_{\mathcal{P}_j}(\mathcal{G}^j_s) \leq 0 \rightarrow E_{\mathcal{P}_i}(\mathcal{G}^j_{l_j-1}) \leq E_{\mathcal{P}_i}(\mathcal{G}^j_s)$ for $s < l_j - 1$. This means that, the last clustering found at the jth RPKM iteration also has the smallest error, with the partition $\mathcal{P}_i$, with respect to the previous clusters obtained at the jth RPKM iteration.

From the chain of inequalities (A.5) and (A.6), we know that

$$E_{\mathcal{P}_h}(\mathcal{G}^{h-1}_{l_{h-1}-1}) \geq E_{\mathcal{P}_h}(\mathcal{G}^h_{l_h-1}) \ \forall h \in \{j+1, \ldots, i-1\}, \quad (A.7)$$

where, if the equality holds, then $a_h = 1$. From Lemma 2, (A.7) implies

$$E_{\mathcal{P}_i}(\mathcal{G}^j_{l_j-1}) \geq E_{\mathcal{P}_i}(\mathcal{G}^{j+1}_{l_{j+1}-1}) \geq \ldots \geq E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) \quad (A.8)$$

In other words, the error with respect to the current partition (or any thinner partition) of the optimal patterns obtained at each RPKM iteration decreases monotonically.

By *reductio ad absurdum*, if we assume that $\mathcal{G}^i_r = \mathcal{G}^j_s$, for some $r \in \{1, \ldots, l_i - 1\}$ and $s \in \{1, \ldots, l_j - 1\}$ and that there exists $j < h < i$ such that $l_h > 1$, then $E_{\mathcal{P}_i}(\mathcal{G}^j_{l_j-1}) > E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) \Rightarrow E_{\mathcal{P}_i}(\mathcal{G}^j_s) > E_{\mathcal{P}_i}(\mathcal{G}^i_r) = E_{\mathcal{P}_i}(\mathcal{G}^j_s)$ $(\Rightarrow\Leftarrow)$. Therefore, from now on we assume $l_{j+1} = \ldots = l_{i-1} = 1$.

Analogously, if we assume that $\mathcal{G}^i_r = \mathcal{G}^j_s$, for some $r \in \{1, \ldots, l_i - 1\}$ and $s \in \{1, \ldots, l_j - 1\}$ and that $l_i > 1$, then $E_{\mathcal{P}_i}(\mathcal{G}^j_{l_j-1}) \geq E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) > E_{\mathcal{P}_i}(\mathcal{G}^i_r) = E_{\mathcal{P}_i}(\mathcal{G}^j_s) \Rightarrow E_{\mathcal{P}_i}(\mathcal{G}^j_s) > E_{\mathcal{P}_i}(\mathcal{G}^j_s)$ $(\Rightarrow\Leftarrow)$.

In the case that $l_{j+1} = \ldots = l_i = 1$, the error associated to $\mathcal{G}^i_r$ satisfies $E_{\mathcal{P}_i}(\mathcal{G}^j_s) \geq E_{\mathcal{P}_i}(\mathcal{G}^j_{l_j-1}) \geq E_{\mathcal{P}_i}(\mathcal{G}^{i-1}_{l_{i-1}-1}) \geq E_{\mathcal{P}_i}(\mathcal{G}^i_r)$, hence the only possible choice is $s = l_j$. □

Theorem 1. *[Upper bound to the number of local WL iterations] An upper bound to the number of Lloyd iterations at the ith RPKM step is given by $l_i \leq \{^{|\mathcal{P}_i|}_K\} - \sum_{j=1}^{i-1}(l_j - 1)$, where $\{^{|\mathcal{P}_i|}_K\}$ is a Stirling number of the second kind.*

Proof. Lemma 3 implies that, at each RPKM iteration, no previous clustering can be repeated (except the last one from the clustering sequence). Therefore, we can eliminate, at least, all clusters generated at the previous RPKM iterations except the last one ($\sum_{j=1}^{i-1}(l_j - 1)$). Moreover, the number of different partitions for $|\mathcal{P}_i|$ observations into K groups is a Stirling number of the second kind, $\{^{|\mathcal{P}_i|}_K\}$ [19], then $l_i \leq \{^{|\mathcal{P}_i|}_K\} - \sum_{j=1}^{i-1}(l_j - 1)$. □

References

[1] D. Aloise, A. Deshpande, P. Hansen, Popat p.: NP-hardness of euclidean sum-of-squares clustering, Mach. Learn. 75 (2009) 245–249.
[2] D. Arthur, S. Vassilvitskii, *k*-means++: the advantages of careful seeding, in: Proceedings of the 18th annual ACM-SIAM Symp. on Disc. Alg, 2007, pp. 1027–1035.
[3] C. Boutsidis, M.W. Mahoney, P. Drineas, Unsupervised feature selection for the *k*-means clustering problem, in: NIPS, 2009, pp. 153–161.
[4] C. Boutsidis, A. Zouzias, P. Drineas, Random projections for *k*-means clustering, in: NIPS, 2010, pp. 298–306.
[5] P.S. Bradley, U.M. Fayyad, Refining initial points for *k*-means clustering, ICML 98 (1998) 91–99.
[6] M.B. Cohen, S. Elder, C. Musco, C. Musco, M. Persu, Dimensionality reduction for *k*-means clustering and low rank approximation, in: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, 2015, pp. 163–172.
[7] I. Davidson, A. Satyanarayana, Speeding up *k*-means clustering by bootstrap averaging, IEEE data mining workshop on clustering large data sets, 2003.
[8] V. Gandhi, J.M. Kang, S. Shekhar, Spatial Databases, Wiley Encyclopedia of Computer Science and Engineering, 2008.
[9] R. Finkel, L. Bentley, Quad trees a data structure for retrieval on composite keys, Acta Inform. 4 (1) (1974) 1–9.
[10] J. Fonollosa, S. Sheik, R. Huerta, S. Marco, Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring, Sens. Actuators B 215 (2015) 618–629.
[11] E. Forgy, Cluster analysis of multivariate data: efficiency vs. interpretability of classifications, Biometrics 21 (1965) 768–769.
[12] Committee on the Analysis of Massive Data, Committee on Applied and Theoretical Statistics, Board on Mathematical Sciences and Their Applications, Division on Engineering and Physical Sciences, National Research Council: Frontiers in Massive Data Analysis, The National Academy Press, 2013.
[13] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[14] R. Dubes, A. Jain, Algorithms for Clustering Data, Prentice Hall, Inc., 1988.
[15] M. Hung, J. Wu, J. Chang, D. Yang, An efficient *k*-means clustering algorithm using simple partitioning, J. Inf. Sci. Eng. 21 (2005) 1157–1177.

[16] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (1999) 264–323.
[17] A.K. Jain, Data clustering: 50 years beyond *k*-means, Pattern Recog. Lett. 31 (8) (2010) 651–666.
[18] L. Kaufman, P. Rousseeuw, Clustering by Means of Medoids, North-Holland, 1987.
[19] T. Karkkainen, S. Ayramo, Introduction to partitioning-based clustering methods with a robust example, Reports of the Department of Mathematical Information Technology Series C. Software and Computational Engineering No. C. 1/2006, 2006. ISBN 951392467X, ISSN 14564378.
[20] S.P. Lloyd, Least squares quantization in PCM, IEEE trans, Inf. Theory 28 (1982) 129–137.
[21] H.S. Park, J. Chi-Hyuck, A simple and fast algorithm for *k*-medoids clustering, Exp. Syst. Appl. 36 (2) (2009) 3336–3341.
[22] J.M. Peña, J.A. Lozano, P. Larrañaga, An empirical comparison of four initialization methods for the *k*-means algorithm, Pattern Recog. Lett. 20 (10) (1999) 1027–1040.
[23] S. Redmond, C. Heneghan, A method for initialising the *k*-means clustering algorithm using kd-trees, J. Pattern Recog. Lett. 28 (8) (2007) 965–973.
[24] D. Sculley, Web-scale *k*-means clustering, in: Proceedings of the 19th international conference on World wide web, 2010, pp. 1177–1178.
[25] A. Vattani, *k*-means requires exponentially many iterations even in the plane, Disc. Comput. Geom. 45 (4) (2011) 596–616.
[26] X. Wu, V. Kumar, J. Ross, J. Ghosh, Q. Yang, H. Motoda, J. McLachlan, A. Ng, B. Liu, P. Yu, Z. Zhou, M. Steinbach, D. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (2007) 1–37.
[27] W. Zhao, H. Ma, Q. He, Parallel k-means clustering based on mapreduce, Cloud Comput. Lect. Notes Comput. Sci. 5931 (2009) 674–679.