Simple and Sharp Analysis of k-means||

Václav Rozhoň [1]

Abstract

We present a simple analysis of k-means|| (Bahmani et al., PVLDB 2012) – a distributed variant of the k-means++ algorithm (Arthur and Vassilvitskii, SODA 2007). Moreover, the bound on the number of rounds is improved from $O(\log n)$ to $O(\log n / \log\log n)$, which we show to be tight.

1. Introduction

Clustering is one of the classical machine learning problems. Arguably the simplest and most basic formalization of clustering is the k-means formulation: we are given a (large) set of points X in the Euclidean space and are asked to find a (small) set of k centers C so as to minimize the sum of the squared distances between each point and the closest center. That is, we minimize

$$\varphi_X(C) = \sum_{x \in X} \min_{c_i \in C} \|x - c_i\|^2,$$

Due to its simplicity, k-means is considered as the problem that tests our understanding of clustering.

The classical, yet still state-of-the-art algorithm k-means++ (Arthur & Vassilvitskii, 2007a) combines two ideas to approach the problem. First, a fast randomized procedure finds a set of k centers that by itself is known to be $O(\log k)$ competitive in expectation with respect to the optimal solution. Then, the classical Lloyd's algorithm (Lloyd, 1982) is run to improve the found solution until a local minimum is achieved.

A significant disadvantage of the k-means++ is the inherent sequential nature of the first, seeding step: one needs to pass through the whole data k times, each time to sample a single center. To overcome this problem, (Bahmani et al., 2012) devised k-means|| : a distributed version of the k-means++ algorithm.

[1]ETH, Zurich. Correspondence to: Václav Rozhoň <rozhonv@ethz.ch>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

In their algorithm one passes through the dataset only few times to extract a set of roughly $O(k)$ candidate centers, from which one later chooses the final k centers by the means of the classical k-means++ algorithm.

Our contribution In this work we first provide a new, simple analysis of k-means|| , thus simplifying known proofs (Bahmani et al., 2012) and (Bachem et al., 2017). In particular, if we denote by μ_X the mean of X and φ^* the optimal solution, we prove in Section 3 that $O(\log \frac{\varphi_X(\mu_X)}{\varphi^*})$ rounds of the k-means|| algorithm suffice to get expected constant approximation guarantee. This matches known guarantees.

In Section 4, we then proceed by refining the analysis to provide a better bound on the number of sampling rounds needed by the algorithm: we prove that $O(\log \frac{\varphi_X(\mu_X)}{\varphi^*} / \log\log \frac{\varphi_X(\mu_X)}{\varphi^*})$ rounds suffice. In Section 4, the second bound is shown to be tight for a wide range of values of k by generalisation of a known lower bound (Bachem et al., 2017).

The first analysis of k-means|| (Bahmani et al., 2012) invokes linear programming duality as a part of the argument. Its second analysis (Bachem et al., 2017) is more similar to ours, as it only relies on basic lemmas known from the analysis of k-means++ (Arthur & Vassilvitskii, 2007b). Our one-page analysis is considerably shorter and, we believe, also simpler. It can be summed up as "view one sampling step of the algorithm as a weighted balls into bins process". We explain this in more detail in Section 3.

2. Background and Notation

We mostly adopt the notation of the paper (Bahmani et al., 2012). Let $X = \{x_1, \ldots, x_n\}$ be a point set in the d-dimensional Euclidean space. We denote the standard Euclidean distance between two points x_i, x_j by $\|x_i - x_j\|$ and for a subset $Y \subseteq X$ we define the distance between Y and $x \in X$ as $d(x, Y) = \min_{y \in Y} \|x - y\|$.

For a subset $Y \subseteq X$ we denote by $\mu(Y)$ its centroid, i.e., $\mu(Y) = \frac{1}{|Y|}\sum_{y \in Y} y$. For a set of points $C = \{c_1, \ldots, c_k\}$ and $Y \subseteq X$ we define the *cost* of Y with respect to C as $\varphi_Y(C) = \sum_{y \in Y} d^2(y, C)$ and use a shorthand $\varphi_x(C)$ for $\varphi_{\{x\}}(C)$ and $\varphi_X(c)$ for $\varphi_X(\{c\})$. It is easy to check that for a given point set A, the center x that minimizes its cost $\varphi_A(x)$ is its mean μ_A.

The goal of the k-means problem is to find a set of *centers* C, $|C| = k$ that minimizes the cost $\varphi_X(C)$ for a given set of points X. The k-means problem is known to be NP-hard (Aloise et al., 2009; Mahajan et al., 2009) and even hard to approximate up to small constant precision (Awasthi et al., 2015; Lee et al., 2017). From now on we fix an optimal solution C^* and denote by φ^* its cost. We assume that minimum distance between two points $x \neq y$ from X is at least 1 and that for the maximum distance Δ between two points from X we have $\Delta = O(\log n)$; this implies $\varphi_X(\mu_X) = \text{poly}(n)$ and hence, the simplified complexities $O(\log n)$ and $O(\log n / \log\log n)$ in abstract.

2.1. k-means++ Algorithm

The classical k-means++ algorithm (Arthur & Vassilvitskii, 2007b; Ostrovsky et al., 2013) computes the centers in k sampling steps. After the first step where the first center is taken from uniform distribution, each subsequent step samples a new point from D^2-*distribution*: if C is the current set of centers, x is being sampled with probability $\frac{\varphi_x(C)}{\varphi_X(C)}$, i.e., we sample the points proportional to their current cost.

The k-means++ algorithm is known to provide an $O(\log k)$ approximation guarantee, in expectation. The analysis crucially makes use of the following two lemmas that we will also use.

The first lemma tells us that if we sample uniformly a random point p from some point set A, we expect the cost $\varphi_A(p)$ to be within a constant factor of the cost $\varphi_A(\mu_A)$ which is the smallest cost achievable with one center. One can think of A as being a cluster of the optimal solution or the whole point set X.

Lemma 1 (Lemma 3.1 in (Arthur & Vassilvitskii, 2007a)). *Let A be an arbitrary set of points. If we sample a random point $p \in A$ according to the uniform distribution, we have $\boldsymbol{E}[\varphi_A(\{p\})] \le 2\varphi_A(\mu_A)$.*

The second lemma ensures that up to a constant factor the same guarantee holds even for the D^2 distribution.

Lemma 2 (Lemma 3.2 in (Arthur & Vassilvitskii, 2007a)). *Let A be an arbitrary set of points, C be an arbitrary set of centers and $p \in A$ be a point chosen by D^2 weighting. Then, $\boldsymbol{E}[\varphi_A(C \cup \{p\})] \le 8\varphi_A(\mu_A)$.*

The analysis of k-means++ from the above two lemmas by a careful inductive argument: important observation is that sampling points proportional to their cost implies that we sample from an optimal cluster with probability proportional to its current cost, hence we preferably sample from costly clusters. Still, there is a small probability in each step that we sample from an already "covered" cluster and this is the reason why the final approximation factor is $O(\log k)$ rather than $O(1)$.

Algorithm 1 k-means|| overseeding

Require data X,# rounds t, sampling factor ℓ

```
1: Uniformly sample $x \in X$ and set $C = \{x\}$.
2: for $i \leftarrow 1, 2, \ldots, t$ do
3:     $C' \leftarrow \emptyset$
4:     for $x \in X$ do
5:         Add $x$ to $C'$ with probability $\min\left(1, \frac{\ell\varphi_x(C)}{\varphi_X(C)}\right)$
6:     end for
7:     $C \leftarrow C \cup C'$
8: end for
9: Return $C$
```

2.2. k-means|| Algorithm

The distributed variant of the k-means++ algorithm called k-means||, was introduced in (Bahmani et al., 2012). The algorithm consists of two parts. In the first, *overseeding* part (Algorithm 1), we proceed in t sequential rounds after sampling uniformly a single center as in the first step of k-means++. In each of the t sampling rounds we sample each point of X with probability $\min\left(1, \frac{\ell\varphi_x(C)}{\varphi_X(C)}\right)$, i.e., $\ell \ge 1$ times bigger than the probability of taking the point in k-means++, independently on the other points.

In the second part of the algorithm we collect the set of sampled centers C and create a new, weighted, instance X' of k-means in which the weight of every center is equal to the number of points of X to which the given center is the closest. The new instance is solved, e.g., with k-means++ as in Algorithm 2. One can prove that finding a set C with $\varphi_X(C) = O(\varphi^*)$ in Algorithm 1 implies that the overall approximation guarantee is, up to a constant, the same as the approximation guarantee of the algorithm used in the second part of the algorithm (see (Bachem et al., 2017), proof of Theorem 1, or (Guha et al., 2003), in general), which is, in this case, $O(\log k)$ in expectation.

Hence, the analysis of Algorithm 2 boils down to bounding the number of steps t needed by the Algorithm 1 to achieve constant approximation guarantee for given sampling factor ℓ. The authors of (Bahmani et al., 2012) prove the following.

Theorem 1 (roughly Theorem 1 in (Bahmani et al., 2012)). *If we choose $t = O(\log \frac{\varphi_X(\mu_X)}{\varphi^*})$ and $\ell \ge k$, Algorithm 1 gives a set C with $\boldsymbol{E}[\varphi_X(C)] = O(\varphi^*)$.*

Their result was later reproved in (Bachem et al., 2017). We

Algorithm 2 k-means|| (Bahmani et al., 2012)

Require data X,# rounds t, sampling factor ℓ

```
1: $B \leftarrow$ Result of Algorithm 1 applied to $(X, t, \ell)$
2: for $c \in B$ do
3:     $w_c \leftarrow$ # of points $x \in X$ whose closest center in $B$ is $c$
4: end for
5: $C \leftarrow$ Run $k$-means++ on the weighted instance $(B, w)$
6: Return $C$
```

provide a new, simple proof in Section 3.

2.3. Other Related Work

k-means++ was introduced in (Arthur & Vassilvitskii, 2007a) and a similar method was studied by (Ostrovsky et al., 2013). This direction led to approximation schemes (Jaiswal et al., 2012; 2015), constant approximation results based on additional local search (Lattanzi & Sohler, 2019; Choo et al., 2020), constant approximation bi-criteria results based on sampling more centers (Aggarwal et al., 2009; Ailon et al., 2009; Wei, 2016), approximate k-means++ based on Markov chains (Bachem et al., 2016b;a) or coresets (Bachem et al., 2018), analysis of hard instances (Arthur & Vassilvitskii, 2007a; Brunsch & Röglin, 2013) or under adversarial noise (Bhattacharya et al., 2019). Consult (Celebi et al., 2013) for an overview of different seeding methods for k-means .

There is a long line of work on a related k-median problem. A k-median algorithm of (Mettu & Plaxton, 2012) is quite similar to k-means|| and its analysis is also quite similar to our analysis. We discuss some other related work on k-means more thoroughly at the end of Section 3.

3. Warm-up: a Simple Analysis

In this section we provide a simple analysis of Algorithm 1 based on viewing the process as a variant of the *balls into bins* problem. Recall that in the most basic version of the balls into bins problem, one throws k balls into k bins, each ball to a uniformly randomly chosen bin, and asks, e.g., what is a probability of a certain bin to be hit by a ball. This is equal to $1 - (1 - 1/k)^k \approx 1 - 1/\mathrm{e}$, hence, we expect a constant proportion of the bins to be hit in a single step.

To see the connection to our problem, we first define the notion of *settled* clusters that is similar to notions used e.g. in (Aggarwal et al., 2009; Lattanzi & Sohler, 2019).

Definition 1 (Settled clusters). *We call a cluster A of the optimum solution C^* settled with respect to current solution C, if $\varphi_A(C) \le 10\varphi_A(C^*)$. Otherwise, we call A unsettled with respect to C.*

We view the clusters of C^* as bins and each sampling round of Algorithm 1 as shooting at each bin and hitting it (i.e., making the cluster settled) with some probability. Intuitively, this probability is proportional to the cost of the cluster, since this is how we defined the probability of sampling any point of X. So, we view the process as a more general and repeated variant of the balls into bins process, where the costs of the clusters act like "weights" of the bins and we sample with probability (roughly) proportional to these weights. We prove now that clusters are being settled with probability roughly proportional to their cost (unless they are very costly).

Lemma 3. *Let C be the current set of sampled centers and let A be an unsettled cluster of the optimum solution. The cluster A is* not *made settled in the next iteration of Algorithm 1 with probability at most*

$$\exp\left(-\frac{\ell\varphi_A(C)}{5\varphi_X(C)}\right).$$

Intuitively, for clusters A with $\varphi_A(C) \le \varphi_X(C)/\ell$ the probability of hitting them in one step is of order $\frac{\ell\varphi_A(C)}{\varphi_X(C)}$ (using that $\mathrm{e}^{-x} \approx 1 - x$ for small positive x), while for more costly clusters the probability of hitting them is lower bounded by some constant.

Proof. If we sample a point c from A according to D^2 weights, we have $\mathbf{E}[\varphi_A(C \cup \{c\})] \le 8\varphi_A^*$ by Lemma 2. Hence, by Markov inequality, A is made settled with probability at least $1 - \frac{8}{10} = \frac{1}{5}$. In other words, there is a subset of points $A' \subseteq A$ with $\varphi_{A'}(C) \ge \varphi_A(C)/5$ such that sampling a point from A' makes A settled. If A' contains a point x with $\varphi_x(C) \ge \frac{\varphi_X(C)}{\ell}$, we sample x and make A settled with probability 1. Otherwise, we have

$$\begin{aligned}
\mathrm{P}(A \text{ does not get settled}) &\le \prod_{x \in A'} (1 - \ell\varphi_x(C)/\varphi_X(C)) \\
&\le \exp(-\sum_{x \in A'} \ell\varphi_x(C)/\varphi_X(C)) \\
&\le \exp(-\ell\varphi_A(C)/(5\varphi_X(C)))
\end{aligned}$$

where we used $1 + x \le \mathrm{e}^x$ and $\varphi_{A'}(C) \ge \varphi_A(C)/5$. □

Similarly to the classical balls into bins problem, we can now observe that the total cost of unsettled clusters drops by a constant factor in each step.

From now on we simplify the notation and write φ_Y^t for the cost of the point set Y after t sampling rounds of Algorithm 1. Moreover, by φ_U^t we denote the total cost of yet unsettled clusters after t sampling rounds.

Proposition 1 (roughly Theorem 2 in (Bahmani et al., 2012))**.** *Suppose that $\varphi_X^t \geq 20\varphi^*$. For $\ell \geq k$ we have*

$$\boldsymbol{E}\left[\varphi_U^{t+1}\right] \leq \left(1 - \frac{1}{50}\right)\varphi_U^t.$$

In other words, the expected cost of yet unsettled clusters drops by a constant factor in each iteration.

Proof. We split the unsettled clusters into two groups: a cluster A with $\varphi_A^t \geq \varphi_U^t/(2k)$ we call *heavy* and otherwise we call it *light*. Note that the probability that a heavy cluster A is not settled in $(t+1)$th iteration is by Lemma 3 bounded by

$$\exp\left(\frac{-k\varphi_A^t}{5\varphi_X^t}\right) \leq \exp\left(\frac{-k\varphi_U^t}{10k\varphi_X^t}\right) \leq \exp\left(\frac{-1}{20}\right) \leq \frac{24}{25}$$

where we used that $\varphi_U^t \geq \varphi_X^t/2$: this holds since otherwise more than half the cost of φ_X^t is formed by settled clusters, hence $\varphi_X^t < 20\varphi^*$, contradicting our assumption $\varphi_X^t \geq 20\varphi^*$. Hence, after the sampling step, a heavy cluster A does not contribute to the overall cost of unsettled clusters with probability at least $1/25$. This implies that the expected drop in the cost of unsettled clusters is at least

$$\varphi_U^t - \mathbf{E}[\varphi_U^{t+1}] \geq \sum_{A \text{ heavy}} \varphi_A^t/25$$
$$= \frac{1}{25}\left(\varphi_U^t - \sum_{A \text{ light}} \varphi_A^t\right) \geq \frac{\varphi_U^t}{50}$$

where we used that the light clusters have total cost of at most $k \cdot \varphi_U^t/(2k) = \varphi_U^t/2$. □

Theorem 1 now follows directly (Bahmani et al., 2012; Bachem et al., 2017) and we prove it here for completeness.

Proof of Theorem 1. From Lemma 1 it follows that after we sample a uniformly random point, we have $\mathbf{E}[\varphi^0] \leq 2\varphi_X(\mu_X)$. Proposition 1 then gives $\mathbf{E}[\varphi_U^{t+1}] \leq \frac{49}{50}\varphi_U^t + 20\varphi^*$. Applying this result T times, we get

$$\mathbf{E}[\varphi_U^T] \leq 2\left(\frac{49}{50}\right)^T \varphi_X(\mu_X) + 20\varphi^* \cdot \sum_{t=0}^{T-1}\left(\frac{49}{50}\right)^t$$
$$\leq 2\left(\frac{49}{50}\right)^T \varphi_X(\mu_X) + 1000\varphi^*$$

Choosing $T = O(\log \frac{\varphi_X(\mu_X)}{\varphi^*})$ and recalling that $\varphi^T \leq \varphi_U^T + 10\varphi^*$ yields the desired claim. □

3.1. Additional Remarks

For the sake of simplicity, we did not optimize constants and analysed Algorithm 1 meaningfully only for the case $\ell = \Theta(k)$. In the following remarks we note how one can extend this (or some previous) analysis and then use it to compare $k-\texttt{means}||$ more carefully to a recent line of work.

Remark 1. *With more care, the approximation factor in Theorem 1 can be made arbitrarily close to* 8*. We omit the proof.*

Remark 2. *With more care, for general $\ell = \alpha k$ one can prove that the number of steps of Algorithm 1 needed to sample a set of points that induce a cost of $O(\varphi')$ is $O(\log_\alpha \frac{\varphi_X(\mu_X)}{\varphi'})$ for $\alpha \geq 1$ and $O(\log \frac{\varphi_X(\mu_X)}{\varphi'}/\alpha)$ for $\alpha \leq 1$. We omit the proof.*

3.2. Similar Algorithms with Additive Error

Remark 2 allows us to make a closer comparison of $k-\texttt{means}||$ with a recent line of work of (Bachem et al., 2016a; 2018) that aims for very fast algorithms that allow for an additive error of $\varepsilon\varphi_X(\mu_X)$.

According to Remark 2, to obtain such a guarantee for the oversampled set of centers, Algorithm 1 needs to set $\ell = O(k)$ and sample for $t = \log(1/\varepsilon)$ steps (this was observed in (Bachem et al., 2017)) or it sets $\ell = O(k/\varepsilon)$ points and sample just once (i.e., $t = 1$). The approximation factor of Algorithm 2 is then multiplied by additional factor of $O(\log k)$ as this is the approximation guarantee of $k-\texttt{means++}$.

An approach of Bachem et al. (Bachem et al., 2018) is similar to $k-\texttt{means}||$ with $t = 1$: there, authors propose a coreset algorithm that samples $\tilde{O}(dk/\varepsilon^2)$ points from roughly the same distribution as the one used in the first sampling step of Algorithm 1. If we use their algorithm by running $k-\texttt{means++}$ on the provided coreset, we get an algorithm with essentially the same guarantees as Algorithm 2 with number of rounds $t = 1$ and $\ell = O(k/\varepsilon)$. The main difference is that in Algorithm 2, the weight of each sampled center used by $k-\texttt{means++}$ subroutine is computed as the number of points for which the center is the closest, whereas in the coreset algorithm, each center is simply given a weight inversely proportional to the probability that the center is sampled. This allows the coreset algorithm to be faster than Algorithm 2 with $t = 1$ and $\ell = O(k/\varepsilon)$, whose time complexity in this case is $O(nk/\varepsilon)$. This is at the expense of higher number of sampled points.

A paper of (Bachem et al., 2016a) uses the Metropolis algorithm on top of the classical $k-\texttt{means++}$ algorithm to again achieve additive $\varepsilon\varphi_X(\mu_X)$ (and multiplicative $O(\log k)$) error, while sampling only $O(\frac{k}{\varepsilon}\log\frac{k}{\varepsilon})$ points from the same distribution as Algorithm 2 with $t = 1$ and

$\ell = O(k/\varepsilon)$. While the number of taken samples is only slightly higher than the one of Algorithm 2 with $t = 1, \ell = O(k/\varepsilon)$, their running time is much better $\tilde{O}(k^2/\varepsilon)$.

We see that the main advantage of k-$\texttt{means}||$ lies in the possibility of running multiple, easily distributed, sampling steps that allow us to achieve strong guarantees.

3.3. MPC Context and a Theoretical Implication

Massively Parallel Computing (MPC) (Dean & Ghemawat, 2004; Karloff et al., 2010) is a distributed model in which the set of input points X is split across several machines, each capable of storing $\tilde{O}(s)$ points for some parameter s. In each round, each machine performs some computation on its part of input and sends $\tilde{O}(s)$ bits to other machines such that each machine receives $\tilde{O}(s)$ bits in total. We want to minimize the number of rounds while keeping the memory s small.

Algorithm 2 with t sampling steps can be implemented in $O(t)$ MPC rounds if $s = \Omega(tk)$. Hence, we get an $O(1)$-approximation algorithm for k-means in $O(\log \frac{\varphi_X(\mu(X))}{\varphi^*}) = O(\log n)$ rounds if each machine has $\tilde{\Theta}(k)$ memory.

In the case of per machine memory $O(n^\alpha k)$ for constant $0 < \alpha \le 1$, we can use k-$\texttt{means}||$ to achieve constant approximation guarantee in only $O(1/\alpha)$ rounds, as was noted by (Ghaffari). A hierarchical clustering scheme of Guha et al. (Guha et al., 2003) in this setting needs $O(1/\alpha)$ rounds to output a set of k centers $\bar{C}$, but at the expense of approximation guarantee: we have

$$\varphi_X(\bar{C}) = 2^{O(1/\alpha)}\varphi_X(C^*).$$

k-$\texttt{means}||$ can now recover this loss in approximation: if we run Algorithm 1 but start not with the trivial $C_0 = \{x\}$ for x sampled uniformly at random, but instead take $C_0 = \bar{C}$, Proposition 1 ensures that

$$t = O(\log \frac{\varphi_X(\bar{C})}{\varphi_X(C^*)}) = O(\log 2^{O(1/\alpha)}) = O(1/\alpha)$$

sampling rounds suffice to get a clustering with $O(1)$-approximation guarantee.

3.4. Submodular Context

One can observe why $O(\log \frac{\varphi_X(\mu_X)}{\varphi^*})$ is a natural bound for the number of rounds by considering a different way of achieving the same round complexity. First, note that after sampling the first point c_1, we have $\mathbf{E}[\varphi_X(c_1)] \le 2\varphi_X(\mu_X) = 2\varphi_X(\mu_X)$ via Lemma 1 with the set A chosen as the whole set X. The process of adding new points to the solution now satisfies a natural law of diminishing returns: for any $C_1 \subseteq C_2 \subseteq X$ and $c \in X$ we have

$$\begin{aligned}&\varphi_X(\{c_1\} \cup C_1 \cup \{c\}) - \varphi_X(\{c_1\} \cup C_1)\\ &\ge \varphi_X(\{c_1\} \cup C_2 \cup \{c\}) - \varphi_X(\{c_1\} \cup C_2)\end{aligned}$$

In other words, the function $\varphi_X(\{c_1\}) - \varphi_X(\{c_1\} \cup C)$ is submodular (see e.g. (Krause & Golovin) for the collection of uses of submodularity in machine learning). Then one can use recent results about distributed algorithms for maximizing submodular functions (see e.g. (Mirzasoleiman et al., 2013; Barbosa et al., 2015a;b; Liu & Vondrak, 2018)) to get that in $O(1)$ distributed rounds, one can find a set of k points C such that $\varphi_X(\{c_1\} \cup C) - \varphi^* \le (\varphi_X(\{c_1\}) - \varphi^*)/2$, i.e., the distance to the best solution drops by a constant factor. Continuing the same process for $\log((\varphi_X(\mu_X))/\varphi^*)$ rounds, one gets the same theoretical guarantees as with running Algorithm 2. However, the advantage of k-$\texttt{means}||$ is its extreme simplicity and speed. Moreover, rather surprisingly, we prove in the next section that the asymptotical round complexity of k-$\texttt{means}||$ is actually slightly better than logarithmic.

4. Sharp Analysis of k-$\texttt{means}||$: Upper Bound

It may seem surprising that the analysis of Theorem 1 can be strengthened, since even for the classical balls into bins problem, where we hit each bin with constant probability, we need $O(\log k)$ rounds to hit all the bins with high probability. However, during our process we can disregard already settled clusters since they are not contributing substantially to the overall cost. If we go back to the classical balls into bins problems and let that process repeat on the same set of bins, with the additional property that in each round we throw each one of k balls to a random bin *out of those that are still empty*, we expect to hit all the bins in mere $O(\log^* n)$ steps (Lenzen & Wattenhofer, 2011). Roughly speaking, this is because of the rapid decrease in the number of bins: after the first round, the probability of a bin remain empty is roughly $\frac{1}{2}$, but after second round it is only roughly $\frac{1}{2^2}$ since the number of bins decreased, in the next iteration it is even roughly $\frac{1}{2^{2^2}}$ and so on. In our weighted case we cannot hope for such a rapid decrease in the number of bins, since the costs of clusters can form a geometric series, in which case we get rid of only a small number of clusters in each step (cf. Section 5) [1].

In this section we show a more careful analysis that bounds

[1]We believe it is an interesting problem to analyze whether there are reasonable assumptions on the data under which the round complexity indeed follows $\log^* n$ behaviour (for example, this is the case if clusters are of same size and lie in vertices of a simplex). This could explain why in practice it is enough to run Algorithm 1 only for few rounds (Bahmani et al., 2012).

the number of necessary steps to

$$O\left(\log\frac{\varphi_X(\mu_X)}{\varphi^*}/\log\log\frac{\varphi_X(\mu_X)}{\varphi^*}\right).$$

In the rest of the paper we use the notation $\gamma = \varphi_X(\mu_X)/\varphi^*$.

Concentration The number of steps $O(\log\gamma/\log\log\gamma)$ suffice even with high probability; to prove it, we first recall the classical Chernoff bounds that are used to argue about concentration around mean.

Theorem 2 (Chernoff bounds)**.** *Suppose $X_1, \dots, X_n$ are independent random variables taking values in $\{0,1\}$. Let X denote their sum. Then for any $0 \le \delta \le 1$ we have*

$$P(X \le (1-\delta)\boldsymbol{E}[X]) \le e^{-\boldsymbol{E}[X]\delta^2/2},$$

$$P(X \ge (1+\delta)\boldsymbol{E}[X]) \le e^{-\boldsymbol{E}[X]\delta^2/3},$$

and for $\delta \ge 1$ we have

$$P(X \ge (1+\delta)\boldsymbol{E}[X]) \le e^{-\boldsymbol{E}[X]\delta/3}.$$

Intuition In the following Proposition 2, a refined version of Proposition 1, we argue similarly, but more carefully, about one sampling step of Algorithm 1. The difference is that we analyze not only the drop in the cost of unsettled clusters, but also the drop in the *number* of unsettled clusters. Here is the intuition.

Let us go back to the proof of Proposition 1, where we distinguished heavy and light clusters. Heavy clusters formed at least constant proportion of the cost of all clusters and every heavy cluster was hit with probability that we lower-bounded by $1/25$. For a light cluster we cannot give a good bound for the probability of hitting it, but since it is not very probable that one light cluster is hit by more than one sampled center, if we denote $\alpha = \left(\sum_{A \text{ light}} \varphi_A^t\right)/\varphi_U^t$, i.e., α is the proportional cost of the light clusters, we expect roughly αk clusters to become settled. On the other hand, we may, optimistically, hope that after one iteration the cost of unsettled clusters drops from φ^t to $\alpha\varphi^t$, since the heavy clusters are hit with high probability. This is not exactly the case, since for a heavy cluster we have only a constant probability of hitting it. But we can consider two cases: either there are lot of heavy clusters and we again make αk clusters settled, or their cost is dominated by few, *massive* clusters, each of which is not settled only with exponentially small probability and, hence, we expect the cost to drop by α factor.

The tradeoff between the behaviour of light and heavy clusters then yields a threshold $\alpha \approx 1/\operatorname{poly}\log\gamma$ that balances the drop of the cost and of the number of unsettled clusters.

Proposition 2. *Suppose that after t steps of Algorithm 1 for $\ell = k$ there are k_t unsettled clusters and their total cost is φ_U^t. Assume that $\varphi_X^t \ge 20\varphi^*$. After the next sampling step, with probability at least $1-\exp(\Theta(k^{0.1}))$, we have that either the number of unsettled clusters decreased by at least $k/(40\sqrt{\log\gamma})$, or the total cost of unsettled clusters decreased from φ_U^t to at most $4\varphi_U^t/\sqrt[3]{\log\gamma}$.*

Proof. Note that $\varphi_X^t \ge 20\varphi^*$ implies $\varphi_U^t \ge \varphi_X^t/2$, since otherwise, more than half the cost of φ_X^t would be formed by settled clusters and, hence, $\varphi_X^t < 20\varphi^*$, a contradiction.

We will say that an unsettled cluster is *heavy* if its cost is at least $\varphi_A^t \ge \varphi_U^t/k$ and *light* otherwise. Let $\alpha = (\sum_{A \text{ light}} \varphi_A^t)/\varphi_U^t$, i.e., the proportional cost of the light clusters. We will distinguish three possible cases.

1. $\alpha \ge 1/\sqrt{\log\gamma}$,
2. $\alpha < 1/\sqrt{\log\gamma}$ and there are at least $k/\sqrt{\log\gamma}$ heavy clusters,
3. $\alpha < 1/\sqrt{\log\gamma}$ and there are less than $k/\sqrt{\log\gamma}$ heavy clusters.

For each case we now prove that with probability $1-\exp(\Omega(-k^{0.1}))$ we either settle at least $k/(40\sqrt{\log\gamma})$ clusters or the total cost of unsettled clusters drops from φ_U^t to $4\varphi_U^t/\sqrt[3]{\log\gamma}$.

1. By Lemma 3, each light cluster A gets settled with probability at least

   $$1 - \mathrm{e}^{-k\varphi_A^t/(5\varphi_X^t)} \ge (k\varphi_A^t)/(20\varphi_U^t),$$

   using $\varphi_U^t \ge \varphi_X^t/2$ and $\mathrm{e}^{-x} \le 1 + x/2$ for $0 \le x \le 1$. If we define X_A to be an indicator of whether a light cluster A got settled in this iteration and $X = \sum_{A \text{ light}} X_A$, we have

   $$\begin{aligned}\mathbf{E}[X] = \sum_{A \text{ light}} \mathbf{E}[X_A] &\ge \sum_{A \text{ light}} \frac{k\varphi_A^t}{20\varphi_U^t}\\ &= \alpha k/20 \ge \frac{k}{20\sqrt{\log\gamma}}\end{aligned}$$

   where we used our assumption on α. Invoking the first bound of Theorem 2, we get

   $$\mathrm{P}(X \le \mathbf{E}[X]/2) \le \mathrm{e}^{-\mathbf{E}[X]/8} \le \mathrm{e}^{-\Theta(k^{0.1})},$$

   using that $k \ge \log\gamma/\log\log\gamma$.

2. We proceed analogously to the previous case. By Lemma 3 we get that each heavy cluster gets settled with probability at least

   $$1 - \mathrm{e}^{-k\varphi_A^t/(5\varphi_X^t)} \ge 1 - \mathrm{e}^{-1/10} \ge \frac{1}{20},$$

using $\varphi_U^t \geq \varphi_X^t/2$ and the definition of heavy cluster.

We define X_A to be an indicator of whether a heavy cluster A got settled in this iteration and $X = \sum_{A \text{ heavy}} X_A$. We have

$$\mathbf{E}[X] = \sum_{A \text{ heavy}} \mathbf{E}[X_A] \geq k/\sqrt{\log\gamma} \cdot \frac{1}{20} = \frac{k}{20\sqrt{\log\gamma}}.$$

Invoking the first bound of Theorem 2, we get

$$\mathrm{P}(X \leq \mathbf{E}[X]/2) \leq \mathrm{e}^{-\mathbf{E}[X]/8} \leq \mathrm{e}^{-\Theta(k^{0.1})},$$

using that $k \geq \log\gamma/\log\log\gamma$.

3. Let $\zeta = \sqrt[10]{\log\gamma}\frac{\varphi_U^t}{k}$. We call a heavy cluster *massive* if its cost is at least ζ. Since we know that there are at most $\frac{k}{\sqrt{\log\gamma}}$ heavy clusters, the total cost of clusters that are heavy but not massive is at most

$$\frac{k}{\sqrt{\log\gamma}} \cdot \sqrt[10]{\log\gamma}\frac{\varphi_U^t}{k} \leq \frac{\varphi_U^t}{\sqrt[3]{\log\gamma}}$$

Hence, the total contribution of clusters that are not massive is at most

$$\frac{\varphi_U^t}{\sqrt[3]{\log\gamma}} + \alpha\varphi_U^t \leq \varphi_U^t\left(\frac{1}{\sqrt[3]{\log\gamma}} + \frac{1}{\sqrt{\log\gamma}}\right) \leq \frac{2\varphi_U^t}{\sqrt[3]{\log\gamma}}.$$

By Lemma 3 each massive cluster is not settled with probability at most $\mathrm{e}^{-k\varphi_A^t/(5\varphi_X^t)}$. Define the random variable X_A to be equal to 0 if a massive cluster A gets settled in this iteration and φ_A^t/ζ otherwise. Let $X = \sum_{A \text{ massive}} X_A$. Note that expected cost of massive clusters that are not settled in this iteration is bounded by $\mathbf{E}[X] \cdot \zeta$.

The value of X is stochastically dominated by the value of a variable X' defined as follows. We first replace each X_A by some variables $X'_{A,1}, \dots, X'_{A,\lceil\varphi_A^t/\zeta\rceil}$, each new variable $X'_{A,i}$ being equal to 1 with probability $\mathrm{e}^{-k\zeta/(10\varphi_X^t)}$ and zero otherwise, independently on the other variables $X'_{A,j}$. Note that the sum $X'_A = X'_{A,1} + \cdots + X'_{A,\lceil\varphi_A^t/\zeta\rceil}$ stochastically dominates the value of X_A, since it attends the value $\lceil\frac{\varphi_A^t}{\zeta}\rceil \geq \frac{\varphi_A^t}{\zeta}$ with probability

$$\prod_{i=1}^{\lceil\varphi_A^t/\zeta\rceil} \mathrm{e}^{-k\zeta/(10\varphi_X^t)} \geq \left(\mathrm{e}^{-k\zeta/(10\varphi_X^t)}\right)^{2\varphi_A^t/\zeta} = \mathrm{e}^{-k\varphi_A^t/(5\varphi_X^t)},$$

and otherwise is nonnegative. Hence, the value $X' = \sum X'_A$ stochastically dominates the value X.

Since the number of variables $X'_{A,i}$ is $\sum_A \lceil\frac{\varphi_A^t}{\zeta}\rceil \leq \frac{2\varphi_X^t}{\zeta}$ we have

$$\begin{aligned}\mathbf{E}[X] \leq \mathbf{E}[X'] &\leq \frac{2\varphi_X^t}{\zeta} \cdot \exp(-k\zeta/(10\varphi_X^t)) \\ &\leq \frac{4\varphi_U^t}{\zeta} \cdot \exp\left(\frac{-k\sqrt[10]{\log\gamma}\varphi_U^t}{20k\varphi_U^t}\right) \\ &= \frac{4\varphi_U^t}{\sqrt[10]{\log\gamma}\varphi_U^t/k} \cdot \exp\left(\frac{-\sqrt[10]{\log\gamma}}{20}\right) \leq \frac{k}{\sqrt{\log\gamma}}\end{aligned}$$

where the last step assumes γ large enough. Finally, we bound

$$\begin{aligned}\mathrm{P}(X \geq 2k/\sqrt{\log\gamma}) &\leq \mathrm{P}(X' \geq 2k/\sqrt{\log\gamma}) \\ &= \mathrm{P}(X' \geq (1+\delta)\mathbf{E}[X'])\end{aligned}$$

for $\delta = \frac{2k}{\sqrt{\log\gamma}\mathbf{E}[X']} - 1 \geq \frac{k}{\sqrt{\log\gamma}\mathbf{E}[X']} \geq 1$ by above bound on $\mathbf{E}[X']$. Applying the third bound in Theorem 2, we conclude that

$$\begin{aligned}\mathrm{P}(X \geq 2k/\sqrt{\log\gamma}) &\leq \exp(-\mathbf{E}[X']\delta/3) \\ &\leq \exp(-k/(3\sqrt{\log\gamma})) \leq \mathrm{e}^{-\Omega(k^{0.1})}.\end{aligned}$$

Hence, with probability $1 - \mathrm{e}^{-\Omega(k^{0.1})}$ the total cost of clusters that remain unsettled after this iteration is bounded by

$$\frac{2\varphi_U^t}{\sqrt[3]{\log\gamma}} + \frac{2k}{\sqrt{\log\gamma}} \cdot \zeta \leq \frac{4\varphi_U^t}{\sqrt[3]{\log\gamma}}$$

□

Theorem 3. *For $\ell = k$, Algorithm 1 achieves a constant approximation ratio for the number of sampling steps $t = O(\min(k, \frac{\log\gamma}{\log\log\gamma}))$ steps with probability $1 - \exp(\Omega(k^{0.1}))$.*

Proof. To see that the number of steps is bounded by k with high probability, note that by Lemma 3 each unsettled cluster is made settled with probability at least $1 - \mathrm{e}^{-k\varphi_A^t/(5\varphi_X^t)}$. So, unless $\varphi_X^t \leq 20\varphi^*$, we have $\varphi_U^t \geq \varphi_X^t/2$ and, hence, with probability at least

$$1 - \prod_{A \text{ unsettled}} \exp(-k\varphi_A^t/10\varphi_U^t) = 1 - \exp(-k/10)$$

we make at least one cluster settled. Union bounding over first k steps of the algorithm, we conclude that the algorithm finishes in at most k steps with probability $1 - \exp(-\Omega(k))$.

Whatever point c_0 is taken uniformly at the beginning of Algorithm 1, for the next iteration we invoke Lemma 3 with $A = X$ (in its formulation we say that $A = X$ is unsettled cluster) to conclude that with probability $1 - \exp(-\Omega(k))$ we have $\varphi^1_X \le 10\varphi_X(\mu_X)$.

We invoke Proposition 2 and union bound over k subsequent iterations of the algorithm to conclude that with probability $1 - \exp(\Omega(k^{0.1}))$, in each sampling step of Algorithm 1 we either make at least $k/(40\sqrt{\log\gamma})$ clusters settled or the cost of unsettled clusters decreases by a factor of $4/\sqrt[3]{\log\gamma}$. The first case can happen at most $O(\sqrt{\log\gamma}) = O(\log\gamma/\log\log\gamma)$ times , whereas the second case can happen at most $O(\log_{\sqrt[3]{\log\gamma}/4}\gamma) = O(\log\gamma/\log\log\gamma)$ times, until we have $\varphi^t_X \le 20\varphi^*$. Hence, the algorithm achieves constant approximation ratio after $O(\log\gamma/\log\log\gamma)$ steps, with probability $1 - \exp(\Omega(k^{0.1}))$. □

Remark 3. *The high probability guarantee in Theorem 3 can be made $1 - \exp(\Omega(k^{1-o(1)}))$. We omit the proof.*

5. Sharp Analysis of k-`means`$||$: Lower Bound

In this section we show that the upper bound of $O(\log\gamma/\log\log\gamma)$ steps is the best possible. Note that (Bachem et al., 2017) proved that for $\ell = k$ there is a dataset X such that $\mathbf{E}[\varphi^t_X] \ge \frac{1}{4}(4kt)^{-t}\mathbf{Var}(X)$ for $t < k-1$. Hence, for $k = O(\text{poly}(\log\gamma))$ we conclude that for the number of steps T necessary to achieve constant approximation we have $(T\text{poly}(\log\gamma))^T \ge \gamma$, implying $T = \Omega(\log\gamma/\log\log\gamma)$. We complement their result by showing that the same lower bound also holds for $k = \Omega(\text{poly}(\log\gamma))$.

In the following theorem we construct an instance where the distance of two different data points can be zero, but it is easy to generalize the result for the case where we have the distance between two different points lower bounded by 1.

Theorem 4. *For any function f with $f(x) = \Omega(\log^{10} x), f(x) = O(x^{0.9})$, there is a dataset X with $|X| = \Theta(k)$ and $k = \Theta(f(\varphi_X(\mu_X)))$ such that $\varphi^* = 0$ and with probability arbitrarily close to 1 Algorithm 1 needs $\Omega(\log(\varphi_X(\mu_X))/\log\log(\varphi_X(\mu_X)))$ iterations to achieve cost zero.*

Proof. First we describe the dataset X. We place $|X|-k+1$ points $x_{0,j}$ for $1 \le j \le |X|-k+1$ to the origin, i.e., $x_{0,j} = \mathbf{0}$. We choose $|X| = \Theta(k)$ to be of such size that we know that with probability at least $1-\varepsilon$, for a given constant ε, the first uniformly chosen center is $\mathbf{0}$.

For each one of the remaining $k-1$ points we consider a new axis orthogonal to the remaining $k-2$ axes and place the point on this axis. For $x_{i,j}$, $1 \le i \le T, 1 \le j \le k/T$, we set $\|x_{i,j} - \mathbf{0}\|^2 = L^{2(T-i+1)}$, for some large enough L and $T = \Theta(L/\log L)$ with the multiplicative constant chosen in such a way that for $\varphi_X(\mu_X) = \Theta(\frac{k}{T}\cdot L^{2T})$ we have $\varphi_X(\mu_X) = \Theta(k\cdot 2^L)$. Note that we need to choose $L = \Omega(\text{poly}\log k)$ to achieve $k = \Theta(f(\varphi_X(\mu_X))) = O((\varphi_X(\mu_X))^{0.9})$. We define $\varphi^t_i = \sum_{j=1}^{k/T}\varphi^t_{x_{i,j}}$.

Conditioning on the first uniformly taken point being $x_{0,j}$ for some j, we prove by induction for $0 \le t \le T$ that with probability at least $1 - t\cdot 1/\text{poly}(k)$, after t sampling steps of the algorithm we have for each $i > t$ that out of k/T points $x_{i,j}$, for some j, at most

$$\frac{k}{T}\cdot\left(\frac{2}{3}\right)^{i-t}$$

of them have been sampled as centers. This will prove the theorem, since it implies that with probability at least $1 - 1/\text{poly}(k)$, after $t = T = \Theta(L/\log L)$ steps the cost φ^t is still nonzero; since we have $k = O((\varphi_X(\mu_X))^{0.9})$, we have

$$\begin{aligned}\Theta(L/\log L) &= \Theta(\log\frac{\varphi_X(\mu_X)}{k}/\log\log\frac{\varphi_X(\mu_X)}{k}) \\ &= \Theta(\log\varphi_X(\mu_X)/\log\log\varphi_X(\mu_X)).\end{aligned}$$

For $t = 0$ the claim we are proving is clearly true. For $t \ge 1$ note that by induction with probability at least $1 - (t-1)/\text{poly}(k)$ we have that at least

$$k/T - k/T\left(\frac{2}{3}\right) = k/(3T)$$

of the points $x_{t,j}$, for some j, were not sampled as centers. Hence, $\varphi^t_X \ge \frac{k}{3T}L^{2(T-t+1)}$ and the probability that each point $x_{i,j}$, $i > t$, is being sampled is bounded by

$$\frac{k\cdot L^{2(T-i+1)}}{kL^{2(T-t+1)}/(3T)} = \frac{3T}{L^{2(i-t)}} \le \frac{1}{L^{i-t}}$$

for L large enough. Hence, for any $i > t$, the expected number of points $x_{i,j}$ that are being hit is bounded by $k/(T\cdot L^{i-t})$. To get concentration around this value for given i, consider two cases.

1. $\frac{k}{3^{i-t}T} \ge k^{0.1}$. Then, by the third bound of Theorem 2 we can bound the probability of taking more than $\frac{k}{3^{i-t}T}$ clusters by $\exp(-\Theta(\frac{k}{3^{i-t}T})) \le \exp(-k^{0.1}) \le 1/\text{poly}(k)$.
2. $\frac{k}{3^{i-t}T} < k^{0.1}$. Using the assumption $k = \Omega(\log^{10}\varphi_X(\mu_X))$, hence $k = \Omega(L^{10})$, we have $3^{i-t} > k^{0.9}/T \ge k^{0.7}$. For L sufficiently large we then have $L^{i-t} \ge \text{poly}(k)$ for any fixed polynomial, hence, the expected number of hits is at most $k/(T\cdot L^{i-t}) \le 1/\text{poly}(k)$, hence, with probability at least $1 - 1/\text{poly}(k)$ there is no hit.

In both cases, conditioning on an event of probability $1 - 1/\text{poly}(k)$, for any $i > t$ the number of points $x_{i,j}$ that were sampled as centers is with probability at least $1 - t/\text{poly}(k)$ bounded by

$$\frac{k}{T}\left(\left(\frac{2}{3}\right)^{i-t+1} + \left(\frac{1}{3}\right)^{i-t}\right) \leq \frac{k}{T} \cdot \left(\frac{2}{3}\right)^{i-t}$$

as needed. □

6. Acknowledgement

I would like to thank my advisor Mohsen Ghaffari for his very useful insights and suggestions regarding the paper. I also thank Davin Choo, Danil Koževnikov, Christoph Grunau, Andreas Krause, and Julian Portmann for useful discussions and anonymous referees for helpful comments.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 853109)

References

Aggarwal, A., Deshpande, A., and Kannan, R. Adaptive sampling for k-means clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 15–28. Springer, 2009.

Ailon, N., Jaiswal, R., and Monteleoni, C. Streaming k-means approximation. In *Advances in neural information processing systems*, pp. 10–18, 2009.

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2): 245–248, May 2009. ISSN 1573-0565. doi: 10.1007/s10994-009-5103-0. URL https://doi.org/10.1007/s10994-009-5103-0.

Arthur, D. and Vassilvitskii, S. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pp. 1027–1035, Philadelphia, PA, USA, 2007a. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5. URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics, 2007b.

Awasthi, P., Charikar, M., Krishnaswamy, R., and Sinop, A. K. The hardness of approximation of euclidean k-means. *arXiv preprint arXiv:1502.03316*, 2015.

Bachem, O., Lucic, M., Hassani, H., and Krause, A. Fast and provably good seedings for k-means. In *Advances in neural information processing systems*, pp. 55–63, 2016a.

Bachem, O., Lucic, M., Hassani, S. H., and Krause, A. Approximate k-means++ in sublinear time. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016b.

Bachem, O., Lucic, M., and Krause, A. Distributed and provably good seedings for k-means in constant rounds. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 292–300. JMLR. org, 2017.

Bachem, O., Lucic, M., and Krause, A. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1119–1127, 2018.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

Barbosa, R., Ene, A., Nguyen, H., and Ward, J. The power of randomization: Distributed submodular maximization on massive datasets. In *International Conference on Machine Learning*, pp. 1236–1244, 2015a.

Barbosa, R., Ene, A., Nguyen, H. L., and Ward, J. A new framework for distributed submodular maximization, 2015b.

Bhattacharya, A., Eube, J., Röglin, H., and Schmidt, M. Noisy, greedy and not so greedy k-means++, 2019.

Brunsch, T. and Röglin, H. A bad instance for k-means++. *Theoretical Computer Science*, 505:19–26, 2013.

Celebi, M. E., Kingravi, H. A., and Vela, P. A. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, 40(1):200–210, 2013.

Choo, D., Grunau, C., Portmann, J., and Rozhoň, V. k-means++: few more steps yield constant approximation, 2020.

Dean, J. and Ghemawat, S. Mapreduce: Simplified data processing on large clusters. 2004.

Ghaffari, M. personal communication.

Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. Clustering data streams: Theory and practice. *IEEE transactions on knowledge and data engineering*, 15(3):515–528, 2003.

Jaiswal, R., Kumar, A., and Sen, S. A simple d^2-sampling based ptas for k-means and other clustering problems, 2012.

Jaiswal, R., Kumar, M., and Yadav, P. Improved analysis of d^2-sampling based ptas for k-means and other clustering problems. *Information Processing Letters*, 115(2):100–103, 2015.

Karloff, H., Suri, S., and Vassilvitskii, S. A model of computation for mapreduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pp. 938–948. SIAM, 2010.

Krause, A. and Golovin, D. Submodular function maximization.

Lattanzi, S. and Sohler, C. A better k-means++ algorithm via local search. In *International Conference on Machine Learning*, pp. 3662–3671, 2019.

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

Lenzen, C. and Wattenhofer, R. Tight bounds for parallel randomized load balancing, 2011.

Liu, P. and Vondrak, J. Submodular optimization in the mapreduce model. *arXiv preprint arXiv:1810.01489*, 2018.

Lloyd, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. ISSN 1557-9654. doi: 10.1109/TIT.1982.1056489.

Mahajan, M., Nimbhorkar, P., and Varadarajan, K. The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pp. 274–285. Springer, 2009.

Mettu, R. and Plaxton, G. Optimal time bounds for approximate clustering, 2012.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems*, pp. 2049–2057, 2013.

Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59(6):1–22, 2013.

Wei, D. A constant-factor bi-criteria approximation guarantee for k-means++. In *Advances in Neural Information Processing Systems*, pp. 604–612, 2016.