Coresets for Clustering with Missing Values

Vladimir Braverman
Johns Hopkins University
vova@cs.jhu.edu

Shaofeng H.-C. Jiang
Peking University
shaofeng.jiang@pku.edu.cn

Robert Krauthgamer
Weizmann Institute of Science
robert.krauthgamer@weizmann.ac.il

Xuan Wu
Johns Hopkins University
wu3412790@gmail.com

Abstract

We provide the first coreset for clustering points in $\mathbb{R}^d$ that have multiple missing values (coordinates). Previous coreset constructions only allow one missing coordinate. The challenge in this setting is that objective functions, like k-MEANS, are evaluated only on the set of available (non-missing) coordinates, which varies across points. Recall that an ϵ-coreset of a large dataset is a small proxy, usually a reweighted subset of points, that $(1+\epsilon)$-approximates the clustering objective for every possible center set.

Our coresets for k-MEANS and k-MEDIAN clustering have size $(jk)^{O(\min(j,k))}(\epsilon^{-1}d\log n)^2$, where n is the number of data points, d is the dimension and j is the maximum number of missing coordinates for each data point. We further design an algorithm to construct these coresets in near-linear time, and consequently improve a recent quadratic-time PTAS for k-MEANS with missing values [Eiben et al., SODA 2021] to near-linear time.

We validate our coreset construction, which is based on importance sampling and is easy to implement, on various real data sets. Our coreset exhibits a flexible tradeoff between coreset size and accuracy, and generally outperforms the uniform-sampling baseline. Furthermore, it significantly speeds up a Lloyd's-style heuristic for k-MEANS with missing values.

1 Introduction

We consider coresets and approximation algorithms for k-clustering problems, particularly k-MEANS[1] and more generally (k,z)-CLUSTERING (see Definition 2.1), for points in $\mathbb{R}^d$ with *missing values (coordinates)*. The presence of missing values in data sets is a common phenomenon, and dealing with it is a fundamental challenge in data science. While data imputation is a very popular method for handling missing values, it often requires prior knowledge which might not be available, or statistical assumptions on the missing values that might be difficult to verify [All01, LR19]. In contrast, our worst-case approach does not requires any prior knowledge. Specifically, in our context of clustering, the distance $\mathrm{dist}(x,c)$ between a clustering center point c and a data point x is evaluated only on the available (i.e., non-missing) coordinates. Similar models that aim to minimize clustering costs using only the available coordinates have been proposed in previous work [HB01, Wag04, CCB16, WLH+19], and some other relevant works were discussed in a survey [HC10].

Clustering under this distance function, which is evaluated only on the available coordinates, is a formidable computational challenge, because distances do not satisfy the triangle inequality, and

[1] In the usual k-MEANS problem (without missing coordinates), the input is a data set $X \subset \mathbb{R}^d$ and the goal is to find a center set $C \subset \mathbb{R}^d$, $|C| = k$ that minimizes the sum of squared distances from every $x \in X$ to C.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

therefore many classical and effective clustering algorithms, such as k-MEANS++ [AV07], cannot be readily applied or even be defined properly. Despite the algorithmic interest in clustering with missing values, the problem is still not well understood and only a few results are known. In a pioneering work, Gao, Langberg and Schulman [GLS08] initiated the algorithmic study of the k-CENTER problem with missing values. They took a geometric perspective and interpreted the k-CENTER with missing values problem as an affine-subspace clustering problem, and followup work [GLS10, LS13] has subsequently improved and generalized their algorithm. Only very recently, approximation algorithms for objectives other than k-CENTER, particularly k-MEANS, were obtained for the limited case of at most one missing coordinate in each input point [MF19] or for constant number of missing coordinates [EFG+21].

We focus on designing coresets for clustering with missing values. Roughly speaking, an ϵ-coreset is a small proxy of the data set, such that the clustering objective is preserved within $(1 \pm \epsilon)$ factor for all center sets (see Definition 2.2 for formal definition). Efficient constructions of small ϵ-coresets usually lead to efficient approximations schemes, since the input size is reduced to that of the coreset, see e.g. [HJLW18, FRS19, MF19]. Moreover, apart from speeding up approximation algorithms in the classical setting (offline computation), coresets can also be applied to design streaming [HM04, FS05, BFL+17], distributed [BEL13, RPS15, BLK18], and dynamic algorithms [Cha09, HK20], which are effective methods/models for dealing with big data, and recently coresets were used even in neural networks [MOB+20].

1.1 Our Results

Coresets. Our main result, stated in Theorem 1.1, is a near-linear time construction of coresets for k-MEANS with missing values. Here, an ϵ-coreset for k-MEANS for a data set X in $\mathbb{R}^d$ with missing coordinates is a weighted subset $S \subseteq X$ with weights $w : S \to \mathbb{R}_+$, such that

$$\forall C \subset \mathbb{R}^d, |C| = k, \qquad \sum_{x \in S} w(x) \cdot \operatorname{dist}^2(x, C) \in (1 \pm \epsilon) \sum_{x \in X} \operatorname{dist}^2(x, C),$$

where $\operatorname{dist}(x, c) := \sqrt{\sum_{i : x_i \text{ not missing}} (x_i - c_i)^2}$, and $\operatorname{dist}(x, C) := \min_{c \in C} \operatorname{dist}(x, c)$; note that the center set C does not contain missing values. More generally, our coreset also works for (k, z)-CLUSTERING, which includes k-MEDIAN (see Definition 2.1 and Definition 2.2). Throughout, we use $\tilde{O}(f)$ to denote $O(f \operatorname{poly} \log f)$.

Theorem 1.1 (Informal version of Theorem 3.1)**.** *There is an algorithm that, given $0 < \epsilon < 1/2$, integers $d, j, k \geq 1$, and a set $X \subset \mathbb{R}^d$ of n points each having at most j missing values, it constructs with constant probability an ϵ-coreset for k-MEANS on X of size $m = (jk)^{O(\min\{j,k\})} \cdot (\epsilon^{-1} d \log n)^2$, and runs in time $\tilde{O}\left((jk)^{O(\min\{j,k\})} \cdot nd + m\right)$.*

Our coreset size is only a low-degree polynomial of d, ϵ and $\log n$, and can thus deal with moderately-high dimension or large data set. The dependence on k (number of clusters) and j (maximum number of missing values per point) is also a low-degree polynomial as long as at least one of k and j is small. Actually, we justify in Theorem 1.2 that this exponential dependence in $\min\{j, k\}$ cannot be further improved, as long as the coreset size is in a similar parameter regime, i.e., the coreset size is of the form $f(j, k) \cdot \operatorname{poly}(\epsilon^{-1} d \log n)$. We provide the proof of Theorem 1.2 in the full version.

Theorem 1.2. *Consider the k-MEANS with missing values problem in $\mathbb{R}^d_?$ where each point can have at most j missing coordinates. Assume there is an algorithm that constructs an ϵ-coreset of size $f(j, k) \cdot \operatorname{poly}(\epsilon^{-1} d \log n)$, then $f(j, k)$ can not be as small as $2^{o(\min(j,k))}$.*

Furthermore, the space complexity of our construction algorithm is near-linear, and since our coreset is clearly mergeable, it is possible to apply the merge-and-reduce method [HM04] to convert our construction into a streaming algorithm of space $\operatorname{poly} \log n$. Prior to our result, the only known coreset construction for clustering with missing values is for the special case $j = 1$ [MF19][2] and has size $k^{O(k)} \cdot (\epsilon^{-2} d \log n)$. Since our coreset has size $\operatorname{poly}(k \epsilon^{-1} d \log n)$ when $j = 1$, it improves the dependence on k over that of [MF19] by a factor of $k^{O(k)}$.

[2] In fact, [MF19] considers a slightly more general setting where the input are arbitrary lines that are not necessarily axis-parallel.

Near-linear time PTAS for k-MEANS with missing values. Very recently, a PTAS for k-MEANS with missing values, was obtained by Eiben, Fomin, Golovach, Lochet, Panolan, and Simonov [EFG+21]. Its time bound is *quadratic*, namely $O(2^{\text{poly}(jk/\epsilon)} \cdot n^2 d)$, and since our coreset can be constructed in near-linear time, we can speedup this PTAS to *near-linear* time by first constructing our coreset and then running this PTAS on the coreset.

Corollary 1.3 (Near-linear time PTAS for k-MEANS with missing values)**.** *There is an algorithm that, given $0 < \epsilon < 1/2$, integers $d, j, k \geq 1$, and a set $X \subset \mathbb{R}^d$ of n points each having at most j missing values, it finds with constant probability a $(1+\epsilon)$-approximation for k-MEANS on X, and runs in time $\tilde{O}\big((jk)^{O(\min\{j,k\})} \cdot nd + 2^{\text{poly}(jk/\epsilon)} \cdot d^{O(1)}\big)$.*

Experiments. We implement our algorithm and validate its performance on various real and synthetic data sets in Section 4. Our coreset exhibits flexible tradeoffs between coreset size and accuracy, and generally outperforms a uniform-sampling baseline and a baseline that is based on imputation, in both error rate and stability, especially when the coreset size is relatively small. In particular, on each data set, a coreset of moderate size 2000 (which is 0.5%-5% of the data sets) achieves low empirical error (5%-20%). We further demonstrate an application and use our coresets to accelerate a Lloyd's-style heuristic adapted to the missing-values setting. The experiments suggest that running the heuristic on top of our coresets gives equally good solutions (error $< 1\%$ relative to running on the original data set) but is much faster (speedup > 5x).

1.2 Technical Overview

Our coreset construction is based on the importance sampling framework introduced by Feldman and Langberg [FL11] and subsequently improved and generalized by [FSS20, BJKW21]. In the framework, one first computes an importance score σ_x for every data point $x \in X$, and then draws independent samples with probabilities proportional to these scores. When no values are missing, the importance scores can be computed easily, even for general metric spaces [VX12b, FSS20, BJKW21]. However, a significant challenge with missing values is that distances do not satisfy the triangle inequality, hence importance scores cannot be easily computed.

We overcome this hurdle using a method introduced by Varadarajan and Xiao [VX12a] for projective clustering (where the triangle inequality similarly does not hold). They reduce the importance-score computation to the construction of a coreset for k-CENTER objective; this method is quite different from earlier approaches, e.g. [FL11, VX12b, FSS20, BJKW21], and yields a coreset for k-MEANS whose size depends linearly on $\log n$ and of course on the size of the k-CENTER coreset. (Mathematically, this arises from the sum of all importance scores.) We make use of this reduction, and thus focus on constructing (efficiently) a small coreset for k-CENTER with missing values.

An immediate difficulty is how to deal with the missing values. We show that it is possible to find a collection of subsets of coordinates $\mathcal{I}$ (so each $I \in \mathcal{I}$ is a subset of $[d]$), such that if we construct k-CENTER coresets S_I on the data set "restricted" to each $I \in \mathcal{I}$, then the union of these S_I's is a k-CENTER coreset for the original data set with missing values. Crucially, we ensure that each "restricted" data set does not contain any missing value, so that it is possible to use a classical coreset construction for k-CENTER. Finally, we show in a technical lemma how to find a collection as necessary of size $|\mathcal{I}| \leq (jk)^{O(\min\{j,k\})}$.

Since a "restricted" data set does not contain any missing values, we can use a classical k-CENTER coreset construction, and a standard construction has size $O(k\epsilon^{-d})$ [AP02], which is known to be tight. We bypass this ϵ^{-d} limitation by observing that actually $\tilde{O}(1)$-coreset for k-CENTER suffices, even though the final coreset error is ϵ. We observe that an $\tilde{O}(1)$-coreset can be constructed using a variant of Gonzalez's algorithm [Gon85].

To implement Gonzalez's algorithm, a key step is to find the *furthest* neighbor of a given subset of at most $O(k)$ points, and a naive implementation of this runs in linear time, which overall yields a quadratic-time coreset construction, because the aforementioned reduction of [VX12a] actually requires $\Theta(n/k)$ successive runs of Gonzalez's algorithm. To resolve this issue, we propose a fully-dynamic implementation of Gonzalez's algorithm so that a furthest-point query is answered in time $\text{poly}(k \log n)$, and the point-set is updated between successive runs instead of constructed from scratch. Our dynamic algorithm is based on a random-projection method that was proposed for furthest-point queries in the streaming setting [Ind03]. Specifically, we project the (restricted) data set onto several random directions, and on each projected (one-dimensional) data set we apply a data structure for intervals.

1.3 Additional Related Work

Coresets for k-MEANS and k-MEDIAN clustering have been studied extensively for two decades, and we only list a few notable results. The first strong coresets for Euclidean k-MEANS and k-MEDIAN were given in [HM04]. In the last decade, most work on coresets for clustering follows the importance sampling framework initiated in [LS10, FL11]. In Euclidean space, recent work showed that coresets for k-MEANS and k-MEDIAN clustering can have size that is independent of the Euclidean dimension [FSS20, SW18, HV20]. Beyond Euclidean space, coresets of size independent of the data-set size were constructed also for many important metric spaces [HJLW18, BJKW21, CASS21]. A more comprehensive overview can be found in recent surveys [Phi17, Fel20].

Recently, attention was given also to non-traditional settings of coresets for clustering, including coresets for Gaussian mixture models (GMM) [LFKF17, FKW19]; simultaneous coresets for a large family of cost functions that include both k-MEDIAN and k-CENTER [BJKW19]; and coresets for clustering under fairness constraints [HJV19]. Also considered were settings that capture uncertainty, for example when each point is only known to lie in a line (i.e., clustering lines) [MF19], and when each point comes from a finite set (i.e., clustering point sets) [JTMF20].

2 Preliminaries

We represent a data point as a vector in $(\mathbb{R} \cup \{?\})^d$, and a coordinate takes "?" if and only if it is missing. Let $\mathbb{R}^d_?$ be a shorthand for $(\mathbb{R} \cup \{?\})^d$. Throughout, we consider a data set $X \subset \mathbb{R}^d_?$. The distance is evaluated only on the coordinates that are present in both x, y, i.e.,

$$\forall x, y \in \mathbb{R}^d_?, \qquad \operatorname{dist}(x, y) := \sqrt{\sum_{i : x_i, y_i \neq ?} (x_i - y_i)^2}.$$

For $x \in \mathbb{R}^d_?$, we denote the set of coordinates that are not missing by $I_x := \{i : x_i \neq ?\}$. For integer $m \geq 1$, let $[m] := \{1, \ldots, m\}$. For two points $p, q \in \mathbb{R}^d_?$ and an index set $I \subseteq I_p \cap I_q$, we define the *I-induced distance* to be $\operatorname{dist}_I(p, q) := \sqrt{\sum_{i \in I} (p_i - q_i)^2}$. A point $x \in \mathbb{R}^d_?$ is called a j-point if it has at most j missing coordinates, i.e., $|I_x| \geq d - j$.

We consider a general k-clustering problem called (k, z)-clustering, which asks to minimize the following objective function. This objective function (and problem) is also called k-MEDIAN when $z = 1$ and k-MEANS when $z = 2$.

Definition 2.1 ((k, z)-CLUSTERING)**.** For data set $X \subset \mathbb{R}^d_?$ and a center set $C \subset \mathbb{R}^d$ containing k (usual) points, let

$$\operatorname{cost}_z(X, C) := \sum_{x \in X} \operatorname{dist}^z(x, C).$$

Definition 2.2 (ϵ-Coreset for (k, z)-CLUSTERING)**.** For data set $X \subset \mathbb{R}^d_?$, we say a weighted set $S \subseteq X$ with weight function $w : S \to \mathbb{R}_+$ is an ϵ-coreset for (k, z)-CLUSTERING, if

$$\forall C \subset \mathbb{R}^d, |C| = k, \qquad \sum_{x \in S} w(x) \cdot \operatorname{dist}^z(x, C) \in (1 \pm \epsilon) \cdot \operatorname{cost}_z(X, C).$$

3 Coresets

Theorem 3.1. *There is an algorithm that, given as input a data set $X \subset \mathbb{R}^d_?$ of size $n = |X|$ consisting of j-points and parameters $k, z \geq 1$ and $0 < \epsilon < 1/2$, constructs with constant probability an ϵ-coreset of size $m = \tilde{O}\left(z^z \cdot \frac{(j+k)^{j+k+1}}{j^j k^{k-z-2}} \cdot \epsilon^{-2} (d \log n)^{\frac{z+2}{2}}\right)$ for (k, z)-CLUSTERING of X, and runs in time $\tilde{O}\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-2}} \cdot nd + m\right)$.*

Theorem 3.1 is the main theorem of this paper, and we only present a sketch of the proof in this section due to the space limitation. Please see the full version for a more detailed and self-contained proof, as well as a complete description of our algorithm. We remark that $\frac{(j+k)^{j+k}}{j^j k^k} \leq (jk)^{O(\min(j,k))}$ which is used in the statement of Theorem 1.1.

As mentioned in Section 1, we use importance sampling method which is a well-known technique for constructing coresets [FL11, FSS20]. A key step is to compute for every data point $x \in X$ an importance score $\sigma_x \geq 0$ that estimates its maximum relative contribution to any solution. The computation of $\{\sigma_x\}$ is standard in metric spaces, see e.g. [FSS20, BJKW21], but this is not applicable for us because distances with missing values do not satisfy the triangle inequality. Hence, we employ an alternative approach proposed by Varadarajan and Xiao [VX12a, Lemma 3.1], which reduces the computation of $\{\sigma_x\}$ to finding coresets for k-CENTER. This coreset concept, adapted to our setting, is defined as follows.

Definition 3.1. An *α-coreset for k-*CENTER of a data set $X \subset \mathbb{R}^d_?$ is a subset $Y \subseteq X$ such that

$$\forall C \subset \mathbb{R}^d, |C| = k, \qquad \max_{x \in X} \operatorname{dist}(x, C) \leq \alpha \cdot \max_{y \in Y} \operatorname{dist}(y, C).$$

We focus on an efficient construction of an $\tilde{O}(1)$-coreset for k-CENTER. The main concern is that the reduction in [VX12a, Lemma 3.1] requires constructing a k-CENTER coreset for multiple data sets. Fortunately, these data sets are related — each data set is a subset of the previous one — and thus to execute the reduction in near-linear time, we need a k-CENTER coreset construction that supports efficient point deletions. Such a dynamic coreset for k-CENTER with missing values is our main technical contribution. We stated it next, and outline its proof in Section 3.1.

Lemma 3.2. *There is a randomized dynamic algorithm with the following guarantees. The input is a dynamic set $X \subset \mathbb{R}^d_?$ of j-points, such that X undergoes q adaptive updates (point insertions and deletions) and the points ever added are fixed in advance (non-adaptively). The algorithm maintains in time $\tilde{O}\left(\frac{(j+k)^{j+k+1}}{j^j k^k} \cdot (j + k \log q)(d + k^2 \log q)\right)$ per update, a subset $Y \subseteq X$ of size $|Y| \leq O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-1}} \cdot \log d\right)$ such that with constant probability, Y is an $O(k\sqrt{d \log q})$-coreset for k-*CENTER *on X after every update.*

3.1 Proof of Lemma 3.2: Dynamic $\tilde{O}(1)$-Coresets for k-Center Clustering

As mentioned, the high level idea is to identify a collection $\mathcal{I}$ of subsets of coordinates (so each $I \in \mathcal{I}$ satisfies $I \subseteq [d]$), construct for each $I_i \in \mathcal{I}$ an α-coreset Y_i (for α determined later) for k-CENTER on the data set X with coordinates *restricted* on I_i (as defined below), and then their union $\bigcup_i Y_i$ would be the overall $\alpha\sqrt{d}$-coreset for k-CENTER on X.

Definition 3.2. For a point $p \in \mathbb{R}^d_?$ and a subset $I \subseteq I_p$, define $p_{|I} \in \mathbb{R}^I$ in the obvious way, by selecting the coordinates $\{p_i\}_{i \in I}$. Define the *I-restricted data set* to be $X_{|I} := \{p_{|I} : p \in X, I \subseteq I_p\}$. Since each vector in $X_{|I}$ arises from a specific vector in X, a subset $Y \subseteq X_{|I}$ corresponds to a specific subset of X, and we shall denote this subset by Y^{-1}.

We observe that $X_{|I} \subset \mathbb{R}^{|I|}$, namely, has no missing values (because of the condition $I \subseteq I_p$). Thus, the metric space on the restricted data set is an ordinary metric space that satisfies the triangle inequality, and so our goal is reduced to constructing k-CENTER coresets for this ordinary setting. However, another key step is to identify a small collection $\mathcal{I}$ such that the union of the coresets restricted on $\mathcal{I}$ yields a coreset. To this end, we consider the so-called (j, k, d)-family of coordinates as in Definition 3.3. We show in Lemma 3.3 that such a family guarantees the correctness of the coreset, and in Lemma 3.4 that a small family exists and moreover can be constructed efficiently.

Definition 3.3. A collection of sets $\mathcal{I} \subset 2^{[d]}$ is called a (j, k, d)-*family* if for every two disjoint subsets $J, K \subset [d], |J| = j, |K| = k$, the family includes $I \in \mathcal{I}$ that misses J and contains K, i.e., $I \cap J = \emptyset$ and $K \subset I$.

Lemma 3.3. *Suppose $\mathcal{I}$ is a (j, k, d)-family Let $X \subseteq \mathbb{R}^d_?$ be a set of j-points, and for every $I \in \mathcal{I}$, let Y_I be an α-coreset for k-*CENTER *on $X_{|I}$. Then $\cup_{I \in \mathcal{I}} Y_I^{-1}$ is an $\alpha\sqrt{d}$-coreset for k-Center on X.*

Proof. It suffices to show that for any center set $C = \{c^1, \dots, c^k\} \subseteq \mathbb{R}^d$ with k points and $x \in X$, if $\mathrm{dist}(x, C) \geq r$ for some $r \geq 0$, then we can find a coreset point $y \in \cup_{I \in \mathcal{I}} Y_I^{-1}$ such that $\mathrm{dist}(y, C) \geq \frac{r}{\alpha\sqrt{d}}$.

For $i \in [k]$, let $t_i \in \arg\max_{t \in I_x} |x_t - c_t^i|$, i.e., t_i is the index of coordinate that contributes the most in distance $\mathrm{dist}(x, c^i)$, so $|x_{t_i} - c_{t_i}^i| \geq \frac{r}{\sqrt{d}}$. Let K be any k-subset such that $K \subseteq I_x$ and $\{t_1, \dots, t_k\} \subseteq K$. Since $\mathcal{I}$ is a (j, k, d)-family and $|I_x| \geq d - j$, by definition, there exists an $I \subseteq \mathcal{I}$ such that $K \subseteq I \subseteq I_x$. We note that

$$\mathrm{dist}(x_{|I}, C_{|I}) = \mathrm{dist}_I(x, C) = \min_{i \in [k]} \mathrm{dist}_I(x, c^i) \geq \min_{i \in [k]} \mathrm{dist}_K(x, c^i) \geq \min_{i \in [k]} |x_{t_i} - c_{t_i}^i| \geq \frac{r}{\sqrt{d}}.$$

Since $I \subseteq I_x$, we know that $x_{|I} \in X_{|I}$. As Y_I is an α-coreset for $X_{|I}$, we know that there exists $y \in Y_I^{-1}$ such that

$$\mathrm{dist}(y, C) \geq \mathrm{dist}_I(y, C) = \mathrm{dist}(y_{|I}, C_{|I}) \geq \frac{\mathrm{dist}(x_{|I}, C_{|I})}{\alpha} \geq \frac{r}{\alpha\sqrt{d}}.$$

□

Lemma 3.4 asserts the existence of a small (j, k, d)-family. We remark that this combinatorial structure has been employed in designing fault-tolerant data structures and algorithms (cf. [DK11, DGR21, KP21]), and they obtained similar bounds although their context and language is different.

Lemma 3.4. *There is a (j, k, d)-family $\mathcal{I}$ of size $|\mathcal{I}| = O\left(\frac{(j+k)^{j+k+1}}{j^j k^k} \log d\right)$. Moreover, there is a randomized algorithm that constructs such $\mathcal{I}$ in time $O(d \cdot |\mathcal{I}|)$ with probability at least $1 - \frac{1}{d^{j+k}}$.*

k-Center coreset for restricted data set via Gonzalez's algorithm. Finally, the k-Center coreset for the restricted data set on each $I \in \mathcal{I}$ is constructed using an approximate version of Gonzalez's algorithm [Gon85], where we first pick an arbitrary data point as the initial coreset, and in every iteration an $\tilde{O}(1)$-approximate furthest neighbor in the dataset is picked into the coreset, and we do this for k times. We show that the resulting coreset, consisting of $k + 1$ points, is an $\tilde{O}(1)$-coreset for k-Center. The assumption that the input forms a metric space is crucial, and this is guaranteed since we always run on a restricted data set that satisfies the triangle inequality.

Dynamic implementation of Gonzalez's algorithm. To make this k-Center coreset construction dynamic, we adapt the random projection technique to Gonzalez's algorithm. We project the (restricted) point set onto several random lines and use a one-dimensional data structure to construct k-Center coreset for each of these (projected) one-dimensional data set. Note that the key step in Gonzalez's algorithm is finding a furthest neighbor, and we can show that our projection method yields an $O(k\sqrt{\log n})$-approximation of the furthest neighbor with high probability.

Lemma 3.5. *Let $A \subset \mathbb{R}^d$, $|A| = n$, $\delta > 0$ and integer $k \geq 1$. Let $\mathcal{V}$ be a collection of $O(k \log n + \log \frac{1}{\delta})$ random vectors, each drawn independently from $\mathcal{N}(0, I_d)$. Then with probability at least $1 - \delta$, for every $P \subseteq A$ and $Q \subseteq A$, $|Q| \leq k$, if $p^\star$ is a furthest point in P from Q, and for every $v \in \mathcal{V}$ we let $\langle p^v, v\rangle$ be a furthest point in $\langle P, v\rangle$ from $\langle Q, v\rangle$, then*

$$\mathrm{dist}(p^\star, Q) \leq O(k\sqrt{\log n}) \cdot \max_{v \in \mathcal{V}} \mathrm{dist}(p^v, Q),$$

where we denote $\langle X, v\rangle := \{\langle x, v\rangle : x \in X\}$.

For each one-dimensional line, we use a balanced search tree structure to support the point update and the furthest neighbor query. All operations can be done in $O(k \log m)$ time where m is the number of currently inserted elements. This combining with the above lemmas implies Lemma 3.2.

4 Experiments

We implement our proposed coreset construction algorithm, and we evaluate its performance on real and synthetic datasets. We focus on k-Means with missing values, and we examine the speedup

Table 1: Parameters of the datasets. n is the number of data points, d is the dimension, k is the number of clusters, j is the maximum number of missing coordinates for each point. n,d,j are given, and k is chosen by us.

Data set	n	d	k	j
Russian housing	30471	4	3	3
KDD cup	50000	31	5	30
Vertical farming	400180	4	2	4
Synthetic	200000	3	3	3

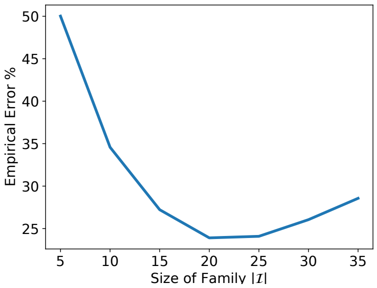


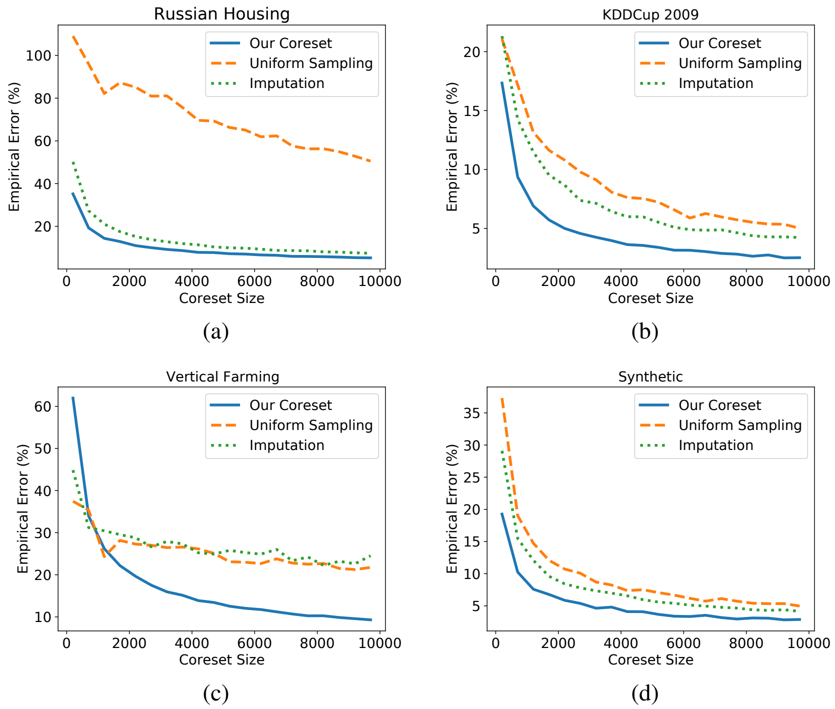
Figure 1: The average empirical error of Russian housing data set with respect to family size $|\mathcal{I}|$ on 10 independent experiments.

for a Lloyd's-style heuristic. In addition to measuring the absolute performance of our coreset, we also compare it with a) uniform sampling baseline, which is a naive way to construct coresets, and b) an imputation-based baseline where missing values are filled in by random values and then a standard importance-sampling coreset construction (cf. [FL11]) is run on top of it. We implement the algorithms using C++ 11, on a laptop with Intel i5-8350U CPU and 8GB RAM.

Datasets. We run our experiments on three real datasets and one synthetic dataset. Below, we briefly describe how we process and choose the attributes of the dataset, and the parameters of the datasets after processing are summarized in Table 1.

1. Russian housing [Rus17] is a dataset on Russian house market. We pick four main numerical attributes of the houses which are the full area, the live area, the kitchen area and the price, and the price attribute is divided by 10^5 so as it lies in the similar range of other attributes. Three columns regarding area contain missing values, and the price column doesn't contain any missing value.
2. KDDCup 2009 [KDD09] is a dataset on customer relationship prediction. We pick 31 numerical attributes that have similar magnitudes. Each column contains missing values.
3. Vertical farming [Sam21] is a dataset about cubes which are used for advanced vertical farming. We include all of four numerical attributes of the dataset. Each column contains missing values.
4. Synthetic dataset. We generate a large synthetic dataset to validate our algorithm's scalability. Data points are randomly generated so that 97% of them are in a square and 3% of them are far away from the square. After that, we delete 25% of attributes at random. We remark that the 3% far away points is to make the dataset less uniform which prevents it from being trivial for clustering.

Implementation notes. In our experiments, we follow a standard practice of fixing coreset size in each experiment (cf. [BBH$^+$20, JTMF20]). Recall that when computing the importance score, our algorithm chooses a family $\mathcal{I}$ of subsets of coordinates and work on each restricted data set $X_{|I}$ for $I \in \mathcal{I}$ (see Section 3.1). For a fixed size coreset, the family size $|\mathcal{I}|$ is a parameter that needs to be optimized. In Figure 1, we plot the empirical error (defined in (1), Section 4.1) for the Russian housing dataset with respect to the family size $|\mathcal{I}|$. Although Lemma 3.4 gives a theoretical upper bound on $|\mathcal{I}|$ but our experiments suggest that a much smaller size $|\mathcal{I}| = 20$ is optimal in this case.



(a)

(b)

(c)

(d)

Figure 2: Accuracy evaluation for the datasets with respect to varing coreset sizes, compared against uniform sampling and imputation baselines.

4.1 Accuracy of Coresets

We evaluate the accuracy versus size tradeoff of our coresets. Since the coreset should preserve the clustering cost for *all* centers, we evaluate the accuracy by testing the *empirical error* on a selected set of centers $\mathcal{C}$. Namely, for a data set X, a coreset $D \subseteq X$ and a collection of center sets $\mathcal{C}$, we define the empirical error of D as

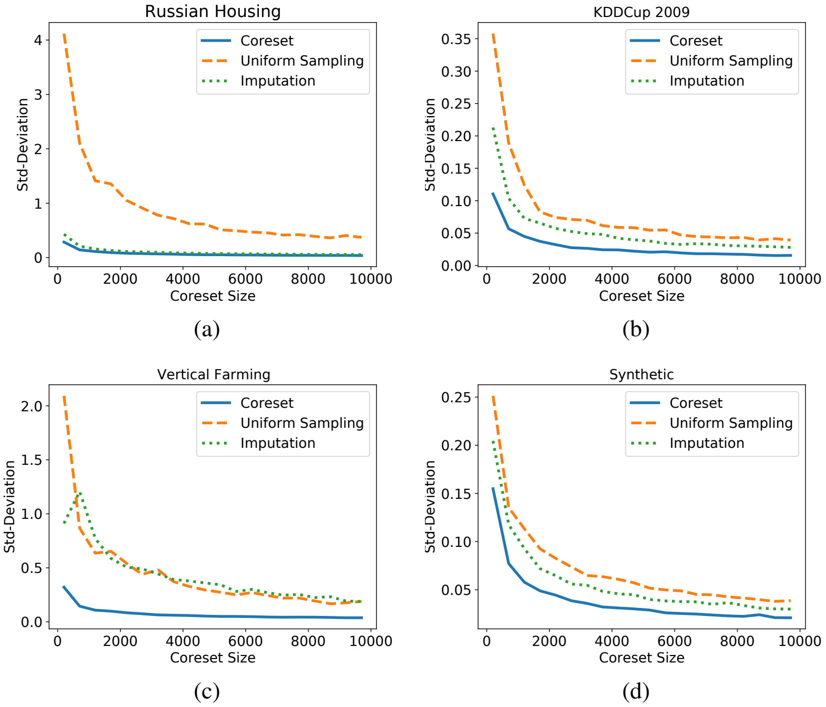
$$\mathrm{err}(D) = \max_{C \in \mathcal{C}} \frac{|\mathrm{cost}(D, C) - \mathrm{cost}(X, C)|}{\mathrm{cost}(X, C)}. \tag{1}$$

We use a randomly selected collection of centers $\mathcal{C}$ that consists of 100 randomly generated k-subset $C \subset \mathbb{R}^d$. Since both the evaluation method and the algorithm has randomness, we run the experiment for $T = 10^3$ times with independent random bits and report the average empirical error to make it stable. We choose 20 different coreset sizes from 200 to 9700 in a step size of 500, and report the corresponding average empirical error.

Results. We report the size versus accuracy tradeoff of our coreset for all four datasets in Figure 2, and record the standard deviation in Figure 3. We compare these results against the abovementioned uniform sampling and imputation baseline. As can be seen from the figures, the accuracy of our coreset improves when the size increases, and we achieve 5%-20% error using only 2000 coreset points (which is within $0.5\% - 5\%$ of the datasets). This 5%-20% error is likely to be enough for practical use, since practical algorithms for k-MEANS are approximation algorithms anyway. Our coresets generally outperform both the uniform sampling and imputation baselines on almost every coreset sample size, and the advantage is more significant when the coreset size is relatively small. Moreover, our coresets have a much lower variance.

4.2 Speedup of Lloyd's-style Heuristic

Coresets often help to speed up existing approximation algorithms. Before our work, the only algorithm for k-MEANS with provable guarantees for multiple missing values was [EFG+21]. Unfortunately, [EFG+21] is not practical even when combined with coresets, since it contains several



(a)

(b)

(c)

(d)

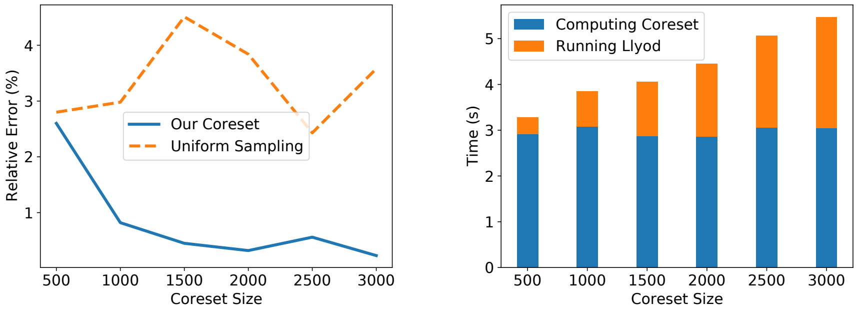
Figure 3: Standard deviation for the size-error evaluation.

enumeration procedures that require $\Theta(\exp(\text{poly}(\epsilon^{-1}jk)))$ time. We consider a variant of Lloyd's heuristic [Llo82] that is adapted to the missing-value setting, and we evaluate its speedup with coresets. The algorithm is essentially the same as the original Lloyd's algorithm, except that the distance as well as the optimal 1-mean for a cluster (which can be computed optimally in $O(d|P|)$ for a cluster P [EFG$^+$21]), is computed differently. We show that our coreset can significantly accelerate this algorithm. In particular, we run the modified Lloyd's heuristic directly on the original dataset, and take its running time and objective value as the comparison reference. Then we run this modified Lloyd's heuristic again, but on top of our coreset and the uniform sampling baseline respectively, and we compare both the speedup and the relative error[3] against the reference. The experiments are run on the Russian housing data set where the number of iterations of the modified Lloyd's is set to $T = 5000$ and the number of clusters is set to a small value $k = 3$ so as the heuristic is likely to find a local minimum faster. Again, to obtain a stable result, we run the experiments for 40 times with independent random bits and report the average relative errors and running time.

Results. The relative error with respect to varying coreset sizes can be found in Figure 4a. We can see that the relative error of Lloyd's algorithm running on our coreset is consistently low, while the uniform sampling baseline has several times higher error and the error does not seem to improve even when improving the size. We note that relative errors for both our coreset and uniform sampling are significantly lower than that we observe from the empirical error in Figure 2a. In fact, they are not necessarily comparable since the empirical error in Figure 2a is always evaluated on a same center, while what we compare in Figure 4a is the center sets found by the modified Lloyd's running on different data sets. This also helps to explain why improving the size of uniform sampling may not result in a better solution, since as shown in Figure 2a, uniform sampling has a large empirical error (around 50%), so a good solution for the uniform sample may not be a good solution for the original data set.

The running time of the modified Lloyd's on top of our coresets can be found in Figure 4b, and the running time of Lloyd's on the original dataset is 22.9s (which is not drawn on the figure). To make a fair comparison, we also take the coreset construction time into account. Note that coreset size is not a dominating factor in the running time of coreset construction, since the majority of time is

[3] For $x \in \mathbb{R}_+$, the relative error of x against a reference $x^\star > 0$ is defined as $\frac{|x - x^\star|}{x^\star}$.



(a)

(b)

Figure 4: Relative error and running time evaluation for the Lloyd's heuristic on the coreset, with respect to varying coreset sizes. The left figure demonstrates the relative error, and the right figure shows the running time of constructing our coreset, and the time for the modified Lloyd's heuristic running on top of our coreset.

spent on computing the importance scores and the coreset size only affects the number of samples. A coreset of size only 1000 can achieve $< 1\%$ error, and the running time of constructing the coreset and applying Lloyd's on top of it are 3s and 0.8s, respectively, which offers more than 5 times of speedup. We remark that our experiments only demonstrate the speedup in a single-machine scenario, and the speedup will increase in the parallel or distributed setting.

5 Conclusion

Our coreset construction builds upon the sensitivity-sampling method (cf. [FL11]). However, a central technical challenge is that the standard method to compute the sensitivity scores breaks, because distances between points with missing values do not satisfy the triangle inequality. We overcome this using another known method, of [VX12a], that requires a coreset for k-CENTER. Our main innovation is a near-linear time algorithm that computes an $O(1)$-approximate k-CENTER coreset for points with missing values. To this end, we need the following key steps, which constitute our main technical contribution.

- We reduce the k-CENTER coreset construction with missing values, to the construction of traditional k-CENTER coresets (i.e., without missing values) on a series of instances. These instances are built by restricting data points with missing values to a carefully-chosen collection of subspaces. The guarantee needed from this collection is a certain combinatorial structure, and we indeed prove it exists.
- The method of Varadarajan and Xiao executes the k-CENTER coreset algorithm many times, and overall takes quadratic time. To improve the running time, we design an efficient dynamic algorithm for the well-known Gonzales' algorithm (which computes an $O(1)$-approximate k-CENTER coreset). The main idea in this dynamic algorithm is to project the data points onto (data-oblivious) random 1D lines, and build on each line a dynamic data structure that supports furthest-neighbor queries (in 1D).

Finally, we implemented our algorithm and the experiments indicate that our algorithm is efficient and accurate enough to be potentially applicable in practice.

Future directions. As an immediate follow-up, one could try to improve our coreset size, e.g., removing the dependence in $\log n$. Our input can be viewed as axis-parallel affine-subspaces. Hence, another an interesting direction is to obtain coresets for the more general setting where the input consists of general affine-subspaces.

Potential negative societal impacts. Our paper focuses on computational issues (improving time and space) of known clustering tasks. Clustering methods in general have potential issues with fairness and privacy, which applies also to our work, but our research is not expected to introduce new negative societal impact beyond what is already known.

Acknowledgments and Disclosure of Funding

The majority of this work was done when Shaofeng Jiang was at Aalto University. This work is partially supported by ONR Award N00014-18-1-2364, by the Israel Science Foundation grant #1086/18, by a Weizmann-UK Making Connections Grant, and by a Minerva Foundation grant.

References

[All01] Paul D Allison. *Missing data*. Sage publications, 2001.

[AP02] Pankaj K. Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA*, pages 1027–1035. SIAM, 2007.

[BBH$^+$20] Daniel Baker, Vladimir Braverman, Lingxiao Huang, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in graphs of bounded treewidth. In *International Conference on Machine Learning*, pages 569–579. PMLR, 2020.

[BEL13] Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In *NIPS*, pages 1995–2003, 2013.

[BFL$^+$17] Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F. Yang. Clustering high dimensional dynamic data streams. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 576–585. PMLR, 2017.

[BJKW19] Vladimir Braverman, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for ordered weighted clustering. In *International Conference on Machine Learning*, pages 744–753. PMLR, 2019.

[BJKW21] Vladimir Braverman, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in excluded-minor graphs and beyond. In *SODA*, pages 2679–2696. SIAM, 2021.

[BLK18] Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-means clustering via lightweight coresets. In *KDD*, pages 1119–1127. ACM, 2018.

[CASS21] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. *STOC*, 2021.

[CCB16] Jocelyn T. Chi, Eric C. Chi, and Richard G. Baraniuk. k-POD: A method for k-means clustering of missing data. *The American Statistician*, 70(1):91–99, 2016. doi:10.1080/00031305.2015.1086685.

[Cha09] Timothy M. Chan. Dynamic coresets. *Discret. Comput. Geom.*, 42(3):469–488, 2009.

[DGR21] Ran Duan, Yong Gu, and Hanlin Ren. Approximate distance oracles subject to multiple vertex failures. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2497–2516. SIAM, 2021.

[DK11] Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 169–178, 2011.

[EFG$^+$21] Eduard Eiben, Fedor V. Fomin, Petr A. Golovach, William Lochet, Fahad Panolan, and Kirill Simonov. EPTAS for k-means clustering of affine subspaces. In *SODA*, pages 2649–2659. SIAM, 2021.

[Fel20] Dan Feldman. Introduction to core-sets: an updated survey. *arXiv preprint arXiv:2011.09384*, 2020.

[FKW19] Dan Feldman, Zahi Kfir, and Xuan Wu. Coresets for Gaussian mixture models of any shape. *arXiv preprint arXiv:1906.04895*, 2019.

[FL11] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *STOC*, pages 569–578. ACM, 2011. https://arxiv.org/abs/1106.1379.

[FRS19] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local search yields a PTAS for k-means in doubling metrics. *SIAM J. Comput.*, 48(2):452–480, 2019.

[FS05] Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *STOC*, pages 209–217. ACM, 2005.

[FSS20] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM J. Comput.*, 49(3):601–657, 2020.

[GLS08] Jie Gao, Michael Langberg, and Leonard J. Schulman. Analysis of incomplete data and an intrinsic-dimension Helly theorem. *Discret. Comput. Geom.*, 40(4):537–560, 2008.

[GLS10] Jie Gao, Michael Langberg, and Leonard J. Schulman. Clustering lines in high-dimensional space: Classification of incomplete data. *ACM Trans. Algorithms*, 7(1):8:1–8:26, 2010.

[Gon85] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[HB01] Richard J Hathaway and James C Bezdek. Fuzzy c-means clustering of incomplete data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(5):735–744, 2001.

[HC10] Ludmila Himmelspach and Stefan Conrad. Clustering approaches for data with missing values: Comparison and evaluation. In *ICDIM*, pages 19–28. IEEE, 2010.

[HJLW18] Lingxiao Huang, Shaofeng H-C Jiang, Jian Li, and Xuan Wu. Epsilon-coresets for clustering (with outliers) in doubling metrics. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 814–825. IEEE, 2018.

[HJV19] Lingxiao Huang, Shaofeng Jiang, and Nisheeth Vishnoi. Coresets for clustering with fairness constraints. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[HK20] Monika Henzinger and Sagar Kale. Fully-dynamic coresets. In *ESA*, volume 173 of *LIPIcs*, pages 57:1–57:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[HM04] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, pages 291–300. ACM, 2004. https://arxiv.org/abs/1810.12826.

[HV20] Lingxiao Huang and Nisheeth K Vishnoi. Coresets for clustering in Euclidean spaces: Importance sampling is nearly optimal. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1416–1429, 2020.

[Ind03] Piotr Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *SODA*, pages 539–545. ACM/SIAM, 2003.

[JTMF20] Ibrahim Jubran, Murad Tukan, Alaa Maalouf, and Dan Feldman. Sets clustering. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4994–5005. PMLR, 13–18 Jul 2020.

[KDD09] Kddcup. https://kdd.org/kdd-cup/view/kdd-cup-2009/Data, 2009.

[KP21] CS Karthik and Merav Parter. Deterministic replacement path covering. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 704–723. SIAM, 2021.

[LFKF17] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training Gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

[Llo82] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

[LR19] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[LS10] 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.

[LS13] Euiwoong Lee and Leonard J. Schulman. Clustering affine subspaces: Hardness and algorithms. In *SODA*, pages 810–827. SIAM, 2013.

[MF19] Yair Marom and Dan Feldman. k-means clustering of lines for big data. In *NeurIPS*, pages 12797–12806, 2019.

[MOB$^+$20] Ben Mussay, Margarita Osadchy, Vladimir Braverman, Samson Zhou, and Dan Feldman. Data-independent neural pruning via coresets. In *ICLR*. OpenReview.net, 2020.

[Phi17] Jeff M Phillips. Coresets and sketches. In *Handbook of discrete and computational geometry*, pages 1269–1288. Chapman and Hall/CRC, 2017.

[RPS15] Sashank J. Reddi, Barnabás Póczos, and Alexander J. Smola. Communication efficient coresets for empirical loss minimization. In *UAI*, pages 752–761. AUAI Press, 2015.

[Rus17] Sberbank russian housing market. https://www.kaggle.com/c/sberbank-russian-housing-market/data, 2017.

[Sam21] Salah Sammari. Vertical farming. https://www.kaggle.com/midouazerty/work-for-parmavir/version/1, 2021.

[SW18] Christian Sohler and David P Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 802–813. IEEE, 2018.

[vH14] Ramon van Handel. Probability in high dimension. Technical report, PRINCETON UNIV NJ, 2014.

[VX12a] Kasturi Varadarajan and Xin Xiao. A near-linear algorithm for projective clustering integer points. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1329–1342. SIAM, 2012.

[VX12b] Kasturi R. Varadarajan and Xin Xiao. On the sensitivity of shape fitting problems. In *FSTTCS*, volume 18 of *LIPIcs*, pages 486–497. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012.

[Wag04] Kiri Wagstaff. Clustering with missing values: No imputation required. In *Classification, clustering, and data mining applications*, pages 649–658. Springer, 2004.

[WLH$^+$19] Siwei Wang, Miaomiao Li, Ning Hu, En Zhu, Jingtao Hu, Xinwang Liu, and Jianping Yin. K-Means clustering with incomplete data. *IEEE Access*, 7:69162–69171, 2019. doi:10.1109/ACCESS.2019.2910287.

Appendices

A Full Version of Section 3

This section is the full version of Section 3. We restate Theorem 3.1 in Theorem A.1 which is the main theorem. Lemma 3.2, Lemma 3.3, and Lemma 3.4 correspond to Lemma A.6, Lemma A.7, and Lemma A.8, respectively. Lemma 3.5 is not used explicitly in the full version but we present it in Section 3 for the sake of presentation, sicne it captures the main idea of A.12 as well as the a central part of the proof for Lemma A.13.

In addition, we present all the missing details of the importance sampling framework (Lemma A.2), the reduction of Varadarajan and Xiao [VX12a] (Lemma A.5), and the Gonzales's algorithm as well as its dynamic implementation (Lemma A.13).

Theorem A.1. *There is an algorithm that, given as input a data set $X \subset \mathbb{R}^d_?$ of size $n = |X|$ consisting of j-points and parameters $k, z \geq 1$ and $0 < \epsilon < 1/2$, constructs with constant probability an ϵ-coreset of size $m = \tilde{O}\left(z^z \cdot \frac{(j+k)^{j+k+1}}{j^j k^{k-z-2}} \cdot \frac{(d \log n)^{\frac{z+2}{2}}}{\epsilon^2}\right)$ for (k, z)-*Clustering *of X, and runs in time $\tilde{O}\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-2}} \cdot nd + m\right)$.*

We remark that $\frac{(j+k)^{j+k}}{j^j k^k} = (jk)^{O(\min(j,k))}$. To see this, assume $j \geq k$ w.l.o.g., so $\frac{(j+k)^j}{j^j} = (1 + \frac{k}{j})^j \leq e^{\frac{k}{j} \cdot j} = e^k$ and $\frac{(j+k)^k}{k^k} \leq (j+k)^k$.

Theorem A.1 is the main theorem of this paper, and we present the proof in this section. As mentioned in Section 1, the coreset is constructed via importance sampling, by following three major steps.

1. For each data point $x \in X$, compute an importance score $\sigma_x \geq 0$.
2. Draw N (to be determined later) independent samples from X, such that $x \in X$ is sampled with probability $p_x \propto \sigma_x$.
3. Denote the sampled (multi)set as S, and for each $x \in S$ define its weight $w(x) := \frac{1}{p_x N}$. Report the weighted set S as the coreset.

The importance score σ_x is usually defined as (an approximation) of the *sensitivity* of x, denoted

$$\sigma^\star_x := \sup_{C \subset \mathbb{R}^d, |C|=k} \frac{\operatorname{dist}^z(x, C)}{\operatorname{cost}_z(X, C)}, \tag{2}$$

which measures the maximum possible relative contribution of x to the objective function.

Usually, there are two main challenges with this approach. First, the sensitivity (2) is not efficiently computable because it requires to optimize over all k-subsets $C \subset \mathbb{R}^d$. Second, one has to determine the number of samples N (essentially the coreset size) based on a probabilistic analysis of the event that S is a coreset. Prior work on coresets has studied these issues extensively and developed a general framework, and we shall use the variant stated in Theorem A.2 below. This framework only needs an approximation to the sensitivities $\{\sigma^\star_x\}_{x \in X}$, more precisely it requires overestimates $\sigma_x \geq \sigma^\star_x$ whose sum $\sum_{x \in X} \sigma_x$ is bounded. Moreover, it relates the number of samples N to a quantity called the *weighted shattering dimension* $\operatorname{sdim}_{\max}$, which roughly speaking measures the complexity of a space (set of points) by the number of distinct ways that metric balls can intersect it. The definition below has an extra complication of a point weight v, which originates from the weight in the importance sampling procedure, and thus we need a uniform upper bound, denoted $\operatorname{sdim}_{\max}$, over all possible weights.[4]

Definition A.1 (Shattering dimension)**.** Given a *weight* function $v : \mathbb{R}^d_? \to \mathbb{R}_+$, let $\operatorname{sdim}_v(\mathbb{R}^d_?)$ be the smallest integer t such that

$$\forall H \subset \mathbb{R}^d_?, |H| \geq 2 \qquad \left|\left\{B^H_v(c, r) : c \in \mathbb{R}^d, r \geq 0\right\}\right| \leq |H|^t,$$

where $B^H_v(c, r) := \{x \in H : v(x) \cdot \operatorname{dist}(x, c) \leq r\}$. Let $\operatorname{sdim}_{\max}(\mathbb{R}^d_?) := \sup_{v : \mathbb{R}^d_? \to \mathbb{R}_+} \operatorname{sdim}_v(\mathbb{R}^d_?)$.

[4] In principle, this uniform upper bound is not necessary, and an upper bound for weights corresponding to the importance score suffices, but a uniform upper bound turns out to be technically easier to deal with.

Strictly speaking, Theorem A.2 has been proposed and proved only for metric spaces, but the proof is applicable also in our setting (where dist need not satisfy the triangle inequality), because it only concerns the *binary* relation between data points and center points (without an indirect use of a third point, e.g., by triangle inequality.)

Theorem A.2 ([FSS20][5])**.** *Let $X \subset \mathbb{R}^d_?$ be a data set, and let $k, z \geq 1$. Consider the importance sampling procedure with importance scores that satisfy $\sigma_x \geq \sigma_x^\star$ for all $x \in X$, and with a sufficiently large number of samples*

$$N = \tilde{O}\left(\epsilon^{-2} k z^z \operatorname{sdim}_{\max}(\mathbb{R}^d_?) \sum_{x \in X} \sigma_x\right).$$

*Then with constant probability it reports an ϵ-coreset for (k, z)-*CLUSTERING.

Proof of Theorem A.1. Because of Theorem A.2, it suffices to bound $\operatorname{sdim}_{\max}(\mathbb{R}^d_?)$, and to provide an efficient algorithm to estimate σ_x whose sum is bounded. These two components are provided in Lemma A.3 and Lemma A.4 stated below (their proofs appear in Sections A.1 and A.2), Plugging these two lemmas into Theorem A.2, the main theorem follows. We provide an outline for the complete algorithm in Algorithm 1. □

Algorithm 1 Main algorithm

```
run Algorithm 3 to obtain $\sigma_x$ for $x \in X$
draw $N := \tilde{O}\left(z^z \cdot \frac{(j+k)^{j+k+1}}{j^j k^{k-z-2}} \cdot \frac{(d \log n)^{\frac{z+2}{2}}}{\epsilon^2}\right)$ independent samples $S$ from $X$, where $x \in X$ is
    sampled with probability $p_x \propto \sigma_x$
for $x \in S$, define weight $w(x) \leftarrow \frac{1}{p_x N}$
return weighted set $S$ with weight $w$ as the coreset
```

Lemma A.3 (Shattering dimension bound)**.** $\operatorname{sdim}_{\max}(\mathbb{R}^d_?) = O(d)$.

Lemma A.4. *There is an algorithm that, given a data set $X \subset \mathbb{R}^d_?$ of n j-points, for (k, z)-*CLUSTERING *computes importance scores $\{\sigma_x\}_{x \in X}$ such that with constant probability,*

- *$\sigma_x \geq \sigma_x^\star$ for all $x \in X$; and*
- *$\sum_{x \in X} \sigma_x \leq O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-z-1}} \cdot \sqrt{d^z \cdot \log^{z+2} n}\right)$,*

and its running time is $\tilde{O}\left(\frac{(j+k)^{j+k+2}}{j^j k^{k-2}} \cdot nd\right)$.

A.1 Proof of Lemma A.3: Shattering Dimension of $\mathbb{R}^d_?$

We now prove Lemma A.3, which asserts that $\operatorname{sdim}_{\max}(\mathbb{R}^d_?) = O(d)$. We remark that the shattering dimension bound for $\mathbb{R}^d$ without missing values has been proved in [FL11, Lemma 16.1] and our proof is actually an extension of it.

Proof of Lemma A.3. Let us verify Definition A.1. Consider $H \subset \mathbb{R}^d_?$ and a weight function $v : \mathbb{R}^d_? \to \mathbb{R}_+$. Recall that given $c \in \mathbb{R}^d$ and $r \geq 0$, we have $B_v^H(c, r) = \{h \in H : v(h) \cdot \mathrm{dist}(h, c) \leq r\}$ and $\mathrm{dist}(h, c)^2 = \sum_{i \in I_h} (h_i - c_i)^2$ for $h \in H$. We need to show that

$$\left|\{B_v^H(c, r) : c \in \mathbb{R}^d, r \geq 0\}\right| \leq |H|^{O(d)}. \tag{3}$$

Observe that

$$h \in B_v^H(c, r) \iff v(h) \cdot \mathrm{dist}(h, c) \leq r \iff -r^2 + \sum_{i \in I_h} (v^2(h) h_i^2 + v^2(h) c_i^2 - 2v^2(h) h_i c_i) \leq 0.$$

[5] Our theorem statement is based on [FSS20, Theorem 31], adapted to our context. One difference is that their theorem is about VC-dimension, but it is also applicable for shattering dimension. Another difference is that we use a more direct terminology that is specialized to metric balls in $\mathbb{R}^d_?$ instead of a general range space.

Next, we write this inequality in an alternative way, that separates terms depending h from those depending on c and r, more precisely as an inner-product $\langle f(h), g(c,r)\rangle \leq 0$ for vectors $f(h), g(c,r) \in \mathbb{R}^{3d+1}$. Now consider $f : H \to \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$ and $g : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$ such that $f(h) = (p, q, t, -1)$, where $p, q, t \in \mathbb{R}^d$ and for $i \in [d]$

$$p_i = \begin{cases} v^2(h) \cdot h_i^2 & \text{if } i \in I_h \\ 0 & \text{otherwise} \end{cases} \qquad q_i = \begin{cases} v^2(h) & \text{if } i \in I_h \\ 0 & \text{otherwise} \end{cases} \qquad t_i = \begin{cases} -2v^2(h) \cdot h_i & \text{if } i \in I_h \\ 0 & \text{otherwise} \end{cases}$$

and $g(c,r) = (y, z, w, r^2)$, where $y, z, w \in \mathbb{R}^d$, $y_i = 1, z_i = c_i^2, w_i = c_i$ for $i \in [d]$. Then we have

$$h \in B_v^H(c,r) \iff \langle f(h), g(c,r)\rangle \leq 0.$$

For a vector $t \in \mathbb{R}^{3d+1}$, let $\mathrm{proj}_-^H(t) := \{h \in H : \langle f(h), t\rangle \leq 0\}$ be the subset of H that has nonpositive inner-product with t (it can be viewed also as projection or a halfspace). Therefore, by (3), we have

$$\left|\{B_v^H(c,r) : c \in \mathbb{R}^d, r \geq 0\}\right| = \left|\{\mathrm{proj}_-^H(g(c,r)) : c \in \mathbb{R}^d, r \geq 0\}\right| \leq \left|\{\mathrm{proj}_-^H(t) : t \in \mathbb{R}^{3d+1}\}\right|.$$

We observe that

$$\left|\{\mathrm{proj}_-^H(t) : t \in \mathbb{R}^{3d+1}\}\right| \leq |H|^{O(d)},$$

since this may be related to the shattering dimension of halfspaces in $\mathbb{R}^{3d+1}$, which is $O(d)$ and is a well-known fact in the PAC learning theory (cf. [vH14, Chapter 7.2]). This concludes the proof of Lemma A.3. □

A.2 Proof of Lemma A.4: Estimating Sensitivity Efficiently

We use a technique introduced by Varadarajan and Xiao [VX12a] that reduces the sensitivity-estimation problem to the problem of constructing a coreset for k-Center clustering. This coreset concept is defined as follows.

Definition A.2. An *α-coreset for k-*Center of a data set $X \subset \mathbb{R}^d_?$ is a subset $Y \subseteq X$ such that

$$\forall C \subset \mathbb{R}^d, |C| = k, \qquad \max_{x \in X} \mathrm{dist}(x, C) \leq \alpha \cdot \max_{y \in Y} \mathrm{dist}(y, C).$$

Note that the error parameter α represents a multiplicative factor, which is slightly different from that of ϵ in ϵ-coreset for (k,z)-Clustering, and roughly corresponds to $\alpha = 1 + \epsilon$. The reasoning is that $\max_{y \in Y} \mathrm{dist}(y, C)$ for $Y \subseteq X$ is always no more than $\max_{x \in X} \mathrm{dist}(x, C)$, and therefore we only need to measure the contraction-side error.

The reduction in Lemma A.5 was presented in [VX12a], and we restate its algorithmic steps in Algorithm 2. This needs access to some Algorithm $\mathcal{A}$ that constructs an α-coreset for k-Center on a point set $X \subset \mathbb{R}^d_?$. Each iteration i calls Algorithm $\mathcal{A}$ to construct a k-Center coreset for the current point set X (which is initially the entire data set), assign sensitivity estimates $O(\alpha^z/i)$ to every coreset point, and then remove these coreset points from X. These iterations are repeated until X is empty.

Algorithm 2 Sensitivity estimation from [VX12a, Lemma 3.1] for data set $X \subset \mathbb{R}^d_?$

```
Require: algorithm $\mathcal{A}$ that constructs $\alpha$-coreset for $k$-Center
$i \leftarrow 1$
while $X \neq \emptyset$ do
    $P \leftarrow \mathcal{A}(X)$
    for $x \in P$ do
        $\sigma_x \leftarrow O(\alpha^z/i)$
    end for
    $X \leftarrow X \setminus P$
    $i \leftarrow i + 1$
end while
```

Lemma A.5 ([VX12a, Lemma 3.1])**.** *Suppose algorithm $\mathcal{A}$ constructs an α-coreset of size $T = T(\alpha, d, j, k)$ for k-*CENTER *an input $X \subset \mathbb{R}^d_?$. Then Algorithm 2 (which makes calls to this Algorithm $\mathcal{A}$) computes sensitivities $\{\sigma_x\}$ for (k, z)-*CLUSTERING *satisfying that $\sigma_x \geq \sigma_x^\star$ for all $x \in X$, and $\sum_{x \in X} \sigma_x \leq \alpha^z \cdot T \log |X|$.*

However, there are two outstanding technical challenges. First, there is no known construction of a small k-CENTER coreset for our clustering with missing values setting. Moreover, as can be seen from Algorithm 2, this reduction executes the k-CENTER coreset construction $\frac{|X|}{T}$ times (where T is the size of the coreset as in Lemma A.5), and when using a naive implementation of the k-CENTER coreset construction, which naturally requires $\Omega(|X|)$ time, results overall in quadratic time, which is not very efficient.

First, to deal with question marks, we employ a certain family $\mathcal{I}$ of subset of coordinates (so each $I \in \mathcal{I}$ is a subset of $[d]$), and we *restrict* the data set X on each $I \in \mathcal{I}$. Each restricted data set (restricted on some I) may be viewed as a data set in $\mathbb{R}^I$, without any question marks. We show that the union of k-CENTER coresets on all restricted data sets with respect all to $I \in \mathcal{I}$, forms a valid k-CENTER coreset for X (which has question marks), provided that the family $\mathcal{I}$ has a certain combinatorial property. Naturally, the size of this coreset for X depends on an upper bound on $|\mathcal{I}|$.

Second, since the choice of family $\mathcal{I}$ is oblivious to the data set, it suffices to design an efficient algorithm for k-CENTER coreset for any restricted data set. We observe that the efficiency bottleneck in Algorithm 2 is the repeated invocation of Algorithm $\mathcal{A}$ to construct a coreset, even though its input changes only a little between consecutive invocations. Hence, we design a dynamic algorithm, that maintains a k-CENTER coreset on the restricted data sets under point updates. Our algorithm may be viewed as a variant of Gonzalez's algorithm [Gon85], and we maintain it efficiently by a random projection idea that was used e.g. in [Ind03]. In particular, we "project" the data points onto several one-dimensional lines in $\mathbb{R}^d$, and we maintain an interval data structure (that is based on balanced trees) to dynamically maintain the result of our variant of Gonzalez's algorithm. We summarize the dynamic algorithm in the following lemma.

Lemma A.6. *There is a randomized dynamic algorithm with the following guarantees. The input is a dynamic set $X \subset \mathbb{R}^d_?$ of j-points, such that X undergoes q adaptive updates (point insertions and deletions) and the points ever added are fixed in advance (non-adaptively). The algorithm maintains in time $\tilde{O}\left(\frac{(j+k)^{j+k+1}}{j^j k^k} \cdot (j + k \log q)(d + k^2 \log q)\right)$ per update, a subset $Y \subseteq X$ of size $|Y| \leq O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-1}} \cdot \log d\right)$ such that with constant probability, Y is an $O(k\sqrt{d \log q})$-coreset for k-*CENTER *on X after every update.*

The proof of the lemma can be found in Section A.3, and here we proceed to the proof of Lemma A.4.

Proof of Lemma A.4. We plug in the dynamic algorithm in Lemma A.6 as $\mathcal{A}$ in Lemma A.5. Specifically, line 3 and 7 of Algorithm 2 are replaced by the corresponding query and update procedure. The detailed description can be found in Algorithm 3.

Algorithm 3 Efficient importance score estimation

```
let $\mathcal{D}$ be the dynamic data structure defined in Algorithm 4, and call $\mathcal{D}$.INIT
$\forall x \in X$, insert $x$ to $\mathcal{D}$
$i \leftarrow 1$
while $X \neq \emptyset$ do
    $P \leftarrow \mathcal{D}$.GET-CORESET
    for $x \in P$ do
        $\sigma_x \leftarrow O(\alpha^z / i)$
    end for
    $\forall x \in P$, remove $x$ from $\mathcal{D}$
    $i \leftarrow i + 1$
end while
return $(\sigma_x : x \in X)$
```

Since $|X| = n$, and each point is inserted and deleted for exactly once, algorithm 2 needs $q = O(n)$ insertions and deletions of points. Moreover, the set of points ever added is just X which is fixed. Thus, α is replaced by $O(k\sqrt{d \log n})$ and T is replaced by $O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-1}} \cdot \log d\right)$. Therefore, for (k, z)-CLUSTERING, this computes σ_x for $x \in X$ such that $\sigma_x \geq \sigma_x^\star$, and that

$$\sum_{x\in X} \sigma_x \leq \alpha^z \cdot T \cdot \log n = O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-z-1}} \cdot \sqrt{d^z \cdot \log^{z+2} n}\right).$$

The total running time is bounded by $\tilde{O}\left(\frac{(j+k)^{j+k+2}}{j^j k^{k-2}} \cdot nd\right)$ for implementing $O(n)$ updates. □

A.3 Proof of Lemma A.6: Dynamic $O(1)$-Coresets for k-Center Clustering

As mentioned, the high level idea is to identify a collection $\mathcal{I}$ of subsets of coordinates (so each $I \in \mathcal{I}$ satisfies $I \subseteq [d]$), construct an α-coreset (a will be determined is the later context) Y_i for k-CENTER on the data set X with coordinates *restricted* on each $I_i \in \mathcal{I}$, and then the union $\bigcup_i Y_i$ would be the overall $\alpha\sqrt{d}$-coreset for k-CENTER on X. The exact definition of restricted data set goes as follows.

Definition A.3. For a point $p \in \mathbb{R}^d_?$ and a subset $I \subseteq I_p$, define $p_{|I} \in \mathbb{R}^I$ in the obvious way, by selecting the coordinates $\{p_i\}_{i\in I}$. Define the *I-restricted data set* to be $X_{|I} := \{p_{|I} : p \in X, I \subseteq I_p\}$. Since each vector in $X_{|I}$ arises from a specific vector in X, a subset $Y \subseteq X_{|I}$ corresponds to a specific subset of X, and we shall denote this subset by Y^{-1}.

We observe that the metric space on the restricted data set becomes a usual metric space, i.e. it satisfies the triangle inequality, and can be realized as a point set in $\mathbb{R}^I$ which does not contain question marks. Therefore, this reduces our goal to constructing k-CENTER coresets for this usual data set. However, the size of the coreset yielded from this approach would depend on the size of the family $\mathcal{I}$. Hence, a key step is to identify a small set $\mathcal{I}$ such that the union of the coreset restricted on $\mathcal{I}$ is an accurate coreset. To this end, we consider the so-called (j, k, d)-family of coordinates as in Definition A.4. This family itself is purely combinatorial, but we will show in Lemma A.7 that such a family actually suffices for the accuracy of the coreset, and we show in Lemma A.8 the existence of a small family.

Definition A.4. A family of sets $\mathcal{I} \subset 2^{[d]}$ is called a (j, k, d)-family if for any $J, K \subset [d]$, $J \cap K = \emptyset$, $|J| = j$, $|K| = k$, there exists an $I \in \mathcal{I}$ such that $I \cap J = \emptyset$ and $K \subset I$.

Lemma A.7. *Suppose $\mathcal{I}$ is a (j, k, d)-family Let $X \subseteq \mathbb{R}^d_?$ be a set of j-points, and for every $I \in \mathcal{I}$, let Y_I be an α-coreset for k-CENTER on $X_{|I}$. Then $\cup_{I\in\mathcal{I}} Y_I^{-1}$ is an $\alpha\sqrt{d}$-coreset for k-Center on X.*

Proof. It suffices to show that for any center set $C = \{c^1, \dots, c^k\} \subseteq \mathbb{R}^d$ with k points and $x \in X$, if $\mathrm{dist}(x, C) \geq r$ for some $r \geq 0$, then we can find a coreset point $y \in \cup_{I\in\mathcal{I}} Y_I^{-1}$ such that $\mathrm{dist}(y, C) \geq \frac{r}{\alpha\sqrt{d}}$.

For $i \in [k]$, let $t_i \in \arg\max_{t\in I_x} |x_t - c^i_t|$, i.e., t_i is the index of coordinate that contributes the most in distance $\mathrm{dist}(x, c^i)$, so $|x_{t_i} - c^i_{t_i}| \geq \frac{r}{\sqrt{d}}$. Let K be any k-subset such that $K \subseteq I_x$ and $\{t_1, \dots, t_k\} \subseteq K$. Since $\mathcal{I}$ is a (j, k, d)-family and $|I_x| \geq d - j$, by definition, there exists an $I \subseteq \mathcal{I}$ such that $K \subseteq I \subseteq I_x$. We note that

$$\mathrm{dist}(x_{|I}, C_{|I}) = \mathrm{dist}_I(x, C) = \min_{i\in[k]} \mathrm{dist}_I(x, c^i) \geq \min_{i\in[k]} \mathrm{dist}_K(x, c^i) \geq \min_{i\in[k]} |x_{t_i} - c^i_{t_i}| \geq \frac{r}{\sqrt{d}}.$$

Since $I \subseteq I_x$, we know that $x_{|I} \in X_{|I}$. As Y_I is an α-coreset for $X_{|I}$, we know that there exists $y \in Y_I^{-1}$ such that

$$\mathrm{dist}(y, C) \geq \mathrm{dist}_I(y, C) = \mathrm{dist}(y_{|I}, C_{|I}) \geq \frac{\mathrm{dist}(x_{|I}, C_{|I})}{\alpha} \geq \frac{r}{\alpha\sqrt{d}}.$$

□

Next, we show the existence of a small (j, k, d)-family. We remark that this combinatorial structure has been employed in designing fault-tolerant data structures and algorithms (cf. [DK11, DGR21, KP21]). Similar bounds were obtained in their different contexts and languages, and here we provide a proof for completeness.

Lemma A.8. *There is a (j,k,d)-family $\mathcal{I}$ of size $O\left(\frac{(j+k)^{j+k+1}}{j^j k^k}\log d\right)$. Moreover, there is a randomized algorithm that constructs $\mathcal{I}$ in time $O(d\cdot|\mathcal{I}|)$ with probability at least $1-\frac{1}{d^{j+k}}$.*

Proof. Set $t=\frac{(j+k)^{j+k+1}}{j^j k^k}\cdot 2\log d$. We add t random sets into $\mathcal{I}$ where each random set is generated by independently including each element of $[d]$ with probability $\frac{k}{j+k}$. For a set $J\subseteq[d], |J|=j$ and a set $K\subseteq[d], |K|=k$ such that $J\cap K=\emptyset$, the probability that a random set generated in the above way contains K but avoids J, is

$$\left(\frac{j}{j+k}\right)^j\cdot\left(\frac{k}{j+k}\right)^k.$$

Since there are at most d^{j+k} tuples of such J and K, by union bound and the choice of t, the probability that $\mathcal{I}$ is a (j,k,d)-family is at least

$$1-d^{j+k}\left(1-(\frac{j}{j+k})^j\cdot(\frac{k}{j+k})^k\right)^t\geq 1-\frac{1}{d^{j+k}}$$

□

Gonzalez's algorithm yields k-Center coreset for restricted data set. Finally, the k-Center coreset for the restricted data set on each $I\in\mathcal{I}$ would be constructed using an approximate version of Gonzalez's algorithm [Gon85]. We note that while Gonzalez's algorithm was originally designed as an approximation algorithm for k-Center, the approximate solution actually serves as a good coreset for k-Center (see Lemma A.9). The assumption that the input forms a metric space is crucial in Lemma A.9, and this is guaranteed since we run this variant of Gonzalez only on a restricted data set which satisfies the triangle inequality.

Lemma A.9 (Approximate Gonzalez)**.** *Let (M,d) be a metric space. Let $A\subset M$ be a set of n points and consider the following variant of Gonzalez's greedy algorithm. Set $B=\{b_0\}$ for an arbitrary $b_0\in A$. Repeat for k times, where each time we add a c-approximation of B's furthest point into B. Precisely, add $b_i\in A$ such that $c\cdot\operatorname{dist}(b_i,B)\geq\max_{a\in A}\operatorname{dist}(a,B)$ into B. Then B is a $(1+2c)$-coreset for k-Center on A.*

Proof. Fix a center set $C=\{c_1,\dots,c_k\}$ with k points and let $r:=\max_{b\in B}\operatorname{dist}(b,C)$. Then we have $\bigcup_{i=1}^k\operatorname{Ball}(c_i,r)$ covers B where $\operatorname{Ball}(x,r)=\{y:\operatorname{dist}(x,y)\leq r\}$ is the ball centered at x with radius r. It suffices to prove that $A\subseteq\bigcup_{i=1}^k\operatorname{Ball}(c_i,(2c+1)r)$.

Since k balls $B(c_1,r),\cdots,B(c_k,r)$ cover B and $|B|=k+1$, by pigeonhole principle, there exists $b_i,b_j\in B, i<j$ that are contained in a same ball $B(c_i,r)$. W.l.o.g., we assume $b_i,b_j\in B(c_1,r)$. Now fix $a\in A\setminus B$, since a has never been added into B, we have

$$\begin{aligned}\operatorname{dist}(a,B)&\leq\operatorname{dist}(a,\{b_1,\dots,b_{j-1}\})\\&\leq c\cdot\operatorname{dist}(b_j,\{b_1,\dots,b_{j-1}\})\\&\leq c\cdot\operatorname{dist}(b_i,b_j)\\&\leq c\cdot(\operatorname{dist}(b_i,c_1)+\operatorname{dist}(b_j,c_1))\\&\leq 2cr.\end{aligned}$$

Thus $A\subseteq\bigcup_{i=1}^{k+1}\operatorname{Ball}(b_i,2cr)\subseteq\bigcup_{i=1}^k\operatorname{Ball}(c_i,(2c+1)r)$. □

Dynamic implementation of Gonzalez's algorithm. To make this k-Center coreset construction dynamic, we adapt the random projection technique to Gonzalez's algorithm, so that it suffices to dynamically execute Gonzalez's algorithm on a set of one-dimensional lines in $\mathbb{R}^d$.

Random projection. We call a sample from the d-dimensional standard normal distribution $N(0, I_d)$ a d-dimensional *random vector* for simplicity. To implement (the variant of) Gonzalez's algorithm as in Lemma A.9 in the dynamic setting, we project the point set to several random vectors and use one dimensional data structure to construct k-CENTER coreset in each of the one dimensional projected data set.

Note that the key step in Gonzalez's algorithm is the furthest neighbor search, and we would show that our projection method eventually yields an $O(k\sqrt{\log n})$-approximation of the furthest neighbor with high probability. The following two facts about normal distribution are crucial in our argument, and Lemma A.12 is our main technical lemma.

Fact A.10. *Let $u \in \mathbb{R}^d$ and let $v \sim N(0, I_d)$ be a random vector, then $\langle u, v/|u|\rangle \sim N(0,1)$.*

Fact A.11. *Let $Z \sim N(0,1)$, then there exists some universal constant $c > 0$ such that $P[|Z| \leq \frac{1}{k}] \leq \frac{c}{k}$, and $P[|Z| \geq t] \leq e^{-c \cdot t^2}$ for any $t > 0$.*

Lemma A.12. *Let $X \subset \mathbb{R}^d, |X| = n$, $\delta > 0$ and integer $k \geq 1$. Let $\mathcal{V}$ be a collection of $t = O(k \log n + \log \delta^{-1})$ random vectors in $\mathbb{R}^d$. Then with probability $1 - \delta$, for every $C \subseteq X, |C| \leq k$ and every $x \in X$, there exists a vector $v \in \mathcal{V}$ such that (i) $|x \cdot v - c \cdot v| \geq \Omega(\frac{1}{k}) \cdot \|c - x\|_2$ for every $c \in C$ and (ii) $|a \cdot v - b \cdot v| \leq O(\sqrt{\log n}) \cdot \|a - b\|_2$ for every $a, b \in X$.*

Proof. Fix a subset $C \subseteq X, |C| \leq k$, a point x and a random vector v. For every $c \in C$, since $(c - x) \cdot v/\|c - x\|_2 \sim N(0,1)$, by Fact A.11, the probability that $|c \cdot v - x \cdot v| \geq \Omega(\frac{1}{k}) \cdot \|c - x\|_2$ is at least $1 - \frac{1}{4k}$. For every $a, b \in X$, since $(a - b) \cdot v/\|a - b\|_2 \sim N(0,1)$, by Fact A.11, the probability that $|a \cdot v - b \cdot v| \leq \sqrt{\log n}\|a - b\|_2$ is at most $\frac{1}{4n^2}$.

Since there are k choices of $c \in C$ and at most n^2 choices of $a, b \in X$, by union bound, with probability at least $1 - k \cdot \frac{1}{4k} - n^2 \cdot \frac{1}{4n^2} = \frac{1}{2}$, the following two events hold, (i) $|x \cdot v - c \cdot v| \geq \Omega(\frac{1}{k}) \cdot \|c - x\|_2$ for every $c \in C$ and (ii) $|a \cdot v - b \cdot v| \leq O(\sqrt{\log n}) \cdot \|a - b\|_2$ for every $a, b \in X$.

Now since $\mathcal{V}$ contains t random vectors, the probability that there exists one vector $v \in \mathcal{V}$ that satisfies (i) and (ii) is at least $1 - \frac{1}{2^t}$.

Finally, by union bound, since there are at most $(n+1)^{k+1}$ choices of $C \subseteq X, |C| = k$ and $x \in X$, the probability such that for every C and x, there exists $v \in \mathcal{V}$ such that (i) and (ii) happen is at least $1 - \frac{(n+1)^{k+1}}{2^t} \geq 1 - \delta$. □

In the next lemma, we present a dynamic algorithm that combines the random projection idea with a one-dimensional data structure. This combining with the (j, k, d)-family idea would immediately imply Lemma A.6.

Lemma A.13. *There is a dynamic algorithm that for every $P \subseteq \mathbb{R}^m$ subject to at most q adaptive point insertions and deletions where the set of points ever added is fixed in advance, and every $\delta > 0$, maintains set $Q \subseteq P$ with $|Q| \leq k + 1$ such that with probability at least $1 - \delta$, Q is an $O(k\sqrt{\log q})$-coreset for k-CENTER on P after every update, in time $O\big((k^2 \log q + m)(k \log q + \log \delta^{-1})\big)$ per update.*

Proof of Lemma A.6. We present our dynamic algorithm in Algorithm 4.

Analysis. Since we pick $\delta = \Theta\left(\frac{1}{|\mathcal{I}|}\right)$ for all $\mathcal{D}_I$'s, with constant probability all data structures $\mathcal{D}_I$'s succeed simultaneously. The running time follows immediately from Lemma A.8 and Lemma A.13. The coreset accuracy follows from Lemma A.7 and Lemma A.13 (noting that we need to suffer a $\sqrt{d}$ factor because of Lemma A.7). □

Proof of Lemma A.13. We assume there is a data structure $\mathcal{T}$ that maintains a set of real numbers and supports the following operations, all running in $O(\log n)$ time where n is the number of elements currently present in the structure.

- REMOVE(x): Remove an element x from the structure.
- ADD(x): Add an element x to the structure.

Algorithm 4 Dynamic k-CENTER coreset with missing values

```
procedure INIT
    let $\mathcal{I}$ be a $(j,k,d)$-family generated by sampling, as in Lemma A.8
                                ▷ $|\mathcal{I}| = O\left(\frac{(j+k)^{j+k+1}}{j^j k^k} \log d\right)$
    $\forall I \in \mathcal{I}$, initialize data structure $\mathcal{D}_I$ using Algorithm 5 (Lemma A.13) with failure probability
    $\delta := \Theta\left(\frac{1}{|\mathcal{I}|}\right)$, and initialize $Y_I = \emptyset$
end procedure
procedure UPDATE($x \in \mathbb{R}^d_?$)
    for $I \in \mathcal{I}$ do
        $\mathcal{D}_I$.UPDATE($x_{|I}$)
        $Y_I \leftarrow \mathcal{D}_I$.GET-CORESET($k$)
    end for                     ▷ we use UPDATE and Get-Coreset in Algorithm 5
end procedure
procedure GET-CORESET
    return $\bigcup_{I \in \mathcal{I}} Y_I^{-1}$                  ▷ as in Lemma A.7
end procedure
```

- UPPERBOUND(x): Return the largest element that is at most x.
- LOWERBOUND(x): Return the smallest element that is at least x.

Note that such $\mathcal{T}$ may be implemented by using a standard balanced binary tree.

Furthest point query. We also need FURTHEST(C) query, where $C \subset \mathbb{R}$ and it asks for an element x that has the largest distance to C (and it should return an arbitrary element if $C = \emptyset$). This FURTHEST(C) can be implemented by using $O(|C|)$ many UPPERBOUND and LOWERBOUND operations, which then takes $O(|C| \log n)$ time in total. To see this, assume $C = \{c_1, \ldots, c_k\}$ where $c_1 \le \ldots \le c_k$ then the clusters partitoned by C is $(-\infty, \frac{1}{2}(c_1 + c_2)], (\frac{1}{2}(c_1 + c_2), \frac{1}{2}(c_2 + c_3)], \cdots, (\frac{1}{2}(c_{k+1} + c_k), +\infty)$ and we can find the potential furthest points in each cluster by querying the following,

$$\begin{aligned}
&\text{UPPERBOUND}(-\infty), \text{LOWERBOUND}\left(\frac{1}{2}(c_1 + c_2)\right), \\
&\text{UPPERBOUND}\left(\frac{1}{2}(c_1 + c_2)\right), \text{LOWERBOUND}\left(\frac{1}{2}(c_2 + c_3)\right) \\
&\ldots \\
&\text{UPPERBOUND}\left(\frac{1}{2}(c_{k+1} + c_k)\right), \text{LOWERBOUND}(+\infty)
\end{aligned}$$

and the furthest point to C among the above $2k = O(|C|)$ many points is what we seek for.

The dynamic algorithm is presented in Algorithm 5. The algorithm samples a set of independent random vectors $\mathcal{V}$ (in a data oblivious way), then creates an above-mentioned interval structure $\mathcal{T}_v$ for each $v \in \mathcal{V}$. When we insert/delete a point x, the update is performed on every $\mathcal{T}_v$ with the projection $\langle x, v \rangle$. The coreset for the current data set P can be computed on the fly by simulating the Gonzalez's algorithm. In particular, this is where the Furthest query is used, and we find an approximate furthest point in P by taking the furthest point in each $\mathcal{T}_v$, and select the one that is the relative furthest in P.

Analysis. Let A be the set of points ever added, so $|A| \le q$. Recall that A is fixed in advance. By applying Lemma A.12 in A, we know that with probability $1 - \delta$, the following event $\mathcal{E}$ happens. For every $C \subseteq A$, $|C| \le k$, every $x \in A$, there exists $v \in \mathcal{V}$, such that

(i) $|\langle c - x, v \rangle| \ge \Omega(\frac{1}{k}) \cdot \|x - c\|_2$ for every $c \in C$, and

(ii) $|\langle a - b, v \rangle| \le O(\sqrt{\log q}) \cdot \|a - b\|_2$ for every $a, b \in A$.

Algorithm 5 Dynamic Gonzalez's algorithm

```
procedure INIT                                    ▷ initialize an empty structure
    $l \leftarrow O(k \log q + \log \delta^{-1})$, and draw $l$ independent random vectors in $\mathbb{R}^m$, denotes as $\mathcal{V}$
    initialize $\mathcal{T}_v$ for each $v \in \mathcal{V}$
end procedure
procedure UPDATE($x$)
    insert/delete $\langle x, v \rangle$ for each $v \in \mathcal{V}$
end procedure
procedure GET-CORESET($k$)
    $Q \leftarrow \emptyset$
    for $i = 1, \dots, k+1$ do
        for $v \in \mathcal{V}$, let $x_v \in P$ satisfy $\langle x_v, v \rangle = \mathcal{T}_v$.FURTHEST($\langle Q, v \rangle$)
                                                  ▷ where $\langle Q, v \rangle := \{\langle x, v \rangle : x \in Q\}$
        $v^\star \leftarrow \arg\max_{v \in \mathcal{V}} \mathrm{dist}(x_v, Q)$
        $Q \leftarrow Q \cup \{x_{v^\star}\}$
    end for
    return $Q$
end procedure
```

Now condition on $\mathcal{E}$. Suppose the current point set is P. Suppose we run the GET-CORESET subroutine and we query $\mathcal{T}_v$.FURTHEST$(\langle Q, v \rangle)$ for some v and Q. Suppose $x \in P \subseteq A$ is the current furthest point to Q. Because of $\mathcal{E}$, there exists a vector $v \in \mathcal{V}$ such that (i) and (ii) hold. By (i), we have that $\mathrm{dist}(\langle x, v \rangle, \langle Q, v \rangle) \geq \Omega(\frac{1}{k}) \cdot \mathrm{dist}(x, Q)$. By (ii), we know that for any $p \in P$ and $c \in Q$, $|\langle p - c, v \rangle| \leq O(\sqrt{\log q}) \|p - c\|_2$, so $\mathrm{dist}(\langle p, v \rangle, \langle Q, v \rangle) \leq O(\sqrt{\log q}) \cdot \mathrm{dist}(p, Q)$. So if $\mathcal{T}_v$.FURTHEST$(\langle Q, v \rangle)$ returns an answer $\langle p, v \rangle$, we know that

$$\mathrm{dist}(p, Q) \geq \frac{\mathrm{dist}(\langle p, v \rangle, \langle Q, v \rangle)}{O(\sqrt{\log q})} \geq \frac{\mathrm{dist}(\langle x, v \rangle, \langle Q, v \rangle)}{O(\sqrt{\log q})} \geq \Omega\left(\frac{1}{k\sqrt{\log q}}\right) \cdot \mathrm{dist}(x, Q).$$

Thus, p is an $O(k\sqrt{\log q})$-approximation of the furthest point to Q. This combining with Lemma A.9. implies the error bound.

Running time. For the running time, we note that for each update of P, we need to update $\mathcal{T}_v$ for each $v \in \mathcal{V}$ accordingly. Thus we need to pay $O(lm)$ time (recalling that $l = O(k \log q + \log \delta^{-1})$ was defined in Algorithm 5) to compute all the inner products and $O(l \log q)$ time to update all $\mathcal{T}_v$'s. The main loop in GET-CORESET requires $O(kl)$ many FURTHEST$(\cdot)$ queries and this runs in $O(k^2 l \log q)$ time in total. In conclusion, the running time of each update (and maintaining coreset) is bounded by

$$O\left((k^2 \log q + m) \cdot l\right) = O\big((k^2 \log q + m)(k \log q + \log \delta^{-1})\big).$$

□

B Lower Bound

We prove the following lower bound to assert the necessity of the exponential dependence on $\min(j, k)$ in our coreset construction Theorem 3.1.

Theorem B.1 (Restatement of Theorem 1.2). *Consider the k-MEANS with missing values problem in $\mathbb{R}^d_?$ where each point can have at most j missing coordinates. Assume there is an algorithm that constructs an ϵ-coreset of size $f(j, k) \cdot \mathrm{poly}(\epsilon^{-1} d \log n)$, then $f(j, k)$ can not be as small as $2^{o(\min(j,k))}$.*

Proof. Consider the following n points instance with $j = k = \Theta(\log n)$, and $d = 2j$. For a subset I of $[d]$, we define a data point $p(I)$ such that $p(I)_i = 1$ if $i \in I$ and $p(I)_i =?$ otherwise. Then we let the data set $P = \{p(I) | I \subseteq [d], |I| = j\}$. We remark that we can make $|P| = \binom{d}{j} = n$ by choosing a proper $j = \Theta(\log n)$.

We prove that any $1/2$-coreset of P should contain every point in P. Let D be such a coreset and assume $p(I) \notin D$, we choose the following $k = j$ centers. For every $i \in I$, we define a center $c^i \in \mathbb{R}^d$ such that the i-th coordinate of c^i is 0 and the other coordinates of c^i are 1. We observe that, for any $i \in I$, $\mathrm{dist}(p(I), c^i) = 1$. Meanwhile for any other $p(I') \neq p(I)$, there must be a $i' \in I \setminus I'$ since $|I| = |I'|$, thus $\mathrm{dist}(p(I'), c^{i'}) = 0$. This should imply that the cost on coreset is 0 while the cost on P is 1 which makes a contradiction.

Since $j = k = \Theta(\log n)$, $d = 2j$, we have $2^{o(\min(j,k))} \cdot \mathrm{poly}(d \log n) = o(n)$. Thus $f(j, k)$ can not be as small as $2^{o(\min(j,k))}$. □