Robust k-means++

Amit Deshpande
Microsoft Research (MSR)
Bengaluru, India
amitdesh@microsoft.com

Praneeth Kacham
Carnegie Mellon University (CMU)
Pittsburgh, USA
praneethkacham@gmail.com

Rameshwar Pratap[*]
IIT Mandi
H.P., India
rameshwar.pratap@gmail.com

Abstract

A good seeding or initialization of cluster centers for the k-means method is important from both theoretical and practical standpoints. The k-means objective is inherently non-robust and sensitive to outliers. A popular seeding such as the k-means++ [3] that is more likely to pick outliers in the worst case may compound this drawback, thereby affecting the quality of clustering on noisy data.

For any $0 < \delta \leq 1$, we show that using a mixture of D^2 [3] and uniform sampling, we can pick $O(k/\delta)$ candidate centers with the following guarantee: they contain some k centers that give $O(1)$-approximation to the optimal robust k-means solution while discarding at most δn more points than the outliers discarded by the optimal solution. That is, if the optimal solution discards its farthest βn points as outliers, our solution discards its $(\beta + \delta)n$ points as outliers. The constant factor in our $O(1)$-approximation does not depend on δ. This is an improvement over previous results for k-means with outliers based on LP relaxation and rounding [7] and local search [17]. The $O(k/\delta)$ sized subset can be found in time $O(ndk)$. Our *robust* k-means++ is also easily amenable to scalable, faster, parallel implementations of k-means++ [5]. Our empirical results show a comparison of the above *robust* variant of k-means++ with the usual k-means++, uniform random seeding, threshold k-means++ [6] and local search on real world and synthetic data.

[*]Corresponding Author.

1 Introduction

The k-means clustering is a popular tool in data analysis and an important objective in statistics, data mining, unsupervised learning, computational geometry, approximation algorithms. The objective of the k-means clustering is to find k centers that minimize the sum of squared distances of all the points to their nearest centers, respectively. Given a set $X \subseteq \mathbb{R}^d$ of n data points and an integer $k > 0$, the k-means objective is to find a set $C \subseteq \mathbb{R}^d$ of k centers that minimizes $\phi_X(C) = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$. The k-means clustering objective and its variants for kernel methods and Bregman divergence are widely used in machine learning and data mining. The most popular algorithm for the k-means remains Lloyd's k-means method [23] that originated as a vector quantization technique for Pulse-Code Modulation (PCM) in signal processing. Lloyd's k-means method [23] is a simple, fast heuristic that starts with any given initial solution and iteratively converges to a locally optimal solution.

However, k-means clustering suffers from two major drawbacks – (a) the k-means objective itself is non-robust and highly sensitive to outliers and, (b) the k-means method does not have a provably good initialization that is also robust to outliers.

Non-robustness of the k-means objective: A common statistical measure of robustness is the breakdown point [18] [1]. The mean or centroid is not a robust statistic and has its breakdown point at zero, that is, even a single outlier can move the mean arbitrarily far. In comparison, median is more robust with a high breakdown point. In that sense, the k-means objective is extremely sensitive to outliers.

[1]Informally, the breakdown point is the minimum fraction of data points which if significantly modified will also significantly modify the output of the estimator. Large breakdown point means better robustness.

Even though the median is a more robust statistic than the mean, the geometric median of a set of points in $\mathbb{R}^d$ is itself a non-trivial computational problem [10]. Thus, we do not have an efficient analog of the k-means method [23] for the k-median problem. In practice, we can remove the outliers first using Principal Component Analysis (PCA), Minimum Covariance Determinant (MCD), or distance to the k-th Nearest Neighbor (k-NN) and then clustering the inliers using k-means but these two objectives are not fully aligned. Thus, despite the non-robustness, we would like to retain the inherently nice properties of the k-means objective for clustering in the presence of outliers. This has lead to the definition of robust versions of the k-means objective such as the trimmed k-means [14].

The *robust k-means clustering* objective we consider in this paper is the same as trimmed k-means that looks at the k-means objective on all but the farthest β fraction of data points according to their distances to their respective centers. In other words, given a set $X \subseteq \mathbb{R}^d$ of n points, an integer $k > 0$, and an outlier parameter $0 < \beta < 1$, the objective of the *robust k-means* is to find a set $C \subseteq \mathbb{R}^d$ of k centers that minimizes $\rho_X(C) = \min_{\substack{Y \subseteq X \\ |Y|=(1-\beta)n}} \sum_{x \in Y} \min_{c \in C} \|x - c\|^2$.

Non-robustness of k-means++: Lloyd's k-means method [23] provably converges to a local optimum, however, in the worst case, it can take exponentially many iterations to do so even in the plane [24]. Uniform sampling is a somewhat popular initialization but provides provable approximation guarantees only when the optimal clusters are somewhat large and balanced. In the worst case, only k-means++ seeding [3] and its faster, scalable versions [5, 4] are known to provide provably good initialization with $O(\log k)$-approximation to the global optimum, in expectation. The k-means++ seeding is based on *D^2-sampling* which builds a partial solution gradually as follows: Pick a point uniformly at random from the given points and make it the first center. Then pick the second point with probability proportional to its squared distance to the first center, and make it the second center. In each step, pick a new point with probability proportional to its squared distance to the nearest center picked so far (known as *D^2-sampling*), and make that point a new center. The k-means++ method applies D^2-sampling k times, adaptively as above, to come up with an initial set of k centers. The practical implementations of the k-means++ method actually pick multiple samples (typically, $O(\log k)$ samples) according to D^2-sampling in each step, and retain the best among them that minimizes the sum of squared distances of all the points to their nearest centers in the current partial solution, respectively.

The D^2-sampling used by the k-means++ seeding inherently has a higher probability of picking outliers as centers, making it extremely sensitive to outliers. Consider the case when 99% of the points form k clusters very close to each other, and the remaining 1% are outliers far from these clusters as well as from each other. In such cases, D^2-sampling is highly likely to pick outliers. The uniform sampling alone, on the other hand, is likely to miss smaller clusters among inliers and does not take into account the inter-point distances.

Compounding the non-robustness of the initialization with the non-robustness of the k-means objective itself can result in poor quality of clustering on noisy data. An important question, therefore, is to come up with provably good seedings or initializations for the k-mean clustering that are robust to outliers.

Our paper addresses this question positively by proposing a simple modification of the k-means++ seeding that is robust to outliers. Instead of using D^2-sampling in each step, we use a mixture of D^2-sampling and uniform sampling to get the best of both worlds.

2 Our results

Assuming that the outliers constitute a constant fraction of the given data, we propose a simple initialization called robust k-means++ using a mixture of D^2 sampling and uniform sampling. Our algorithm runs in $O(ndk)$ time, outputs $O(k)$ clusters, discards marginally more points than the optimal number of outliers, and comes with a provable $O(1)$ approximation guarantee. Our algorithm can be modified to output exactly k clusters instead of $O(k)$ clusters, while keeping its running time linear in n and d. The pseudocode of our algorithm is presented in 1, and the precise statement of our approximation guarantee appears in Theorem 1. In addition, Table 1 shows a detailed comparison of our results with previous work.

Our empirical results in Section 5 show the advantage of robust k-means++ over the k-means++ or D^2 sampling, uniform random seeding, and a state of the art local search algorithm for k-means with outliers [17], on standard real world and synthetic data sets used in previous work.

Result	Approximation guarantee	No. of clusters in the output	No. of outliers discarded	Practical algorithm	Running time
This paper	5	$O(kn/z)$	$O(z)$	Yes	$O(ndk)$
	5	k	$O(z)$	Yes	$O(ndk) \times (kn/z)^k$
[6]	$(64+\epsilon)$	$(1+c)k$	$\frac{(1+c)(1+\epsilon)z}{c(1-\mu)}$	Yes	$\tilde{O}(ndk)$
[6]	$O(\log k)$	k	$O(z \log k)$	Yes	$\tilde{O}(ndk)$
[19]	$(53.002+\epsilon)$	k	z	No	$n^{O(1/\epsilon^3)}$
[13]	$(1+\epsilon)$	$k(1+\epsilon)$	z	No	$kn^{d^{O(d)}}$
[17]	274	k	$O(zk \log n)$	Yes	$O(\frac{1}{\epsilon}k^2(k+z)^2\log(n\Delta)+nz)$, Δ:=largest pairwise distance
[9]	$O(1)$	k	z	No	poly(n,k)
[7]	$4(1+1/\epsilon)$	k	$(1+\epsilon)z$	No	$n^{O(1)}$

Table 1: Comparison with related work on k-means clustering with outliers, where n = no. of data points, d = dimension, k = desired or given no. of clusters, and z = desired or given no. of outliers. In [6], c is a given parameter, and $\mu > 0$ is an arbitrary constant. Their algorithm crucially require an initial guess of the optimal clustering cost. [7, 9] are about a closely related problem called k-median clustering with outliers.

3 Related work

The k-means++ method, in its spirit, is a randomized variant of a 2-approximation algorithm for the k-center clustering problem that picks the farthest data point from the current partial solution in each step [16]. A simple and clever modification of the farthest point algorithm gives a 3-approximation for the robust k-center problem with outliers [7]. For k-median with outliers, they give a $4(1+1/\epsilon)$-approximation in polynomial time that discards at most $(1+\epsilon)$ times more outliers than the optimal robust k-median solution. To compare with our results, their approximation depends on the additional number of points discarded, whereas our approximation is a fixed constant factor that does not. Gupta et al. [17] proposed a local-search algorithm for the robust k-means problem that gives a $O(1)$-approximation while discarding $O(zk \log n)$ outliers, which is unreasonably large when the number of outliers is a constant fraction of n. The number of extra outliers deleted by our algorithm is δn, which is comparatively very small, as for example we can choose $\delta = 1/\text{poly}(k)$. We trade-off by picking $O(k/\delta)$ centers as opposed to k centers picked by both these algorithms [7, 17].

Recently, Bhaskara *et. al.* [6] suggest a bi-criteria approximation algorithm for the problem. They show that a thresholding operation on the D^2 sampling distribution makes it robust to outliers. However, their algorithms crucially require an initial guess of the optimal clustering cost. With this assumption they showed the following two results. In the first, their algorithm output $(1+c)k$ centers and obtain $(64+\epsilon)$ approximations by discarding $\frac{(1+c)(1+\epsilon)z}{c(1-\mu)}$ outliers in the running time $\tilde{O}(ndk)$, where c is some parameter, and μ is a constant. Whereas our algorithm offers 5 approximation factor, in a faster running time, while outputting $O(kn/z)$ cluster centers and discarding $O(z)$ outliers. In the second result, their algorithm output exactly k centers and obtain $O(\log k)$ approximations by discarding $O(z \log k)$ outliers in the running time $\tilde{O}(ndk)$. Whereas our algorithm offers 5 approximation factor while outputting k cluster centers and discarding $O(z)$ outliers. However, theoretical bounds of our running time is exponential in k in this case, while for experiments we use a weighted k-means++ as a heuristic.

[9] gives a polynomial time constant factor approximation to k-means and k-median with outliers that does not discard extra outliers, however, the algorithm is not practical or scalable compared to k-means++ and its variants. There is an important line of work [12, 20, 11] based on coresets and robust coresets for clustering problems that give $(1+\epsilon)$-approximation to robust k-median and various other clustering problems by constructing small-sized coresets via sampling. Many of these algorithms are relevant to our work and can be thought of as extensions of sampling-based seeding techniques with the emphasis on getting nearly optimal or $(1+\epsilon)$-approximately optimal solutions in running time linear in the number of points n and the dimension d. Scalable k-means++ [5] has been empirically observed to be more robust to outliers than the k-means++ but there is no theoretical understanding of this aspect.

[8] propose a natural modification of the Lloyd's k-means method in the presence of outliers, and show its local convergence. Another recent paper revisits the robust or trimmed k-means objective, proposes a different variant and studies its consistency and robustness properties [15].

Krishnaswamy *et al.* [19] give (roughly) 53-approximation algorithm for k-means with z outliers. Their algorithm outputs exactly k centers and does not discard any extra points as outliers, and use iterative rounding of LP relaxation with polynomial running time. However their algorithm is not designed to be practical. Friggstad *et al.* [13] give $(1+\epsilon)$-approximation to k-means with z outliers in a fixed dimensional Euclidean space. Their algorithm gives $(1+\epsilon)k$ centers and discards exactly z outliers. They use local search and have running time doubly exponential in the dimension making it impractical. k-means clustering with outliers problem has also been studied in different formulations. We list a one such recent results as follows: [22] considered the problem of joint cluster analysis and outlier detection and propose an algorithm that achieves simultaneous consensus clustering and outlier detection.

Our proposed distribution – a mixture of D^2-sampling and uniform sampling – is used as a *proposal distribution* in the initial step of the recent MCMC-based fast and provably good seeding for the k-means [4]. However, instead of employing this adaptively, they only use it in the first step and their final result is a $O(\log k)$-approximation, in expectation, along the lines of the inductive analysis of the k-means++. On the other hand, our analysis uses ideas from [1, 2] that provide a different analysis of the k-means++ method to give bi-criteria $O(1)$-approximations with high probability guarantees. We summarise a comparison of the related work in Table 1.

Algorithm 1 Robust k-means++

```
**Input:** a set $X \subseteq \mathbb{R}^d$ of $n$ points, an integer $k > 0$, an outlier parameter $0 \leq \beta \leq 1$, and an error parameter $0 < \delta \leq 1$.
**Output:** a set $S \subseteq X$ of size $O(k/\delta)$
Initialize $S_0 \leftarrow \emptyset$
**for** $i = 1$ **to** $t = O(k)$ **do**
    **for** $l = 1$ **to** $m = O(1/\delta)$ **do**
        Pick a point $x_l \in X$ from the following distribution:
        $\mathsf{Pr}\,(\text{picking } x_l) = \frac{\phi_{\{x_l\}}(S_{i-1})}{2\ \phi_X(S_{i-1})} + \frac{1}{2n}$
        (Note: For $i = 1$ step, the distribution is uniform.)
    **end for**
    $S_i \leftarrow S_{i-1} \cup \{x_1, x_2, \dots, x_m\}$
    $i \leftarrow i + 1$
**end for**
$S \leftarrow S_t$
**return** $S$
```

4 Analysis of robust k-means++

We start with a word of notations used in the analysis.

Notation: Given a set $X \subseteq \mathbb{R}^d$ of n points, an integer $k > 0$, and an outlier parameter $0 < \beta < 1$, recall that the objective of the *robust k-means* is to find a set $C \subseteq \mathbb{R}^d$ of k centers that minimizes

$$\rho_X(C) = \min_{\substack{Y \subseteq X \\ |Y| = (1-\beta)n}} \sum_{x \in Y} \min_{c \in C} \|x - c\|^2 .$$

In other words, $\rho_X(C)$ is the squared distance error only over the nearest $(1-\beta)n$ points instead of the entire data. We denote by $\phi_A(C) = \sum_{x \in A} \min_{c \in C} \|x - c\|^2$ the contribution of points in a subset $A \subseteq X$. Let C_{OPT} be the set of optimal k centers for the robust k-means problem and let Y_{OPT} be the optimal subset of inliers, then $\rho(C_{\text{OPT}}) = \phi_{Y_{\text{OPT}}}(C_{\text{OPT}})$, since the error or the potential is measured only over the inliers. In the optimal solution, each point of Y_{OPT} is assigned to its nearest center in C_{OPT}. This induces a natural partition of the inliers Y_{OPT} as $A_1 \cup A_2 \cup \cdots \cup A_k$ into disjoint subsets, with means $\mu_1, \mu_2, \dots, \mu_k$, respectively, while $X \setminus Y_{\text{OPT}}$ are the outliers. Therefore,

$$\rho(C_{\text{OPT}}) = \phi_{Y_{\text{OPT}}}(C_{\text{OPT}}) = \sum_{j=1}^{k} \phi_{A_j}(\{\mu_j\}) .$$

Now we show the analysis of our sampling scheme. One simple approach to obtain a set S that contains k points which give $O(1)$-factor approximation for k-means with outliers problem, is to run any c-approximate k-means algorithm with $(k+\beta n)$ centers, where βn is the number of outliers. It is easy to show the cost of such a solution is less than $O(c)$ times the cost of the optimal solution of k-means with βn outliers. This approach picks $(k + \beta n)$ points, which is large for high values of β. In what follows, we present our algorithm which is simple, accurate, and samples a significantly lesser number of points.

Theorem 1. *Let $S \subseteq X$ be the subset of $O(k/\delta)$ points picked by the robust k-means++ seeding Algorithm 1. Then, with constant probability, S contains some k centers that give 5-approximation to $\rho(C_{OPT})$ while discarding $(\beta + \delta)n$ points, slightly more than the βn outliers discarded by the optimal solution. The probability can be boosted to $1 - \eta$, by repeating it $O(\log(1/\eta))$ and picking the best solution, where $\eta \in (0, 1)$. The running time of our algorithm is $O(nkd/\delta)$. The subset of k centers guaranteed as above can be recovered by iterating over all possible subsets of size k, and picking the best solution in time $nd \cdot 2^{O(k \log(1/\delta))}$.*

Remark 1. *$\beta \in [0, 1]$ is an outlier parameter given as input to Algorithm 1. δ is another parameter introduced for our bi-criteria approximation. Theorem 1 holds for any $\beta \in (0, 1)$ and $\delta \in [0, 1 - \beta]$. For example, we can always choose $\delta = \beta/100$ so that our algorithm discards at most $1.01\beta n$ points, in other words, only 1% more outliers than the optimal solution. Moreover, when outliers constitute a very tiny (not even a constant) fraction of the data, e.g., say $\beta = 1/\mathrm{poly}(k)$, our algorithm can still output $\mathrm{poly}(k)$ centers that give $O(1)$-approximation while discarding only 1% more outliers than the optimal solution.*

Remark 2. *The analysis of Theorem 1 shown for a $(1/2, 1/2)$ mixture of uniform and D^2 sampling distribution used in Algorithm 1. We note that the analysis of the theorem can be extended for any mixture $(\alpha, 1 - \alpha)$ of D^2 and uniform sampling distribution, where $\alpha \in (0, 1)$. In our experiments, we use $\alpha \in \{0, 1/4, 1/2, 3/4, 1\}$, and notice that our proposal outperforms with respect to the baselines on at least one of the values of α.*

Our analysis is simple, largely based on ideas from [1]. However, we need to carefully combine the advantages of D^2-sampling and uniform sampling, especially when considering conditional probabilities.

Now we analyze the effect of sampling from a mixture of D^2-sampling and uniform sampling in each step. Let S_{i-1} be the set of $(i-1)m$ centers obtained after the $(i-1)$-th iteration of the outer loop of our algorithm, where $m = O(1/\delta)$. In step i, we define

$$\mathsf{Good}_i = \{A_j \ : \ \phi_{A_j}(S_{i-1}) \leq 5 \ \phi_{A_j}(\{\mu_j\})\},$$

$$\mathsf{Bad}_i = \{A_1, A_2, \ldots, A_k\} \setminus \mathsf{Good}_i.$$

In other words, Good_i consists of the optimal clusters A_j for which the current cluster centers in S_{i-1} already give a good approximation. First we show that when we sample from a mixture of D^2-sampling and uniform sampling with equal weights in the i-th step, there is a reasonable probability that we pick a point from some cluster in Bad_i and moving that cluster to Good_{i+1} for the subsequent step. We summarise this in the lemma below whose proof is deferred to appendix due to space limitations.

Lemma 2. *In the outer loop of Algorithm 1, one of the following two possibilities must hold.*

1. *$\sum_{j \,:\, A_j \in \mathsf{Bad}_i} |A_j| \leq \delta n$, in which case, S_{i-1} contains some k centers that give 5-approximation to robust k-means when we remove their farthest $(\beta + \delta)n$ points.*
2. *The total number of bad clusters decreases with at least a constant probability, that is, $\mathsf{Pr}(|\mathsf{Bad}_{i+1}| < |\mathsf{Bad}_i|) = \Omega(1)$.*

Proof. Consider any cluster $A_j \in \mathsf{Bad}_i$. Let μ_j be the mean of A_j and define $B_j \subseteq A_j$ as the subset of points in A_j *close* to μ_j as follows. Define the root-mean-square radius of A_j as $r_j = \sqrt{\phi_{A_j}(\{\mu_j\}) / |A_j|}$ and define $B_j = \left\{x \in A_j \ : \ \|x - \mu_j\| \leq \sqrt{2}\ r_j\right\}$.

In step i of Algorithm 1, either $\sum_{j \,:\, A_j \in \mathsf{Bad}_i} |A_j| \leq \delta n$, in which case, $Y' = \cup_{j \,:\, A_j \in \mathsf{Good}_i} A_j$ has size $|Y'| \geq (1 - \beta - \delta)n$ and cost

$$\begin{aligned}\phi_{Y'}(S_{i-1}) &= \sum_{j \,:\, A_j \in \mathsf{Good}_i} \phi_{A_j}(S_{i-1}) \\ &\leq 5 \sum_{j \,:\, A_j \in \mathsf{Good}_i} \phi_{A_j}(\{\mu_j\}) \\ &\leq 5\ \phi_{Y_{\mathrm{OPT}}}(C_{\mathrm{OPT}}) = 5\ \rho(C_{\mathrm{OPT}}),\end{aligned}$$

that is, we get a 5-approximation to robust k-means while discarding at most $(\beta + \delta)n$ points.

Otherwise, if $\sum_{j \,:\, A_j \in \mathsf{Bad}_i} |A_j| \geq \delta n$, then we show that $\mathsf{Pr}(x \in B_j \text{ for some } A_j \in \mathsf{Bad}_i)$ is not too small, and moreover, picking $x \in B_j$ makes the cluster $A_j \in \mathsf{Good}_{i+1}$ for the subsequent $(i+1)$-th step of Algorithm 1. To begin with,

$$\begin{aligned}&\mathsf{Pr}(x \in A_j \text{ for some } A_j \in \mathsf{Bad}_i) \\ &= \frac{\sum_{j \,:\, A_j \in \mathsf{Bad}_i} \phi_{A_j}(S_{i-1})}{2\ \phi_X(S_{i-1})} + \frac{\sum_{j \,:\, A_j \in \mathsf{Bad}_i} |A_j|}{2n} \\ &\geq \frac{\sum_{j \,:\, A_j \in \mathsf{Bad}_i} |A_j|}{2n} \geq \frac{\delta}{2}. \qquad (1)\end{aligned}$$

Moreover, we have

$$\begin{aligned}&\mathsf{Pr}(x \in B_j \mid x \in A_j \text{ and } A_j \in \mathsf{Bad}_i) \qquad (2) \\ &= \frac{\dfrac{\phi_{B_j}(S_{i-1})}{2\ \phi_X(S_{i-1})} + \dfrac{|B_j|}{2n}}{\dfrac{\phi_{A_j}(S_{i-1})}{2\ \phi_X(S_{i-1})} + \dfrac{|A_j|}{2n}} \geq \min\left\{\frac{\phi_{B_j}(S_{i-1})}{\phi_{A_j}(S_{i-1})}, \frac{|B_j|}{|A_j|}\right\} \\ &\qquad (3)\end{aligned}$$

Observe that B_j contains at least half of the points in A_j, that is, $|B_j| / |A_j| \geq 1/2$, for all j. Otherwise, $\sum_{x \notin B_j} \|x - \mu_j\|^2 > |A_j|/2 \cdot 2r_j^2 = \phi_{A_j}(\{\mu_j\}) = \sum_{x \in A_j} \|x - \mu_j\|^2$, which is a contradiction.

Let $p \in S_{i-1}$ be the point nearest to the mean μ_j of A_j and let $\|p - \mu_j\| = d$. For any $A_j \in \mathsf{Bad}_i$, $5\ \phi_{A_j}(\{\mu_j\}) \leq \phi_{A_j}(S_{i-1}) \leq \phi_{A_j}(\{p\}) = \phi_{A_j}(\{\mu_j\}) + |A_j| \|p - \mu_j\|^2$, by the parallel axis theorem (see Proposition 1 of [1]), which implies that

$\|p - \mu_j\| = d \geq 2r_j$. Now

$$\phi_{B_j}(S_{i-1}) = \sum_{x \in B_j} \min_{s \in S_{i-1}} \|x - s\|^2$$
$$\geq \sum_{x \in B_j} \min_{s \in S_{i-1}} (\|s - \mu_j\| - \|\mu_j - x\|)^2 (\text{triangle inequality})$$
$$\geq |B_j| \; (\|p - \mu_j\| - \sqrt{2}\; r_j)^2 \geq \frac{|A_j|}{2} \; (d - \sqrt{2}\; r_j)^2$$

The last inequality holds because $|B_j| \geq |A_j|/2$. On the other hand, $\phi_{A_j}(S_{i-1}) \leq \phi_{A_j}(\{p\}) = \phi_{A_j}(\{\mu_j\}) + |A_j| \|p - \mu_j\|^2 = |A_j| (r_j^2 + d^2)$. Therefore,

$$\frac{\phi_{B_j}(S_{i-1})}{\phi_{A_j}(S_{i-1})} \geq \frac{(d - \sqrt{2}\; r_j)^2}{2\;(r_j^2 + d^2)} \geq \frac{(2 - \sqrt{2})^2}{10} \geq \frac{1}{30},$$

because the ratio above is an increasing function of d for $d \geq 2r_j$. Hence, we can bound (3) as

$$\Pr(x \in B_j \mid x \in A_j \text{ and } A_j \in \mathsf{Bad}_i)$$
$$\geq \min\left\{\frac{\phi_{B_j}(S_{i-1})}{\phi_{A_j}(S_{i-1})}, \frac{|B_j|}{|A_j|}\right\} \geq \min\left\{\frac{1}{30}, \frac{1}{2}\right\} = \frac{1}{30}.$$

Combining this with (1), we get

$$\Pr(x \in B_j \text{ for some } A_j \in \mathsf{Bad}_i) \geq \delta/60 = \Omega(\delta).$$

Moreover, for $x \in B_j$,

$$\phi_{A_j}(S_{i-1} \cup \{x\}) \leq \phi_{A_j}(\{x\})$$
$$= \phi_{A_j}(\{\mu_j\}) + |A_j| \|x - \mu_j\|^2$$
$$\leq \phi_{A_j}(\{\mu_j\}) + |A_j| 2r_j^2 \leq 3\; \phi_{A_j}(\{\mu_j\}),$$

and when the algorithm updates $S_i \leftarrow S_{i-1} \cup \{x\}$, we have $A_j \in \mathsf{Good}_{i+1}$, and the probability of this is at least $\Omega(\delta)$ as shown above. By repeating it $O(1/\delta)$ times the inner loop of the algorithm boosts this probability to $\Omega(1)$. □

To prove Theorem 1, we use the following well known facts about super-martingales.

Definition 3. *A sequence of real valued random variables $J_0, J_1, \ldots, J_t$ is called a* super-martingale *if for every $i > 1$, $\mathsf{E}[J_i \mid J_0, \ldots, J_{i-1}] \leq J_{i-1}$.*

Super-martingales have the following concentration bound.

Theorem 4. (Azuma-Hoeffding inequality) *If $J_0, J_1, \ldots, J_t$ is a super-martingale with $J_{i+1} - J_i \leq 1$, then $\Pr(J_t \geq J_0 + \epsilon) \leq \exp(-\epsilon^2/2t)$.*

Proof. (Proof of Theorem 1) Recall that by Lemma 2, in each step of the inner loop of Algorithm 1,

- Either $\sum_{j \,:\, A_j \in \mathsf{Bad}_i} |A_j| \leq \delta n$, in which case, S_{i-1} contains a subset of k centers that give 5-approximation to robust k-means when we remove the farthest $(\beta + \delta)n$ points.
- Or the total number of bad clusters decreases with at least a constant probability, that is, $\Pr(|\mathsf{Bad}_{i+1}| < |\mathsf{Bad}_i|) = \Omega(1)$.

For each step define an indicator variable X_i as follows.

$$X_i = \begin{cases} 1 & \text{if } |\mathsf{Bad}_{i+1}| = |\mathsf{Bad}_i| \\ 0 & \text{otherwise.} \end{cases}$$

and let $\Pr(X_i = 0) \geq \theta = \Omega(1)$. Define $J_i = \sum_{1 \leq j \leq i} (X_j - (1 - \theta))$. Then $J_{i+1} - J_i \leq 1$ and

$$\mathsf{E}[J_i \mid J_0, \ldots, J_{i-1}] = \mathsf{E}[J_{i-1} + X_i - (1 - \theta) \mid J_0, \ldots, J_{i-1}]$$
$$= J_{i-1} + \mathsf{E}[X_i \mid J_0, \ldots, J_{i-1}] - (1 - \theta)$$
$$\leq J_{i-1},$$

which means that $J_1, J_2, \ldots, J_t$ is a super-martingale. So using Theorem 4 we get the following bound: $\Pr(J_t \geq J_0 + \epsilon) \leq \exp(-\epsilon^2/2t)$, which translates to $\Pr\left(\sum_{i=1}^{t} (1 - X_i) \geq \theta t - \epsilon\right) \geq 1 - \exp(-\epsilon^2/2t)$. Choosing $t = (k + \sqrt{k})/\theta$ and $\epsilon = \sqrt{k}$, we obtain,

$$\Pr\left(\sum_{i=1}^{(k+\sqrt{k})/\theta} (1 - X_i) \geq k\right) \geq 1 - \exp(-\theta/4).$$

Therefore,

$$\Pr\Big(\text{there are no bad clusters after } (k + \sqrt{k})/\theta \text{ steps}\Big)$$
$$\geq 1 - \exp(-\theta/4),$$

or equivalently, $\Pr(\phi(S) \leq 5\; \phi(C_{OPT}))$, with high probability. □

5 Experiments

Hardware description. We performed our experiments on a machine having the following configuration: CPU: Intel(R) Core(TM) i5-3320M CPU @ 2.70GHz x 4; Memory: 8 GB.

We study the performance of robust k-means++ to find k initial centers, wherein each iteration we use a $(1 - \alpha, \alpha)$ mixture of D^2-sampling and uniform sampling, and the algorithm marks the farthest z points as outliers. Though we proved that $\alpha = 1/2$ works well in case of outliers, we use several values of $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$ in experiments. Although our cost guarantee holds after removing $(\beta + \delta)n$ fraction of points as outliers, we only remove the number of points removed by other algorithms for a fair comparison. Following is the experimental procedure we used to compare with an algorithm removing z outliers.

1. Sample 1 center uniformly at random.
2. In each iteration, pick $1/\delta$ centers using $(1-\alpha, \alpha)$ mixture distribution of D^2 and uniform. Repeat this for $k-1$ iterations.
3. Find z outliers corresponding to these centers.
4. Using the inliers from the previous step, we calculate the weights of each of these $O(k/\delta)$ centers, which is equal to the number of points in their respective cluster. We solve this weighted k-means instance using k-means++ sampling. We use this procedure as a heuristic to extract k good centers, and refer as *weighted k-means++*.
5. Farthest points from these centers are considered final outliers.

The minimum, maximum, and average values are over 10 repetitions. We compare the performance of our algorithm with: (a) LSO (local search algorithm for k-means with outliers) [17], (b) k-means++ [3], (c) random seeding [23], and (d)TKM + + (Greedy Sampling for Approximate Clustering in the Presence of Outliers) [6]. TKM + + [6] requires two parameters to derive the probability distribution – an initial guess of optimal clustering cost, and error tolerance parameter in clustering cost. In their paper they did not mentioned any principal way of guessing the clustering cost. For our empirical comparison we used, the cost k-means++ results as initial guess and for the error parameter we used values from $\in \{0.1, 0.5, 1, 2, 5, 10\}$, and report the best result. We use *precision* and *recall* as our evaluation metric. If z^* is the set of true outliers and z is the set of outliers reported by the algorithm, then *precision*:=$|z^* \cap z|/|z|$, and *recall*:=$|z^* \cap z|/|z^*|$.

5.1 Results on real world data sets

Results on `KDDCup Full` dataset: `KDDFull`[21] This dataset is from 1999 kddcup competition and contains instances describing connections of sequences of tcp packets, and have about 4.9M data points. We only consider the 34 numerical features of this dataset. We also normalize each feature so that it has zero mean and unit standard deviation. There are 23 classes in this dataset, 98.3% points of the dataset belong to 3 classes (`normal` 19.6%, `neptune` 21.6%, and `smurf` 56.8%). We consider small clusters as outliers and there are 45747 outliers.

We run robust k-means++ on the `KDDFull` dataset with $k = 3$, and $\delta = \{0.1, 0.15\}$ and considering 45747 points as outliers. We compare its performance with LSO, vanilla k-means++ and random initialisation, and we delete the same number of outliers, and note the values of precision, recall, clustering cost and running time. We summarise our empirical findings in Table 2,3.

Insight. We noticed that our clustering cost is significantly better than that of k-means++, TKM + + and random seeding. Further, on `max` and `avg` values of precision and recall we obtained advantage over majority of the values as compared to TKM + +, k-means++ and random seeding. However, our running time is slightly off comparable to baselines.

Result	α	Cost			Time(s)
		Min	Mean	Med.	
This work δ=0.05	0	**2.8**	**2.8**	**2.8**	150
δ=0.1	0	**2.8**	**2.8**	**2.8**	295
δ=0.15	0	**2.8**	**2.8**	**2.8**	395
TKM + +		2.83	4.49	4.49	233
KM + +		**2.8**	4.22	4.44	120
RAND		2.83	3.38	2.85	70

Table 2: Robust k-means++ on `KDDCup Full` dataset with $k = 3$. We consider 45747 points as outliers. The cost of Robust k-means++ is same for all other values of $\alpha = \{0.25, 0.5, 0.75, 0.1\}$. LSO doesn't stop after 8 hours. All costs are multiplicative of 10^7.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
	0	0.61	0.61	0.61	0.61	0.61	0.61
	0.25	0.63	0.59	0.61	0.63	0.59	0.61
This work	0.5	**0.64**	**0.62**	0.61	**0.64**	**0.62**	0.61
δ=0.1	0.75	0.63	0.59	0.61	0.63	0.59	0.61
	1	0.61	0.61	0.61	0.61	0.61	0.61
	0	**0.64**	0.61	0.61	**0.64**	0.61	0.61
	0.25	**0.64**	0.60	0.61	**0.64**	0.60	0.61
This work	0.5	0.61	0.56	0.61	0.61	0.56	0.61
δ=0.15	0.75	0.63	0.60	0.61	0.63	0.60	0.61
	1	0.63	0.61	0.61	0.63	0.61	0.61
TKM + +		0.63	0.55	0.60	0.63	0.55	0.60
KM + +		0.63	0.61	0.62	0.63	0.61	0.62
RAND		0.63	0.56	**0.63**	0.63	0.56	**0.63**

Table 3: Result on `KDDCup` dataset with $k = 3$ and 45747 outliers. LSO doesn't stop after 8 hours.

Results on `Shuttle` dataset. `Shuttle` training data set from UCI Machine Learning Repository [21] contains 43,500 points. It has 7 classes in total. The two smallest classes contain only 17 points and we would like to detect these as outliers. The choice of `Shuttle` data set was because the local search algorithm of [17] reported particularly poor precision and recall values on it.

We run robust k-means++ on the `Shuttle` dataset with $k \in \{5, 10, 15\}$, and $\delta = \{0.05, 0.1\}$. In order to have a fair basis of comparison among all the candidate algorithms, we delete the same number of outliers as of mentioned in LSO (see Table 4 of [17]), and note the values of precision, recall, and cost. We summarise our empirical findings in Tables 4, 5, 6, 7.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	**0.28**	0.17	**0.26**	**0.35**	0.21	**0.32**
	0.25	**0.28**	0.09	0.02	**0.35**	0.11	0.02
	0.5	**0.28**	0.12	0.04	**0.35**	0.15	0.05
	0.75	**0.28**	0.13	0.07	**0.35**	0.12	0.07
	1	0.19	**0.19**	0.19	0.23	**0.23**	0.23
TKM + +		0.25	0.17	0.23	0.29	0.21	0.29
KM + +		**0.28**	0.16	0.23	**0.35**	0.20	0.29
RAND		0.19	**0.19**	0.19	0.23	**0.23**	0.23
LSO		–	0.176	–	–	0.212	–

Table 4: Robust k-means++ on `Shuttle` dataset with $k = 5$ and $\delta = 0.05$. We delete the farthest 21 point similar to LSO.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	0.20	0.16	**0.17**	0.41	0.32	**0.35**
	0.25	**0.29**	**0.176**	0.16	**0.58**	**0.353**	0.32
	0.5	0.20	0.14	0.16	0.41	0.29	0.32
	0.75	0.23	0.14	0.14	0.47	0.29	0.29
	1	0.176	0.15	0.15	0.353	0.30	0.29
TKM + +		0.26	0.14	0.11	0.52	0.29	0.23
KM + +		0.26	0.15	0.14	0.52	0.31	0.29
RAND		0.14	0.14	0.14	0.29	0.29	0.29
LSO		–	**0.176**	–	–	**0.353**	–

Table 5: Robust k-means++ on `Shuttle` dataset with $k = 10$ and $\delta = 0.05$. We delete the farthest 34 point similar to LSO.

Insight. In almost every scenario robust k-means++ outperforms random initialisation. On the comparison with k-means++ and TKM + + our algorithm gives comparable/better performance for small values of k. However, when the value of k is large $k = 15$, on most of the cases robust k-means++ outperforms both k-means++ and TKM + +. We compare the performance of robust k-means++ with LSO. Here again, we notice that for a small value of k such as $k \in \{5, 10\}$, robust k-means++ give similar performance as of LSO, while simultaneously outperforming on some values of α. However, for a large value of k, it significantly outperforms LSO on the most values of α.

5.2 Results on Synthetic Data Sets

Dataset. We generate synthetic dataset in the similar way as used in k-means++ [3] and LSO [17]. We discuss it as follows. We pick $k+z$ uniformly random points from a large d-dimensional hyper-cube of side

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	0.25	0.15	0.15	0.76	0.47	0.47
	0.25	0.25	0.15	0.14	0.76	0.47	0.44
	0.5	0.25	0.21	**0.23**	0.76	0.65	0.64
	0.75	**0.27**	**0.22**	0.22	**0.82**	**0.67**	**0.67**
	1	0.17	0.14	0.13	0.52	0.44	0.41
TKM + +		0.25	0.17	0.19	0.76	0.52	0.58
KM + +		0.25	0.17	0.17	0.76	0.52	0.5
RAND		0.13	0.13	0.13	0.41	0.41	0.41
LSO		–	0.181	–	–	0.553	–

Table 6: Robust k-means++ on `Shuttle` dataset with $k = 15$ and $\delta = 0.05$. We delete the farthest 51 point similar to LSO.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	**0.29**	0.20	0.21	**0.88**	0.61	0.64
	0.25	0.27	**0.21**	**0.22**	0.82	**0.64**	**0.67**
	0.5	**0.29**	**0.21**	**0.22**	**0.88**	**0.64**	**0.67**
	0.75	0.25	0.16	0.16	0.76	0.48	0.5
	1	0.17	0.13	0.13	0.52	0.40	0.41
TKM + +		0.25	0.17	0.19	0.76	0.52	0.58
KM + +		0.25	0.17	0.17	0.76	0.52	0.5
RAND		0.13	0.13	0.13	0.41	0.41	0.41
LSO		–	0.182	–	–	0.553	–

Table 7: Robust k-means++ on `Shuttle` dataset with $k = 15$ and $\delta = 0.1$. We delete the farthest 51 point similar to LSO.

length $s = 100$. We use k points from them as means and pick n/k points around each of them from a random Gaussian of unit variance. This gives a data set of $n + z$ points with n points clustered into k clusters and the remaining z as outliers.

Empirical Evaluation. We first run robust k-means++ with the values of $k \in \{10, 20\}$, $\alpha \in \{0, 0.25, 0.5, 1\}$, $\delta \in \{0.05, 0.1\}$ on the synthetic datasets with values $n = 10^4, d = 15$, and the number of outliers $\{25, 50, 100\}$. In all the cases, similar to LSO (see Table 1 of [17]), robust k-means++ achieves both precision and recall 1.

We further perform experiments on synthetic datasets with values $n = 1000, d = 2, k = 20$ and the number of outliers $25, 50, 100$. We summarised our results in Tables 8,9,10,11.

Efficiency of robust k-means++ As mentioned earlier, the algorithm first sample a set of size $O(k/\delta)$, which contains k points that give $O(1)$-approximation for the problem. We empirically find a set of k points using *weighted k-means++* that gives better/comparable performance as compare to other algorithms. The running time of the algorithm is linear in n, k, d. However, for LSO it is $O(\frac{1}{\epsilon}k^2(k+z)^2 \log(n\Delta) + nz)$, where Δ is the largest distance between pair of points. Their time complexity is of order k^4, and moreover, when $z = \Omega(n)$ it becomes quadratic in n. Empirically, we noticed

We defer the benchmark on the clustering cost and running time on `Shuttle` and `Synthetic` datasets the appendix due to space limit.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	0.96	0.85	0.84	0.96	0.85	0.84
	0.25	0.92	0.82	0.84	0.92	0.82	0.84
	0.5	0.88	0.82	0.84	0.88	0.82	0.84
	0.75	0.96	0.88	0.88	0.96	0.88	0.88
	1	**1**	0.9	**0.92**	**1**	0.9	**0.92**
TKM + +		0.80	0.65	0.64	0.80	0.65	0.64
KM + +		0.96	0.51	0.48	0.96	0.51	0.48
RAND		0.4	0.07	0.04	0.4	0.07	0.04
LSO		–	**0.94**	–	–	**0.94**	–

Table 8: Robust k-means++ on Synthetic dataset with $\delta = 0.1$. $n = 1000, d = 2, k = 20$. We delete the farthest 25 point similar to LSO.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	0.86	0.83	0.84	0.86	0.83	0.84
	0.25	0.88	0.84	0.83	0.88	0.84	0.83
	0.5	0.88	0.82	0.83	0.88	0.82	0.83
	0.75	0.94	0.89	0.88	0.94	0.89	0.88
	1	**1**	0.9	**0.92**	**1**	0.9	**0.92**
TKM + +		0.90	0.86	0.5	0.90	0.86	0.5
KM + +		0.84	0.5	0.5	0.84	0.5	0.5
RAND		0.48	0.07	0.04	0.48	0.07	0.04
LSO		–	**0.91**	–	–	**0.91**	–

Table 9: Robust k-means++ on Synthetic dataset with $\delta = 0.1$. $n = 1000, d = 2, k = 20$. We delete the farthest 50 point similar to LSO.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	0.80	0.77	0.77	0.97	0.92	0.93
	0.25	0.80	0.77	0.77	0.97	0.93	0.93
	0.5	0.81	0.78	0.77	0.98	0.94	0.93
	0.75	0.81	**0.79**	**0.79**	0.98	0.95	**0.95**
	1	**0.83**	**0.79**	**0.79**	**0.99**	**0.95**	0.94
TKM + +		0.75	0.49	0.54	0.90	0.59	0.65
KM + +		0.63	0.37	0.32	0.76	0.44	0.39
RAND		0.33	0.21	0.251	0.40	0.26	0.25
LSO		–	0.72	–	–	0.91	–

Table 10: Robust k-means++ on Synthetic dataset with $\delta = 0.1$. $n = 1000, d = 2, k = 20$. We delete the farthest 120 point similar to LSO.

Result	α	Precision			Recall		
		Max.	Avg.	Med.	Max.	Avg.	Med.
This work	0	0.79	0.77	0.77	0.95	0.93	**0.93**
	0.25	**0.80**	**0.78**	**0.78**	**0.97**	**0.94**	**0.93**
	0.5	0.79	0.77	0.77	0.95	0.93	**0.93**
	0.75	0.78	0.76	0.77	0.94	0.92	0.92
	1	**0.80**	0.77	0.77	**0.97**	0.92	0.92
TKM + +		0.75	0.49	0.54	0.90	0.59	0.65
KM + +		0.62	0.33	0.30	0.74	0.39	0.36
RAND		0.33	0.2	0.2	0.39	0.23	0.23
LSO		–	0.72	–	–	0.91	–

Table 11: Robust k-means++ on Synthetic dataset with $\delta = 0.05$. $n = 1000, d = 2, k = 20$. We delete the farthest 120 point similar to LSO.

that our running time of our algorithm is slightly off compare to k-means++ and TKM + +.

6 Conclusion and open questions

In this work, we present a robust sampling algorithm for k-means clustering. Our major contribution lies in coming up with a simple and intuitive tweak to the k-means++ sampling algorithm, which makes it robust to the outliers. We empirically evaluated our algorithm on synthetic as well real-world datasets, and showed that our proposed method seems to be better than k-means++, random initialization, and [17], when the value of k and/or the fraction of outliers is large, and when these outliers are very far away from the inliers. Our work leaves the possibility of several open questions. We discuss them as follows:

Our algorithm samples a set of $O(k/\delta)$ points which contain some k centers that gives $O(1)$-approximation for the k-means problem with outliers. For analysis purpose, we generate all possible subsets of size k, and pick the best solution, which has time complexity exponential in k. For experiments, we use *weighted k-means++* as a heuristic to find a set of k-points, which gives good results. Coming up with an algorithm that can pick k points from the $O(k/\delta)$ sized set in time polynomial in k, and linear in n, d, and simultaneously offers a $O(1)$-factor approximation, is an open question of this work.

We analyzed the bounds of our algorithm for $\alpha = 1/2$. However, we performed experiments with several values of $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. We notice that in most of the cases at least on one values of α, we outperformed *w.r.t.* other candidate algorithms. For a given distribution of points, finding the optimal value of α is another open question of this work. Finally, given the simplicity, accuracy, and efficiency of our algorithm, we hope that it will be adopted in practice.

References

[1] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '09 / RANDOM '09, pages 15–28, 2009.

[2] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. In *Advances in Neural Information Processing Systems 22.*, pages 10–18, 2009.

[3] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding.

In *Proceedings of*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[4] Olivier Bachem, Mario Lucic, Seyed Hamed Hassani, and Andreas Krause. Fast and provably good seedings for k-means. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 55–63, 2016.

[5] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, March 2012.

[6] Aditya Bhaskara, Sharvaree Vadgama, and Hong Xu. Greedy sampling for approximate clustering in the presence of outliers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11146–11155. Curran Associates, Inc., 2019.

[7] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th SODA, Jan 7-9, 2001, Washington, DC, USA.*, pages 642–651, 2001.

[8] Sanjay Chawla and Aristides Gionis. k-means–: A unified approach to clustering and outlier detection. In *SDM*, pages 189–197. SIAM, 2013.

[9] Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *SODA*, volume 8, pages 826–835, 2008.

[10] Michael B. Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of*, STOC '16, pages 9–21, 2016.

[11] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2011.

[12] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A ptas for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 11–18. ACM, 2007.

[13] Zachary Friggstad, Kamyar Khodamoradi, Mohsen Rezapour, and Mohammad R. Salavatipour. Approximation schemes for clustering with outliers. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM, SODA 2018, New Orleans, LA, USA, January 7-10*, pages 398–414, 2018.

[14] Luis Angel Garcia-Escudero, Luis Ángel García-Escudero, and Alfonso Gordaliza. Robustness properties of k means and trimmed k means. *Journal of the American Statistical Association*, 94:956–969, 1999.

[15] Alexandros Georgogiannis. Robust k-means: a theoretical revisit. In *Advances in Neural Information Processing Systems 29.*

[16] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[17] Shalmoli Gupta, Ravi Kumar, Kefu Lu, Benjamin Moseley, and Sergei Vassilvitskii. Local search methods for k-means with outliers. *PVLDB*, 10(7):757–768, 2017.

[18] P.J. Huber, J. Wiley, and W. InterScience. *Robust statistics.* Wiley New York, 1981.

[19] Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 646–659, 2018.

[20] Michael Langberg and Leonard J Schulman. Universal ε-approximators for integrals. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 598–607. SIAM, 2010.

[21] M. Lichman. Uci machine learning repository. 2013.

[22] Hongfu Liu, Jun Li, Yue Wu, and Yun Fu. Clustering with outlier removal. *CoRR*, abs/1801.01899, 2018.

[23] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 2006.

[24] Andrea Vattani. K-means requires exponentially many iterations even in the plane. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 324–332, 2009.