Fast Noise Removal for k-Means Clustering

Sungjin Im
University of California at Merced

Mahshid Montazer Qaem
University of California at Merced

Benjamin Moseley
Carnegie Mellon University

Xiaorui Sun
University of Illinois at Chicago

Rudy Zhou
Carnegie Mellon University

Abstract

This paper considers k-means clustering in the presence of noise. It is known that k-means clustering is highly sensitive to noise, and thus noise should be removed to obtain a quality solution. A popular formulation of this problem is called *k-means clustering with outliers*. The goal of k-means clustering with outliers is to discard up to a specified number z of points as noise/outliers and then find a k-means solution on the remaining data. The problem has received significant attention, yet current algorithms with theoretical guarantees suffer from either high running time or inherent loss in the solution quality. The main contribution of this paper is two-fold. Firstly, we develop a simple greedy algorithm that has *provably* strong worst case guarantees. The greedy algorithm adds a simple preprocessing step to remove noise, which can be combined with any k-means clustering algorithm. This algorithm gives the first pseudo-approximation-preserving reduction from k-means with outliers to k-means without outliers. Secondly, we show how to construct a coreset of size $O(k \log n)$. When combined with our greedy algorithm, we obtain a scalable, near linear time algorithm. The theoretical contributions are verified experimentally by demonstrating that the algorithm quickly removes noise and obtains a high-quality clustering.

Proceedings of the 23[rd]International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

1 Introduction

Clustering is a fundamental unsupervised learning method that offers a compact view of data sets by grouping similar input points. Among various clustering methods, k-means clustering is one of the most popular clustering methods used in practice, which is defined as follows: given a set X of n points in Euclidean space[1] $\mathbb{R}^d$ and a target number of clusters k, the goal is to choose a set C of k points from $\mathbb{R}^d$ as centers, so as to minimize the ℓ_2-loss, i.e., the sum of the squared distances of every point $x \in X$ to its closest center in C.

Due to its popularity, k-means clustering has been extensively studied for decades both theoretically and empirically, and as a result, various novel algorithms and powerful underlying theories have been developed. In particular, because the clustering problem is NP-hard, several constant-factor approximation algorithms have been developed (Charikar and Guha, 1999; Kanungo et al., 2004; Kumar et al., 2004; Feldman et al., 2007), meaning that their output is always within an $O(1)$ factor of the optimum. One of the most successful algorithms used in practice is k-means++ (Arthur and Vassilvitskii, 2007). The algorithm k-means++ is a preprocessing step used to set the initial centers when using Lloyd's algorithm (Lloyd, 1982). Lloyd's algorithm is a simple local search heuristic that alternates between updating the center of every cluster and reassigning points to their closest centers. k-means++ has a provable approximation guarantee of $O(\log k)$ by carefully choosing the initial centers.

k-means clustering is highly sensitive to noise, which is present in many data sets. Indeed, it is not difficult to see that the k-means clustering objective can vary significantly even with the addition of a single point

[1]The input space can be extended to an arbitrary metric space.

that is far away from the true clusters. In general, it is a non-trivial task to filter out noise; without knowing the true clusters, we cannot identify noise, and vice versa. While there are other clustering methods, such as density-based clustering (Ester et al., 1996), that attempt to remove noise, they do not replace k-means clustering because they are fundamentally different than k-means.

Consequently, there have been attempts to study k-means clustering in the presence of noise. The following problem formulation is the most popular formulation in the theory (Chen, 2008; Charikar et al., 2001; McCutchen and Khuller, 2008; Guha et al., 2017), machine learning (Malkomes et al., 2015; Chawla and Gionis, 2013; Li and Guo, 2018) and database communities (Gupta et al., 2017). Note that traditional k-means clustering is a special case of this problem when $z = 0$. Throughout, for $x, y \in \mathbb{R}^d$, we let $d(x, y)$ denote the ℓ_2 distance between x and y. For a subset of points Y, let $d(x, Y) := \min_{y \in Y} d(x, y)$.

Definition 1 (k-Means with Outliers)**.** *In this problem we are given as input a subset X of n points in $\mathbb{R}^d$, a parameter $k \in \mathbb{N}$ (number of centers), and a parameter $z \in \mathbb{N}$ (number of outliers). The goal is to choose a collection of k centers, $C \subseteq \mathbb{R}^d$, to minimize: $\sum_{x \in X_z(C)} d^2(x, C)$, where $X_z(C) \subseteq X$ is the subset of $n - z$ input points with the smallest distances to C.*

Because this problem generalizes k-means clustering, it is NP-hard, and in fact, turns out to be significantly more challenging. The only known constant approximations (Chen, 2008; Krishnaswamy et al., 2018) are highly sophisticated and are based on complicated local search or linear program rounding. They are unlikely to be implemented in practice due to their runtime and complexity. Therefore, there have been strong efforts to develop simpler algorithms that offer good approximation guarantees when allowed to discard more than z points as outliers (Charikar et al., 2001; Meyerson et al., 2004; Gupta et al., 2017), or heuristics (Chawla and Gionis, 2013). Unfortunately, the aforementioned algorithms with theoretical guarantees suffer from either high running time or inherent loss in solution quality.

In contrast , concurrently with our work, Bhaskara et al. (2019) developed a simple algorithm for k-means with outliers that gives a good approximation guarantee when allowed to discard more than z outliers or use more than k centers, which we discuss in more detail when we describe our results.

1.1 Our Results and Contributions

The algorithmic contribution of this paper is two-fold, and further these contributions are validated by experiments. In this section, we state our contribution and discuss it in detail compared to the previous work.

Simple Preprocessing Step for Removing Outliers with Provable Guarantees: In this paper we develop a simple preprocessing step, which we term NK-MEANS, to effectively filter out outliers. NK-MEANS stands for noise removal for k-means. Our proposed preprocessing step can be combined with *any* algorithm for k-means clustering. Despite the large amount of work on this problem, we give the *first* reduction to the standard k-means problem. In particular, NK-MEANS can be combined with the popular k-means++. The algorithm is the fastest known algorithm for the k-means with outliers problem. Its speed and simplicity gives it the potential to be used in practice. Formally, given an α-approximation for k-means clustering, we give an algorithm for k-means with outliers that is guaranteed to discard up to $O(kz)$ points such that the cost of remaining points is at most $O(\alpha)$ times the optimum that discards up to exactly z points. While the theoretical guarantee on the number of outliers is larger than z on worst-case inputs, we show that NK-MEANS removes at most $O(z)$ outliers under the assumption that every cluster in an optimal solution has at least $3z$ points. We believe that this assumption captures most practical cases since otherwise significant portions of the true clusters can be discarded as outliers. In actual implementation, we can guarantee discarding exactly z points by discarding the farthest z points from the centers we have chosen. It is worth keeping in mind that all (practical) algorithms for the problem discard more than z points to have theoretical guarantees (Charikar et al., 2001; Meyerson et al., 2004; Gupta et al., 2017; Bhaskara et al., 2019).

When compared to the concurrent work of Bhaskara et al. (2019), our work differs in two main ways. Firstly, our algorithm consists of a pre-processing step that can be combined with any k-means algorithm, such as k-means++, while theirs consists of a modification to the standard k-means++ algorithm. Further, our algorithm throws away a multiplicative $O(k)$-factor extra outliers in the worst case and uses no extra centers, while in their work they prove a trade-off between extra outliers and extra centers. In particular, they obtain a $O(\log k)$ approximation for k-means with outliers using an extra multiplicative $O(\log k)$ outliers and no extra centers, and a $O(1)$-approximation using an extra

multiplicative $O(1)$ outliers and centers.

New Coreset Construction: When the data set is large, a dominant way to speed up clustering is to first construct a coreset and then use the clustering result of the coreset as a solution to the original input. Informally, a set of (weighted) points Y is called a coreset of X if a good clustering of Y is also a good clustering of X (see Section 4.1 for the formal definition of coreset.)

The idea is that if we can efficiently construct such Y, which is significantly smaller than X, then we can speed up any clustering algorithm with little loss of accuracy. In this paper, we give an algorithm to construct a coreset of size $O(k \log n)$ for k-means with outliers. Importantly, the coreset size is independent of z and d - the number of outliers and dimension, respectively.

Experimental Validation: Our new coreset enables the implementation and comparison of all potentially practical algorithms, which are based on primal-dual (Charikar et al., 2001), uniform sampling (Meyerson et al., 2004), or local search (Chawla and Gionis, 2013; Gupta et al., 2017). It is worth noting that, to the best of our knowledge, this is the first paper to implement the primal-dual based algorithm (Charikar et al., 2001) and test it for large data sets. We also implemented natural extensions of k-means++ and our algorithm NK-MEANS. We note that for fair comparison, once each algorithm chose the k centers, we considered all points and discarded the farthest z points. Our experiments show that our NK-MEANS consistently outperforms other algorithms for both synthetic and real-world data sets with little running time overhead as compared to k-means++.

1.2 Comparison to the Previous Work

Algorithms for k-Means with Outliers: To understand the contribution of our work, it is important to contrast the algorithm with previous work. We believe a significant contribution of our work is the algorithmic simplicity and speed as well as the theoretical bounds that our approach guarantees. In particular, we will discuss why the previous algorithms are difficult to use in practice.

The first potentially practical algorithm developed is based on primal-dual (Charikar et al., 2001). Instead of solving a linear program (LP) and converting the solution to an integer solution, the primal-dual approach only uses the LP and its dual to guide the algorithm. However, the algorithm does not scale well and is not easy to implement. In particular, it involves increasing variables uniformly, which requires $\Omega(n^2)$ running time and extra care to handle precision issues of fractional values. As mentioned before, this algorithm was never implemented prior to this paper. Our experiments show that this algorithm considerably under-performs compared to other algorithms.

The second potentially practical algorithm is based on uniform sampling (Meyerson et al., 2004). The main observation of Meyerson et al. (2004) is that if every cluster is large enough, then a small uniform sample can serve as a coreset. This observation leads to two algorithms for k-means clustering with outliers: (i) (implicit) reduction to k-means clustering via conservative uniform sampling and (ii) (explicit) aggressive uniform sampling plus primal-dual (Charikar et al., 2001). In (i) it can be shown that a constant approximate k-means clustering of a uniform sample of size $n/(2z)$ is a constant approximation for k-means clustering with outliers, under the assumption that every cluster has size $\Omega(z \log k)$. Here, the main idea is to avoid any noise by sampling conservatively. Although this assumption is reasonable as discussed before, the real issue is that conservative uniform sampling doesn't give a sufficiently accurate sketch to be adopted in practice. For example, if there are 1% noise points, then the conservative uniform sample has only 50 points. In (ii), a more aggressive uniform sampling is used and followed by the primal dual (Charikar et al., 2001). It first obtains a uniform sample of size $\Theta(k(n/z) \log n)$; then the (expected) number of outliers in the sample becomes $\Theta(k \log n)$. This aggressive uniform sampling turns out to have very little loss in terms of accuracy. However, as mentioned before, the primal-dual algorithm under-performs compared to other algorithms in speed and accuracy.

Another line of algorithmic development has been based on local search (Chawla and Gionis, 2013; Gupta et al., 2017). The algorithm in Chawla and Gionis (2013) guarantees the convergence to a local optimum, but has no approximation guarantees. The other algorithm (Gupta et al., 2017) is an $O(1)$-approximation but theoretically it may end up with discarding $O(kz \log n)$ outliers. These local search algorithms are considerably slower than our method and the theoretical guarantees require discarding many more points.

To summarize, there is a need for a fast and effective algorithm for k-means clustering with outliers.

Coresets for k-Means with Outliers: The other main contribution of our work is a coreset for k-means with outliers of size $O(k \log n)$ - independent of the number of outliers z and dimension d.

The notion of coreset we consider is related to the concept of a *weak coreset* in the literature - see e.g. Feldman and Langberg (2011) for discussion of weak coresets and other types of coresets. Previous coreset constructions (some for stronger notions of coreset) have polynomial dependence on the number of outliers z (Gupta et al., 2017), inverse polynomial dependence on the fraction of outliers $\frac{z}{n}$ (Meyerson et al., 2004; Huang et al., 2018), or polynomial dependence on the dimension d (Huang et al., 2018). Thus, all coresets constructed in the previous work can have large size for some value of z, e.g. $z = \Theta(\sqrt{n})$, or for large values of d. In contrast, our construction is efficient for *all* values of $z \in [0, n]$ and yields coresets of size with no dependence on d or z.

1.3 Overview of Our Algorithms: NK-means and SampleCoreset

Our preprocessing step, NK-means, is reminiscent of density-based clustering. Our algorithm tags an input point as light if it has relatively few points around it. Formally, a point is declared as light if it has less than $2z$ points within a certain distance threshold r, which can be set by binary search. Then a point is discarded if it only has light points within distance r. We emphasize that the threshold is chosen by the algorithm, not by the algorithm user, unlike in density-based clustering. While our preprocessing step looks similar to the algorithm for k-center clustering (Charikar et al., 2001), which optimizes the ℓ_∞-loss, we find it surprising that a similar idea can be used for k-means clustering.

It can take considerable time to label each point light or not. To speed up our algorithm, we develop a new corest construction for k-means with outliers. The idea is relatively simple. We first use aggressive sampling as in Meyerson et al. (2004). The resulting sample has size $O(\frac{kn}{z} \log n)$ and includes $O(k \log n)$ outliers with high probability. Then we use k-means++ to obtain $O(k \log n)$ centers. As a result, we obtain a high-quality coreset of size $O(k \log n)$. Interestingly, to our best knowledge, combining aggressive sampling with another coreset for k-means with outliers has not been considered in the literature.

1.4 Other Related Work

Due to the vast literature on clustering, we refer the reader to Aggarwal and Reddy (2013); Kogan et al. (2006); Jain et al. (1999) for an overview and survey of the literature. k-means clustering can be generalized by considering other norms of loss, and such extensions have been studied under different names. When the objective is ℓ_1-norm loss, the problem is called k-medians. The k-median and k-mean clustering problems are closely related, and in general the algorithm and analysis for one can be readily translated into one for the other with an $O(1)$ factor loss in the approximation ratio. Constant approximations are known for k-medians and k-means based on linear programming, primal-dual, and local search (Arya et al., 2004; Charikar et al., 2002; Charikar and Guha, 1999). While its approximation ratio is $O(\log k)$, the k-means++ algorithm is widely used in practice for k-means clustering due to its practical performance and simplicity. When the loss function is ℓ_∞, the problem is known as k-centers and a 3-approximation is known for k-centers clustering with outliers (Charikar et al., 2001). For recent work on these outlier problems in distributed settings, see Malkomes et al. (2015); Li and Guo (2018); Guha et al. (2017); Chen et al. (2018).

2 Preliminaries

In this paper we will consider the Euclidean k-means with outliers problem as defined in the introduction. Note that the ℓ_2-distance satisfies the *triangle inequality*, so for all $x, y, z \in \mathbb{R}^d$, $d(x, z) \leq d(x, y) + d(y, z)$. Further, the *approximate triangle inequality* will be useful to our analyses (this follows from the triangle inequality): $d^2(x, z) \leq 2d^2(x, y) + 2d^2(y, z) \quad \forall x, y, z \in \mathbb{R}^d$. Given a set of centers $C \subset \mathbb{R}^d$, we say that the *assignment cost* of $x \in X$ to C is $d^2(x, C)$. For k-means with outliers, a set, C, of k centers naturally defines a clustering of the input points X as follows:

Definition 2 (Clustering)**.** *Let $C = \{c_1, \ldots, c_k\} \subset \mathbb{R}^d$ be a set of k centers. A clustering of X defined by C is a partition $C_1 \cup \cdots \cup C_k$ of $X_z(C)$ satisfying: For all $x \in X_z$ and $c_i \in C$, $x \in C_i \iff d(x, C) = d(x, c_i)$, where ties between c_i's are broken arbitrarily but consistently.*

In summary, for the k-means with outliers problem, given a set C of k centers, we assign each point in X to its closest center in C. Then we exclude the z points of X with the highest assignment cost from the objective function (these points are our outliers.) This procedure defines a clustering of X with outliers.

Notations: For $n \in \mathbb{N}$, we define $[n] := \{1, \ldots, n\}$. Recall that as in the introduction, for any finite $Y \subset \mathbb{R}^d, x \in \mathbb{R}^d$, we define: $d(x, Y) := \min_{y \in Y} d(x, y)$. For any $x \in \mathbb{R}^d, X \subseteq \mathbb{R}^d, r > 0$, we define the X-ball centered at x with radius r by $B(x, r) := \{y \in X \mid d(x, y) \leq r\}$. For a set of k centers, $C \subset \mathbb{R}^d$, and $z \in \mathbb{N}$, we define the *z-cost* of C by $f_z^X(C) := \sum_{x \in X_z(C)} d^2(x, C)$. Recall that we define $X_z(C) \subset X$ to be the subset of points of X excluding the z points with highest assignment costs. Thus the z-cost of C

is the cost of clustering X with C while excluding the z points with highest assignment costs. As shorthand, when $z = 0$ – so when we consider the k-means problem without outliers – we will denote the 0-cost of clustering X with C by $f^X(C) := f_0^X(C)$. Further, we say a set of k centers C^* is an *optimal z-solution* if it minimizes $f_z^X(C)$ over all choices of k centers, C. Then we define $Opt(X, k, z) := f_z^X(C^*)$ to be the optimal objective value of the k-means with outliers instance (X, k, z). Analogously, for the k-means without outliers problem, we denote the optimal objective value of the k-means instance (X, k) by $Opt(X, k)$.

3 NK-MEANS Algorithm

In this section, we will describe our algorithm, NK-MEANS, which turns a k-means algorithm without outliers to an algorithm for k-means with outliers in a black box fashion. We note that the algorithm naturally extends to k-medians with outliers and general metric spaces. For the remainder of this section, let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $k \in \mathbb{N}$, and $z \in \mathbb{N}$ define an instance of k-means with outliers.

Algorithm Intuition: The guiding intuition behind our algorithm is as follows: We consider a ball of radius $r > 0$ around each point $x \in X$. If this ball contains many points, then x is likely not to be an outlier in the optimal solution.

More concretely, if there are more than $2z$ points in x's ball, then at most z of these points can be outliers in the optimal solution. This means that the majority of x's neighbourhood is real points in the optimal solution, so we can bound the assignment cost of x to the optimal centers. We call such points *heavy*.

There are 2 main steps to our algorithm. First, we use the concept of heavy points to decide which points are real points and those that are outliers. Then we run a k-means approximation algorithm on the real points.

Formal Algorithm: Now we formally describe our algorithm NK-MEANS. As input, NK-MEANS takes a k-means with outliers instance (X, k, z) and an algorithm for k-mean without outliers, $\mathcal{A}$, where $\mathcal{A}$ takes an instance of k-means as input.

We will prove that if $\mathcal{A}$ is an $O(1)$-approximation for k-means and the optimal clusters are sufficiently large with respect to z, then NK-MEANS outputs a good clustering that discards $O(z)$ outliers. More precisely, we will prove the following theorem about the performance of NK-MEANS:

Theorem 1. *Let C be the output of* NK-MEANS$(X, k, z, \mathcal{A})$. *Suppose that $\mathcal{A}$ is an α-approximation for k-means. If every cluster in the clustering defined by C^* has size at least $3z$, then $f_{2z}^X(C) \le 9\alpha \cdot Opt(X, k, z)$.*

Corollary 1. *Let C be the output of* NK-MEANS$(X, k, z, \mathcal{A})$. *Suppose that $\mathcal{A}$ is an α-approximation. Then $f_{3kz+2z}^X(C) \le 9\alpha \cdot Opt(X, k, z)$.*

In other words, NK-MEANS gives a pseudo-approximation-preserving reduction from k-means with outliers to k-means, where any α approximation for k-means implies a 9α pseudo-approximation for k-means with outliers that throws away $3kz + 2z$ points as outliers.

Algorithm 1 for k-means with outliers

NK-MEANS$(X, k, z, \mathcal{A})$

1: Suppose we know the optimal objective value $Opt := Opt(X, k, z)$
2: Initialize $r \leftarrow 2(Opt/z)^{1/2}$, $Y \leftarrow \emptyset$
3: **for** each $x \in X$ **do**
4: Compute $B(x, r)$
5: **if** $|B(x, r)| \ge 2z$ **then**
6: Mark x as *heavy*
7: **end if**
8: **end for**
9: **for** each $x \in X$ **do**
10: **if** $B(x, r)$ contains no heavy points **then**
11: Update $Y \leftarrow Y \cup \{x\}$
12: **end if**
13: **end for**
14: Output $C \leftarrow \mathcal{A}(X \setminus Y, k)$

3.1 Implementation Details

Here we describe a simple implementation of NK-MEANS that achieves runtime $O(n^2 d) + T(n)$ assuming we know the optimal objective value, Opt, where $T(n)$ is the runtime of the algorithm $\mathcal{A}$ on inputs of size n. This assumption can be removed by running that algorithm for many guesses of Opt, say by trying all powers of 2 to obtain a 2-approximation of Opt for the correct guess.

For our experiments, we implement the loop in Line 3 by enumerating over all pairs of points and computing their distance. This step takes time $O(n^2 d)$. We implement the loop in Line 9 by enumerating over all elements in $B(x, r)$ and checking if it is heavy for each $x \in X$. This step takes $O(n^2)$. Running $\mathcal{A}$ on $(X \setminus Y, k)$ takes $T(n)$ time. We summarize the result of this section in the following lemma:

Lemma 1. *Assuming that we know Opt and that $\mathcal{A}$ takes time $T(n)$ on inputs of size n, then* NK-MEANS *can be implemented to run in time $O(n^2 d) + T(n)$.*

4 Coreset of Near Linear Size in k

In this section we develop a general framework to speed up any k-means with outliers algorithm, and we apply this framework to NK-MEANS to show that we can achieve near-linear runtime. In particular, we achieve this by constructing what is called a *coreset* for the k-means with outliers problem of size $O(k \log n)$, which is *independent* of the number of outliers, z.

4.1 Coresets for k-Means with Outliers

Our coreset construction will leverage existing constructions of coresets for k-means with outliers. A coreset gives a good summary of the input instance in the following sense:

Definition 3 (Coreset for k-Means with Outliers)**.** [2] *Let (X, k, z) be an instance of k-means with outliers and Y be a (possibly weighted) subset of $\mathbb{R}^d$. We say the k-means with outliers instance (Y, k, z') is an (α, β)-coreset for X if for any set $C \subset \mathbb{R}^d$ of k-centers satisfying $f^Y_{\kappa_1 z'}(C) \leq \kappa_2 Opt(Y, k, z')$ for some $\kappa_1, \kappa_2 > 0$, we have $f^X_{\alpha\kappa_1 z}(C) \leq \beta\kappa_2 Opt(X, k, z)$.*

In words, if (Y, k, z') is an (α, β) coreset for (X, k, z), then running any (κ_1, κ_2)-approximate k-means with outliers algorithm on (Y, k, z') (meaning the algorithm throws away $\kappa_1 z'$ outliers and outputs a solution with cost at most $\kappa_2 Opt(Y, k, z'))$ gives a $(\alpha\kappa_1, \beta\kappa_2)$-approximate solution to (X, k, z).

Note that if Y is a weighted set with weights $w : Y \to \mathbb{R}_+$, then the k-means with outliers problem is analogously defined, where the objective is a weighted sum of assignment costs: $\min_C \sum_{y \in Y_z(C)} w(y) d^2(y, C)$. Further, note that NK-MEANS generalizes naturally to weighted k-means with outliers with the same guarantees.

The two coresets we will utilize for our construction are K-MEANS++ (Aggarwal et al., 2009) and Meyerson's sampling coreset (Meyerson et al., 2004). The guarantees of these coresets are as follows:

Theorem 2 (K-MEANS++)**.** *Let* K-MEANS++(X, k) *denote running* K-MEANS++ *on input points X to obtain a set $Y \subset X$ of size k. Further, let $Y_1, \ldots Y_k$ be the clustering of X with centers $y_1, \ldots, y_k \in Y$, respectively. We define a weight function $w : Y \to \mathbb{R}_+$ by $w(y_i) = |Y_i|$ for all $y_i \in Y$. Suppose $Y =$* K-MEANS++$(X, 32(k + z))$*. Then with probability at least* 0.03*, the instance (Y, k, z) where Y has weights w is an $(1, 124)$-coreset for the k-means with outliers instance (X, k, z).*

Theorem 3 (Sampling)**.** *Let S be a sample from X, where every $x \in X$ is included in S independently with probability $p = \max(\frac{36}{z} \log(\frac{4nk^2}{z}), 36\frac{k}{z} \log(2k^3))$. Then with probability at least $1 - \frac{1}{k^2}$, the instance $(S, k, 2.5pz)$ is a $(16, 29)$-coreset for (X, k, z).*

Observe that K-MEANS++ gives a coreset of size $O(k + z)$, and uniform sampling gives a coreset of size $O(\frac{kn}{z} \log n)$ in expectation. If z is small, then K-MEANS++ gives a very compact coreset for k-means with outliers, but if z is large – say $z = \Omega(n)$ – then K-MEANS++ gives a coreset of linear size. However, the case where z is large is exactly when uniform sampling gives a small coreset.

In the next section, we show how we can combine these two coresets to construct a small coreset that works for all z.

4.2 Our Coreset Construction: SAMPLECORESET

Using the above results, our strategy is as follows: Let (X, k, z) be an instance of k-means with outliers. If $p > 1$, then we can show that $z = O(k \log n)$, so we can simply run K-MEANS++ on the input instance to get a good coreset. Otherwise, z is large, so we first subsample approximately $\frac{kn}{z}$ points from X. Let S denote the resulting sample. Then we compute a coreset on S of size $32(k + 2.5pz)$, where we scale down the number of outliers from X proportionally.

Algorithm 2 formally describes our coreset construction. We will prove that SAMPLECORESET outputs with constant probability a good coreset for the k-means with outliers instance (X, k, z) of size $O(k \log n)$. In particular, we will show:

Theorem 4. *With constant probability,* SAMPLECORESET *outputs an $(O(1), O(1))$-coreset for*

Algorithm 2 Coreset Constuction for k-Means with Outliers

SAMPLECORESET(X, k, z)

1: Let $p = \max(\frac{36}{z} \log(\frac{4nk^2}{z}), 36\frac{k}{z} \log(2k^3))$.
2: **if** $p > 1$ **then**
3: Output $Y \leftarrow$ K-MEANS++$(X, 32(k + z))$.
4: **else**
5: Let S be a sample drawn from X, where each $x \in X$ is included in S independently with probability p.
6: Output $Y \leftarrow$ K-MEANS++$(S, 32(k + 2.5pz))$
7: **end if**

[2] Note that our definition of coreset is parameterized by the *number* of outliers, z, in contrast to previous work such as Meyerson et al. (2004) and Huang et al. (2018), whose constructions are parametereized by the *fraction* of outliers, z/n.

the k-means with outliers instance (X, k, z) of size $O(k \log n)$ in expectation.

4.3 A Near Linear Time Algorithm for k-Means With Outliers

Using SAMPLECORESET, we show how to speed up NK-MEANS to run in near linear time: Let Y be the result of SAMPLECORESET(X, k, z). Then, to choose k centers we run NK-MEANS$(Y, k, z, \mathcal{A})$ if $p > 1$; otherwise, run NK-MEANS$(Y, k, 2.5pz, \mathcal{A})$, where $\mathcal{A}$ is any $O(1)$-approximate k-means algorithm with runtime $T(n)$ on inputs of size n.

Theorem 5. *There exists an algorithm that outputs with a constant probability an $O(1)$-approximate solution to k-means with outliers while discarding $O(kz)$ outliers in expected time $O(kdn \log^2 n) + T(k \log n)$.*

5 Experiment Results

This section presents our experimental results. The main conclusions are:

- Our algorithm NK-MEANS almost always has the best performance and finds the largest proportion of ground truth outliers. In the cases where NK-MEANS is not the best, it is competitive within 5%.
- Our algorithm results in a stable solution. Algorithms without theoretical guarantees have unstable objectives on some experiments.
- Our coreset construction SAMPLECORESET allows us to run slower, more sophisticated, algorithms with theoretical guarantees on large inputs. Despite their theoretical guarantees, their practical performance is not competitive.

The experiments shows that for a modest overhead for preprocessing, NK-MEANS makes k-means clustering more robust to noise.

Algorithms Implemented: Our new coreset construction makes it feasible to compare many algorithms for large data sets. Without this, most known algorithms for k-means with outliers become prohibitively slow even on modestly sized data sets. In our experiments, the coreset construction we utilize is SAMPLECORESET. More precisely, we first obtain a uniform sample by sampling each point independently with probability $p = \min\{\frac{2.5k \log n}{z}, 1\}$. Then, we run k-means++ on the sample to choose $k + pz$ centers – the resulting coreset is of size $k + pz$.

Next we describe the algorithms tested. Besides the coreset construction, we use k-means++ to mean running k-means++ and then Lloyd's algorithm for brevity. For more details, see Supplementary Material E. In the following, "on coreset" refers to running the algorithm on the coreset as opposed to the entire input. For fair comparison, we ensure each algorithm discards *exactly* z outliers regardless of the theoretical guarantee. At the end of each algorithm's execution, we discard the z farthest points from the chosen k centers as outliers.

Algorithms Tested:

1. **NK-MEANS (plus k-means++ on coreset)**: We use NK-MEANS with k-means++ as the input $\mathcal{A}$. The algorithm requires a bound on the objective Opt. For this, we considered powers of 2 in the range of $[n \min_{u,v \in X} d^2(u, v), n \max_{u,v \in X} d^2(u, v)]$.
2. **k-means++ (on the original input)**: Note this algorithm is not designed to handle outliers.
3. **k-means++ (on coreset)**: Same note as the above.
4. **Primal-dual algorithm of Charikar et al. (2001) (on coreset)**: A sophisticated algorithm based on constructing an approximate linear program solution.
5. **Uniform Sample (conservative uniform sampling plus k-means++)**: We run k-means++ on a uniform sample consisting of points sampled with probability $1/(2z)$.
6. **k-means– (Chawla and Gionis, 2013) on coreset**: This algorithm is a variant of the Lloyd's algorithm that executes each iteration of Lloyd's excluding the farthest z points.
7. **Local search of Gupta et al. (2017) (on coreset)** : This is an extension of the well-known k-means local search algorithm.

Experiments: We now describe our experiments which were done on both synthetic and real data sets.

Synthetic Data Experiments We first conducted experiments with synthetic data sets of various parameters. Every data set has n equal one million points and $k, d \in \{10, 20\}$ and $z \in \{10000, 50000\}$. Then we generated k random Gaussian balls. For the ith Gaussian we choose a center c_i from $[-1/2, 1/2]^d$ uniformly at random. These are the true centers. Then, we add n/k points drawn from $\mathcal{N}(c_i, 1)$ for the ith Gaussian. Next, we add noise. Points that are noise were sampled uniformly at random either from the same range $[-1/2, 1/2]^d$ or from a larger range $[-5/2, 5/2]^d$ depending on the experiment. We tagged the farthest z points from the centers $\{c_1, \ldots, c_k\}$ as ground truth outliers. We consider all possible 16 combinations of k, d, z values and the noise range.

	Primal-Dual	k-means–	Local Search	Uniform Sample	NK-MEANS
run time $>$ 4hrs	9/16	1/16	8/16	0/16	0/16
precision $<$ 0.8	2/16	0/16	0/16	4/16	0/16
total failure	11/16	1/16	8/16	4/16	0/16

Table 1: Failure rates due to high run time or low precision.

Each experiment was conducted 3 times, and we chose the result with the minimum objective and measured the total running time over all 3 runs. We aborted the execution if the algorithm failed to terminate within 4 hours. All experiments were performed on a cluster using a single node with 20 cores at 2301MHz and RAM size 128GB. Table 1 shows the number of times each algorithm aborted due to high run time. Also we measured the recall, which is defined as number of ground truth outliers reported by the algorithm, divided by z, the number of points discarded. The recall was the same as the precision in all cases, so we use precision in the remaining text. We choose 0.8 as the threshold for the acceptable precision and counted the number of inputs for which each algorithm had precision lower than 0.8. Our algorithm NK-MEANS, k-means++ on coreset, and k-means++ on the original input all had precision greater than 0.99 for all data sets and always terminated within 4 hours. The k-means++ results are excluded from the table. Details of the quality and runtime are deferred to the Supplementary Material E.

Real Data Experiments For further experiments, we used real data sets. We used the same normalization, noise addition method and the same value of $k = 10$ in all experiments. The data sets are SKIN-Δ, SUSY-Δ, and POWER-Δ. We normalized the data such that the mean and standard deviation are 0 and 1 on each dimension, respectively. Then we randomly sampled $z = 0.01n$ points uniformly at random from $[-\Delta, \Delta]^d$ and added them as noise. We discarded data points with missing entries.

Real Data Sets:

1. **SKIN-Δ** (ski). $n = 245057$, $d = 3$, $k = 10$, $z = 0.01n$. Only the first 3 features were used.
2. **SUSY-Δ** (sus). $n = 5$M, $d = 18$, $k = 10$, $z = 0.01n$.
3. **POWER-Δ** (pow). $n = 2049280$, $d = 7$, $k = 10$, $z = 0.01n$. Out of 9 features, we dropped the first 2, date and time, that denote when the measurements were made.
4. **KDDFULL** (kdd). $n = 4898431$, $d = 34$, $k = 3$, $z = 45747$. Each instance has 41 features and we excluded 7 non-numeric features. This data set has 23 classes and 3 classes account for 98.3% of the data points. We considered the other 45747 data points as ground truth outliers.

Table 2 shows our experiment results for the above real data sets. Due to their high failure rate observed in Table 1 and space constraints, we excluded the primal-dual, local search, and conservative uniform sampling algorithms from Table 2; all results can be found in Supplementary Material E. As before, we executed each algorithm 3 times. It is worth noting that NK-MEANS is the *only* algorithm with the worst case guarantees shown in Table 2. This gives a candidate explanation for the stability of our algorithm's solution quality across all data sets in comparison to the other algorithms considered.

The result shows that our algorithm NK-MEANS has the best objective for all data sets, except within 5% for SKIN-5. Our algorithm is always competitive with the best precision. For KDDFULL where we didn't add artificial noise, NK-MEANS significantly outperformed other algorithms in terms of objective. We can see that NK-MEANS pays extra in the run time to remove outliers, but this preprocessing enables stability, and competitive performance.

6 Conclusion

This paper presents a near linear time algorithm for removing noise from data before applying a k-means clustering. We show that the algorithm has provably strong guarantees on the number of outliers discarded and approximation ratio. Further, NK-MEANS gives the first pseudo-approximation-preserving reduction from k-means with outliers to k-means without outliers. Our experiments show that the algorithm is the fastest among algorithms with provable guarantees and is more accurate than state-of-the-art algorithms. It is of interest to determine if the algorithm achieves better guarantees if data has more structure such as being in low dimensional Euclidean space or being assumed to be well-clusterable (Braverman et al., 2011).

7 Acknowledgments

S. Im and M. Montazer Qaem were supported in part by NSF grants CCF-1617653 and 1844939. B. Moseley and R. Zhou were supported in part by NSF grants

	Skin-5	Skin-10	Susy-5	Susy-10	Power-5	Power-10	KddFull
NK-means	1 **0.8065** 56	**1** **0.9424** 56	**1** 0.8518 1136	**1** 0.9774 1144	**1** 0.6720 363	**1** 0.9679 350	**1** 0.6187 1027
k-means–	0.9740 0.7632 86	1.5082 0.9044 89	1.2096 0.8151 672	1.1414 0.9753 697	1.0587 0.6857 291	1.0625 0.9673 251	2.0259 **0.6436** 122
k-means++ coreset	1.0641 0.7653 39	1.4417 0.9012 37	1.0150 **0.8622** 462	1.0091 **0.9865** 465	1.0815 **0.7247** 177	1.0876 **0.9681** 142	1.5825 0.3088 124
k-means++ original	**0.9525** 0.7775 34	1.6676 0.8975 43	1.0017 0.8478 6900	1.0351 0.9814 6054	1.0278 0.7116 689	1.0535 0.9649 943	1.5756 0.3259 652

Table 2: Experiment results on real data sets with $\Delta = 5, 10$. The top, middle, bottom in each entry are the objective (normalized relative to NK-means), precision, and run time (sec.), resp. Bold indicates the best in the category.

CCF-1725543, 1733873, 1845146, a Google Research Award, a Bosch junior faculty chair and an Infor faculty award.

References

Kdd cup 1999 data. URL `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

Individual household electric power consumption data set. URL `https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption`.

Skin segmentation data set. URL `https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation`.

SUSY data set. URL `https://archive.ics.uci.edu/ml/datasets/SUSY`.

Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *RANDOM*, pages 15–28, 2009.

Charu C. Aggarwal and Chandan K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman and Hall/CRC, 2013.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.

Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33 (3):544–562, 2004.

Aditya Bhaskara, Sharvaree Vadgama, and Hong Xu. Greedy sampling for approximate clustering in the presence of outliers. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 11146–11155, 2019.

Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on well-clusterable data. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 26–40, 2011.

Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 378–388. IEEE, 1999.

Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, pages 642–651, 2001.

Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

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

Jiecao Chen, Erfan Sadeqi Azer, and Qin Zhang. A practical algorithm for distributed clustering and outlier detection. In *Advances in Neural Information Processing Systems*, pages 2248–2256, 2018.

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

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discover-

ing clusters in large spatial databases with noise. In *AAAI*, pages 226–231, 1996.

Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 569–578. ACM, 2011. doi: 10.1145/1993636.1993712. URL https://doi.org/10.1145/1993636.1993712.

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.

Sudipto Guha, Yi Li, and Qin Zhang. Distributed partial clustering. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 143–152. ACM, 2017.

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. doi: 10.14778/3067421.3067425. URL http://www.vldb.org/pvldb/vol10/p757-Lu.pdf.

Lingxiao Huang, Shaofeng H.-C. Jiang, Jian Li, and Xuan Wu. Epsilon-coresets for clustering (with outliers) in doubling metrics. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 814–825. IEEE Computer Society, 2018. doi: 10.1109/FOCS.2018.00082. URL https://doi.org/10.1109/FOCS.2018.00082.

A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264–323, 1999.

Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28 (2-3):89–112, 2004.

Jacob Kogan, Charles Nicholas, Marc Teboulle, et al. *Grouping multidimensional data*. Springer, 2006.

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 Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 646–659, 2018. doi: 10.1145/3188745.3188882. URL https://doi.org/10.1145/3188745.3188882.

Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time (1+ ε) -approximation algorithm for k-means clustering in any dimensions. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 454–462, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2228-9. doi: 10.1109/FOCS.2004.7. URL http://dx.doi.org/10.1109/FOCS.2004.7.

Shi Li and Xiangyu Guo. Distributed k-clustering for data with heavy noise. In *Advances in Neural Information Processing Systems*, pages 7838–7846, 2018.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–136, 1982.

Gustavo Malkomes, Matt Kusner, Wenlin Chen, Kilian Weinberger, and Benjamin Moseley. Fast distributed k-center clustering with outliers on massive data. In *NIPS*, pages 1063–1071, 2015.

Richard Matthew McCutchen and Samir Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *APPROX*, pages 165–178, 2008.

Adam Meyerson, Liadan O'callaghan, and Serge Plotkin. A k-median algorithm with running time independent of data size. *Machine Learning*, 56(1-3): 61–87, 2004.

Supplementary Material for: Fast Noise Removal for k-Means Clustering

A Analysis of NK-MEANS

The goal of this section is to prove Theorem 1. For the remainder of this section, let C^* denote the optimal z-solution and C denote the output of NK-MEANS$(X, k, z, \mathcal{A})$. Again, let $Opt := Opt(X, k, z)$.

We first show the benefits of optimal clusters having size at least $3z$.

Claim 1. *For each optimal center $c^* \in C^*$, let $x(c^*) \in X$ be the closest input point to c^*. If the cluster defined by $c^* \in C^*$ has size at least $3z$, then $d(x(c^*), c^*) \leq (\frac{Opt}{3z})^{1/2}$.*

Proof. Assume for contradiction that $d(x(c^*), c^*) > (\frac{Opt}{3z})^{1/2}$. Thus for each input point $x \in X$ that is assigned to center c^* in the optimal solution, we have $d^2(x, c^*) > \frac{Opt}{3z}$. There are at least $3z$ such points, so we can lower bound the assignment cost of these points by $3z(\frac{Opt}{3z}) = Opt$. This is a contradiction. □

Lemma 2. *If the cluster defined by $c^* \in C^*$ in the optimal solution has size at least $3z$, then $x(c^*)$ is heavy.*

Proof. Assume for contradiction that $x(c^*)$ is light, so $|B(x(c^*), r)| < 2z$. However, at least $3z$ points are assigned to c^* in the optimal solution, so there are at least $z + 1$ such points outside of $B(x(c^*), r)$.

Let $x \notin B(x(c^*), r)$ be such a point that is assigned to c^* in the optimal solution. By the triangle inequality, we have:

$$d(x, x(c^*)) \leq d(x, c^*) + d(x(c^*), c^*)$$

, which implies $d(x, c^*) \geq r - d(x(c^*), c^*) \geq 2(Opt/z)^{1/2} - (1/3)^{1/2}(Opt/z)^{1/2} \geq (Opt/z)^{1/2}$.

We conclude that for at least $z + 1$ points assigned to c^* in the optimal solution, their assignment costs are each at least Opt/z. This is a contradiction. □

Now using this result, we can upper bound the number of outliers required by NK-MEANS$(X, k, z, \mathcal{A})$ to remain competitive with the optimal z-solution (we will show that this quantity is upper bounded by the size of Y at the end of NK-MEANS$(X, k, z, \mathcal{A})$.)

Lemma 3. *At the end of* NK-MEANS$(X, k, z, \mathcal{A})$, *$|Y| \leq 3z\#\{$optimal clusters of size less than $3z\} + 2z$.*

Proof. Let $C^* = \{c_1^*, \ldots, c_k^*\} \subset X$.

For each $x \in X$, we will classify points into two types:

1) $d(x, C^*) \leq (Opt/z)^{1/2}$:

   We have that x satisfies $d(x, C^*) = d(x, c^*) \leq (Opt/z)^{1/2}$ for some $c^* \in C^*$. If the cluster defined by c^* has size at least $3z$, then by Lemma 2, $x(c^*)$ is heavy.

   Further, $d(x, x(c^*)) \leq d(x, c^*) + d(x(c^*), c^*) \leq (Opt/z)^{1/2} + (1/3)^{1/2}(Opt/z)^{1/2} \leq 2(Opt/z)^{1/2}$, so $x(c^*) \in B(x, r)$. Thus, we will not add x to Y if its nearest optimal cluster has size at least $3z$.

2) $d(x, C^*) > (Opt/z)^{1/2}$:

   We claim that there are at most $2z$ such $x \in X$ satisfying $d(x, C^*) > (Opt/z)^{1/2}$. Assume for contradiction that there are at least $2z + 1$ points $x \in X$ with $d(x, C^*) > (Opt/z)^{1/2}$. At most z of these points can be outliers, so the optimal solution must cluster at least $z + 1$ of these points. Thus we can lower bound the assignment cost of these points to C^* by:

   $$(z + 1)r^2 = (z + 1)(Opt/z) > Opt$$

   This is a contradiction.

We conclude that Y includes no points of type 1 from clusters of size at least $3z$, at most $3z$ points from each cluster of size less than $3z$, and at most $2z$ points of type 2. □

Corollary 2. *If every optimal cluster has size at least $3z$, then at the end of* NK-MEANS$(X, k, z, \mathcal{A})$, *$|Y| \leq 2z$.*

It remains to bound the $|Y|$-cost of C. Recall that the $|Y|$-cost of C is the cost of clustering X with C excluding the $|Y|$ points of largest assignment cost.

Intuitively, we do not need to worry about the points in X that are clustered in both the $|Y|$-solution C and the z-solution C^* – so the points in $X_{|Y|}(C) \cap X_z(C^*)$, because such points are paid for in both solutions.

We must take some care to bound the cost of the points in X that are clustered by the $|Y|$-solution C but are outliers in the z-solution C^*, because such points could have unbounded assignment costs to C^*. Here we will use the following property of heavy points:

Lemma 4. *Let $x \in X$ be a heavy point. Then there exists some optimal center $c^* \in C^*$ such that $d(x, c^*) \leq 2r$.*

Proof. Assume for contradiction that $d(x, c^*) > 2r$ for every $c^* \in C^*$. However, x is heavy, so $|B(x, r)| \geq 2z$. At least z points in $B(x, r)$ must be clustered by the optimal z-solution C^*.

Consider any such $x' \in B(x,r) \cap X_z(C^*)$. By the triangle inequality, we have

$$2r < d(x, C^*) \leq d(x, x') + d(x', C^*) \leq r + d(x', C^*)$$

This implies $d(x', C^*) > r$. Thus we can lower bound the assignment cost to C^* of all points in $B(x,r) \cap X_z(C^*)$ by:

$$\sum_{x' \in B(x,r) \cap X_z(C^*)} d^2(x', C^*) > zr^2 = 4Opt$$

This is a contradiction. □

Now we are ready to prove the main theorem of this section.

Proof of Theorem 1. By Corolloary 2, we have $f^X_{2z}(C) \leq f^{X \setminus Y}(C)$.

Further, by construction, C is an α-approximate k-means solution on $X \setminus Y$. Then

$$f^{X \setminus Y}(C) \leq \alpha f^{X \setminus Y^*} \leq \alpha f^{X \setminus Y}(C^*),$$

so it suffices to show that $f^{X \setminus Y}(C^*) \leq 9 \cdot Opt$.

We will consider two types of points:

1) $x \in (X \setminus Y) \cap X_z(C^*)$, so points in $X \setminus Y$ that are also clustered in the optimal z-solution C^*:

   We have

$$\sum_{x \in (X \setminus Y) \cap X_z(C^*)} d^2(x, C^*) \leq \sum_{x \in X_z(C^*)} d^2(x, C^*) = f^X_z(C^*).$$

2) $x \in (X \setminus Y) \cap (X \setminus X_z(C^*))$, so points in $X \setminus Y$ that are outliers in the optimal z-solution C^*:

   Observe that by definition, $|X \setminus X_z(C^*)| = z$, so there are at most z such x. By Lemma 2, for each such $x \in (X \setminus Y) \cap (X \setminus X_z(C^*))$, we have $d^2(x, C^*) \leq 4r^2$. Thus,

$$\sum_{x \in (X \setminus Y) \cap (X \setminus X_z(C^*))} d^2(x, C^*) \leq z(4r^2) = 8 f^X_z(C^*).$$

We conclude that $f^{X \setminus Y}(C^*) \leq f^X_z(C^*) + 8 f^X_z(C^*) = 9 f^X_z(C^*)$, as required. □

B Analysis of Coreset Construction and Near Linear Time Algorithm

The goal of this section is to prove Theorems 4 and 5. In our proof, we will use Theorems 2 and 3. For proofs of these theorems, see Sections C and D.

Proof of Theorem 4. We consider 2 cases: $p > 1$ and $p \leq 1$.

If $p > 1$, then $Y = \text{K-MEANS++}(X, 32(k + z))$. Because $p > 1$, we have $\max(36 \log(\frac{4nk^2}{z}), 36k \log(2k^3)) > z \Rightarrow z = O(k \log n)$. Then $|Y| = O(k + z) = O(k \log n)$, as required. Further, by Theorem 2, (Y, k, z) is a $(1, 124)$-coreset for (X, k, z) with constant probability.

Otherwise, if $p \leq 1$, then $Y = \text{K-MEANS++}(S, 32(k + 2.5pz))$. Thus, $|Y| = O(k + pz) = O(k \log n)$, as required. By Theorem 3, with probability at least $1 - \frac{1}{k^2}$, $(S, k, 2.5pz)$ is an $(16, 29)$-coreset for (X, k, z). For the remainder of this analysis, we assume this condition holds. We also know that $(Y, k, 2.5pz)$ is a $(1, 124)$-coreset for $(S, k, 2.5pz)$ with constant probability. Assume this holds for the remainder of the analysis.

Let C be a set of k centers satisfying $f^Y_{2.5\kappa_1 pz}(C) \leq \kappa_2 Opt(Y, k, 2.5pz)$. Because $(Y, k, 2.5pz)$ is an $(1, 124)$-coreset for $(S, k, 2.5pz)$, this implies:

$$f^S_{2.5\kappa_1 pz}(C) \leq 124\kappa_2 Opt(S, k, 2.5pz)$$

Because $(S, k, 2.5pz)$ is an $(16, 29)$-coreset for (X, k, z), we conclude:

$$f^X_{16\kappa_1 z}(C) \leq 29 \cdot 124\kappa_2 Opt(X, k, z)$$

Thus $(Y, k, 2.5pz)$ is an $(O(1), O(1))$-coreset for (X, k, z). □

Proof of Theorem 5. The approximation guarantees follow directly from Theorems 1 and 4.

To analyze the runtime, note that we can compute S in time $O(n)$. It is known that $\text{K-MEANS++}(X, k)$ takes $O(kdn)$ time (Arthur and Vassilvitskii, 2007; Aggarwal et al., 2009). Thus the runtime of SAMPLECORESET is dominated by the runtime of K-MEANS++ in both cases when $p > 1$ and $p \leq 1$, which takes $O((k \log n)dn)$ time.

Note that Y has size $O(k \log n)$ in expectation, so by Lemma 1, NK-MEANS can be implemented to run in time $O(k^2 d \log^2 n) + T(k \log n)$ on Y in expectation. □

C Proof of Theorem 2

Our proof of Theorem 2 relies on the following lemma which is implicit in Aggarwal et al. (2009):

Lemma 5. *Let* $Y = \text{K-MEANS++}(X, 32k)$. *Then* $f^X(Y) \leq 20 \cdot Opt(X, k)$ *with probability at least* 0.03.

Corollary 3. *Let* $Y = \text{K-MEANS++}(X, 32(k + z))$. *Then* $f^X(Y) \leq 20 \cdot Opt(X, k, z)$ *with probability at least* 0.03.

Proof. Let C^* be the optimal solution to the k-means with outliers instance (X, k, z). Note that $X \setminus X_z(C^*)$ is the set of outliers in the optimal solution, so $|X \setminus X_z(C^*)| \leq z$.

Then we have $f_z^X(C^*) \geq f^X(C^* \cup (X \setminus X_z(C^*)) \geq Opt(X, k+z)$. Combining this inequality with the above lemma gives the desired result. □

Using the above corollary, we can prove Theorem 2 by a moving argument:

Proof of Theorem 2. Let $Y = \text{K-MEANS++}(X, 32(k+z))$ with weights w as defined in the theorem statement. By the above corollary, we have $f^X(Y) \leq 20 \cdot Opt(X, k, z)$ with constant probability. We assume for the remainder of the proof that this condition holds.

Let C be any set of k centers such that $f^Y_{\kappa_1 z}(C) \leq \kappa_2 Opt(Y, k, z)$. We wish to bound $f^X_{\kappa_1 z}(C)$.

Note that by definition of w, $\sum_{y \in Y} w(y) = n$, and each weight is an integer. Thus for the remainder of the proof we interpret Y as a multiset such that for each $y \in Y$, there are $w(y)$ copies of y in the multiset.

It follows, we can associate each $x \in X$ with a unique $y(x) \in Y$ such that $d^2(x, y(x)) = d^2(x, Y)$ (so $y(x)$ is a unique copy of the center that x is assigned to in the clustering of X with centers Y.)

Now we partition $X_{\kappa_1 z}(C)$ into two sets:

$$X' := \{x \in X \mid x \in X_{\kappa_1 z}(C), y(x) \in Y_{\kappa_1 z}(C)\}$$

$$X'' := \{x \in X \mid x \in X_{\kappa_1 z}(C), y(x) \notin Y_{\kappa_1 z}(C)\}$$

For each $x \in X_{\kappa_1 z}(C)$, we want to bound its assignment cost. There are two cases:

1) $x \in X'$:

   We can bound $d^2(x, C) \leq 2d^2(x, y(x)) + 2d^2(y(x), C)$. Note that $y(x) \in Y_{\kappa_1 z}(C)$, so we can bound the assignment cost $d^2(y(x), C)$.

2) $x \in X''$:

   Note that because $y(x) \notin Y_{\kappa_1 z}(C)$, and $X_{\kappa_1 z}(C), Y_{\kappa_1 z}(C)$ are the same size, we can associate $y(x)$ with a unique element in $Y_{\kappa_1 z}(C)$, say $y(x') \in Y_{\kappa_1 z}(C)$ such that $x' \notin X_{\kappa_1 z}(C)$.

   Note that x' is not assigned in the $\kappa_1 z$-solution C, but x is assigned, so we can bound:

   $$d^2(x, C) \leq d^2(x', C) \leq 2d^2(x', y(x')) + 2d^2(y(x'), C)$$

By summing over all $x \in X_{\kappa_1 z}(C)$ and applying the above bounds, we have:

$$\begin{aligned} f^X_{\kappa_1 z}(C) &= \sum_{x \in X'} d^2(x, C) + \sum_{x \in X''} d^2(x, C) \\ &\leq 2 \sum_{x \in X'} (d^2(x, y(x)) + d^2(y(x), C)) \\ &\quad + 2 \sum_{x \in X''} (d^2(x', y(x')) + d^2(y(x'), C)) \\ &= 2f^X(Y) + 2f^Y_{\kappa_1 z}(C) \\ &\leq 2 \cdot 20 \cdot Opt(X, k, z) + 2\kappa_2 Opt(Y, k, z) \end{aligned}$$

An analogous argument gives that $Opt(Y, k, z) \leq 2 \cdot 20 \cdot Opt(X, k, z) + 2 \cdot Opt(X, k, z)$.

We conclude that $f^X_{\kappa_1 z}(C) \leq (40 + 84\kappa_2) Opt(X, k, z)$, where we may assume $\kappa_2 \geq 1$. This gives the desired result. □

D Proof of Theorem 3

The proof of Theorem 3 closely follows Meyerson et al. (2004) and is given here for completeness. Note that the key difference is that rather than sampling elements uniformly from X with replacement as in Meyerson et al. (2004), instead we sample each element of X independently with probability p. In this section let $C^* = \{c_1^*, \ldots, c_k^*\}$ be an optimal z-solution on X with clusters $C_1^*, \ldots, C_k^*$ such that C_i^* is the set of points assigned to center c_i^*. Let $n_i := |C_i^*|$ denote the size of cluster i.

Further, let S be a sample drawn from X of size s as in SAMPLECORESET, and let C a set of k centers satisfying $f^S_{2.5\kappa_1 pz}(C) \leq \kappa_2 Opt(S, k, 2.5pz)$ for some constants $\kappa_1, \kappa_2 > 0$.

The goal of this section is to prove that the sample S gives a good coreset of X for the k-means with outliers problem. We begin with some definitions that will be useful to our analysis:

Definition 4 (Large/Small Clusters)**.** *We say a cluster C_i^* of the optimal solution is **large** if $|C_i^*| \geq \frac{z}{k}$ and **small** otherwise.*

Definition 5 (Covered/Uncovered Clusters)**.** *Let $A := S_{2.5pz}(C^*) \cap S_{2.5\kappa_1 pz}(C)$, so A is the set of points in S that are assigned in the $2.5pz$-solution C^* and in the $2.5\kappa_1 pz$-solution C.*

*We say a large cluster C_i^* is **covered** if $|C_i^* \cap A| \geq \frac{1}{2}|C_i^* \cap S|$ and **uncovered** otherwise.*

Intuitively, in our analysis we want to show that most of the large clusters are covered, because the large clusters make up the majority of the points. In order to obtain a good summary of the whole point set, it suffices to obtain a good summary of the large clusters.

We quantify this notion of a good summary by defining a division of X into bins with respect to the centers, C^*.

Definition 6 (Bin Division)**.** *Let $b \in \mathbb{N}$. The b-bin division of X with respect to a set of k centers, C, is a partition $B_1, \dots B_b$ of X such that B_1 contains the $\frac{n}{b}$ points in X with the smallest assignment costs to C^*, B_2 contains the next $\frac{n}{b}$ cheapest points, and so on. More formally, the partition $B_1, \dots, B_b$ satisfies:*

- $|B_i| = \frac{n}{b}$ *for all* $i \in [b]$
- $\max_{x \in B_i} d^2(x, C) \leq \min_{x \in B_{i+1}} d^2(x, C)$ *for all* $i \in [b-1]$

For the remainder of this section, let $B_1, \dots, B_b$ denote the bin division of X with respect to the optimal z-solution C^*, where $b = \frac{n}{z}$ (so each bin has size z.)

The following lemma shows that our sample size is sufficiently large to obtain a good representation of each bin and each large cluster.

Lemma 6. *With probability at least $1 - \frac{1}{k^2}$, the following both hold:*

1) For all $i \in [b]$, $|S \cap B_i| \in [0.75pz, 1.25pz]$

2) For every large cluster C_i^*, $|S \cap C_i^*| \geq 0.75pn_i$

Proof. We will use the following standard Chernoff bounds, where $X = \sum_{i \in [n]} X_i$ is the sum of n i.i.d. random variables $X_i \sim Ber(p)$. For any $\delta \geq 0$, $Pr(|X - pn| \geq \delta pn) \leq 2exp(\frac{-\delta^2 pn}{2+\delta})$ and $Pr(X \geq (1-\delta)pn) \leq exp(\frac{-\delta^2 pn}{2+\delta})$.

We bound the failure probability for each bin and each large cluster. For all $i \in [b]$, we have $Pr(||S \cap B_i| - pz| \geq \frac{1}{4}pz) \leq exp(-\frac{1}{36}pz)$. For all large clusters C_i^*, we have $Pr(|S \cap C_i^*| \leq (1 - \frac{1}{4})pn_i) \leq exp(-\frac{1}{36}pn_i) \leq exp(-\frac{1}{36}p\frac{z}{k})$.

Now by union bounding over the failure events for each bin and large cluster, the probability that Condition 1 or 2 does not hold is upper bounded by:

$$\frac{n}{z} 2 \cdot exp(-\frac{1}{36}pz) + k \cdot exp(-\frac{1}{36}p\frac{z}{k})$$

Because $p \geq \frac{36}{z}\log(\frac{4nk^2}{z})$ and $p \geq 36\frac{k}{z}\log(2k^3)$, the first and second terms are both upper bounded by $\frac{1}{2k^2}$, respectively. ☐

For the remainder of this section, we assume that both Conditions 1 and 2 hold. Now we will formalize the idea that it suffices to get a good representation of the large clusters, because we can simply throw away the remaining points as outliers by increasing the number of outliers by a constant factor.

Lemma 7. *Let $X' = X_{covered}$, where $X_{covered}$ is the union of all covered large clusters (so X' excludes all small clusters, all uncovered clusters, and all outliers in the optimal z-solution on X.)*

Then $f^X_{(9+7\kappa_1)z}(C) \leq f^{X'}(C)$.

Proof. It suffices to show that $|X_{small} \cup (X \setminus X_z(C^*)) \cup X_{uncovered}| \leq (9 + 7\kappa_1)z$.

By definition $|X_{small}| \leq z$ (because each small cluster has at most $\frac{z}{k}$ points and there are at most k small clusters), and $|X \setminus X_z(C^*)| \leq z$, so it remains to bound the size of $X_{uncovered}$.

Recall that $A = S_{2.5pz}(C^*) \cap S_{2.5\kappa_1 pz}(C)$, so $|A| \geq s - 2.5pz - 2.5\kappa_1 pz$. This implies $|S \setminus A| \leq 2.5pz(1+\kappa_1)$.

By definition, a cluster C_i^* is uncovered if $|C_i^* \cap A| < \frac{1}{2}|C_i^* \cap S|$. This implies $|C_i^* \cap S| \leq 2|C_i^* \cap (S \setminus A)|$

By summing over all uncovered clusters, we have:

$$|X_{uncovered} \cap S|$$
$$\leq 2|X_{uncovered} \cap (S \setminus A)| \leq 2|S \setminus A| \leq 5pz(1 + \kappa_1)$$

Further, by Condition 2, for every large cluster C_i^*, we have $|S \cap C_i^*| \geq 0.75pn_i$, which gives $n_i \leq \frac{4}{3}\frac{1}{p}|S \cap C_i^*|$. This holds for every uncovered cluster, so we can upper bound:

$$|X_{uncovered}| \leq \frac{4}{3}\frac{1}{p}|X_{uncovered} \cap S| \leq \frac{20}{3}z(1 + \kappa_1)$$

Combining these bounds gives:

$$\begin{aligned} & |X_{small} \cup X_{uncovered} \cup (X \setminus X_z(C^*))| \\ \leq \;& z + z + \frac{20}{3}z(1 + \kappa_1) \\ = \;& (2 + \frac{20}{3}(1 + \kappa_1))z \leq (9 + 7\kappa_1)z \end{aligned}$$

☐

Now it suffices to bound the cost of clustering all the covered clusters, which we do so with a standard moving argument:

Lemma 8. $f^{X'}(C) \leq 2f_z^X(C^*) + \frac{32}{3}\frac{1}{p}(f^S_{2.5pz}(C^*) + f^S_{2.5\kappa_1 pz}(C))$

Proof. By the approximate triangle inequality:

$$f^{X'}(C) \leq 2f^{X'}(C^*) + 2\sum_{i \in [k]} n_i d^2(c_i^*, C),$$

where the first term accounts for moving each point in X' to its closest center in C^*, and the second term accounts for moving each point from its respective center in C^* to the nearest center in C.

Note that $X' \subset X_z(C^*)$, so $f^{X'}(C^*) \leq f^{X_z(C^*)}(C^*) = f_z^X(C^*)$.

It remains to bound $2\sum_{i\in[k]} n_i d^2(c_i^*, C)$. By a standard averaging argument, for any covered cluster C_i^*:

$$d^2(c_i^*, C) \leq \frac{1}{|C_i^* \cap A|} \sum_{x\in C_i^*\cap A} (2d^2(x, c_i^*) + 2d^2(x, C))$$

Because C_i^* is covered, $\frac{1}{|C_i^*\cap A|} \leq \frac{2}{|C_i^*\cap S|}$. Further, by Condition 2, we have $|C_i^* \cap S| \geq 0.75pn_i$.

Using these results, we can bound $2\sum_{i\in[k]} n_i d^2(c_i^*, C) \leq \frac{32}{3}\frac{1}{p}\sum_{i\in[k]}(Q_i + R_i)$, where we define $Q_i := \sum_{x\in C_i^*\cap A} d^2(x, c_i^*)$ and $R_i := \sum_{x\in C_i^*\cap A} d^2(x, C)$.

Recall that $A = S_{2.5pz}(C^*) \cap S_{2.5\kappa_1 pz}(C)$, so:

$$\sum_{i\in[k]} Q_i = \sum_{x\in X'\cap A} d^2(x, C^*) \leq \sum_{x\in A} d^2(x, C^*)$$
$$\leq f_{2.5pz}^S(C^*)$$

Analogously we can show $\sum_{i\in[k]} R_i \leq f_{2.5\kappa_1 pz}^S(C)$. Combining these bounds gives the desired result. □

Note that by definition of C, we have $f_{2.5\kappa_1 pz}^S(C) \leq \kappa_2 Opt(S, k, 2.5pz) \leq \kappa_2 f_{2.5pz}^S(C^*)$.

We require one more lemma to prove Theorem 3, because we must relate the $2.5pz$-cost of clustering S with C^* to the z-cost of clustering X with C^*. To do this we use the fact that the bins are approximately equally-represented:

Lemma 9. $f_{2.5pz}^S(C^*) \leq 1.25pf_z^X(C^*)$

Proof. By Condition 1, no bin contributes more than $1.25pz$ elements to S, so $S_{2.5pz}(C^*)$ excludes all of $S \cap B_{b-1}$, $S \cap B_b$.

Thus $S_{2.5pz}(C^*) \subset \bigcup_{i\in[b-2]} S \cap B_i$, so we have $f_{2.5pz}^S(C^*) \leq \sum_{i\in[b-2]} f^{S\cap B_i}(C^*)$.

Further, by definition of the bin division $B_1, \ldots, B_b$, we have $f_z^X(C^*) = \sum_{i\in[b-1]} f^{B_i}(C^*)$, and for any $i \in [b-2]$:

$$\max_{x\in S\cap B_i} d^2(x, C^*) \leq \min_{x\in B_{i+1}} d^2(x, C^*)$$

Our strategy will be to charge each point in $S \cap B_i$ to a point in B_{i+1}. Observe $|B_i| = z$ for all i and by Condition 1, $|S \cap B_i| \leq 1.25pz$. This implies $f^{S\cap B_i}(C^*) \leq 1.25pf^{B_{i+1}}(C^*)$ for all $i \in [b-2]$.

We conclude:

$$f_{2.5pz}^S(C^*) \leq \sum_{i\in[b-2]} f^{S\cap B_i}(C^*)$$
$$\leq 1.25p\sum_{i\in[b-2]} f^{B_{i+1}}(C^*) \leq 1.25pf_z^X(C^*)$$

□

Now we are ready to put these lemmas together to prove Theorem 3:

Proof of Theorem 3. By Lemma 6, with probability at least $1 - \frac{1}{k^2}$, both Conditions 1 and 2 hold. For the remainder of the proof, suppose both conditions hold.

Now by chaining Lemmas 7 and 8, we have:

$$f_{(9+7\kappa_1)z}^X(C) \leq f^{X'}(C) \leq 2f_z^X(C^*)$$
$$+\frac{32}{3}\frac{1}{p}(f_{2.5pz}^S(C^*) + f_{2.5\kappa_1 pz}^S(C))$$

Applying the definition of S, $f_{2.5\kappa_1 pz}^S(C) \leq \kappa_2 f_{2.5pz}^S(C^*)$:

$$f_{(9+7\kappa_1)z}^X(C) \leq f^{X'}(C) \leq 2f_z^X(C^*)$$
$$+\frac{32}{3}\frac{1}{p}(1+\kappa_2)f_{2.5pz}^S(C^*)$$

Finally, we apply Lemma 9 to obtain:

$$f_{(9+7\kappa_1)z}^X(C) \leq 2f_z^X(C^*) + \frac{32}{3}\frac{1}{p}(1+\kappa_2)(1.25pf_z^X(C^*))$$
$$= (\frac{46}{3} + \frac{40}{3}\kappa_2)Opt(X, k, z)$$

We may assume $\kappa_1, \kappa_2 \geq 1$, which completes the proof. □

E Other Experiment Results

E.1 Algorithms Implemented

We discuss each algorithm's implementation in more detail. When we ran k-means++, Lloyd's, k-means–, we terminated the execution when the objective improves less than a 1.00001 factor.

1. NK-MEANS (plus k-means++ on coreset). We added NK-MEANS to k-means++ as a preprocessing step. See Algorithm 1 for its pseudo-code. Since we had to guess the value of *Opt*, we considered all possible values that are power of 2 in the range of $[n \min_{u,v\in X} d^2(u, v), n \max_{u,v\in X} d^2(u, v)]$. Occasionally, when z was almost as big as n/k, NK-MEANS discarded almost all points – such cases

were considered as failure. However, if the guessed value of Opt is sufficiently large, NK-MEANS discards no points. Therefore, essentially this algorithm should be as good as running k-means++ on the coreset directly.

2. k-means++ (on the original input). The coreset is not used in this algorithm. So, we run k-means++ on the original input.

3. k-means++ (on coreset). This algorithm runs k-means++ on the coreset.

4. Primal-dual of Charikar et al. (2001) (on coreset). The primal-dual algorithm Charikar et al. (2001) is executed on the coreset. This algorithm is quite involved, and therefore, we only provide the parameters we chose to run the algorithm. For the whole algorithm description, see Charikar et al. (2001). The algorithm requires us to guess the value of Opt. As in the implementation of NK-MEANS, we considered all possible values that are power of 2 in the range of $[n \min_{u,v \in X} d^2(u,v), n \max_{u,v \in X} d^2(u,v)]$. This algorithm is based on a reduction to the facility location problem where one is allowed to choose as many centers as needed, but has to pay a (uniform) cost for using each center. Thus, another binary search is needed on the facility (center) opening cost. Each outlier cost is set to $Opt/(2z)$.

5. Uniform Sample (conservative uniform sampling plus k-means++): k-means++ was executed on a uniform sample consisting of points sampled with probability $1/(2z)$.

6. k-means– Chawla and Gionis (2013) on coreset. This algorithm is a variant of the Lloyd's algorithm that executes each iteration of Lloyd's excluding the farthest z points. That is, the algorithm repeats the following: it bring back all input points, excludes the farthest z points from the current centers, reassigns each remaining point to the closest center, and then recomputes the center of each cluster.

7. Local search of Gupta et al. (2017) (on coreset). In principle, this algorithm may end up with discarding $\Omega(zk \log n)$ points. However, it was observed that it never discarded more than $2z$ points in experimentation. We adopt the practical implementation of the algorithm described in Gupta et al. (2017). When the algorithm converges we enforce the farthest z points to be the outliers.

E.2 Experiment Results

In this section we present all experiment results.

(d, k, z)	NK-MEANS	Primal Dual	coreset k-means++	original k-means++	Local Search	k-means–	Uniform Sample
(10,10,10K)	333	$>$ 4hrs	223	110	347	357	14
(10,10, 50K)	89	5400	33	126	143	$>$ 4hrs	14
(10, 20, 10K)	864	$>$ 4hrs	667	249	3712	$>$ 4hrs	20
(10, 20, 50K)	214	$>$ 4hrs	102	718	3355	$>$ 4hrs	20
(20, 10, 10K)	6576	5759	306	141	500	519	24
(20, 10, 50K)	145	5855	58	180	269	$>$ 4hrs	24
(20, 20, 10K)	1590	$>$ 4hrs	1232	270	6698	7787	36
(20, 20, 50K)	361	$>$ 4hrs	203	1173	5278	$>$ 4hrs	36

Table 3: The running time (sec.) for synthetic data sets with noise sampled from $[-1/2, 1/2]^d$.

(d, k, z)	NK-MEANS	Primal Dual	coreset k-means++	original k-means++	Local Search	k-means–	Uniform Sample
(10,10,10K)	285	11271	129	406	247	4039	13
(10,10, 50K)	126	7638	35	772	154	154	14
(10, 20, 10K)	860	$>$ 4hrs	585	900	3966	3970	20
(10, 20, 50K)	280	$>$ 4hrs	103	1742	$>$ 4hrs	$>$ 4hrs	20
(20, 10, 10K)	415	13973	153	557	328	5658	25
(20, 10, 50K)	220	11356	60	1032	269	$>$ 4hrs	25
(20, 20, 10K)	1235	$>$ 4hrs	742	1050	6278	6629	36
(20, 20, 50K)	474	$>$ 4hrs	184	2079	4967	$>$ 4hrs	36

Table 4: The running time (sec.) for synthetic data sets with noise sampled from $[-5/2, 5/2]^d$.

(d, k, z)	NK-MEANS	Primal Dual	coreset k-means++	original k-means++	Local Search	k-means–	Uniform Sample
(10,10,10K)	1.0002	-	12.6865	1.0318	1.2852	1.2852	1.1501
(10,10, 50K)	1.7474	61.2273	1.7475	1.8293	1.2351	-	144.8253
(10, 20, 10K)	1.0002	-	8.3791	1.0141	1.2949	-	1.4066
(10, 20, 50K)	6.9265	-	33.7886	29.9914	1.2444	-	352.9934
(20, 10, 10K)	1.0002	94.9299	48.5067	1.0392	1.1547	1.1547	1.1499
(20, 10, 50K)	2.4857	1.1048	2.4857	2.6371	1.1293	-	76.9656
(20, 20, 10K)	1.0002	-	1.0422	1.0369	1.1532	1.1532	1.3261
(20, 20, 50K)	1.8725	-	1.8768	42.7528	1.1166	-	491.5800

Table 5: The objective value for synthetic data sets with noise sampled from $[-1/2, 1/2]^d$.

(d, k, z)	NK-MEANS	Primal Dual	coreset k-means++	original k-means++	Local Search	k-means–	Uniform Sample
(10,10,10K)	1.0002	1.1138	67.8622	62.6428	1.1152	1.1152	1.1363
(10,10, 50K)	1.0017	82.2102	168.1054	101.0163	1.1678	1.1678	107.6996
(10, 20, 10K)	1.0002	-	58.4918	74.9860	1.1264	1.1264	1.3842
(10, 20, 50K)	1.0018	-	172.8086	188.8177	-	-	348.5592
(20, 10, 10K)	1.0002	1.0410	128.6235	45.5434	1.0432	1.0432	1.1440
(20, 10, 50K)	1.0013	1.0932	159.8269	151.0288	1.0907	-	217.1559
(20, 20, 10K)	1.0002	-	119.9495	98.5209	1.0404	1.0404	1.3629
(20, 20, 50K)	1.0016	-	200.1431	242.0977	1.0535	-	454.5076

Table 6: The objective value for synthetic data sets with noise sampled from $[-5/2, 5/2]^d$.

(d, k, z)	NK-MEANS	Primal Dual	coreset k-means++	original k-means++	Local Search	k-means–	Uniform Sample
(10,10,10K)	1	-	0.9999	1	1	1	1
(10,10, 50K)	1	0.7586	1	1	1	-	0.5983
(10, 20, 10K)	1	-	0.9999	1	1	-	1
(10, 20, 50K)	0.9998	-	0.9956	0.9980	1	-	0.0768
(20, 10, 10K)	1	0.7070	1	1	1	1	1
(20, 10, 50K)	1	1	1	1	1	-	0.8830
(20, 20, 10K)	1	-	1	1	1	1	1
(20, 20, 50K)	1	-	1	0.9999	1	-	0.2312

Table 7: The precision/recall value for synthetic data sets with noise sampled from $[-1/2, 1/2]^d$.

(d, k, z)	NK-MEANS	Primal Dual	coreset k-means++	original k-means++	Local Search	k-means–	Uniform Sample
(10,10,10K)	1	1	1	1	1	1	1
(10,10, 50K)	1	1	1	1	1	1	1
(10, 20, 10K)	1	-	1	1	1	1	1
(10, 20, 50K)	1	-	1	1	-	-	0.9999
(20, 10, 10K)	1	1	1	1	1	1	1
(20, 10, 50K)	1	1	1	1	1	-	1
(20, 20, 10K)	1	-	1	1	1	1	1
(20, 20, 50K)	1	-	0.9999	1	1	-	1

Table 8: The precision/recall value for synthetic data sets with noise sampled from $[-5/2, 5/2]^d$.

	NK-MEANS	k-means–	coreset k-means++	original k-means++	Uniform Sample	Primal Dual	Local Search
SKIN-5	1 0.8065 56	0.9740 0.7632 86	1.0641 0.7653 39	0.9525 0.7775 34	1.1273 0.7575 1	1.1934 0.7636 1274	0.9746 0.7632 86
SKIN-10	1 0.9424 56	1.5082 0.9044 89	1.4417 0.9012 37	1.6676 0.8975 43	1.4108 0.9346 1	1.1197 0.9473 1931	1.5082 0.9044 89
SUSY-5	1 0.8518 1136	1.2096 0.8151 672	1.0150 0.8622 462	1.0017 0.8478 6900	1.1816 0.7622 97	1.2093 0.8558 4261	1.1337 0.8151 672
SUSY-10	1 0.9774 1144	1.1414 0.9753 697	1.0091 0.9865 465	1.0351 0.9814 6054	1.1611 0.9816 98	1.2474 0.9808 5075	1.1414 0.9753 697
POWER-5	1 0.6720 363	1.0587 0.6857 291	1.0815 0.7247 177	1.0278 0.7116 689	1.2814 0.7943 19	1.2655 0.6481 2494	1.0587 0.6857 291
POWER-10	1 0.9679 350	1.0625 0.9673 251	1.0876 0.9681 142	1.0535 0.9649 943	1.2408 0.9821 19	1.2299 0.9634 3097	1.0625 0.9673 251
KDDFULL	1 0.6187 1027	2.0259 0.6436 122	1.5825 0.3088 124	1.5756 0.3259 652	1.1527 0.5855 104	2.6394 0.5947 844	2.0259 0.6436 122

Table 9: Experiment results on real-world data sets with $\Delta = 5, 10$. The top, middle, bottom in each entry are the objective (normalized relative to NK-MEANS), precision, and run time (sec.), resp.