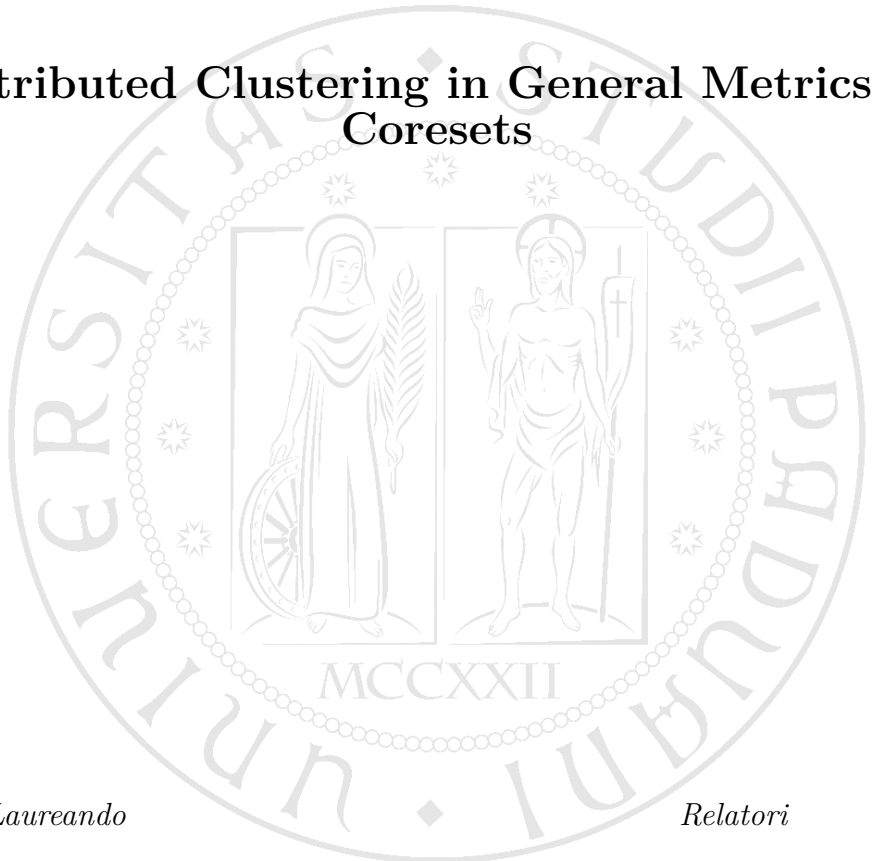
**Università degli Studi di Padova**

Dipartimento di ingegneria dell'informazione

*Corso di laurea magistrale in ingegneria informatica*

# Distributed Clustering in General Metrics via Coresets



*Laureando*

**Alessio Mazzetto**

*Matricola*

**1177943**

*Relatori*

**Prof. Andrea Pietracaprina**

**Prof. Geppino Pucci**

Anno Accademico 2018-2019

# Abstract

Center-based clustering is a fundamental primitive for data analysis and becomes very challenging for large datasets. In this thesis, we focus on the popular $k$-center, $k$-median, and $k$-means variants which, given a set $P$ of points from a metric space and a parameter $k < |P|$, require to identify a set $S$ of $k$ centers minimizing respectively the maximum distance, the sum of the distances, and the sum of the squared distances of all points in $P$ from their closest centers. Our specific focus is on general metric spaces for which it is reasonable to require that the centers belong to the input set (i.e., $S \subseteq P$). We present a general coreset construction primitive which allows us to design coreset-based 2-round distributed approximation algorithms for the above problems using the MapReduce computational model. The algorithms are rather simple and obliviously adapt to the intrinsic complexity of the dataset, captured by the doubling dimension $D$ of the metric space. Remarkably, the algorithms attain approximation ratios that can be made arbitrarily close to those achievable by the best known polynomial-time sequential approximations, and they are very space efficient for small $D$, requiring local memory sizes substantially sublinear in the input size. While in the literature similar results were already achieved for the $k$-center problem, to the best of our knowledge, no previous distributed approaches were able to attain similar quality-performance guarantees for $k$-median and $k$-means in general metric spaces. Moreover, we address the NP-hard computational problem of the estimation of the doubling dimension of a set of points with two novel methods: a sequential algorithm, whose performance is slightly better than known approaches, and a 2-round MapReduce algorithm, which is more suitable to tackle large datasets, but yields a poorer approximation.

# Table of contents

# Chapter 1

# Introduction

## 1.1 Introduction

Clustering is a fundamental primitive in the realms of data management and machine learning, with applications in a large spectrum of domains such as database search, bioinformatics, pattern recognition, networking, operations research, and many more [18]. A prominent clustering subspecies is *center-based clustering*, whose goal is to partition a set of data items into $k$ groups, where $k$ is an input parameter, according to a notion of similarity, captured by a given measure of closeness to suitably chosen representatives, called centers. There is a vast and well-established literature on sequential strategies for different instantiations of center-based clustering [3]. However, the explosive growth of data that needs to be processed often rules out the use of these sequential strategies, which are often impractical on large data sets, due to their time and space requirements. Therefore, it is of paramount importance to devise efficient distributed clustering strategies tailored to the typical computational frameworks for big data processing, such as MapReduce [26].

In this thesis, we focus on the *$k$-center*, *$k$-median*, and *$k$-means* clustering problems. Given a set $P$ of points in a general metric space and a positive integer $k \leq |P|$, the $k$-center problem requires to find a subset $S \subseteq P$ of $k$ points, called *centers*, so that the maximum distance between a point of $P$ to its closest center is minimized. In the $k$-median problem, we want to find the set $S \subseteq P$ which minimizes the sum of the distances between the points of $P$ to their respective closest center; similarly, in the $k$-means problem, we seek to minimize the sum of the squared distances. Once $S$ is determined, the association of each point to the closest center naturally defines a clustering of $P$. While scarcely meaningful for general metric spaces, for Euclidean spaces the widely studied *continuous* variant of the $k$-center, $k$-median, and $k$-means

problems removes the constraint that $S$ is a subset of $P$, hence allowing a much richer choice of centers from the entire space. These clustering problems are the most popular instantiations of center-based clustering, whose efficient solution in the realm of big data has attracted vast attention in the recent literature [10, 5, 6, 32, 7]. One of the reference models for big data computing, also adopted in most of the aforementioned works, is MapReduce [9, 29, 26], where a set of processors with limited-size local memories process data in a sequence of parallel rounds. Efficient MapReduce algorithms should aim at minimizing the number of rounds while using substantially sublinear local memory.

A natural approach to solving large instances of combinatorial optimization problems relies on the extraction of a much smaller "summary" of the input instance, often dubbed *coreset* in the literature [16], which embodies sufficient information to enable the computation of a good approximate solution of the whole input. This approach is profitable whenever the (time and space) resources needed to compute the coreset are considerably lower than those required to compute a solution by working directly on the input instance. Coresets with different properties have been studied in the literature to solve different variants of the aforementioned clustering problems [28].

For the Euclidean Space $\mathbb{R}^d$, the value $d$ describes the "complexity" or "dimensionality" of a space. Likewise, in the case of a general metric space, a widely adopted notion to measure its complexity is its *doubling dimension*. The concept of doubling dimension naturally extends to characterize any subset of points of a metric space, as the subset can be seen as a metric space on its own. While the doubling dimension of the input data is generally unknown, the design of algorithms whose performance is parametrized by this dimensionality has gathered momentum in recent years, as they generally feature unmatched performance for low doubling dimensions [7, 17].

This thesis features novel coreset-based space/round-efficient MapReduce algorithms for $k$-center, $k$-median, and $k$-means. In addition, it presents new sequential and MapReduce algorithms to estimate the doubling dimension of an input set of points.

## 1.2 Related work

The $k$-center, $k$-median, and $k$-means clustering problems in general metric spaces have been extensively studied, and constant approximation algorithms are known for these NP-hard problems [3, 12]. Lately, there has been growing interest towards the development of distributed algorithms to attack these problems in the big data scenario (see [32, 7] and references therein). While straightforward parallelizations of known

iterative sequential strategies tend to be inefficient due to high round complexity, the most relevant efforts to date rely on distributed constructions of coresets of size much smaller than the input, upon which a sequential algorithm is then run to obtain the final solution.

For $k$-center, in [10] it is presented a randomized MapReduce algorithm which provides a 10-approximation. This result is improved in the work in [27], which features a deterministic 2-round 4-approximation MapReduce algorithm with local memory $O\left(\sqrt{|P|k}\right)$. Lately, Ceccarello et al. [7] have presented a deterministic 2-round $(2+\epsilon)$-approximation algorithm with local memory $O\left(\sqrt{|P|k}(16/\epsilon)^{\mathtt{dd}(P)}\right)$, where $\epsilon \in (0,1)$ is a user chosen parameter which represents the accuracy-space tradeoff, and $\mathtt{dd}(P)$ is the doubling dimension of the input dataset.

For $k$-median, Ene et al. [10] present a randomized MapReduce algorithm which computes a coreset of size $O(k^2|P|^\delta)$ in $O(1/\delta)$ rounds, for any $\delta < 1$. By using a $\alpha$-approximation algorithm on this coreset, a weak $(10\alpha+3)$-approximate solution is obtained. In the paper, the authors claim that their approach extends also to the $k$-means problem, but do not provide the analysis. For this latter problem, in [5] a parallelization of the popular $k$-means++ algorithm by [1] is presented, which builds an $O(k\log|P|)$-size coreset for $k$-means in $O(\log|P|)$ rounds. By running an $\alpha$-approximation algorithm on the coreset, the returned solution features an $O(\alpha)$ approximation ratio. A randomized MapReduce algorithm for $k$-median has been recently presented in [32], where the well known local-search PAM algorithm [23] is employed to extract a small family of possible solutions from random samples of the input. A suitable refinement of the best solution in the family is then returned. While extensive experiments support the effectiveness of this approach in practice, no tight theoretical analysis of the resulting approximation quality is provided.

In the continuous setting, Balcan et al. [6] present randomized 2-round algorithms to build coresets in $\mathbb{R}^d$ of size $O\left(\frac{kd}{\epsilon^2}+Lk\right)$ for $k$-median, and $O\left(\frac{kd}{\epsilon^4}+Lk\log(Lk)\right)$ for $k$-means, for any choice of $\epsilon \in (0,1)$, where the computation is distributed among $L$ processing elements. By using an $\alpha$-approximation algorithm on the coresets, the overall approximation factor is $\alpha+O(\epsilon)$. For $k$-means, a recent improved construction yields a coreset which is a factor $O(\epsilon^2)$ smaller and features very fast distributed implementation [4]. It is not difficult to show that a straightforward adaptation of these algorithms to general spaces (hence in a non-continuous setting) would yield $(c\cdot\alpha+O(\epsilon))$-approximations, with $c \geq 2$, thus introducing a non-negligible gap with respect to the quality of the best sequential approximations.

The problem of determining the doubling dimension $\mathtt{dd}(P)$ of an input set of points $P$ has already been studied by the literature, and proved to be NP-hard [13]. Also in [13], it is presented a 2-approximation algorithm to estimate $\mathtt{dd}(P)$ which has time complexity $O\left(|P|^3 2^{O(\mathtt{dd}(P))}\right)$. This algorithm is based on the point hierarchy in [24], which does not provide a precise bound on the term $O(\mathtt{dd}(P))$ of the time complexity. A faster algorithm with time complexity $O\left(2^{O(\mathtt{dd}(P))}|P|\log|P|\right)$ is known [17], although it has worse $O(1)$-approximation factor.

Finally, it is worth mentioning that there is a rich literature on sequential coreset constructions for $k$-median and $k$-means which mostly focus on the continuous case in Euclidean spaces [11, 16, 15, 19, 31, 8]. We do not review the results in these works since our focus is on distributed algorithms in general metric sapces.

## 1.3 Contribution

We devise new distributed coreset constructions and show how to employ them to yield accurate space-efficient 2-round MapReduce algorithms for $k$-center, $k$-median and $k$-means. Our coresets are built in a *composable* fashion [20] in the sense that they are obtained as the union of small local coresets computed in parallel (in a single MapReduce round) on distinct subsets of a partition of the input. The final solution is obtained by running a sequential approximation algorithm on the coreset in a second MapReduce round. The memory requirements of our algorithms are analyzed in terms of the desired approximation guarantee, and of the doubling dimension $\mathtt{dd}(P)$ of the metric space induced by the input set of points $P$, a parameter which generalizes the dimensionality of Euclidean spaces to general metric spaces and is thus related to the increasing difficulty of spotting good clusterings as the parameter $\mathtt{dd}(P)$ grows.

Let $\alpha$ denote the best approximation ratio attainable by a sequential algorithm for either $k$-median or $k$-means on general metric spaces. Our main results are 2-round $(\alpha + O(\epsilon))$-approximation MapReduce algorithms for $k$-median and $k$-means, which require $O\left(\sqrt{|P|k}(c/\epsilon)^{2\mathtt{dd}(P)}\log^2|P|\right)$ local memory, where $c > 0$ is a suitable constant that will be specified in the analysis, and $\epsilon \in (0,1)$ is a user-defined precision parameter. To the best of our knowledge, these are the first MapReduce algorithms for $k$-median and $k$-means in general metric spaces which feature approximation guarantees that can be made arbitrarily close to those of the best sequential algorithms, and run in few rounds using local space substantially sublinear for low-dimensional spaces. In fact, prior to our work existing MapReduce algorithms for $k$-median and $k$-means in general

metric spaces either exhibited approximation factors much larger than $\alpha$ [10, 5], or missed a tight theoretical analysis of the approximation factor [32].

For $k$-center, we develop a 2-round $(2+\epsilon)$-approximation MapReduce algorithm which requires $O\left(\sqrt{|P|k}(16/\epsilon)^{\mathtt{dd}(P)}\right)$ local memory. For comparison, the best sequential algoritm achieves a 2-approximation, and the problem is NP-hard to approximate within a smaller factor. Even if similar results performance-wise were already obtained in previous works [7], this result substantiates the generality of our method.

Our algorithms revolve around novel coreset constructions somehow inspired by those proposed in [16] for Euclidean spaces. As a fundamental tool, the constructions make use of a procedure that, starting from a set of points $P$ and a set of centers $C$, produces a (not much) larger set $C'$ such that for any point $x \in P$ its distance from $C'$ is significantly smaller than its distance from $C$. Simpler versions of our constructions can also be employed to attain 2-round MapReduce algorithms for the continuous versions of $k$-median and $k$-means, featuring $\alpha+O(\epsilon)$ approximation ratios and $O\left(\sqrt{|P|k}(c/\epsilon)^{\mathtt{dd}(P)}\log|P|\right)$ local space requirements. While similar approximation guarantees have already been achieved in the literature [6, 4] using even smaller local space, this result provides further evidence of the general applicability of our novel approach.

We want to point out that a very desirable feature of our MapReduce algorithms is that they do not require a priori knowledge of the doubling dimension $\mathtt{dd}(P)$ and, in fact, they naturally adapt to the dimensionality of the dataset.

Finally, we present novel sequential and MapReduce algorithms to estimate the doubling dimension $\mathtt{dd}(P)$ of a set of points $P$. In particular, we devise a sequential 2-approximation algorithm which slightly improves the time complexity to $O\left(|P|^3 2^{2\mathtt{dd}(P)}\right)$ from the previously best-known [13] complexity $O\left(|P|^3 2^{O(\mathtt{dd}(P))}\right)$. The super-cubic complexity of the algorithm makes it unusable for large sets of points. In order to be able to tackle larger datasets, we exhibit a novel 2-round MapReduce algorithm with local memory $O(|P|/L+L)$, which returns an estimate $D$ of the doubling dimension such that $\mathtt{dd}(P) \leq D \leq 8 \cdot \mathtt{dd}(P) + \log_2 L$, where $L$ is a user-defined parameter. As far as we know, no previous work has ever tackled the distributed computation of the doubling dimension for large set of points.

## 1.4 MapReduce

Many algorithms presented in this thesis are designed for the *MapReduce* model of computation which has become a de facto standard for big data algorithmics in recent

years. A MapReduce algorithm [9, 29, 26] executes in a sequence of parallel *rounds*. In a round, a multiset $X$ of key-value pairs is first transformed into a new multiset $X'$ of key-value pairs by applying a given *map function* (simply called *mapper*) to each individual pair, and then into a final multiset $Y$ of pairs by applying a given *reduce function* (simply called *reducer*) independently to each subset of pairs of $X'$ having the same key. The model features two parameters, $M_L$, the *local memory* available to each mapper/reducer, and $M_A$, the *aggregate memory* across all mappers/reducers. In our algorithms, mappers are straightforward constant-space transformations, thus the memory requirements will be related to the reducers.

## 1.5 Organization of the thesis

The rest of the thesis is organized as follows. Chapter 2 introduces some preliminary concepts, formally introduces the notion of doubling dimension and investigate some of its properties, and presents two algorithms which can be used to estimate this dimensionality. Chapter 3 describes the $k$-center, $k$-median, and $k$-means clustering problems, introduces various properties of coresets that are needed to achieve our results, and presents our novel coreset constructions primitive. Based on the latter, Chapter 4 derives the MapReduce algorithms for the three clustering problems. Finally, Chapter 5 offers some concluding remarks and briefly discusses possible future work.

# Chapter 2

# Doubling Dimension

The doubling dimension is a notion that captures the intrinsic complexity of a generic metric space. In our work, we will develop coreset construction algorithms to efficiently solve the $k$-center, $k$-median, and $k$-means clustering problems which are sensitive to the dimensionality of the space. We remark that the task of spotting a small coreset which "summarizes" the structure of the input set of points can become arbitrarily hard for generic metric spaces. For the sake of explaination, consider a finite set $P$ such that any two different points in $P$ are at distance 1. Any subset of $P$ (i.e. any candidate coreset) gives no information on the structure or properties of the other points, as they all have the same distance 1 from the subset. As such, it is inherently hard to develop an algorithm that can extract a small-sized and meaningful coreset from this specific $P$.

It turns out that the concept of doubling dimension is very convenient to determine how succint a summary (to be used for clustering) we may expect to extract from a pointset. In this context, the concept of doubling dimension turns out to be very convenient. In our coreset construction algorithm, we exploit the properties expressed by the definition of this dimensionality, so that the output coreset size is a function of the (possibly unknown) doubling dimension $\mathtt{dd}(P)$ of the input $P$. The idea is that we can extract coresets of small size as far as $\mathtt{dd}(P)$ is low. Note that this algorithm doesn't aim to be efficient for any input. Consider again the set $P$ of equi-distant points. As we will see later in this Chapter, this set has a doubling dimension $\mathtt{dd}(P) = \log_2 |P|$, which is the highest possible, and so our coreset construction algorithm will possibly return a very large-sized coreset. However, this is not really a problem, as the high doubling dimension also implies that the set of points is indeed hard to cluster. In fact, with reference again to the previous example, there is no meaningful choice of $k$ centers from $P$. In other words, a large doubling dimension value indicates that

it is difficult to find significant representatives, whether they are cluster centers or points of a coreset. If our coreset construction algorithm fails to return a small coreset because of the dimensionality of the input, then it is also improbable that a meaningful center-based clustering exists.

Although our coreset construction is oblivious to the value of the doubling dimension, the problem of estimating the value of this dimensionality is undoubtedly interesting. The ability to systematically and efficiently estimate the doubling dimension of real datasets would make it possible to give concrete evidence of the effectiveness of this type of approach.

In this chapter, after introducing some preliminary concepts, we provide a formal definition of doubling dimension and an overview of its properties. Then, we present two novel algorithms for estimating the doubling dimension.

## 2.1 Preliminaries

A metric space is an ordered pair $(\mathcal{M}, d)$, where $\mathcal{M}$ is a set and $d$ is a metric. A metric is a function $d : \mathcal{M} \times \mathcal{M} \to \mathcal{R}$ such that for any $x, y, z \in \mathcal{M}$, the following properties hold: $(i)$ $d(x,y) = 0 \iff x = y$ (identity); $(ii)$ $d(x,y) = d(y,x)$ (simmetry); $(iii)$ $d(x,y) \geq 0$ (non-negativity); and $(iv)$ $d(x,z) \leq d(x,y) + d(y,z)$ (triangle inequality). The function $d$ is also called distance. Throughout this thesis, input pointsets will always consist of points which share the same distance function $d$. We introduce the two following useful notations. For any sets of points $P$,$Q$, and $x \in P$, we define $d(x,Y) := \min_{y \in Y} d(x,y)$ and $x^Y := \arg\min_{y \in Y} d(x,y)$. Note that $d(x,Y) = d(x,x^Y)$.

In what follows, we will also consider weighted sets of points. In a weighted set, every point is assigned a non-negative real value according to a weight function $w$. Formally, given a set of points $P$, a function $w : P \to \mathbb{R}$ is said to be a weight function for $P$ if it is non-negative. We will use the notation $P_w$ to denote a weighted set of points, where $w$ is its weight function. The computation of the distance of two points is not affected by their weights. Note that an unweighted set of points $P$ can be considered weighted with unitary weights, that is, its implicit weight function is $w(p) = 1$, $\forall x \in P$.

## 2.2 Definition of doubling space

The doubling dimension is a value which can be used to describe a specific property of a set of points. To give a formal definition, we first need to introduce some preliminary

notation. Let $P$ be a set of points. Given $r \geq 0$ and a point $x$, we define the *ball of radius $r$ centered at $x$*, $\texttt{ball}_P(x,r) := \{y \in P : d(x,y) \leq r\}$. That is, $\texttt{ball}_P(x,r)$ is the subset of $P$ at distance at most $r$ from $x$. A ball that contains a set of points is said to *cover* those points. A set of points $C$ is said to be an $r$-cover of $P$ if $P \subseteq \cup_{c \in C} \texttt{ball}_P(c,r)$. Given a set of points $P$, we define $\texttt{m-cover}(P,r)$ to be the set of points $T \subseteq P$ of minimum cardinality such that $T$ is an $r$-cover of $P$.

▶ **Definition 2.2.1.** *Let $P$ be a set of points. Its doubling dimension $\texttt{dd}(P)$ is defined as*

$$\texttt{dd}(P) = \sup_{x \in P, r \geq 0} \log_2 |\texttt{m-cover}(\texttt{ball}_P(x,r), \frac{r}{2})|$$

An alternative and common way to describe the doubling dimension of a set of points is the following: any ball of radius $r$ centered in a point $x \in P$ has an $r/2$-cover $C \subseteq P$ of cardinality at most $2^{dd(P)}$.

We observe that a set infinite pointset can have a finite doubling dimension. For example, it is easy to show that $\texttt{dd}(\mathbb{R}) \leq 1$. In fact, for any $x \in \mathbb{R}$ and $r \geq 0$, the set $\{x + r/2, x - r/2\}$ is a $r/2$-cover of $\texttt{ball}_P(x,r)$. However, there exist sets of points which have infinite doubling dimension. For example, consider a set of infinite points $P$ such that for any $x, y \in P$, $x \neq y$, $d(x,y) = 1$, that is any point in $P$ is at the same distance from all the others. In this case, for $r = 1$ and for any $x \in P$, we have that $\texttt{ball}_P(x,1) = P$. Clearly, a $1/2$-cover of $P$ must contain all points in $P$, and it is thus infinite. Indeed, assume that it exists a set $C$ which is a $1/2$-cover of $P$ and does not contain a point $x \in P$. This implies that it exists a point $y \in C$, $y \neq x$, such that $d(x,y) \leq 1/2$, which yields a contradiction.

In the examples above, we have shown that a set with an infinite number of points can have either a finite or infinite doubling dimension. The following result formally states that if a set of points is finite, its doubling dimension cannot be larger that the base-2 logarithm of its cardinality.

▶ **Lemma 2.2.2.** *Let $P$ be a set of points such that $|P| < \infty$. Then, $\texttt{dd}(P) \leq \log_2 |P|$.*

**Proof.** For any $x \in P$ and $r \geq 0$, we have that $|\texttt{m-cover}(\texttt{ball}_P(x,r), \frac{r}{2})| \leq |P|$, as $\texttt{ball}_P(x,r)$ is a $r/2$-cover of $\texttt{ball}_P(x,r)$, and $|\texttt{ball}_P(x,r)| \leq |P|$. The statement immediately follows. ◀

The bound of the previous lemma is strict, as shown by the next example. Consider again a set $P$ such that for any $x, y \in P$, $x \neq y$, $d(x,y) = 1$, however let $P$ have finite cardinality this time. Arguing as in the previous case, for any $x \in P$, we have that any

$1/2$-cover of $\texttt{ball}_P(x,1) = P$ must contain all points in $P$. Hence, $\texttt{dd}(P) \geq \log_2 |P|$. Since by Lemma 2.2.2, $\texttt{dd}(P) \leq \log_2 |P|$, we have that $\texttt{dd}(P) = \log_2 |P|$ for this particular $P$.

The definition of doubling dimension can be recursively applied in order to describe the number of points required at most to have an $r/2^i$-cover of any ball of radius $r$ centered at a point of $P$, for any integer $i \geq 1$.

▶ **Lemma 2.2.3.** *Let $P$ be a set of points. For any $r \geq 0$, $x \in P$, and integer $i \geq 1$, there exists a set $C$ such that $C$ is an $r/2^i$-cover of $\texttt{ball}_P(x,r)$ and $|C| \leq 2^{i \cdot \texttt{dd}(P)}$.*

**Proof.** Consider the tree of height $i$ constructed as follows. Any node is labelled by a real value, denoting the radius, and a point. The root is labeled by $r$ and $x$. For any internal node $u$, let $S$ denote the set of its children. If $u$ is labelled by the value of the radius $r$, then any $v \in S$ is labelled by radius $r/2$. Moreover, the union of the point labels of $S$ must be a minimum cardinality $r/2$-cover of $\texttt{ball}_P(y,r)$, where $y$ is the point label of $u$. Then, by the definition of doubling dimension, every internal node has at most $2^{\texttt{dd}(P)}$ children. By induction, it is easy to prove that the leaves are labeled with radius $r/2^i$. Let $L$ denote the union of the point labels of the leaves. By induction, we can easily show that $\texttt{ball}_P(x,r) \subseteq \cup_{y \in L} \texttt{ball}_P(y, r/2^i)$. Hence $L$ is a $r/2^i$-cover of $\texttt{ball}_P(x,r)$. Since every internal node can have at most $2^{\texttt{dd}(P)}$ children, and the tree has height $i$, the maximum number of leaves is $(2^{\texttt{dd}(P)})^i$. ◀

It is of theorethical interest to study the properties of the doubling dimension when partitioning sets of points. We will extensively use the properties featured by the following lemmas throughout this and next chapters.

▶ **Lemma 2.2.4.** *Let $S \subseteq P$. If $\texttt{dd}(P) < \infty$, then $\texttt{dd}(S) \leq 2 \cdot \texttt{dd}(P)$.*

**Proof.** Fix arbitrarily $x \in S$ and $r \geq 0$. By Lemma 2.2.3, there exists a set $C \subseteq P$ such that $C$ is an $r/4$-cover of $\texttt{ball}_P(x,r)$ and $|C| \leq 2^{2\texttt{dd}(P)}$. We now construct a set $C'$ as follows. For any $c \in C$, consider $\texttt{ball}_P(c, r/4)$. There are two cases. In the first case, $\texttt{ball}_P(c,r/4) \cap S$ is empty, and we don't add any point to $C'$. In the second case, there exists $y \in \texttt{ball}_P(c,r/4) \cap S$, and we add $y$ to $C'$. It's easy to see that $\texttt{ball}_P(c,r/4) \subseteq \texttt{ball}_P(y,r/2)$. We claim that $C'$ is an $r/2$-cover of $\texttt{ball}_S(x,r)$. In fact, $\texttt{ball}_S(x,r) \subseteq \cup_{c \in C} \texttt{ball}_S(c,r/4) \subseteq \cup_{c \in C'} \texttt{ball}_S(c,r/2)$, where the last relation is due to how $C'$ is constructed. Since for any point in $C$, we add at most one point to $C'$, it holds that $|C'| \leq |C| \leq 2^{2\texttt{dd}(P)}$. It immediately follows that $\texttt{dd}(S) \leq 2 \cdot \texttt{dd}(P)$. ◀

The following proposition is convenient to prove the next lemma.

▶ **Proposition 2.2.5.** *Consider a point $x \notin P$ and a radius $r \geq 0$. Then, there exists a set $C \subseteq P$ such that $C$ is an $r/2$-cover of $\texttt{ball}_P(x,r)$ and $|C| \leq 2^{2\,\texttt{dd}(P)}$.*

**Proof.** Consider an arbitrary point $y \in \texttt{ball}_P(x,r) \subseteq P$. The following relation surely holds: $\texttt{ball}_P(x,r) \subseteq \texttt{ball}_P(y,2r)$. In fact, for any $z \in \texttt{ball}_P(x,r)$, we have that $d(z,y) \leq d(z,x) + d(x,y) \leq 2r$, so $z \in \texttt{ball}_P(y,2r)$. By Lemma 2.2.3, there exists a set $C$ such that $C$ is a $(2r)/4$-cover of $\texttt{ball}_P(y,2r)$ and $|C| \leq 2^{2\texttt{dd}(P)}$. As $\texttt{ball}_P(x,r) \subseteq \texttt{ball}_P(y,2r)$, the set $C$ is also an $r/2$-cover of $\texttt{ball}_P(x,r)$, and the statement immediately follows. ◀

▶ **Lemma 2.2.6.** *Let $P_1, ..., P_L$ be a partition of $P$ and $c \geq 2$. If $\texttt{dd}(P) < \infty$, then*

$$\texttt{dd}(P) \leq \log_2 \sum_{\ell=1}^{L} 2^{c \cdot \texttt{dd}(P_\ell)} \leq \log_2 L + 2c \cdot \texttt{dd}(P)$$

**Proof.** The second inequality is trivial from Lemma 2.2.4. We want to show the first inequality. Fix a value of radius $r$ and a point $x \in P$. For any $\ell = 1, ..., L$, let $C_\ell$ be the set of Proposition 2.2.5 which is a $r/2$-cover of $\texttt{ball}_{P_\ell}(x,r)$. Let $C = \cup_{\ell=1}^{L} C_\ell$. The set $C$ is an $r/2$-cover of $\texttt{ball}_P$, and by the union bound, $|C| \leq \sum_{\ell=1}^{L} 2^{2\texttt{dd}(P_\ell)} \leq \sum_{\ell=1}^{L} 2^{c \cdot \texttt{dd}(P_\ell)}$. The statement immediately follows. ◀

The definition of doubling dimension in Definition 2.2.1 requires to compute the maximum over an infinite number of elements, even if the considered set of points is finite. This is due to the fact that in principle, all real values of the radius $r$ must be checked. However, one can compute the doubling dimension by only focusing on specific values of the radius. This fact is shown in the following lemma and it is crucial to the development of algorithms which estimate the doubling dimension.

▶ **Lemma 2.2.7.** *Let $P$ be a set of points. Then,*

$$\texttt{dd}(P) = \sup_{x,y \in P} \log_2 \left| \texttt{m-cover}\left( \texttt{ball}_P(x, d(x,y)), \frac{d(x,y)}{2} \right) \right|$$

**Proof.** Let $D$ be the right-side of the equation in the statement of the lemma. For every point $x \in P$, we are testing only $O(|P|)$ values of the radius. By comparison with Definition 2.2.1, we are restricting the set on which the sup is taken, therefore $D \leq \texttt{dd}(P)$.

To complete the proof, we will now show that $D \geq \texttt{dd}(P)$. In order to do so, we will demonstrate that for any $x \in P$ and $r \geq 0$, there exists $y \in P$ such that:

$$(1) \qquad \left|\texttt{m-cover}\left(\texttt{ball}_P(x,r), \frac{r}{2}\right)\right| \leq \left|\texttt{m-cover}\left(\texttt{ball}_P(x,d(x,y)), \frac{d(x,y)}{2}\right)\right|$$

Fix a point $x \in P$ and a value of the radius $r$. Let $y$ be the farthest point from $x$ such that $d(x,y) \leq r$. For construction, it holds that $\texttt{ball}_P(x,d(x,y)) = \texttt{ball}_P(x,r)$. Suppose by contradiction that the inequality (1) does not hold. As by hypothesis $d(x,y) \leq r$, the set $\texttt{m-cover}\left(\texttt{ball}_P(x,d(x,y)), \frac{d(x,y)}{2}\right)$ is also a $r/2$-cover of $\texttt{ball}_P(x,r)$, which is an absurd as any $r/2$-cover of $\texttt{ball}_P(x,r)$ should have a bigger cardinality. ◀

## 2.3 Estimation of the doubling dimension

Given a set of points $P$, it is NP-hard to compute its doubling dimension [13]. The proof is by reduction from the *dominating set* problem, and this reduction preserves the hardness of approximation. As the authors of [13] argue, this implies it is NP-hard to approximate the value $2^{\texttt{dd}(P)}$ within a factor $\Omega(\ln|P|)$. However, it is possible to find constant approximation algorithms for the doubling dimension. We say that an algorithm $(\alpha,\beta)$-approximates the doubling dimension, if for any input set of points $P$, it computes a value $D$ such that $\texttt{dd}(P) \leq D \leq \alpha \cdot \texttt{dd}(P) + \beta$. Always in [13], the authors present a $(2,0)$-approximate algorithm which has time complexity $O\left(|P|^3 \cdot 2^{O(\texttt{dd}(P))}\right)$. We will slightly lower this complexity to $O\left(|P|^3 \cdot 4^{\texttt{dd}(P)}\right)$, while preserving the same approximation. It is possible to sacrifice the quality of the estimation in order to reduce this time complexity. In [17], an $(O(1),0)$-approximate algorithm is presented which runs in $O\left(2^{O(\texttt{dd}(P))}|P|\log|P|\right)$. The approximation term $(O(1),0)$ is not satisfying as it does not give precise bounds on how much worse the approximation could be. In this section, we will see a parallel approach to lower the time complexity, at the cost of a moderately worse approximation

We will now describe the algorithm `SimpleEstimateDD`, which $(2,0)$-approximates the doubling dimension. The pseudocode of the algorithm can be found in Algorithm 1. The algorithm is based on the result of Lemma 2.2.7. We iterate over all possible pairs of points $x,y \in P$, and for each pair we compute a set $C$ which is a $d(x,y)/2$-cover of $\texttt{ball}_P(x,d(x,y))$. Ideally, we would like to be able to compute $\texttt{m-cover}\left(\texttt{ball}_P(x,d(x,y)), \frac{d(x,y)}{2}\right)$, but this is not possible unless P = NP (remember that it NP-hard to calculate the doubling dimension of a set of points). However, the strategy adopted in the while loop of Algorithm 1 computes a set $C$ whose cardinality

**Algorithm 1:** `SimpleEstimateDD`$(P)$

```
1  D ← 0
2  foreach x ∈ P do
3  |  foreach y ∈ P do
4  |  |  T ← ball_P(x, d(x,y))
5  |  |  C ← ∅
6  |  |  while T ≠ ∅ do
7  |  |  |  p ← arbitrary point in T
8  |  |  |  C ← C ∪ {p}
9  |  |  |  foreach z ∈ T do
10 |  |  |  |  if d(p,z) ≤ d(x,y)/2 then
11 |  |  |  |  |  T ← T − {z}
12 |  |  |  |  end
13 |  |  |  end
14 |  |  end
15 |  |  D ← max{D, log_2 |C|}
16 |  end
17 end
18 return D
```

is not substantially bigger. In order to do so, we must guarantee that the points in $C$ are spread apart, more specifically that for any $u, v \in C$, $u \neq v$, $d(u,v) > d(x,y)/2$.

▶ **Lemma 2.3.1.** *Let $x, y \in P$. Let $C \subseteq \texttt{ball}_P(x, d(x,y))$ be a $d(x,y)/2$-cover of $\texttt{ball}_P(x, d(x,y))$, and suppose that for any $u, v \in C$, $u \neq v$, $d(u,v) > d(x,y)/2$. Then $|C| \leq 2^{2\texttt{dd}(P)}$.*

**Proof.** Let $r = d(x,y)$. By Lemma 2.2.3, there exists a set $C'$ which is a $r/4$-cover of $\texttt{ball}_P(x, d(x,y))$ and $|C'| \leq 2^{2\texttt{dd}(P)}$. Also, for any point $u \in C'$, there can be at most one point $v \in C$ such that $d(u,v) \leq r/4$. This claim can be simply proven as follows. By construction, any two different points in $C$ must be at distance greater than $r/2$. Suppose that together with $v$ there is another point $w \in C$ such that $d(u,w) \leq r/4$. Then by the triangle inequality, $d(v,w) \leq d(v,u) + d(u,w) \leq r/2$, which is a contradiction. Since $C'$ is a $r/4$-cover of $C \subseteq \texttt{ball}_P(x, d(x,y))$, we conclude that $|C| \leq |C'|$. ◀

▶ **Lemma 2.3.2.** *The algorithm `SimpleEstimateDD` $(2,0)$-approximates the doubling dimension.*

**Proof.** Let $P$ be the input set. Let $C_{xy}$ be the set $C$ computed in the While loop of the algorithm for $x, y \in P$. By construction, the set $C_{xy}$ satisfies the hypothesis of

Lemma 2.3.1. We can write:

$$\texttt{dd}(P) = \sup_{x,y \in P} \log_2 \left| \texttt{m-cover}\left( \texttt{ball}_P(x, d(x,y)), \frac{d(x,y)}{2} \right) \right| \leq \sup_{x,y \in P} \log_2 |C_{xy}|$$

The first equality is due to Lemma 2.2.7, and the second inequality is due to the optimality of `m-cover`. Also, by Lemma 2.3.1, we have that $|C_{xy}| \leq 2^{2\texttt{dd}(P)}$. Thus:

$$\texttt{dd}(P) \leq \sup_{x,y \in P} \log_2 |C_{xy}| \leq 2 \cdot \texttt{dd}(P).$$

The statement follows by observing that $\sup_{x,y \in P} \log_2 |C_{xy}|$ is the output $D$ of the algorithm. ◀

The time complexity of the algorithm is $O\left(|P|^3 \cdot 4^{\texttt{dd}(P)}\right)$. In fact, there are $O\left(|P|^2\right)$ iterations of the first two *foreach*, and each of these iterations requires at most $O\left(|C| \cdot |P|\right)$ operations, with $|C| \leq 2^{2\texttt{dd}(P)}$ by Lemma 2.3.1.

The algorithm presents a high time complexity and therefore cannot be used with a large number of points. However, we are mostly interested in the estimation of the doubling dimension in the case of big datasets. In fact, if the points are few, the doubling dimension is always relatively low thanks to Lemma 2.2.2.

The strategy we adopt is to partition the points into different subsets and estimate the doubling dimension in each of these subsets in parallel. Then, we collect these local estimantes and use Lemma 2.2.6 to determine the final estimate on the whole set of points. The 2-rounds MapReduce algorithm `MR-EstimateDD` uses the approach just described. Formally, this algorithm receives in input a set of points $P$, and is parametrized by an integer $L$. The algorithm performs the following operations:

1. Partition the set $P$ into $L$ equally-sized subsets $P_1, ..., P_L$, and assign each to a different reducer.
2. For $\ell = 1, ..., L$, the reducer whose input is $P_\ell$ computes $D_\ell \leftarrow \texttt{SimpleEstimateDD}(P_\ell)$.
3. Gather all values $D_\ell$ and return $D = \log_2 \sum_{\ell=1}^{L} 2^{2D_\ell}$

The algorithm requires two rounds: a first round for the first two steps, and a second round for the third step. The local memory required by the algorithm is $O(L + |P|/L)$, where the term is $O(L)$ is due to Step 3, and the term $O(|P|/L)$ is due to the fact that `SimpleEstimateDD` requires linear working memory with respect to the input size. The aggregate memory is $O(|P|)$.

▶ **Lemma 2.3.3.** *The algorithm* `MR-EstimateDD` $(8, \log_2 L)$*-approximates the doubling dimension.*

**Proof.** Let $D$ be the value returned by `MR-EstimateDD` with input $P$. By Lemma 2.3.2, for any $\ell = 1, ..., L$, it holds that $\mathtt{dd}(P_\ell) \leq D_\ell \leq 2 \cdot \mathtt{dd}(P_\ell)$. By Lemma 2.2.6 with $c = 2$, we have that

$$D = \log_2 \sum_{\ell=1}^{L} 2^{2D_\ell} \geq \log_2 \sum_{\ell=1}^{L} 2^{2\mathtt{dd}(P_\ell)} \geq \mathtt{dd}(P)$$

By setting $c = 4$, Lemma 2.2.6 also implies that:

$$D = \log_2 \sum_{\ell=1}^{L} 2^{2D_\ell} \leq \log_2 \sum_{\ell=1}^{L} 2^{4\mathtt{dd}(P_\ell)} \leq 8\mathtt{dd}(P) + \log_2 L$$

◀

In practice, $L$ must be carefully chosen based on the input size and the resources available. Focusing only on efficiency, the value $L = \sqrt{|P|}$ would yield the lowest local memory. However, the choice $L = \sqrt{|P|}$ gives an $\left(8, \frac{1}{2}\log_2(|P|)\right)$-approximate algorithm, which has a very poor approximation, if we consider that $\mathtt{dd}(P) \leq \log_2 |P|$ by Lemma 2.2.2. As we are interested in obtaining a good approximation, one should reduce $L$ as much as possible.

The two algorithms `SimpleEstimateDD` and `MR-EstimateDD` prove the main result of this chapter, condensed in the following theorem.

▶ **Theorem 2.3.4.** *There exists an algorithm which* $(2,0)$*-approximates the doubling dimension with time complexity* $O\left(|P|^3 4^{\mathtt{dd}(P)}\right)$.
*There exists a* 2*-rounds MapReduce algorithm which* $(8, \log_2 L)$*-approximates the doubling dimension with* $O(|P|/L + L)$ *local memory and* $O(|P|)$ *aggregate memory.*

# Chapter 3

# $k$-center, $k$-median, $k$-means and coresets

In this chapter, we formally introduce the $k$-center, $k$-median, and $k$-means clustering problems. For each of these three problems, we give multiple definition of coresets with different properties. In the next chapter, these kinds of coresets will play a fundamental role in the design and analysis of the final clustering MapReduce algorithms. The latter are all based on the same common coreset construction primitive which is presented in the last section of this chapter.

## 3.1 $k$-center

Let $X$ and $Y$ be two set of points, we define $r_X(Y) = \max_{x \in X} d(x, Y)$. This value is also referred to as radius (of $Y$ with respect to $X$). In the *$k$-center* problem, we are given in input an instance $\mathcal{I} = (P, k)$, with $P$ a set of points and $k$ a positive integer. A set $S \subseteq P$ is a solution of $\mathcal{I}$ if $|S| \leq k$. The objective is to find the solution $S$ with minimum radius $r_P(S)$. Given an instance $\mathcal{I}$ of the $k$-center problem, we denote with $\text{opt}_{\mathcal{I}}$ its optimal solution. Moreover, for $\alpha \geq 1$, we say that $S$ is an *$\alpha$-approximate solution* for $\mathcal{I}$ if its radius is within a factor $\alpha$ from the radius of $\text{opt}_{\mathcal{I}}$. In this case, the value $\alpha$ is also called approximation factor. An *$\alpha$-approximation algorithm* computes an $\alpha$-approximate solution for any input instance.

The $k$-center problem is NP-hard to approximate within a factor $2 - \epsilon$, for any $\epsilon > 0$ [12]. Always in [12], a 2-approximation algorithm is presented with time complexity $O(|P| \cdot k)$. In the literature, this algorithm is referred to as Gonzalez's algorithm, and it is based on a *farthest-first traversal*. A farthest-first traversal is a set of points where the first point is selected arbitrarily, and each successive point is as far as possible

from the set of previously selected points. Gonzalez's algorithm exploits the fact that a farthest-first traversal $S$ of size $k$ built from the input set of points $P$ is a 2-approximate solution of the instance $\mathcal{I} = (P,k)$. The pseudocode of the algorithm can be found in Algorithm 2.

**Algorithm 2:** `Gonzalez`$(P,k)$

```
1 S ← { arbitrary point in P}
2 for i ← 2 to k do
3 |   p ← arg max_{x∈P} d(x,S)
4 |   S ← S ∪ {p}
5 end
6 return S
```

The algorithm can be implemented to run in $O(|P| \cdot k)$ time by using an auxiliary data structure (e.g. a vector) that stores for any $x \in P$, the point in $S$ which is nearest to $x$. This auxiliary data structure must be kept up do date at every iteration.

▶ **Lemma 3.1.1.** *Let $\mathcal{I} = (P,k)$ be a $k$-center instance. The set $S$ returned by the execution of `Gonzalez`$(P,k)$ is a 2-approximate solution of $\mathcal{I}$.*

**Proof.** Let $z = \arg\max_{x \in P} d(x,S)$ and $r = d(z,S) = r_P(S)$. For the greedy criterion with which the points to be added to S are chosen, we have that for any $x, y \in S$, $x \neq y$, it holds $d(x,y) \geq r$. Also, $d(z,x) \geq r$ for any $x \in R$. Hence, $S \cup \{z\}$ are $k+1$ points which are at least at distance $r$ from one another. By the pigeonhole principle, there must exist a point $o^* \in \text{opt}_{\mathcal{I}}$ such that there are at least two points $x, y \in (S \cup \{z\})$ for which $d(x,o^*) = d(x,\text{opt}_{\mathcal{I}})$ and $d(y,o^*) = d(y,\text{opt}_{\mathcal{I}})$. By triangle inequality, we conclude:

$$r_P(S) = r \leq d(x,y) \leq d(x,o^*) + d(o^*,y) \leq 2 \cdot r_P(\text{opt}_{\mathcal{I}})$$

◀

The coresets summarize the input instance based on some properties. The property that is used must be useful in describing the input instance entirely, and it must be possible to exploit it when the coreset is used. In this thesis, following the ideas in [7], we have chosen to use the following definition of coreset for the $k$-center problem.

▶ **Definition 3.1.2.** *Let $\mathcal{I} = (P,k)$ be a $k$-center instance. A set of points $C$ is an $\epsilon$-bounded coreset of $\mathcal{I}$ if it exists a map $\tau : P \to C$ such that $d(x,\tau(x)) \leq \epsilon \cdot r_P(\text{opt}_{\mathcal{I}})$ for any $x \in P$.*

The coreset property stated in the previous definition allows to obtain a relationship between solutions computed from the reduced instance $\mathcal{I}' = (C_w, k)$ and $\text{opt}_{\mathcal{I}}$.

▶ **Lemma 3.1.3.** *Let $\mathcal{A}$ be an $\alpha$-approximation algorithm for the $k$-center problem. Let $\mathcal{I} = (P,k)$ be a $k$-center instance. Let $C$ be an $\epsilon$-bounded coreset of $\mathcal{I}$. Let $S$ be the solution returned by the execution of $\mathcal{A}$ on the instance $\mathcal{I}' = (C,k)$. Then, $r_P(S) \leq (2\alpha+\epsilon)\cdot r_P(\text{opt}_{\mathcal{I}})$.*

**Proof.** Let $\tau$ be the map from $P$ to $C$ of Definition 3.1.2. By triangle inequality, it holds that:

$$r_P(S) = \max_{x\in P} d(x,S) \leq \max_{x\in P} d(x,\tau(x)) + \max_{x\in P} d(\tau(x),S)$$

By definition of $\epsilon$-bounded coreset, we have that $\max_{x\in P} d(x,\tau(x)) \leq \epsilon\cdot r_P(S)$. Also, we observe that $\max_{x\in P} d(\tau(x),S) = \max_{x\in C} d(x,S)$. As $S$ is an $\alpha$-approximate solution of $\mathcal{I}'$, it results that $\max_{x\in C} d(x,S) \leq \alpha\cdot r_C(\text{opt}_{\mathcal{I}'})$. Now, let $G = \{x^C : x \in \text{opt}_{\mathcal{I}}\}$. By optimality of $\text{opt}_{\mathcal{I}'}$, we have that $r_C(\text{opt}_{\mathcal{I}'}) \leq r_C(G)$. Using triangle inequality, we obtain:

$$r_C(G) = \max_{x\in C} d(x,G) \leq \max_{x\in C} \left[d(x,x^{\text{opt}_{\mathcal{I}}}) + d(x^{\text{opt}_{\mathcal{I}}},G)\right] \leq 2\max_{x\in C} d(x,x^{\text{opt}_{\mathcal{I}}})$$

In the last inequality, we used the fact that $d(x^{\text{opt}_{\mathcal{I}}},G) \leq d(x,x^{\text{opt}_{\mathcal{I}}})$ by construction of $G$. Since $C \subseteq P$, we have that $\max_{x\in C} d(x,x^{\text{opt}_{\mathcal{I}}}) \leq \max_{x\in P} d(x,x^{\text{opt}_{\mathcal{I}}}) \leq r_P(\text{opt}_{\mathcal{I}})$. Wrapping it up, it results that $r_P(S) \leq (2\alpha+\epsilon) r_P(\text{opt}_{\mathcal{I}})$. ◀

It is important to highlight that the use of a coreset, together with a straightforward application of Lemma 3.1.3, worsens the approximation factor of the algorithm by a factor of approximately 2. However, if we restrict to the Gonzalez's algorithm, we can obtain a better result.

▶ **Lemma 3.1.4.** *Let $\mathcal{A}$ be the Gonzalez's algorithm. Let $\mathcal{I} = (P,k)$ be a $k$-center instance. Let $C$ be an $\epsilon$-bounded coreset of $\mathcal{I}$. Let $S$ be the solution returned by the execution of $\mathcal{A}$ on the instance $\mathcal{I}' = (C,k)$. Then, $r_P(S) \leq (2+\epsilon)\cdot r_P(\text{opt}_{\mathcal{I}})$.*

**Proof.** Proceeding as in the proof of Lemma 3.1.3, we have that:

$$r_P(S) \leq \epsilon\cdot r_P(\text{opt}_{\mathcal{I}}) + r_C(S)$$

Let $z$ be the farthest point to $S$ in $C$ and let $r = d(z,S) = r_C(S)$. Arguing as in the proof of Lemma 3.1.1, it results that $S\cup\{z\}$ are $k+1$ points which are at least at distance $r$

from one another. By the pigeonhole principle, there must exist a point $o^* \in \text{opt}_{\mathcal{I}}$ such that there are at least two points $x, y \in (S \cup \{z\})$ for which $d(x, o^*) = d(x, \text{opt}_{\mathcal{I}})$ and $d(y, o^*) = d(y, \text{opt}_{\mathcal{I}})$. By triangle inequality, we conclude:

$$r_C(S) = r \leq d(x, y) \leq d(x, o^*) + d(o^*, y) \leq 2 \cdot r_P(\text{opt}_{\mathcal{I}})$$

◀

The lemma above shows how the Gonzalez's algorithm fits well with the definition of $\epsilon$-bounded coreset. The properties of the Gonzalez's algorithm lead to a final approximation factor which is much lower than in the general case. If we were to apply Lemma 3.1.3, we would obtain $4 + \epsilon$, which is much higher than the factor $2 + \epsilon$ of Lemma 3.1.4.

Bounded coresets have the nice property to be *composable*. That is, we can partition the input points into different subsets and compute a bounded coreset separately in each subset: the union of those coresets is a bounded coreset of the input instance. This property, which is formally stated in the following lemma, is crucial to develop efficient MapReduce algorithms for the clustering problems

▶ **Lemma 3.1.5.** *Let $\mathcal{I} = (P, k)$ be an instance of $k$-center. Let $P_1, \ldots, P_L$ be a partition of $P$. For $\ell = 1, \ldots, L$, let $C_\ell$ be an $\epsilon$-bounded coreset of $\mathcal{I}_\ell = (P_\ell, k)$. Then $C = \cup_\ell C_\ell$ is a $2\epsilon$-bounded coreset of $\mathcal{I}$.*

**Proof.** For $\ell = 1, \ldots L$, let $\tau_\ell$ be the map of Definition 3.1.2 from $P_\ell$ to $C_\ell$. Now, for any $x \in P$, let $\ell$ be the integer such that $x \in P_\ell$; we define $\tau(x) = \tau_\ell(x)$. By construction, we have that:

$$\max_{x \in P} d(x, \tau(x)) = \max_{\ell = 1, \ldots, L} \max_{x \in P_l} d(x, \tau_\ell(x)) \leq \epsilon \cdot \max_{\ell = 1, \ldots, L} r_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$$

Fix a value of $\ell$. We rephrase the last part of the proof of Lemma 3.1.3 in order to show that $r_{P_\ell}(\text{opt}_{\mathcal{I}_\ell}) \leq 2 r_P(\text{opt}_{\mathcal{I}})$ for any $\ell$. Let $G = \{x^{P_\ell} : x \in \text{opt}_{\mathcal{I}}\}$. By optimality of $\text{opt}_{\mathcal{I}_\ell}$, we have that $r_{P_\ell}(\text{opt}_{\mathcal{I}_\ell}) \leq r_{P_\ell}(G)$. Using triangle inequality, we obtain:

$$r_{P_\ell}(G) = \max_{x \in P_\ell} d(x, G) \leq \max_{x \in P_\ell} \left[ d(x, x^{\text{opt}_{\mathcal{I}}}) + d(x^{\text{opt}_{\mathcal{I}}}, G) \right] \leq 2 \max_{x \in P_\ell} d(x, x^{\text{opt}_{\mathcal{I}}})$$

In the last inequality, we used the fact that $d(x^{\text{opt}_{\mathcal{I}}}, G) \leq d(x, x^{\text{opt}_{\mathcal{I}}})$ by construction of $G$. Since $P_\ell \subseteq P$, we have that $\max_{x \in P_\ell} d(x, x^{\text{opt}_{\mathcal{I}}}) \leq \max_{x \in P} d(x, x^{\text{opt}_{\mathcal{I}}}) \leq r_P(\text{opt}_{\mathcal{I}})$. ◀

## 3.2 $k$-median and $k$-means

Let $X_w$ and $Y$ be two weighted set of points. We define $\nu_{X_w}(Y) = \sum_{x \in X_w} w(x)d(x,Y)$ and $\mu_{X_w}(Y) = \sum_{x \in X_w} w(x)d(x,Y)^2$. The values $\nu_{X_w}(Y)$ and $\mu_{X_w}(Y)$ are also referred to as *costs*.

In the *$k$-median problem* (resp., *$k$-means problem*), we are given in input an instance $\mathcal{I} = (P,k)$, with $P \subseteq \mathcal{M}$ and $k$ a positive integer. A set $S \subseteq P$ is a solution of $\mathcal{I}$ if $|S| \leq k$. The objective is to find the solution $S$ with minimum cost $\nu_P(S)$ (resp., $\mu_P(S)$). Following the lines of the definitions given for $k$-center in the previous section, we say that the optimal solution of an instance $\mathcal{I}$ of one of these two problems is $\text{opt}_{\mathcal{I}}$. Moreover, for $\alpha \geq 1$, we say that $S$ is an *$\alpha$-approximate solution* for $\mathcal{I}$ if its cost is within a factor $\alpha$ from the cost of $\text{opt}_{\mathcal{I}}$. We will also use the notion of *bi-criteria approximation* [30]. For $k$-median and $k$-means, an $(\alpha,\beta)$ bi-criteria approximation algorithm returns a set of $\geq \beta k$ centers ($\beta \geq 1$) such that its cost is within a factor $\alpha$ from the cost of the optimal solution with $k$ centers. In other words, in a bi-criteria approximation we are allowed to select more centers than $k$ in order to obtain a smaller cost. The $k$-median and $k$-means problems are immediately generalized to the case of weighted instances $(P_w,k)$. In fact, all known approximation algorithms can be straightforwardly adapted to handle weighted instances, keeping the same approximation quality.

Observe that the squared distance does not satisfy the triangle inequality. During the analysis, we will use the following weaker bound.

▶ **Proposition 3.2.1.** *Let $x,y$ and $z$ be three points. For every $c > 0$ we have that $d(x,y)^2 \leq (1+1/c)d(x,z)^2 + (1+c)d(z,y)^2$.*

**Proof.** Let $a,b$ be two real numbers. Since $(a/\sqrt{c} - b \cdot \sqrt{c})^2 \geq 0$, we obtain that $2ab \leq a^2/c + c \cdot b^2$. Hence, $(a+b)^2 \leq (1+1/c)a^2 + (1+c)b^2$. The proof follows since $d(x,y)^2 \leq [d(x,z) + d(z,y)]^2$ by triangle inequality. ◀

Both the $k$-median and the $k$-means problems are NP-hard, and they are also hard to approximate respectively within a factor $1 + \frac{2}{e}$ [21] and 1.0013 [25]. Now, we will briefly present the local search algorithm which can be used to obtain a constant-approximation factor for both $k$-median and $k$-means [2, 14]. The algorithm starts with an arbitrary set of $k$ points of the input set $P$ as the solution, and iteratively tries to improve it until a local minimum is reached. The operation allowed to change the solution at each iteration is called $t$-swap. A $t$-swap is an operation that changes $t$ points of the actual solution $S$ with $t$ distinct points of $P - S$. At each iteration, the algorithm finds the $t$-swap that leads to the lowest cost. If no $t$-swap lowers the cost

with respect to the actual solution, then the algorithm terminates. At each iteration, the number of possible $t$-swaps is $\binom{|P|-k}{t}\binom{k}{t}$.

▶ **Theorem 3.2.2.** ***(Theorem 3.5 of [14])***. *The local search algorithm with $t$-swaps returns:*

1. $(3+\frac{2}{t})$*-approximate solution for $k$-median*
2. $(5+\frac{4}{t})$*-approximate solution for $k$-means*

As discussed in the previous section, a coreset is a small (weighted, in the case of $k$-median and $k$-means) subset of the input which summarizes the whole data. The concept of summarization can be captured with the following definition, which is commonly adopted to describe coresets for $k$-means and $k$-median (e.g., [16, 11, 19]).

▶ **Definition 3.2.3.** *A weighted set of points $C_w$ is an $\epsilon$-approximate coreset of an instance $\mathcal{I}=(P,k)$ of $k$-median (resp., $k$-means) if for any solution $S$ of $\mathcal{I}$ it holds that $|\nu_P(S)-\nu_{C_w}(S)|\leq\epsilon\cdot\nu_P(S)$ (resp., $|\mu_P(S)-\mu_{C_w}(S)|\leq\epsilon\cdot\mu_P(S)$).*

Informally, the cost of any solution is approximately the same if computed with respect to the $\epsilon$-approximate coreset rather than with respect to the full set of points. In the thesis, we will also make use of the following different notion of coreset (already used in [16, 10]), which upper bounds the aggregate "proximity" of the input points from the coreset as a function of the optimal cost.

▶ **Definition 3.2.4.** *Let $\mathcal{I}=(P,k)$ be an instance of $k$-median (resp., $k$-means). A set of points $C_w$ is an $\epsilon$-bounded coreset of $\mathcal{I}$ if it exists a map $\tau:P\to C_w$ such that $\sum_{x\in P}d(x,\tau(x))\leq\epsilon\cdot\nu_P(\mathrm{opt}_{\mathcal{I}})$ (resp., $\sum_{x\in P}d(x,\tau(x))^2\leq\epsilon\cdot\mu_P(\mathrm{opt}_{\mathcal{I}})$) and for any $x\in C_w$, $w(x)=|\{y\in P:\tau(y)=x\}|$. We say that $C_w$ is weighted according to $\tau$.*

The above two kinds of coresets are related, as shown in the following two lemmas.

▶ **Lemma 3.2.5.** *Let $C_w$ be an $\epsilon$-bounded coreset of a $k$-median instance $\mathcal{I}=(P,k)$. Then $C_w$ is also an $\epsilon$-approximate coreset of $\mathcal{I}$.*

**Proof.** Let $\tau$ be the map of the definition of $\epsilon$-bounded coreset. Let $S$ be a solution of $\mathcal{I}$. Using triangle inequality, we can easily see that $d(x,S)-d(x,\tau(x))\leq d(\tau(x),S)$ and $d(\tau(x),S)\leq d(\tau(x),x)+d(x,S)$ for any $x\in P$. Summing over all points in $P$, we obtain that

$$\nu_P(S)-\sum_{x\in P}d(x,\tau(x))\leq\nu_{C_w}(S)\leq\sum_{x\in P}d(x,\tau(x))+\nu_P(S)$$

To conclude the proof, we observe that $\sum_{x\in P}d(x,\tau(x))\leq\epsilon\cdot\nu_P(\mathrm{opt}_{\mathcal{I}})\leq\epsilon\cdot\nu_P(S)$. ◀

▶ **Lemma 3.2.6.** *Let $C_w$ be an $\epsilon$-bounded coreset of a $k$-means instance $\mathcal{I} = (P,k)$. Then $C_w$ is also an $(\epsilon + 2\sqrt{\epsilon})$-approximate coreset of $\mathcal{I}$.*

**Proof.** Let $\tau$ be the map of the definition of $\epsilon$-bounded coreset. Let $S$ be a solution of $\mathcal{I}$. We want to bound the quantity $|\mu_P(S) - \mu_{C_w}(S)| = \sum_{x\in P} |d(x,S)^2 - d(\tau(x),S)^2|$. We rewrite $|d(x,S)^2 - d(\tau(x),S)^2|$ as $[d(x,S) + d(\tau(x),S)] \cdot |d(x,S) - d(\tau(x),S)|$. By triangle inequality, we have that $d(x,S) \leq d(x,\tau(x)) + d(\tau(x),S)$ and $d(\tau(x),S) \leq d(\tau(x),x) + d(x,S)$. By combining these two inequalities, it results that $|d(x,S) - d(\tau(x),S)| \leq d(x,\tau(x))$. Moreover, $d(x,S) + d(\tau(x),S) \leq 2d(x,S) + d(x,\tau(x))$. Hence

$$\begin{aligned} |\mu_P(S) - \mu_{C_w}(S)| &\leq \sum_{x\in P} d(x,\tau(x)) \left[2d(x,S) + d(x,\tau(x))\right] \\ &\leq \epsilon \cdot \mu_P(S) + 2\sum_{x\in P} d(x,\tau(x)) d(x,S) \end{aligned}$$

where we used the fact that $\sum_{x\in P} d(x,\tau(x))^2 \leq \epsilon \cdot \mu_P(\text{opt}_{\mathcal{I}}) \leq \epsilon \cdot \mu_P(S)$. We now want to bound the sum over the products of the two distances. Arguing as in the proof of Proposition 3.2.1, we can write:

$$2\sum_{x\in P} d(x,\tau(x)) d(x,S) \leq \sqrt{\epsilon} \cdot \sum_{x\in P} d(x,S)^2 + \frac{1}{\sqrt{\epsilon}} \sum_{x\in P} d(x,\tau(x))^2 \leq 2\sqrt{\epsilon} \cdot \mu_P(S)$$

To wrap it up, it results that $|\mu_P(S) - \mu_{C_w}(S)| \leq (\epsilon + 2\sqrt{\epsilon}) \cdot \mu_P(S)$. ◀

In our work, we will build coresets by working in parallel over a partition of the input instance. The next lemma provides known results on the relations between the optimal solution of the whole input points and the optimal solution of a subset of the input points.

▶ **Lemma 3.2.7.** *Let $C_w \subseteq P$. Let $\mathcal{I} = (P,k)$ and $\mathcal{I}' = (C_w,k)$. Then: (a) $\nu_{C_w}(\text{opt}_{\mathcal{I}'}) \leq 2\nu_{C_w}(\text{opt}_{\mathcal{I}})$; and (b) $\mu_{C_w}(\text{opt}_{\mathcal{I}'}) \leq 4\mu_{C_w}(\text{opt}_{\mathcal{I}})$.*

**Proof.** We first prove point $(b)$. Let $X = \{x^{C_w} : x \in \text{opt}_{\mathcal{I}}\}$. The set $X$ is a solution of $\mathcal{I}'$. By optimality of $\text{opt}_{\mathcal{I}'}$, we have that $\mu_{C_w}(\text{opt}_{\mathcal{I}'}) \leq \mu_{C_w}(X)$. Also, by triangle inequality, it holds that $\mu_{C_w}(X) \leq \sum_{x\in C_w} w(x) \left[d(x,\text{opt}_{\mathcal{I}}) + d(x^{\text{opt}_{\mathcal{I}}},X)\right]^2$. We observe that $d(x^{\text{opt}_{\mathcal{I}}},X) \leq d(x,\text{opt}_{\mathcal{I}})$ by definition of $X$. Thus, we obtain that $\mu_{C_w}(\text{opt}_{\mathcal{I}'}) \leq 4\mu_{C_w}(\text{opt}_{\mathcal{I}})$. The proof of $(a)$ follows the same lines with a factor 2 less since we do not square. ◀

As for $k$-center, bounded coresets for the $k$-median and $k$-means problems are composable, and we will extensively use this property in the development of efficient MapReduce algorithms for those clustering problems.

▶ **Lemma 3.2.8.** *Let $\mathcal{I} = (P, k)$ be an instance of $k$-median (resp., $k$-means). Let $P_1, \ldots, P_L$ be a partition of $P$. For $\ell = 1, \ldots, L$, let $C_{w,\ell}$ be an $\epsilon$-bounded coreset of $\mathcal{I}_\ell = (P_\ell, k)$. Then $C_w = \cup_\ell C_{w,\ell}$ is a $2\epsilon$-bounded coreset (resp., a $4\epsilon$-bounded coreset) of $\mathcal{I}$.*

**Proof.** We prove the lemma for $k$-median. The proof for $k$-means is similar. For $\ell = 1, \ldots, L$, let $\tau_\ell$ be the map from $P_\ell$ to $C_{w,\ell}$ of Definition 3.2.4. Now, for any $x \in P$, let $\ell$ be the integer such that $x \in P_\ell$; we define $\tau(x) = \tau_\ell(x)$.

$$\sum_{x \in P} d(x, \tau(x)) \leq \sum_{\ell=1}^{L} \sum_{x \in P_\ell} d(x, \tau_\ell(x)) \leq \epsilon \sum_{\ell=1}^{L} \nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell}) \leq 2\epsilon \cdot \nu_P(\text{opt}_{\mathcal{I}})$$

In the last inequality, we used the fact that $\nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell}) \leq 2\nu_{P_\ell}(\text{opt}_{\mathcal{I}})$ from Lemma 3.2.7. ◀

We will also need the following additional characterization of a representative subset of the input, originally introduced in [16].

▶ **Definition 3.2.9.** *Let $\mathcal{I} = (P, k)$ be an instance of $k$-median (resp., $k$-means). A set $C$ is said to be an $\epsilon$-centroid set of $\mathcal{I}$ if there exists a subset $X \subseteq C$, $|X| \leq k$, such that $\nu_P(X) \leq (1+\epsilon)\nu_P(\text{opt}_{\mathcal{I}})$ (resp., $\mu_P(X) \leq (1+\epsilon)\mu_P(\text{opt}_{\mathcal{I}})$).*

Simply put, a centroid set contains a solution which is almost as good as $\text{opt}_{\mathcal{I}}$ when the cost is calculated on the whole input $P$.

## 3.3 Coreset building primitive

Our coreset constructions are based on a suitable point selection algorithm called `CoverWithBalls`, somewhat inspired by the exponential grid construction used in [16] to build $\epsilon$-approximate coresets for $k$-median and $k$-means in $\mathbb{R}^d$ for the continuous case. For ease of explaination, we will give the intuition of the algorithm restricting our focus on $k$-median. However, the same algorithm will be a fundamental building block of all our final MapReduce clustering algorithms. Suppose that we want to build an $\epsilon$-bounded coreset of a $k$-median instance $\mathcal{I} = (P, k)$ and that a $\beta$-approximate solution $T$ for $\mathcal{I}$ is available. A simple approach would be to find a set $C_w$ such that for any $x$ in $P$ there exists a point $\tau(x) \in C$ for which $d(x, \tau(x)) \leq (\epsilon/2\beta) \cdot d(x, T)$. Indeed, if $C_w$ is weighted according to $\tau$, it can be seen that $C_w$ is an $\epsilon$-bounded coreset of $\mathcal{I}$. The set $C_w$ can be constructed greedily by iteratively selecting an arbitrary point $p \in P$, adding it to $C_w$, and discarding all points $q \in P$ (including $p$) for which the

aforementioned property holds with $\tau(q) = p$. The construction ends when all points of $P$ are discarded. However, note that the points of $P$ which are already very close to $T$, say at a distance $\leq R$ for a suitable tolerance threshold $R$, do not contribute much to $\nu_P(T)$, and so to the sum $\sum_{x \in P} d(x, \tau(x))$. For these points, we can relax the constraint and discard them from $P$ as soon their distance to $C_w$ becomes at most $(\epsilon/2\beta) \cdot R$. This relaxation is crucial to bound the size of the returned set as a function of the doubling dimension of the space. Algorithm `CoverWithBalls` implements the

**Algorithm 3:** `CoverWithBalls`$(P, T, R, \epsilon, \beta)$

```
1  C_w ← ∅
2  while P ≠ ∅ do
3  |   p ⟵ arbitrarily selected point in P
4  |   C_w ⟵ C_w ∪ {p}, w(p) ⟵ 0
5  |   foreach q ∈ P do
6  |   |   if d(p,q) ≤ ε/(2β) max{R, d(q,T)} then
7  |   |   |   remove q from P
8  |   |   |   w(p) ⟵ w(p) + 1      /* (i.e. τ(q) = p, see Lemma 3.3.1) */
9  |   |   end
10 |   end
11 end
12 return C_w
```

coreset construction discussed above. The algorithm receives in input two sets of points, $P$ and $T$, and three positive real parameters $R$, $\epsilon$, and $\beta$, with $\epsilon < 1$ and $\beta \geq 1$ and outputs a weighted set $C_w \subseteq P$ which satisfies the property stated in the following lemma.

▶ **Lemma 3.3.1.** *Let $C_w$ be the output of `CoverWithBalls`$(P, T, R, \epsilon, \beta)$. $C_w$ is weighted according to a map $\tau : P \to C_w$ such that, for any $x \in P$, $d(x, \tau(x)) \leq \epsilon/(2\beta) \max\{R, d(x, T)\}$.*

**Proof.** For any $x \in P$, we define $\tau(x)$ as the point in $C_w$ which caused the removal of $x$ from $P$ during the execution of the algorithm. The statement immediately follows. ◀

While in principle the size of $C_w$ can be arbitrarily close to $|P|$, the next theorem shows that this is not the case for low dimensional spaces, as a consequence of the fact that there cannot be too many points which are all far from one another. We first need a technical lemma. A set of points $X$ is said to be an *$r$-clique* if for any $x, y \in X$, $x \neq y$, it holds that $d(x, y) > r$. We have:

▶ **Lemma 3.3.2.** *Let $0 < \epsilon < 1$. Let $\mathcal{M}$ be a metric space with doubling dimension $\mathtt{dd}(\mathcal{M})$. Let $X \subseteq \mathcal{M}$ be an $\epsilon \cdot r$-clique and assume that $X$ can be covered by a ball of radius $r$ centered at a point of $\mathcal{M}$. Then, $|X| \leq (4/\epsilon)^{\mathtt{dd}(\mathcal{M})}$.*

**Proof.** Applying Lemma 2.2.3, we observe that the ball of radius $r$ centered at $x$ which covers $X$ can be covered by $2^{j \cdot \mathtt{dd}(\mathcal{M})}$ balls of radius $2^{-j} \cdot r$, where $j$ is any non negative integer. Let $i$ be the least integer for which $2^{-i} \cdot r \leq \epsilon/2 \cdot r$ holds. Any of the $2^{i \cdot \mathtt{dd}(\mathcal{M})}$ balls with radius $2^{-i} \cdot r$ can contain at most one point of $X$, since $X$ is an $\epsilon \cdot r$-clique. Thus $|X| \leq 2^{i \cdot \mathtt{dd}(\mathcal{M})}$. As $i = 1 + \lceil \log_2(1/\epsilon) \rceil$, we finally obtain that $|X| \leq (4/\epsilon)^{\mathtt{dd}(\mathcal{M})}$. ◀

▶ **Theorem 3.3.3.** *Let $C_w$ be the set returned by the execution of $\mathtt{CoverWithBalls}(P, T, R, \epsilon, \beta)$. Suppose that $P \cup T \subseteq \mathcal{M}$, and that $\mathcal{M}$ is a metric space with doubling dimension $\mathtt{dd}(\mathcal{M})$. Let $c$ be a real value such that, for any $x \in P$, $c \cdot R \geq d(x, T)$. Then,*

$$|C_w| \leq |T| \cdot \left[ (8\beta/\epsilon)^{\mathtt{dd}(\mathcal{M})} + \lceil \log_2 c \rceil (16\beta/\epsilon)^{\mathtt{dd}(\mathcal{M})} \right]$$

**Proof.** Let $T = \{t_1, \ldots, t_{|T|}\}$ be the set in input to the algorithm. For any $i$, $1 \leq i \leq |T|$, let $P_i = \{x \in P : x^T = t_i\}$ and $B_i = \{x \in P_i : d(x, T) \leq R\}$. In addition, for any integer value $j \geq 0$ and for any feasible value of $i$, we define $D_{i,j} = \{x \in P_i : 2^j \cdot R < d(x, T) \leq 2^{j+1} \cdot R\}$. We observe that for any $j \geq \lceil \log_2 c \rceil$, the sets $D_{i,j}$ are empty, since $d(x, T) \leq c \cdot R$. Together, the sets $B_i$ and $D_{i,j}$ are a partition of $P_i$.

For any $i$, let $C_i = C_w \cap B_i$. We now want to show that the set $C_i$ is a $\epsilon/(2\beta) \cdot R$-clique. Let $c_1, c_2$ be any two different points in $C_i$ and suppose, without loss of generality, that $c_1$ was added first to $C_w$. Since $c_2$ was not removed from $P$, this means that $d(c_1, c_2) > \epsilon/(2\beta) \cdot \max\{d(c_2, T), R\} \geq \epsilon/(2\beta)R$, where we used the fact that $d(c_2, T) \leq R$ since $c_2$ belongs to $B_i$. Also, the set $C_i \subseteq B_i$ is contained in a ball of radius $R$ centered in $t_i$, thus we can apply Lemma 3.3.2 and bound its size, obtaining that $|C_i| \leq (8\beta/\epsilon)^{\mathtt{dd}(\mathcal{M})}$.

For any $i$ and $j$, let $C_{i,j} = C_w \cap D_{i,j}$. We can use a similar strategy to bound the size of those sets. We first show that the sets $C_{i,j}$ are $\frac{\epsilon}{4\beta} \cdot 2^{j+1}R$-cliques. Let $c_1, c_2$ be any two different points in $C_{i,j}$ and suppose, without loss of generality, that $c_1$ was added first to $C_w$. Since $c_2$ was not removed from $P$, this means that $d(c_1, c_2) > \epsilon/(2\beta) \cdot \max\{d(c_2, T), R\} \geq \epsilon/(4\beta)2^{j+1}R$, where we used the fact that $d(c_2, T) > 2^j \cdot R$ since $c_2$ belongs to $D_{i,j}$. Also, the set $C_{i,j} \subseteq D_{i,j}$ is contained in a ball of radius $2^{j+1}R$ centered in $t_i$, thus we can apply Lemma 3.3.2 and obtain that $|C_{i,j}| \leq (16\beta/\epsilon)^{\mathtt{dd}(\mathcal{M})}$. Since the sets $C_i$ and $C_{i,j}$ partition $C_w$, we can bound the size of $C_w$ as the sum of the

bounds of the size of those sets. Hence:

$$|C_w| \leq \sum_{i=1}^{|T|} |C_i| + \sum_{i=1}^{|T|} \sum_{j=0}^{\lceil \log_2 c \rceil - 1} |C_{i,j}| \leq |T| \cdot \left[ (8\beta/\epsilon)^{\mathtt{dd}(\mathcal{M})} + \lceil \log_2 c \rceil (16\beta/\epsilon)^{\mathtt{dd}(\mathcal{M})} \right]$$

◀

# Chapter 4

# MapReduce algorithms for clustering via coreset

In this chapter, we present the coreset-based MapReduce algorithms for the $k$-center, $k$-median and $k$-means clustering problems. All the algorithms use the primitive of Section 3.3 as a fundamental building block. The chapter starts with the $k$-center problem, which has the simplest coreset construction (Section 4.1) and final MapReduce algorithm (Section 4.2). In Section 4.3, a similar construction is applied for $k$-median, however it does not lead to an approximation factor which can be made arbitrarily close to the best known sequential algorithm for the problem. In order to achieve this result, we improve the coreset construction in Section 4.4, and use an analogous method for $k$-means in Section 4.5. Finally, in Section 4.6, we present the final MapReduce algorithms for $k$-median and $k$-means.

## 4.1 Coreset construction for $k$-center

In this section we present a 1-round MapReduce algorithm that builds a coreset $C \subseteq P$ of a $k$-center instance $\mathcal{I} = (P, k)$. The algorithm is parametrized by a value $\epsilon \in (0, 1)$, which represents a tradeoff between coreset size and accuracy.

This MapReduce algorithm operates as follows. The set $P$ is partitioned into $L$ equally-sized subsets $P_1, ...P_L$. In parallel, on each $k$-center instance $\mathcal{I}_\ell = (P_\ell, k)$, with $\ell = 1, ..., L$, these computations are performed:

1. $T_\ell \longleftarrow$ `Gonzalez`$(P, k)$
2. $R_\ell \longleftarrow r_{P_\ell}(T_\ell)$
3. $C_\ell \longleftarrow$ `CoverWithBalls`$(P_\ell, T_\ell, R_\ell, \epsilon, 2)$

The set $C = \cup_{\ell=1}^{L} C_\ell$ is the output of the algorithm.

In Step 3, the output of the algorithm `CoverWithBalls` should be a weighted set. However, for the $k$-center problem, the weights are not needed and can be dropped. As the Gonzalez's algorithm requires only linear space in the input, the entire coreset construction can be implemented in a single MapReduce round using $O(|P|/L)$ local memory and $O(|P|)$ aggregate memory.

▶ **Lemma 4.1.1.** *For $\ell = 1, \ldots L$, the set $C_\ell$ is an $\epsilon/2$-bounded coreset of $\mathcal{I}_\ell$.*

**Proof.** Fix a value of $\ell$. By Lemma 3.3.1, there exists a map $\tau_\ell : P_\ell \to C_\ell$, such that for any $x \in P_\ell$, it holds that $d(x, \tau(x)) \leq (\epsilon/4)\max\{R_\ell, d(x, T_\ell)\}$. For our choice of $R_\ell$, it holds that $\max\{R_\ell, d(x, T_\ell)\} = R_\ell$. Hence, we have that:

$$\max_{x \in P_\ell} d(x, \tau_\ell(x)) \leq \frac{\epsilon}{4} R_\ell \leq \frac{\epsilon}{2} r_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$$

In the last inequality, we used the fact that $R_\ell = r_{P_\ell}(T_\ell)$, and that $T_\ell$ is a 2-approximate solution of $\mathcal{I}_\ell$ (Lemma 3.1.1). ◀

As noted in the proof of the previous lemma, the algorithm `CoverWithBalls` can be simplified in this case, as for any $\ell$, and $x \in P_\ell$, it holds that $\max\{R_\ell, d(x, T_\ell)\} = R_\ell$ (line 6 of the pseudocode of `CoverWithBalls`).

The next lemma is immediate consequence of Lemma 4.1.1 and Lemma 3.1.5.

▶ **Lemma 4.1.2.** *Let $\mathcal{I} = (P, k)$ be a $k$-center instance. The set $C$ returned by the above MapReduce algorithm is an $\epsilon$-bounded coreset of $\mathcal{I}$.*

It is possible to bound the size of $C$ in function of the doubling dimension $\mathtt{dd}(P)$.

▶ **Lemma 4.1.3.** *Let $\mathcal{I} = (P, k)$ be a $k$-center instance. The set $C$ returned by the above MapReduce algorithm has size $|C| = O\left(L \cdot k \cdot (16/\epsilon)^{\mathtt{dd}(P)}\right)$.*

**Proof.** Fix a value of $\ell$. For any $x \in P_\ell$, we have that $1 \cdot R_\ell \geq d(x, T_\ell)$, and $T_\ell \cup P_\ell \subseteq P$. We can invoke Theorem 3.3.3 with $c = 1$ and $\mathcal{M} = P$, thus $|C_\ell| = O\left(k \cdot (16/\epsilon)^{\mathtt{dd}(P)}\right)$. As $C = \cup_{\ell=1}^{L} C_\ell$, we prove the statement with a union bound. ◀

## 4.2 MapReduce algorithm for $k$-center

In this section, we present a 2-rounds MapReduce algorithm which computes a $(2+\epsilon)$-approximate solution of an instance $\mathcal{I} = (P, k)$. The algorithm operates as follows: in the first round we compute a coreset $C$ by using the algorithm described in Section 4.1, and in the second round we use the Gonzalez's algorithm on the coreset $C$ to

obtain the final solution. By setting $L = \sqrt{|P|/k}$, and combining Lemma 4.1.3 and Lemma 3.1.4, we obtain the next theorem.

▶ **Theorem 4.2.1.** *Let $\mathcal{I} = (P,k)$ be an instance of $k$-center. For any $\epsilon \in (0,1)$, the 2-round MapReduce algorithm described above computes a $(2+\epsilon)$-approximate solution of $\mathcal{I}$ using local space $O(\sqrt{|P|k}(16/\epsilon)^{\mathtt{dd}(P)})$.*

In principle, it is possible to use another $\alpha$-approximation algorithm for the $k$-center in problem in the second round of the algorithm. However, in this case we would need to use Lemma 3.1.3, and the the final solution would be a $(2\alpha+\epsilon)$-approximate solution. For this reason, it is advantageous to use the Gonzalez's algorithm.

## 4.3 A first approach to coreset construction for $k$-median

In this section we present a 1-round MapReduce algorithm that builds a weighted coreset $C_w \subseteq P$ of a $k$-median instance $\mathcal{I} = (P,k)$. The algorithm is parametrized by a value $\epsilon \in (0,1)$, which represents a tradeoff between coreset size and accuracy. The returned coreset has the following property. Let $\mathcal{I}' = (C_w,k)$. If we run an $\alpha$-approximation algorithm on $\mathcal{I}'$, then the returned solution is a $(2\alpha+O(\epsilon))$-approximate solution of $\mathcal{I}$. Building on this construction, in the next section we will obtain a better coreset which allows us to reduce the final approximation factor to the desired $\alpha+O(\epsilon)$ value. The coreset construction algorithm operates as follows. The set $P$ is partitioned into $L$ equally-sized subsets $P_1,\ldots,P_L$. In parallel, on each $k$-median instance $\mathcal{I}_\ell = (P_\ell,k)$, with $\ell = 1,\ldots,L$, the following operations are performed:

1. Compute a set $T_\ell$ of $m \geq k$ points such that $\nu_{P_\ell}(T_\ell) \leq \beta \cdot \nu_{P_\ell}(\mathrm{opt}_{\mathcal{I}_\ell})$.
2. $R_\ell \longleftarrow \nu_{P_\ell}(T_\ell)/|P_\ell|$.
3. $C_{w,\ell} \longleftarrow \texttt{CoverWithBalls}(P_\ell, T_\ell, R_\ell, \epsilon, \beta)$.

The set $C_w = \cup_{\ell=1}^{L} C_{w,\ell}$ is the output of the algorithm.

In Step 1, the set $T_\ell$ can be computed through a sequential (possibly bi-criteria) approximation algorithm for $m$-median, with a suitable $m \geq k$, to yield a small value of $\beta$. If we assume that such an algorithm requires space linear in $P_\ell$, the entire coreset costruction can be implemented in a single MapReduce round, using $O(|P|/L)$ local memory and $O(|P|)$ aggregate memory. For example, using one of the known linear-space, constant-approximation algorithms (e.g., local search [2]), we can get $\beta = O(1)$ with $m = k$.

▶ **Lemma 4.3.1.** *For $\ell = 1, \ldots, L$, $C_{w,\ell}$ is an $\epsilon$-bounded coreset of the $k$-median instance $\mathcal{I}_\ell$.*

**Proof.** Fix a value of $\ell$. Let $\tau_\ell$ be the map between the points in $C_{w,\ell}$ and the points in $P_\ell$ of Lemma 3.3.1. The set $C_{w,\ell}$ is weighted according to $\tau_\ell$. Also, it holds that:

$$\sum_{x \in P_\ell} d(x, \tau_\ell(x)) \leq \frac{\epsilon}{2\beta} \sum_{x \in P_\ell} (R_\ell + d(x, T_\ell)) \leq \frac{\epsilon}{2\beta} \left( R_\ell \cdot |P_\ell| + \nu_{P_\ell}(T_\ell) \right) \leq \epsilon \cdot \nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$$

◀

By combining Lemma 4.3.1 and Lemma 3.2.8, the next lemma immediately follows.

▶ **Lemma 4.3.2.** *Let $\mathcal{I} = (P, k)$ be a $k$-median instance. The set $C_w$ returned by the above MapReduce algorithm is a $2\epsilon$-bounded coreset of $\mathcal{I}$.*

It is possible to bound the size of $C_w$ as a function of the doubling dimension $\texttt{dd}(P)$. For any $\ell = 1, \ldots, L$ and $x \in P_\ell$, it holds that $R_\ell \cdot |P_\ell| = \nu_{P_\ell}(T_\ell) \geq d(x, T_\ell)$, thus we can bound the size of $C_{w,\ell}$ by using Theorem 3.3.3 (with $c = |P| \geq |P_\ell|$ and $\mathcal{M} = P$) Since $C_w$ is the union of those sets, this argument proves the following lemma.

▶ **Lemma 4.3.3.** *Let $\mathcal{I} = (P, k)$ be a $k$-median instance. Then, the set $C_w$ returned by the above MapReduce algorithm has size $|C_w| = O\left( L \cdot m \cdot (16\beta/\epsilon)^{\texttt{dd}(P)} \log |P| \right)$*

Let $S$ be an $\alpha$-approximate solution of $\mathcal{I}' = (C_w, k)$, with constant $\alpha$. We will now show that $\nu_P(S)/\nu_P(\text{opt}_{\mathcal{I}}) = 2\alpha + O(\epsilon)$. Let $\tau$ be the map of from $P$ to $C_w$ (see Lemma 3.3.1). By triangle inequality, $\nu_P(S) \leq \sum_{x \in P} d(x, \tau(x)) + \nu_{C_w}(S)$. We have that $\sum_{x \in P} d(x, \tau(x)) \leq 2\epsilon \cdot \nu_P(\text{opt}_{\mathcal{I}})$ since, by Lemma 4.3.2, $C_w$ is a $2\epsilon$-bounded coreset. By the fact that $S$ is an $\alpha$-approximate solution of $\mathcal{I}'$ and by Lemma 3.2.7, we have that $\nu_{C_w}(S) \leq \alpha \cdot \nu_{C_w}(\text{opt}_{\mathcal{I}'}) \leq 2\alpha \cdot \nu_{C_w}(\text{opt}_{\mathcal{I}})$. By Lemma 3.2.5, $C_w$ is also a $2\epsilon$-approximate coreset of $\mathcal{I}$, thus $\nu_{C_w}(\text{opt}_{\mathcal{I}}) \leq (1 + 2\epsilon)\nu_P(\text{opt}_{\mathcal{I}})$. Putting it all together, we have that $\nu_P(S)/\nu_P(\text{opt}_{\mathcal{I}}) \leq 2\alpha(1 + 2\epsilon) + 2\epsilon = 2\alpha + O(\epsilon)$. We observe that the factor 2 is due to the inequality which relates $\text{opt}_{\mathcal{I}}$ and $\text{opt}_{\mathcal{I}'}$, namely $\nu_{C_w}(\text{opt}_{\mathcal{I}'}) \leq 2\nu_{C_w}(\text{opt}_{\mathcal{I}})$. In the next section, we will show how to get rid of this factor.

**Application to the continuous case** The same algorithm of this section can also be used to build a $O(\epsilon)$-bounded (Definition 3.2.4) coreset in the continuous scenario where centers are not required to belong to $P$. It is easy to verify that the construction presented in this section also works in the continuous case, with the final approximation factor improving to $(\alpha + O(\epsilon))$. Indeed, we can use the stronger

inequality $\nu_{C_w}(\text{opt}_{\mathcal{I}'}) \leq \nu_{C_w}(\text{opt}_{\mathcal{I}})$, as $\text{opt}_{\mathcal{I}}$ is also a solution of $\mathcal{I}'$, which allows us to avoid the factor 2 in front of $\alpha$. While the same approximation guarantee has already been achieved in the literature using more space-efficient but randomized coreset constructions [6, 4], as mentioned in the introduction, this result provides evidence of the general applicability of our approach.

## 4.4 Coreset construction for $k$-median

In this section, we present a 1-round MapReduce algorithm which computes a weighted subset which is both an $O(\epsilon)$-bounded coreset and an $O(\epsilon)$-centroid set of an input instance $\mathcal{I} = (P, k)$ of $k$-median. The algorithm is similar to the one of the previous section but applies `CoverWithBalls` twice in every subset of the partition. This idea is inspired by the strategy presented in [16] for $\mathbb{R}^d$, where a double exponential grid construction is used to ensure that the returned subset is a centroid set. Specifically, our MapReduce algorithm partitions $P$ into $L$ equally-sized subsets $P_1, \ldots, P_L$. In parallel, on each $k$-median instance $\mathcal{I}_\ell = (P_\ell, k)$, with $\ell = 1, \ldots, L$, the following steps are performed:

1. Compute a set $T_\ell$ of $m$ points such that $\nu_{P_\ell}(T_\ell) \leq \beta \cdot \nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$.
2. $R_\ell \longleftarrow \nu_{P_\ell}(T_\ell)/|P_\ell|$.
3. $C_{w,\ell} \longleftarrow$ `CoverWithBalls`$(P_\ell, T_\ell, R_\ell, \epsilon, \beta)$.
4. $E_{w,\ell} \longleftarrow$ `CoverWithBalls`$(P_\ell, C_{w,\ell}, R_\ell, \epsilon, \beta)$.

The set $E_w = \cup_{\ell=1}^{L} E_{w,\ell}$ is the output of the algorithm. The computation of $T_\ell$ can be accomplished as described in the previous section. The local memory of the algorithm is $O(|P|/L)$ and the aggregate memory is $O(|P|)$.

The following two lemmas characterize the properties and the size of $E_w$.

▶ **Lemma 4.4.1.** *Let $\mathcal{I} = (P, k)$ be a $k$-median instance. Then, the set $E_w$ returned by the above MapReduce algorithm is both a $2\epsilon$-bounded coreset and a $7\epsilon$-centroid set of $\mathcal{I}$.*

**Proof.** The first three steps of the algorithm are in common with the algorithm of Section 4.4. By Lemma 4.3.1, for $\ell = 1, ..., L$, the sets $C_{w,\ell}$ are $\epsilon$-bounded coresets of $\mathcal{I}_\ell$. Let $C_w = \cup_{\ell=1}^{L} C_{w,\ell}$. By Lemma 3.2.8, the set $C_w$ is a $2\epsilon$-bounded coreset of $\mathcal{I}$, and also, by Lemma 3.2.5, a $2\epsilon$-approximate coreset. Let $\tau(x)$ and $\tau_\ell(x)$ be the maps respectively from $P$ to $C_w$, and from $P_\ell$ to $C_{w,\ell}$, as specified in Definition 3.2.4. It holds that $\nu_P(C_w) \leq \sum_{x \in P} d(x, \tau(x)) \leq 2\epsilon \cdot \nu_P(\text{opt}_{\mathcal{I}})$ and that $\nu_{P_\ell}(C_{w,\ell}) \leq \sum_{x \in P_\ell} d(x, \tau_\ell(x)) \leq \epsilon \cdot \nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$. Let $\phi_\ell$ be the map of Lemma 3.3.1 from the points in $P_\ell$ to the points in $E_{w,\ell}$. By reasoning as in the proof of Lemma 4.3.1, we obtain that $\sum_{x \in P_\ell} d(x, \phi_\ell(x)) \leq$

$\epsilon/(2\beta)\left[|P_\ell|\cdot R_\ell+\nu_{P_\ell}(C_{w,\ell})\right]$. Since $\beta\geq 1$ and $|P_\ell|\cdot R_\ell=\nu_{P_\ell}(T_\ell)\leq\beta\cdot\nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$, we conclude that $\sum_{x\in P_\ell}d(x,\phi_\ell(x))\leq\epsilon\cdot\nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$. Therefore, $E_{w,\ell}$ is a $\epsilon$-bounded coreset of $\mathcal{I}_\ell$ and, by Lemma 3.2.8, $E_w$ is a $2\epsilon$-bounded coreset.

We now show that $E_w$ is a $7\epsilon$-centroid set of $\mathcal{I}$. Let $X=\{x^{E_w}:x\in\text{opt}_{\mathcal{I}}\}$. We will prove that $\nu_P(X)\leq(1+7\epsilon)\nu_P(\text{opt}_{\mathcal{I}})$. By triangle inequality, we obtain that:

$$\nu_P(X)=\sum_{x\in P}d(x,X)\leq\sum_{x\in P}d(x,\tau(x))+\sum_{x\in P}d(\tau(x),X)$$

The first term of the above sum can be bounded as $\sum_{x\in P}d(x,\tau(x))\leq 2\epsilon\cdot\nu_P(\text{opt}_{\mathcal{I}})$, since $C_w$ is a $2\epsilon$-bounded coreset. Also, we notice that the second term of the sum can be rewritten as $\sum_{x\in P}d(\tau(x),X)=\sum_{x\in C_w}w(x)d(x,X)$, due to the relation between $\tau$ and $w$. By triangle inequality, we obtain that:

$$\sum_{x\in C_w}w(x)d(x,X)\leq\sum_{x\in C_w}w(x)d(x,x^{\text{opt}_{\mathcal{I}}})+\sum_{x\in C_w}w(x)d(x^{\text{opt}_{\mathcal{I}}},X)$$

Since $C_w$ is a $2\epsilon$-approximate coreset, we can use the bound $\sum_{x\in C_w}w(x)d(x,x^{\text{opt}_{\mathcal{I}}})=\nu_{C_w}(\text{opt}_{\mathcal{I}})\leq(1+2\epsilon)\nu_P(\text{opt}_{\mathcal{I}})$. Fix a point $x\in C_w$ and let $\ell$ be the index such that $x^{\text{opt}_{\mathcal{I}}}\in P_\ell$. By definition of $X$, we have that $d(x^{\text{opt}_{\mathcal{I}}},X)\leq d(x^{\text{opt}_{\mathcal{I}}},\phi_\ell(x^{\text{opt}_{\mathcal{I}}}))$, and by definition of $\phi_\ell$, $d(x^{\text{opt}_{\mathcal{I}}},\phi_\ell(x^{\text{opt}_{\mathcal{I}}}))\leq\epsilon/(2\beta)\cdot\left[R_\ell+d(x^{\text{opt}_{\mathcal{I}}},C_{w,\ell})\right]$. We have:

$$\begin{aligned}\sum_{x\in C_w}w(x)d(x^{\text{opt}_{\mathcal{I}}},X)&\leq\frac{\epsilon}{2\beta}\sum_{\ell=1}^{L}\sum_{x\in C_{w,\ell}}w(x)\left[R_\ell+d(x^{\text{opt}_{\mathcal{I}}},C_{w,\ell})\right]\\&\leq\frac{\epsilon}{2\beta}\left(\sum_{\ell=1}^{L}|P_\ell|\cdot R_\ell+\sum_{\ell=1}^{L}\sum_{x\in C_{w,\ell}}w(x)d(x,\text{opt}_{\mathcal{I}})\right)\end{aligned}$$

In the last inequality we used the simple observation that for any $x\in C_{w,\ell}$, $d(x^{\text{opt}_{\mathcal{I}}},C_{w,\ell})\leq d(x,x^{\text{opt}_{\mathcal{I}}})=d(x,\text{opt}_{\mathcal{I}})$. As argued previously in the proof, $|P_\ell|\cdot R_\ell\leq\nu_{P_\ell}(T_\ell)\leq\beta\cdot\nu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})\leq 2\beta\cdot\nu_{P_\ell}(\text{opt}_{\mathcal{I}})$, where the last inequality follows from Lemma 3.2.7. Also, $\sum_{x\in C_{w,\ell}}w(x)d(x,\text{opt}_{\mathcal{I}})=\nu_{C_{w,\ell}}(\text{opt}_{\mathcal{I}})$. By using the fact that $\sum_{\ell=1}^{L}\nu_{P_\ell}(\text{opt}_{\mathcal{I}})=\nu_P(\text{opt}_{\mathcal{I}})$, and that $\sum_{\ell=1}^{L}\nu_{C_{w,\ell}}(\text{opt}_{\mathcal{I}})=\nu_{C_w}(\text{opt}_{\mathcal{I}})\leq(1+2\epsilon)\nu_P(\text{opt}_{\mathcal{I}})$, we finally obtain:

$$\sum_{x\in C_w}w(x)d(x^{\text{opt}_{\mathcal{I}}},X)\leq\frac{\epsilon}{2\beta}(2\beta+1+2\epsilon)\nu_P(\text{opt}_{\mathcal{I}})\leq 3\epsilon\cdot\nu_P(\text{opt}_{\mathcal{I}})$$

We conclude that $\nu_P(X)\leq(2\epsilon+1+2\epsilon+3\epsilon)\nu_P(\text{opt}_{\mathcal{I}})=(1+7\epsilon)\cdot\nu_P(\text{opt}_{\mathcal{I}})$ ◀

▶ **Lemma 4.4.2.** *Let $\mathcal{I} = (P,k)$ be a $k$-median instance. Let $E_w$ be the set returned by the above MapReduce algorithm with input $\mathcal{I}$. Then $|E_w| = O\left(L \cdot m \cdot (16\beta/\epsilon)^{2\mathtt{dd}(P)} \log^2 |P|\right)$.*

**Proof.** From the previous section, we know that $|C_{w,\ell}| = O\left(m \cdot (16\beta/\epsilon)^{\mathtt{dd}(P)} \log |P|\right)$. Since $C_{w,\ell}$ is an $\epsilon$-bounded coreset of $\mathcal{I}_\ell$, for every $x \in P_\ell$ we have that $\epsilon |P_\ell| \cdot R_\ell = \epsilon \cdot \nu_{P_\ell}(T_\ell) \geq \epsilon \cdot \nu_{P_\ell}(\text{opt}_\mathcal{I}) \geq \nu_{P_\ell}(C_{w,\ell}) \geq d(x, C_{w,\ell})$. The lemma follows by applying Theorem 3.3.3 to bound the sizes of the sets $E_{w,\ell}$. ◀

We are now ready to state the main result of this section.

▶ **Theorem 4.4.3.** *Let $\mathcal{I} = (P,k)$ be a $k$-median instance and let $E_w$ be the set returned by the above MapReduce algorithm for a fixed $\epsilon \in (0,1)$. Let $\mathcal{A}$ be an $\alpha$-approximation algorithm for the $k$-median problem, with constant $\alpha$. If $S$ is the solution returned by $\mathcal{A}$ with input $\mathcal{I}' = (E_w, k)$, then $\nu_P(S)/\nu_P(\text{opt}_\mathcal{I}) \leq \alpha + O(\epsilon)$.*

**Proof.** Let $\tau$ be the map from $P$ to $E_w$ of Definition 3.2.4. By triangle inequality, it results that $\nu_P(S) \leq \sum_{x \in P} d(x, \tau(x)) + \nu_{E_w}(S)$. The set $E_w$ is a $2\epsilon$-bounded coreset of $\mathcal{I}$, so we have that $\sum_{x \in P} d(x, \tau(x)) \leq 2\epsilon \cdot \nu_P(\text{opt}_\mathcal{I})$. Since $\mathcal{A}$ is an $\alpha$-approximation algorithm, we have that $\nu_{E_w}(S) \leq \alpha \cdot \nu_{E_w}(\text{opt}_{\mathcal{I}'})$. As $E_w$ is also a $7\epsilon$-centroid set, there exists a solution $X \subseteq E_w$ such that $\nu_P(X) \leq (1+7\epsilon)\nu_P(\text{opt}_\mathcal{I})$. We obtain that $\nu_{E_w}(\text{opt}_{\mathcal{I}'}) \leq \nu_{E_w}(X) \leq (1+2\epsilon)(1+7\epsilon)\nu_P(\text{opt}_\mathcal{I})$. In the last inequality, we used the fact that $E_w$ is a $2\epsilon$-approximate coreset of $\mathcal{I}$ due to Lemma 3.2.5. To wrap it up, $\nu_P(X)/\nu_P(\text{opt}_\mathcal{I}) \leq \alpha(1+7\epsilon)(1+2\epsilon) + 2\epsilon = \alpha + O(\epsilon)$. ◀

## 4.5 Coreset construction for $k$-means

In this section, we present a 1-round MapReduce algorithm to compute a weighted subset $E_w$ which is both an $O(\epsilon^2)$-approximate coreset and an $O(\epsilon)$-centroid set of an instance $\mathcal{I}$ of $k$-means and then show that an $\alpha$-approximate solution of $\mathcal{I}' = (E_w, k)$ is an $(\alpha + O(\epsilon))$-approximate solution of $\mathcal{I}$. The algorithm is an adaptation of the one devised in the previous section for $k$-median, with suitable tailoring of the parameters involved to account for the presence of squared distances in the objective function of $k$-means. As before, the input set $P$ is partitioned into $L$ subsets $P_1, \ldots, P_L$ and then, in parallel for each $k$-means instance $\mathcal{I}_\ell = (P_\ell, k)$, the following operations are performed:

1. Compute a set $T_\ell$ of $m$ points such that $\mu_{P_\ell}(T_\ell) \leq \beta \cdot \mu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})$.
2. $R_\ell \longleftarrow \sqrt{\mu_{P_\ell}(T_\ell)/|P_\ell|}$.

3. $C_{w,\ell} \longleftarrow$ `CoverWithBalls`$(P_\ell, T_\ell, R_\ell, \sqrt{2}\epsilon, \sqrt{\beta})$.
4. $E_{w,\ell} \longleftarrow$ `CoverWithBalls`$(P_\ell, C_{w,\ell}, R_\ell, \sqrt{2}\epsilon, \sqrt{\beta})$.

The set $E_w = \cup_{\ell=1}^{L} E_{w,\ell}$ is the output of the algorithm. As for $k$-median, the set $T_\ell$ can be computed through a sequential (possibly bi-criteria) approximation algorithm for $m$-means, with a suitable $m \geq k$, to yield a small value of $\beta$. If the computation of $T_\ell$ requires linear space in $|P_\ell|$ (e.g., using the local search algorithm [14, 22]), then the MapReduce algorithm uses $O(|P|/L)$ local memory and $O(|P|)$ aggregate memory. The analysis follows the lines of the one carried out for the $k$-median coreset construction.

The following lemma establishes the properties of each $C_{w,\ell}$.

▶ **Lemma 4.5.1.** *For $\ell = 1, \ldots, L$, $C_{w,\ell}$ is an $\epsilon^2$-bounded coreset of the $k$-means instance $\mathcal{I}_\ell$.*

**Proof.** Fix a value of $\ell$. Let $\tau_\ell$ be the map between the points in $C_{w,\ell}$ and the points in $P_\ell$ of Lemma 3.3.1. The set $C_{w,\ell}$ is weighted according to $\tau_\ell$. Also, it holds that:

$$\sum_{x \in P_\ell} d(x, \tau_\ell(x))^2 \leq \frac{\epsilon^2}{2\beta} \sum_{x \in P_\ell} \left[ R_\ell^2 + d(x, T_\ell)^2 \right] \leq \frac{\epsilon^2}{2\beta} \left[ R_\ell^2 \cdot |P_\ell| + \mu_{P_\ell}(T_\ell) \right] \leq \epsilon^2 \cdot \mu_{P_\ell}(\mathrm{opt}_{\mathcal{I}_\ell})$$

◀

Next, in the following two lemmas, we characterize the properties and the size of $E_w$.

▶ **Lemma 4.5.2.** *Let $\mathcal{I} = (P, k)$ be a $k$-means instance and assume that $\epsilon$ is a positive value such that $\epsilon + \epsilon^2 \leq 1/8$. Then, the set $E_w$ returned by the above MapReduce algorithm is both a $4\epsilon^2$-bounded coreset and a $27\epsilon$-centroid set of $\mathcal{I}$.*

**Proof.** By Lemma 4.5.1, $C_{w,\ell}$ is an $\epsilon^2$-bounded coreset of $P_\ell$, thus $\mu_{P_\ell}(C_{w,\ell}) \leq \epsilon^2 \cdot \mu_{P_\ell}(\mathrm{opt}_{\mathcal{I}_\ell})$. Let $\phi_\ell$ be the map of Lemma 3.3.1 from the points in $P_\ell$ to the points in $E_{w,\ell}$. We have that $\sum_{x \in P_\ell} d(x, \phi_\ell(x))^2 \leq \epsilon^2/(2\beta) \left( |P_\ell| \cdot R_\ell^2 + \mu_{P_\ell}(C_{w,\ell}) \right)$. Using the fact that $|P_\ell| \cdot R_\ell^2 = \mu_{P_\ell}(T_\ell) \leq \beta \cdot \mu_{P_\ell}(\mathrm{opt}_{\mathcal{I}_\ell})$, we conclude that $\sum_{x \in P_\ell} d(x, \phi_\ell(x))^2 \leq \epsilon^2 \cdot \mu_{P_\ell}(\mathrm{opt}_{\mathcal{I}_\ell})$. Therefore, $E_{w,\ell}$ is an $\epsilon^2$-bounded coreset of $\mathcal{I}_\ell$ and, by Lemma 3.2.8, $E_w$ is a $4\epsilon^2$-bounded coreset of $\mathcal{I}$.

We now show that $E_w$ is a centroid set of $\mathcal{I}$. Let $X = \{x^{E_w} : x \in \mathrm{opt}_{\mathcal{I}}\}$ and let $C_w = \cup_\ell C_{w,\ell}$. By Lemma 4.5.1 and Lemma 3.2.8, $C_w$ is a $4\epsilon^2$-bounded coreset. Also by Lemma 3.2.6, $C_w$ is a $\gamma$-approximate coreset of $\mathcal{I}$, with $\gamma = 4(\epsilon + \epsilon^2) \leq 1/2$. Hence, $\mu_P(X) \leq 1/(1-\gamma) \cdot \mu_{C_w}(X)$. By Proposition 3.2.1, we have:

$$\mu_{C_w}(X) = \sum_{x \in C_w} w(x) d(x, X)^2 \leq (1+\epsilon) \mu_{C_w}(\mathrm{opt}_{\mathcal{I}}) + (1 + 1/\epsilon) \sum_{x \in C_w} w(x) d(x^{\mathrm{opt}_{\mathcal{I}}}, X)^2$$

Since $C_w$ is a $\gamma$-approximate coreset, it holds that $\mu_{C_w}(\text{opt}_{\mathcal{I}}) \leq (1+\gamma)\mu_P(\text{opt}_{\mathcal{I}})$. By reasoning as in the proof of Lemma 4.4.1, we have that $\sum_{x\in C_w} w(x)d(x^{\text{opt}_{\mathcal{I}}}, X)^2 \leq (5\epsilon^2/2+\gamma\epsilon^2/2)\mu_P(\text{opt}_{\mathcal{I}})$. Putting it all together, we conclude:

$$\mu_P(X)/\mu_P(\text{opt}_{\mathcal{I}}) \leq \left(1+\gamma+5\epsilon^2/2+\gamma\epsilon^2/2+7\epsilon/2+3\gamma\epsilon/2\right)/(1-\gamma).$$

Since $\gamma \leq 1/2$, we have that $1/(1-\gamma) \leq 1+2\gamma$. By using the constraint on $\epsilon$ and the definition of $\gamma$, after some tedious computations, we obtain $\mu_P(X)/\mu_P(\text{opt}_{\mathcal{I}}) \leq 1+27\epsilon$. ◀

▶ **Lemma 4.5.3.** *Let $\mathcal{I} = (P,k)$ be a $k$-means instance. Let $E_w$ be the set returned by the above MapReduce algorithm with input $\mathcal{I}$. Then, $|E_w| = O\left(L \cdot m \cdot (8\sqrt{2\beta}/\epsilon)^{2\,\mathtt{dd}(P)} \log^2 |P|\right)$*

**Proof.** For any $\ell = 1,\ldots,L$ and $x \in P_\ell$, it holds that $R_\ell \cdot \sqrt{|P_\ell|} = \sqrt{\mu_{P_\ell}(T_\ell)} \geq d(x,T_\ell)$. By using Theorem 3.3.3, we obtain that $|C_{w,\ell}| = O\left(m \cdot (8\sqrt{2\beta}/\epsilon)^{\mathtt{dd}(P)} \log |P|\right)$. Also, for any $x \in P_\ell$ we have that $\sqrt{\epsilon|P_\ell|} \cdot R_\ell = \sqrt{\epsilon \cdot \mu_{P_\ell}(T_\ell)} \geq \sqrt{\epsilon \cdot \mu_{P_\ell}(\text{opt}_{\mathcal{I}_\ell})} \geq \sqrt{\mu_{P_\ell}(C_{w,\ell})} \geq d(x, C_{w,\ell})$. Thus, the lemma follows by applying Theorem 3.3.3 to bound the sizes of the sets $E_{w,\ell}$. ◀

We are now ready to state the main result of this section.

▶ **Theorem 4.5.4.** *Let $\mathcal{I} = (P,k)$ be a $k$-means instance and let $E_w$ be the set returned by the above MapReduce algorithm for a fixed positive $\epsilon$ such that $\epsilon+\epsilon^2 \leq 1/8$. Let $\mathcal{A}$ be an $\alpha$-approximation algorithm for the $k$-means problem, with constant $\alpha$. If $S$ is the solution returned by $\mathcal{A}$ with input $\mathcal{I}' = (E_w,k)$, then $\mu_P(S)/\mu_P(\text{opt}_{\mathcal{I}}) \leq \alpha + O(\epsilon)$.*

**Proof.** By Lemma 4.5.2 and Lemma 3.2.6, $E_w$ is a $(4\epsilon^2+4\epsilon)$-approximate coreset of $\mathcal{I}$. Therefore, $\mu_P(S) \leq (1/(1-4\epsilon-4\epsilon^2)) \cdot \mu_{E_w}(S)$. Since $\mathcal{A}$ is an $\alpha$-approximation algorithm, $\mu_{E_w}(S) \leq \alpha \cdot \mu_{E_w}(\text{opt}_{\mathcal{I}'})$. Also, $E_w$ is a $27\epsilon$-centroid set, thus there exists a solution $X \subseteq E_w$ such that $\mu_P(X) \leq (1+27\epsilon) \cdot \mu_P(\text{opt}_{\mathcal{I}})$. We have that $\mu_{E_w}(\text{opt}_{\mathcal{I}'}) \leq \mu_{E_w}(X) \leq (1+4\epsilon+4\epsilon^2) \cdot \mu_P(X) \leq (1+4\epsilon+4\epsilon^2)(1+27\epsilon) \cdot \mu_P(\text{opt}_{\mathcal{I}})$, where the second inequality follows again from the fact that $E_w$ is a $(4\epsilon^2+4\epsilon)$-approximate coreset of $\mathcal{I}$. Because of the constraints on $\epsilon$, we have that $1/(1-4\epsilon-4\epsilon^2) \leq 1+8\epsilon+8\epsilon^2$. Therefore, it finally results that $\mu_P(S)/\mu_P(\text{opt}_{\mathcal{I}}) \leq \alpha \cdot (1+8\epsilon+8\epsilon^2)(1+4\epsilon+4\epsilon^2)(1+27\epsilon) = \alpha + O(\epsilon)$. ◀

As noted in Section 4.3, a simpler version of this algorithm can be employed if we restrict our attention to the continuous case. Indeed, if we limit the algorithm to the first three steps, and output the set $C_w = \cup_\ell C_{w,\ell}$, it is easy to show that an

$\alpha$-approximate algorithm executed on the coreset $C_w$ returns a $(\alpha+O(\epsilon))$-approximate solution.

## 4.6 MapReduce algorithms for $k$-median and $k$-means

Let $\mathcal{I}=(P,k)$ be a $k$-median (resp., $k$-means) instance. We can compute an approximate solution of $\mathcal{I}$ in two MapReduce rounds: in the first round, a weighted coreset $E_w$ is computed using the algorithm described in Section 4.4 (resp., Section 4.5), while in the second round the final solution is computed by running a sequential approximation algorithm for the weighted variant of the problem on $E_w$. Suppose that in the first round we use a linear-space algorithm to compute the sets $T_\ell$ of size $m=O(k)$, and cost at most a factor $\beta$ times the optimal cost, and that in the second round we run a linear-space $\alpha$-approximation algorithm on $E_w$, with constant $\alpha$. Setting $L=\sqrt{|P|/k}$ we obtain the following theorem as an immediate consequence of Lemmas 4.4.2 and 4.5.3, and Theorems 4.4.3 and 4.5.4.

▶ **Theorem 4.6.1.** *Let $\mathcal{I}=(P,k)$ be an instance of $k$-median (resp., $k$-means). For any $\epsilon\in(0,1)$ (with $\epsilon+\epsilon^2\leq 1/8$ for $k$-means) the 2-round MapReduce algorithm described above computes an $(\alpha+O(\epsilon))$-approximate solution of $\mathcal{I}$ using local space $O\left(\sqrt{|P|k}(16\beta/\epsilon)^{2\mathtt{dd}(P)}\log^2|P|\right)$ (resp., $O\left(\sqrt{|P|k}(8\sqrt{2\beta}/\epsilon)^{2\mathtt{dd}(P)}\log^2|P|\right)$).*

Note that for a wide range of the relevant parameters, the local space of the MapReduce algorithms is substantially sublinear in the input size. As concrete instantiations of the above result, both the $T_\ell$'s and the final solution may be obtained through the local search algorithm, which features approximations $\alpha=3+2/t$ for $k$-median, and $\alpha=5+4/t$ for $k$-means, where $t$ is the number of simultaneous swaps allowed (see Theorem 3.2.2). With this choice, the result of the above theorem holds with $\beta=\alpha=O(1)$. Alternatively, for the $T_\ell$'s we could use $k$-means++ [5] as a bi-criteria approximation algorithm (e.g, see [33]), which yields a smaller $\beta$, at the expense of a slight, yet constant, increase in the size $m$ of the $T_\ell$'s. For larger $\mathtt{dd}(P)$, this might be a better choice as the coreset size (hence the local memory) is linear in $m$ and $\beta^{2\mathtt{dd}(P)}$ (resp., $\beta^{\mathtt{dd}(P)}$). Moreover, bi-criteria approximations are usually faster to compute than actual solutions.

# Chapter 5

# Conclusion

We presented deterministic, distributed coreset constructions that can be used in conjunction with sequential approximation algorithms for $k$-median and $k$-means in general metric spaces to obtain the first space-efficient, 2-round MapReduce algorithms for the two problems, which are almost as accurate as their sequential counterparts. The constructions for the two problems are based on a uniform strategy, and crucially leverage the properties of spaces of bounded doubling dimension, specifically those related to ball coverings of sets of points. The same construction strategy can be applied to obtain a 2-round MapReduce algorithm for the $k$-center problem, whose performance is on par with state-of-the-art approaches. One attractive feature of our constructions is their simplicity, which makes them amenable to fast practical implementations. Moreover, we designed a new sequential algorithm to estimate the doubling dimension of an input point set, slightly improving existing approaches. However, this estimation still has high time complexity, and in practice it is unfeasible on large datasets. To address this problem, we presented a novel MapReduce algorithm, which sacrifices the quality of the approximation in order to parallelize the computation.

**Future work.** The performance of our clustering algorithms depends on the doubling dimension $\texttt{dd}(P)$ of the input. Given a point set, its doubling dimension can range from 0 to $\log_2 |P|$. This being the case, the estimation of the average doubling dimension of real datasets appears to be a very interesting problem. If this value happened to be low, it would strengthen the results achieved in this thesis, and provide evidence of the applicability of methods based on the doubling dimension. In our case, we would obtain coresets for $k$-median and $k$-means with sizes substantially linear in $O(1/\epsilon^{O(1)})$, which is on par with state-of-the-art approaches for the continuous variants in the Euclidean Space. The MapReduce algorithm in Chapter 2 aims at solving this problem. In future work, we would like to perform pratical experimentations in order to evaluate

its performance and provide a first answer to the problem. Furthermore, it is an open question whether it is possible to obtain a lower approximation factor than 2 on the estimation of the doubling dimension.

It would be also interesting to generalize our approach for the $(k, z)$-clustering problem. The goal is to identify a set of $k$ centers such that the sum of the $z$-th power of the distances from any point of the input to its closest center is minimized ($k$-median for $z = 1$, $k$-means for $z = 2$, and $k$-center for $z = \infty$). The definition of bounded coreset and our coreset construction primitive could be easily adapted for this problem, however the analysis of the approximation factor obtained by working with the coreset remains unclear.

# Bibliography


[1] Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *Proc. 18th ACM-SIAM SODA*, pages 1027–1035.

[2] Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. (2004). Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562.

[3] Awasthi, P. and Balcan, M. (2015). Center based clustering: A foundational perspective. In *Handbook of cluster analysis*. CRC Press.

[4] Bachem, O., Lucic, M., and Krause, A. (2018). Scalable k -means clustering via lightweight coresets. In *Proc. 24th ACM KDD*, pages 1119–1127.

[5] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. (2012). Scalable k-means++. *PVLDB*, 5(7):622–633.

[6] Balcan, M., Ehrlich, S., and Liang, Y. (2013). Distributed k-means and k-median clustering on general communication topologies. In *Proc. 27th NIPS*, pages 1995–2003.

[7] Ceccarello, M., Pietracaprina, A., and Pucci, G. (2019). Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially. *PVLDB*, 12(7).

[8] Cohen, E., Chechik, S., and Kaplan, H. (2018). Clustering small samples with quality guarantees: Adaptivity with one2all PPS. In *Proc. 32nd AAAI*, pages 2884–2891.

[9] Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113.

[10] Ene, A., Im, S., and Moseley, B. (2011). Fast Clustering Using MapReduce. In *Proc. 17th ACM KDD*, pages 681–689.

[11] Feldman, D. and Langberg, M. (2011). A unified framework for approximating and clustering data. In *Proc. 43rd ACM STOC*, pages 569–578.

[12] Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306.

[13] Gottlieb, L. and Krauthgamer, R. (2014). Proximity algorithms for nearly doubling spaces. *SIAM J. Discrete Math*, 27(4):1759–1769.

[14] Gupta, A. and Tangwongsan, K. (2008). Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554.

[15] Har-Peled, S. and Kushal, A. (2005). Smaller coresets for k-median and k-means clustering. In *Proc. 21st SCG*, pages 126–134.

[16] Har-Peled, S. and Mazumdar, S. (2004). On coresets for k-means and k-median clustering. In *Proc. 36th ACM STOC*, pages 291–300.

[17] Har-Peled, S. and Mendel, M. (2006). Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184.

[18] Hennig, C., Meila, M., Murtagh, F., and Rocci, R. (2015). *Handbook of cluster analysis*. CRC Press.

[19] Huang, L., Jiang, S., Li, J., and Wu, X. (2018). Epsilon-coresets for clustering (with outliers) in doubling metrics. In *Proc. 59th IEEE FOCS*, pages 814–825.

[20] Indyk, P., Mahabadi, S., Mahdian, M., and Mirrokni, V. (2014). Composable Core-sets for Diversity and Coverage Maximization. In *Proc. 33rd ACM PODS*, pages 100–108.

[21] Jain, K., Mahdian, M., and Saberi, A. (2002). A new greedy approach for facility location problems. In *Proc. 34th STOC*, pages 731–740.

[22] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). A local search approximation algorithm for k-means clustering. In *Proc. 18th SCG*, pages 10–18.

[23] Kaufmann, L. and Rousseeuw, P. (1987). Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416.

[24] Krauthgamer, R. and Lee, J. R. (2004). Navigating nets: Simple algorithms for proximity search. In *Proc. 15th SODA*, pages 798–807.

[25] Lee, E., Schmidt, M., and Wright, J. (2017). Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43.

[26] Leskovec, J., Rajaraman, A., and Ullman, J. (2014). *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press.

[27] Malkomes, G., Kusner, M. J., Chen, W., Weinberger, K. Q., and Moseley, B. (2015). Fast distributed k-center clustering with outliers on massive data. In *Proc. 28th NIPS*, pages 1063–1071.

[28] Phillips, J. M. (2016). Coresets and sketches. *Handbook of Discrete and Computational Geometry, 3rd Ed.*

[29] Pietracaprina, A., Pucci, G., Riondato, M., Silvestri, F., and Upfal, E. (2012). Space-Round Tradeoffs for MapReduce Computations. In *Proc. 26th ACM ICS*, pages 235–244.

[30] Serna, M., Shaltiel, R., Jansen, K., and Rolim, J. (2009). Approximation, randomization, and combinatorial optimization. algorithms and techniques. *Lecture Notes in Computer Science*.

[31] Sohler, C. and Woodruff, D. P. (2018). Strong coresets for k-median and subspace approximation: Goodbye dimension. In *Proc. 59th IEEE FOCS*, pages 802–813.

[32] Song, H., Lee, J., and Han, W. (2017). PAMAE: parallel $k$-medoids clustering with high accuracy and efficiency. In *Proc. 23rd ACM KDD*, pages 1087–1096.

[33] Wei, D. (2016). A constant-factor bi-criteria approximation guarantee for k-means++. In *Proc. 30th NIPS*, pages 604–612.