Sets Clustering

Ibrahim Jubran [*1] **Murad Tukan** [*1] **Alaa Maalouf** [*1] **Dan Feldman** [1]

Abstract

The input to the *sets-k-means* problem is an integer $k \geq 1$ and a set $\mathcal{P} = \{P_1, \cdots, P_n\}$ of fixed sized sets in $\mathbb{R}^d$. The goal is to compute a set C of k centers (points) in $\mathbb{R}^d$ that minimizes the sum $\sum_{P \in \mathcal{P}} \min_{p \in P, c \in C} \|p - c\|^2$ of squared distances to these sets. An ε-*core-set* for this problem is a weighted subset of $\mathcal{P}$ that approximates this sum up to $1 \pm \varepsilon$ factor, for *every* set C of k centers in $\mathbb{R}^d$. We prove that such a core-set of $O(\log^2 n)$ sets always exists, and can be computed in $O(n \log n)$ time, for every input $\mathcal{P}$ and every fixed $d, k \geq 1$ and $\varepsilon \in (0, 1)$. The result easily generalized for any metric space, distances to the power of $z > 0$, and M-estimators that handle outliers. Applying an inefficient but optimal algorithm on this coreset allows us to obtain the first PTAS ($1 + \varepsilon$ approximation) for the sets-k-means problem that takes time near linear in n. This is the first result even for sets-mean on the plane ($k = 1$, $d = 2$). Open source code and experimental results for document classification and facility locations are also provided.

1. Introduction

In machine learning it is common to represent the input as a set of n points (database records) $P = \{p_1, \cdots, p_n\}$ in the Euclidean d-dimensional space $\mathbb{R}^d$. That is, an $n \times d$ real matrix whose rows correspond to the input points. Every point corresponds to e.g. the GPS address of a person (Liao et al., 2006; Nguyen et al., 2011), a pixel/feature in an image (Tuytelaars et al., 2008), “bag of words” of a document (Mladenic, 1999), or a sensor’s sample (Dunia et al., 1996). Arguably, the most common statistics of such a set is its mean (center of mass) which is the center $c \in \mathbb{R}^d$ that minimizes its sum of squared distances $\sum_{p \in P} \tilde{D}(p, c) = \sum_{p \in P} \|p - c\|^2$ to the input points in P.

[*]Equal contribution [1]Robotics & Big Data Lab, Department of Computer Science, University of Haifa, Israel. Correspondence to: Ibrahim Jubran <ibrahim.jub@gmail.com>.

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

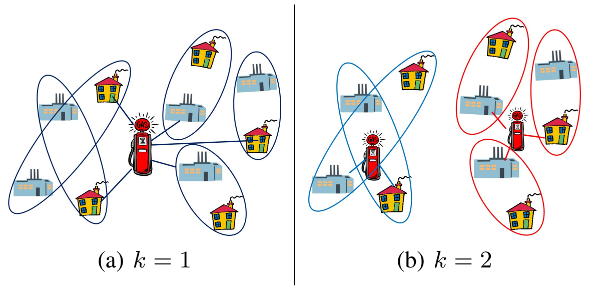


Figure 1. **sets-k-means for pairs on the plane.** The input is a set of $n = 5$ pairs of points that correspond to home/work addresses. (left) The distance from a gas station (in red) to a person is the smaller between its distance to its home and work address (line segments). (right) For $k = 2$ gas stations, each person will choose its closest gas station; see real word database at section 6.

Here, $\tilde{D}(p, c) := \|p - c\|^2$ is the squared distance between a point $p \in P$ to the center $c \in \mathbb{R}^d$. More generally, in unsupervised learning, for a given integer (number of clusters) $k \geq 1$, the *k-means* of the set P is a set $C = \{c_1, \cdots, c_k\}$ of k centers (points in $\mathbb{R}^d$) that minimizes the sum of squared distances

$$\sum_{p \in P} \tilde{D}(p, C) = \sum_{p \in P} \min_{c \in C} \|p - c\|^2 ,$$

where $\tilde{D}(p, C) := \min_{c \in C} \tilde{D}(p, c)$ denotes the squared distance from each point $p \in P$ to its nearest center in C. The k-means clustering is probably the most common clustering objective function, both in academy and industry as claimed in (Hartigan, 1975; Arthur & Vassilvitskii, 2006; Berkhin, 2002; Wu et al., 2008).

However, in the real-world, every database’s record actually links to another database table, a GPS location may correspond to multiple GPS locations (e.g. home/work), every image consists of a set of pixels/features, every document contains a set of paragraphs, and a sensor’s sample may actually be a distribution over some possible values (Li et al., 2010; 2008; Dunia et al., 1996; Xiao et al., 2007). This motivates the following title and subject of this paper.

Sets Clustering. Along this paper, the input is not a set of points, but rather a set $\mathcal{P} = \{P_1, \cdots, P_n\}$ of sets in $\mathbb{R}^d$ (or any other metric space; see Section 2), each of size m, denoted as m-sets. A natural generalization of the mean of a set P is what we defined as the *sets-mean* of our set $\mathcal{P}$ of sets. The sets-mean is the point $c \in \mathbb{R}^d$ that minimizes its sum of squared distances

$$\sum_{P \in \mathcal{P}} \tilde{D}(P, c) = \sum_{P \in \mathcal{P}} \min_{p \in P} \|p - c\|^2, \qquad (1)$$

to the nearest point in each set. Here, $\tilde{D}(P, c) := \min_{p \in P} \tilde{D}(p, c)$.

More generally, the *sets-k-means* C of $\mathcal{P}$ is a set of k points in $\mathbb{R}^d$ that minimizes its sum of squared distances

$$\sum_{P \in \mathcal{P}} \tilde{D}(P, C) = \sum_{P \in \mathcal{P}} \min_{p \in P, c \in C} \|p - c\|^2, \qquad (2)$$

to the nearest point in each set. Here, $\tilde{D}(P, C) := \min_{p \in P, c \in C} \tilde{D}(p, c)$ is the closest distance between a pair in $P \times C$.

Example. Suppose that we want to place a gas station that will serve n people whose home addresses are represented by n GPS points (on the plane). The mean is a natural candidate since it minimizes the sum of squared distances from the gas station to the people; see (Jubran et al., 2019). Now, suppose that the ith person for every $i \in \{1, \cdots, n\} = [n]$ is represented by a pair $P_i = \{h, w\}$ of points on the plane: home address h and work address w; see Fig. 1. It would be equally as convenient for a resident if the gas station was built next to his work address rather than his home address. Hence, the sets-mean of the addresses $\mathcal{P} = \{P_1, \cdots, P_n\}$, as defined in the previous page, minimizes the sum of squared distances from the gas station to the nearest address of each person (either home or work). The sets-k-means is the set $C \subseteq \mathbb{R}^d$ of k gas stations that minimizes the sum of squared Euclidean distances from each person to its nearest gas station as in (2).

1.1. Applications

From a theoretical point of view, sets clustering is a natural generalization of points clustering. The distance $\tilde{D}(P, C)$ between sets generalizes the distance $\tilde{D}(p, C) = \min_{c \in C} \tilde{D}(p, c)$ between a point and a set, as used e.g. in k-means clustering of points.

Clustering Shapes (Srivastava et al., 2005). The first sets clustering related result appeared only recently in (Marom & Feldman, 2019) for the special case where each of the n input sets is a line (an infinite set) in $\mathbb{R}^d$. However, in this paper every input set is a finite and arbitrary set in a general metric space.

It is therefore not surprising that many of the numerous applications for points clustering can be generalized to sets clustering. Few examples are given below.

Facility locations (Cohen-Addad et al., 2019; Blelloch & Tangwongsan, 2010; Ahmadian et al., 2013). The above gas station example immediately implies applications for Facility Location problems.

Natural Language Processing (Collobert et al., 2011). A disadvantage of the common "bag of words" model is that the order of words in a document does not change its representation (Spanakis et al., 2012). Sets clustering can help partially overcome this issue by considering the document as the set of vectors corresponding to each of its paragraphs, as illustrated in Fig. 5.

Hierarchical clustering (Abboud et al., 2019; Murtagh, 1983). Here, the goal is to compute a tree of clusters. The leaves of this tree are the input points, and the next level represent their clustering into n sets. In the next level, the goal is to cluster these n sets into k sets.

Probabilistic databases (Suciu et al., 2011). Here, each data sample corresponds to a finite distribution over possible values. E.g. a sample that was obtained from a sensor with a known noise model. Algorithm for computing the minimum enclosing ball (1-center) for sets (distributions) was suggested in (Munteanu et al., 2014) using coresets, as defined in section 1.4.

1.2. Why is it Hard?

Computing the k-means of points in $\mathbb{R}^d$ $(m = 1)$ is already NP-hard when k is not fixed, even for $d = 2$. It can be solved in $n^{O(dk)}$ time using exhaustive search as explained in (Inaba et al., 1994). Multiplicative $(1 + \varepsilon)$ approximation is also NP-hard for constant a $\varepsilon > 0$ (Lee et al., 2017).

For fixed k, deterministic constant factor approximation can be computed in time $O(ndk)$ by constructing coresets (see Section 1.4) of size $m = O(k/\varepsilon^3)$ (Braverman et al., 2016; Feldman & Langberg, 2011), on which the optimal exhaustive search is then applied. In practice, it has efficient approximation algorithms with provable guarantees, such as k-means++ (Arthur & Vassilvitskii, 2006) which yields $O(\log k)$ approximation, using D^2 sampling.

The mean $(k = 1)$ $\sum_{p \in P} p/n$ of a set P of n points in $\mathbb{R}^d$ can be computed in linear $O(nd)$ time. However, we could not find in the literature an algorithm for computing even the sets-mean in (1) for n pairs of points on the plane $(m = d = 2)$.

Separability. The clusters in the k-means problem are *separable*: the minimum enclosing ball of each cluster consists only of the points in this cluster. Fundamental results in computational geometry (Toth et al., 2017) (chapter 28) or PAC-learning theory (Shalev-Shwartz & Ben-David, 2014) prove that there are only $n^{O(1)}$ partitions of P into k such clusters that can be covered by balls. On the contrary, even in the case of sets-mean $(k = 1)$, the union of n representative points from each pair is not separable from the other n

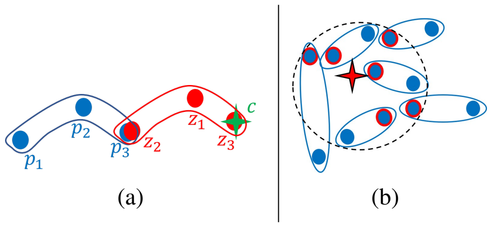


Figure 2. **Why is the sets clustering problem hard?** 2(a): The space is non-metric. Two m-sets $P = \{p_1, p_2, p_3\}$ and $Z = \{z_1, z_2, z_3\}$ in $\mathbb{R}^d$ for $m = 3$ and $c \in \mathbb{R}^d$ that do not satisfy the triangle inequality since $\tilde{D}(P, Z) = 0, \tilde{D}(Z, c) = 0$ but $\tilde{D}(P, c) \neq 0$. 2(b): Separability. For $d = 2$, a set of $n = 6$ pairs (blue ellipses) and their optimal mean (red star). There is no ball that separates the closest n points (1 from each set) which are closest to the optimal mean (red circles), from the other n points (solid blue circles).

points (that are not served by the center); see Fig 2.

Non-metric space. The generalization of the k-means distance function to sets in (2) is not a metric space, i.e., does not satisfy the triangle inequality, even approximately. For example, two input sets might have zero distance between them while one is very far and the other is very close to a center point; see Fig 2.

Furthermore, one may suggest to replace every input set $P \subseteq \mathbb{R}^d$ of size $|P| = m$ with m points in $\mathbb{R}^d$, thus reducing the sets-k-means problem to an instance of the well known k-means problem. However, in theory, Fig. 2(a) gives a simple example where the cost of sets-k-means is 0 and the k-means cost of flattening the sets may be arbitrarily large.

1.3. How Hard?

The previous section may raise the suspicion that sets-k-means is NP-hard, even for $k = 1$ and $d = 2$. However, this is not the case. In Section 4.4, we present a simple theorem for computing the *exact* (optimal) sets-k-means for any input set $\mathcal{P}$ of n sets, each of size m. This takes time polynomial in n, i.e., $n^{O(1)}$, for every constant integers $k, d, m \geq 1$. The theorem is based on a generic reduction for the case of $k = m = 1$. Unfortunately, the constants that are hidden in the $O(1)$ notation above make our algorithm impractical for even modest values of k. This motivates the construction of the first *coreset for sets*, which is the main technical result of this paper.

1.4. Sets Coresets

Coreset (or core-set) is a modern data summarization paradigm (Maalouf et al., 2019a; Bachem et al., 2017a; Phillips, 2016) that was originated from computational geometry (Agarwal et al., 2005). Usually, the input for a *coreset construction algorithm* is an approximation error $\varepsilon \in (0, 1)$, a set $\mathcal{P}$ of n items (called points), and a loss $\sum_{P \in \mathcal{P}} \tilde{D}(P, \cdot)$ that we wish to minimize over a (usually infinite) set $\mathcal{C}$ of feasible queries (solutions). The output is a (sub)set $\mathcal{S} \subseteq \mathcal{P}$ and a weights function $v : \mathcal{S} \to [0, \infty)$, which is called an ε-*coreset* for the tuple $(\mathcal{P}, \mathcal{C}, \tilde{D})$ if

$$\left| \sum_{P \in \mathcal{P}} \tilde{D}(P, C) - \sum_{S \in \mathcal{S}} v(S) \tilde{D}(S, C) \right| \leq \varepsilon \sum_{P \in \mathcal{P}} \tilde{D}(P, C),$$

for *every* query $C \in \mathcal{C}$. In particular, an optimal solution of the coreset is an approximated optimal solution to the original problem. If $|\mathcal{S}| \ll |\mathcal{P}|$, i.e., the size of the coreset $\mathcal{S}$ is smaller than $\mathcal{P}$ by orders of magnitude, then we can run a possibly inefficient algorithm on $\mathcal{S}$ to compute an approximation solution to $\mathcal{P}$. In this paper, unlike previous papers, $\mathcal{P}$ is a set of sets of size m (rather than points) in $\mathbb{R}^d$ and $\mathcal{C} = \left\{ C \subseteq \mathbb{R}^d \middle| |C| = k \right\}$.

Why coresets? Applying the above optimal exhaustive search on such a coreset would reduce the running time from $n^{O(1)}$ to time near linear in n conditioned upon: (i) every such input $\mathcal{P}$ has a coreset $\mathcal{S}$ of size, say, $|\mathcal{S}| \in (\log n)^{O(1)}$, and (ii) this coreset can be computed in near linear time, say $O(n \log n)$.

However, such a coreset construction for a problem has many other applications, including handling big streaming dynamic distributed data in parallel. Here, streaming means maintaining the sets-k-means of a (possibly infinite) stream of sets, via one pass and using only logarithmic memory and update time per new set. Dynamic data supports also deletion of sets. Distributed data means that the input is partitioned among $M \geq 2$ machines, where the running time reduces by a factor of M (Régin et al., 2013). Many surveys explain how to obtain those applications, given an efficient construction of a small coreset as suggested in our paper. Due to lack of space we do not repeat them here and refer the reader to e.g. (Feldman, 2020).

The recent result above (Marom & Feldman, 2019) for k-means of lines (infinite sets) is obtained via coresets. We do not know any coresets for finite sets except for singletons ($m = 1$). This coreset, that is called coreset for k-means (of points) is one of the fundamental and most researched coresets in this century: (Har-Peled & Mazumdar, 2004; Chen, 2006; Frahling & Sohler, 2008; Chen, 2009; Fichtenberger et al., 2013; Bachem et al., 2015; Barger & Feldman, 2016; Bachem et al., 2017b; Feldman et al., 2017; Bachem et al., 2018; Huang et al., 2018). Coresets for fair clustering of points, which preserve sets-related properties of the input points, were suggested in (Schmidt et al., 2019).

A natural open question is **"does a small coreset exist for the sets-k-means problem of any input?"**.

1.5. Main Contributions

In this paper we suggest the first $(1+\varepsilon)$ approximation for the sets-k-means problem, by suggesting the first coreset for sets. More precisely, we provide

(i): A proof that an ε-coreset $\mathcal{S}$ of size $|\mathcal{S}| = O(\log^2 n)$ exists for *every* input set $\mathcal{P}$ of n sets in $\mathbb{R}^d$, each of size m. This holds for every constants $d, k, m \geq 1$. $\mathcal{S}$ can be computed in time $O(n \log n)$; see exact details in Theorem 4.2. We also provide a lower bound for the size of such a coreset; see Section 4.3.

(ii): An algorithm that computes an optimal solution for the sets-k-means of such $\mathcal{P}$ in $n^{O(1)}$ time. See Theorem 4.4.

(iii): Combining the above results implies the first PTAS ($(1+\varepsilon)$-approximation) for the sets-k-means of any such input set $\mathcal{P}$, that takes $O(n \log n)$ time; see Corollary 4.5.

(iv): Extensions for (i) from the Euclidean distance in $\mathbb{R}^d$ to any metric space $(\mathcal{X}, \tilde{D})$, distances to the power of $\ell > 0$, and M-estimators that are robust to outliers. See Section 2.

(v): Experimental results on synthetic and real-world datasets show that our coreset performs well also in practice.

(vi): Open source implementation for reproducing our experiments and for future research (Jubran et al., 2020).

1.6. Novelty

Our coreset construction needs to characterize which of the input items are similar, and which are dissimilar, in some sense. To this end, we first suggest a similarity measure for sets and then present our novel non-uniform sampling scheme for sets, which we call *onion sampling*.

Recursive similarity. When $m = 1$, items are similar if their mutual distance is small. When $m \geq 2$, we propose a recursive and abstract similarity measure, which requires all the m items in the first set to be "close" to the m items in the second set, for some ordering of the items inside each set; see Algorithm 1.

Onion Sampling. Recall that the D^2 sampling assigns each input point with probability that is proportional to its distance to the k-means of the input (or its approximation), which reflects its importance. When we try to generalize D^2 to handle sets rather than points, it is not clear what to do when one point in an input m-set is close to the approximated center and the other one is far, as in Fig. 2(a). In particular, if the optimal sum of squared distances is zero, the coreset in the in k-means problem is trivial (the k points). This is not the case for the sets-k-mean (even for $k = 1$).

To this end, we suggest an iterative and non-trivial alternative sampling scheme called onion sampling. In each iteration we apply an algorithm which characterizes "recursively similar" input sets, as described above, which form an "onion layer". We assign those sets the same sampling probability, which is inversely proportional to the number of those items, and peal this layer off. We continue until we have pealed off the entire onion (input). Finally, we prove that a random sample according to this distribution yields a coreset for the sets clustering problem; see Algorithm 2.

2. Definitions

In (2) we define sets-k-means for points in $\mathbb{R}^d$. However, our coreset construction holds for any metric space, or general (non-distance) loss functions as in Table 1.

Definition 2.1 (Loss function $\tilde{D}$). *Let* $\text{lip} : [0,\infty) \to [0,\infty)$ *be a non-decreasing function that satisfies the following* r-log-log Lipschitz condition*: There is a constant* $0 < r < \infty$ *such that for every* $x, z > 0$ *we have* $\text{lip}(zx) \leq z^r \text{lip}(x)$*. Let* $(\mathcal{X}, D)$ *be a metric space, and* $\tilde{D} : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to [0,\infty)$ *be a function that maps every two subsets* $P, C \subseteq \mathcal{X}$ *to*

$$\tilde{D}(P,C) := \min_{p \in P, c \in C} \text{lip}(\tilde{D}(p,c)).$$

For $p, b \in \mathcal{X}$*, denote* $\tilde{D}(p,C) := \tilde{D}(\{p\}, C)$*, and* $\tilde{D}(P,b) := \tilde{D}(P,\{b\})$*, for short. For an integer* $k \geq 1$ *define* $\mathcal{X}_k := \{C \subseteq \mathcal{X} \mid |C| = k\}$.

Although $(\mathcal{X}, \tilde{D})$ is not necessarily a metric space, the triangle inequality is approximated as follows.

Lemma 2.2 (Lemma 2.1 (ii) in (Feldman & Schulman, 2012))**.** *Let* $(\mathcal{X}, \tilde{D})$ *and* $r > 0$ *be as defined in Definition 2.1. Let* $\rho = \max\{2^{r-1}, 1\}$*. Then the function* $\tilde{D}$ *satisfies the weak triangle inequality for singletons, i.e., for every* $p, q, c \in \mathcal{X}$*,* $\tilde{D}(p,q) \leq \rho(\tilde{D}(p,c) + \tilde{D}(c,q))$.

Table 1. Example loss functions as in Definition 2.1. Let $\delta > 0$ be a constant and let $(\mathcal{X}, \tilde{D})$ be a metric space where $\mathcal{X} = \mathbb{R}^d$ and $\tilde{D}(p,c) = \|p - c\|$ for every $p, c \in \mathbb{R}^d$.

Optimization Problem	$\text{lip}(x)$	$\tilde{D}(P,C)$	ρ
sets-k-median	x	$\min_{p \in P, c \in C} \|p - c\|$	1
sets-k-means	x^2	$\min_{p \in P, c \in C} \|p - c\|^2$	2
sets-k-means with M-estimators	$\begin{cases} \frac{1}{2}x^2 & \text{if } x \leq \delta \\ \delta(\lvert x \rvert - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$	$\min_{p \in P, c \in C} \begin{cases} \frac{1}{2}\|p - c\|^2 & \text{if } \|p - c\| \leq \delta \\ \delta(\|p - c\| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$	2
ℓ_ψ norm	x	$\min_{p \in P, c \in C} \|p - c\|_\psi$	$\max\{2^{\frac{1}{\psi}}, 1\}$

Notation. For the rest of the paper we denote $[n] = \{1, \cdots, n\}$ for an integer $n \geq 1$. Unless otherwise stated, let $(\mathcal{X}, \tilde{D})$ be as in Definition 2.1.

As discussed in Section 1, the input set for the sets clustering problem is a set of finite and equal sized sets as follows.

Definition 2.3 ((n,m)-set)**.** *An* m*-set* P *is a set of* m distinct *points in* $\mathcal{X}$*, i.e.* $P \subseteq \mathcal{X}$ *and* $|P| = m$*. An* (n,m)-set *is a set* $\mathcal{P} = \{P \mid P \subseteq \mathcal{X}, |P| = m\}$ *such that* $|\mathcal{P}| = n$.

In what follows we define the notion of robust approximation. Informally, a robust median for an optimization problem at hand is an element b that approximates the optimal value of this optimization problem, with some leeway on the number of input elements considered.

Definition 2.4 (Robust approximation)**.** *Let $\mathcal{P}$ be an (n,m)-set, $\gamma \in (0, \frac{1}{2}]$, $\tau \in (0, 1/10)$, and $\alpha \geq 1$. Let $(\mathcal{X}, \tilde{D})$ be as in Definition 2.1. For every $C \in \mathcal{X}_k$, we define* $\text{closest}(\mathcal{P}, C, \gamma)$ *to be the set that is the union of $\lceil \gamma |\mathcal{P}| \rceil$ sets $P \in \mathcal{P}$ with the smallest values of $\tilde{D}(P, C)$, i.e.,*

$$\text{closest}(\mathcal{P}, C, \gamma) \in \underset{\mathcal{Q} \subseteq \mathcal{P} : |Q| = \lceil \gamma |\mathcal{P}| \rceil}{\arg\min} \sum_{P \in \mathcal{Q}} \tilde{D}(P, C).$$

*The singleton $\{b\} \in \mathcal{X}_1$ is a (γ, τ, α)-*median *for $\mathcal{P}$ if*

$$\sum_{P \in \text{closest}(\mathcal{P}, \{b\}, (1-\tau)\gamma)} \tilde{D}(P, b) \leq \alpha \cdot \min_{b' \in \mathcal{X}} \sum_{P \in \text{closest}(\mathcal{P}, \{b'\}, \gamma)} \tilde{D}(P, b').$$

Given an m-set P, and a set $\mathcal{B}$ of $|\mathcal{B}| = j \leq m$ points, in what follows we define the projection of P onto $\mathcal{B}$ to be the set P after replacing j of its points, which are the closest to the points of $\mathcal{B}$, by the points of $\mathcal{B}$. We denote by $\overline{\text{proj}}(P, \mathcal{B})$ the remaining "non-projected" points of P.

Definition 2.5 (Set projection)**.** *Let $m \geq 1$ be an integers, P be an m-set, $(\mathcal{X}, \tilde{D})$ be as in Definition 2.1, $j \in [m]$, and let $\mathcal{B} = \{b_1, \cdots, b_j\} \in \mathcal{X}_j$. Let $p_1 \in P$ denote the closest point to b_1 i.e., $p_1 \in \arg\min_{p \in P} \tilde{D}(p, b_1)$. For every integer $i \in \{2, \cdots, j\}$ recursively define $p_i \in P$ to be the closest point to b_i, excluding the $i-1$ points that were already chosen, i.e., $p_i \in \underset{p \in P \setminus \{p_1, \cdots, p_{i-1}\}}{\arg\min} \tilde{D}(p, b_i)$. We denote **(i):** $\{(p_1, b_1), \cdots, (p_j, b_j)\}$ by* $\text{closepairs}(P, \mathcal{B})$*, **(ii):** the $m - j$ points from P that are not among the closest points to $\mathcal{B}$ by $\overline{\text{proj}}(P, \mathcal{B}) = P \setminus \{p_1, \cdots, p_j\}$, and **(iii):** the* projection *of P onto $\mathcal{B}$ by $\text{T}(P, \mathcal{B}) = \{b_1, \cdots, b_j\} \cup \big(P \setminus \{p_1, \cdots, p_j\}\big)$. For $X = \emptyset$, we define $\overline{\text{proj}}(P, X) = \text{T}(P, X) = P$*

3. Sensitivity Based Coreset

A common technique to compute coresets is the approach of non-uniform sampling, which is also called sensitivity sampling (Langberg & Schulman, 2010; Braverman et al., 2016), and was widely used lately to construct coresets for Machine Learning problems; see e.g., (Huggins et al., 2016; Munteanu et al., 2018; Maalouf et al., 2019b; Bachem et al., 2017a). Intuitively, the sensitivity of an element $P \in \mathcal{P}$ represents the importance of P with respect to the other elements, and the specific optimization problem at hand; see definition and details in Theorem 3.1. Suppose that we computed an upper bound $s(P)$ for the sensitivity of every element $P \in \mathcal{P}$. Then a coreset is now simply a random (sub)sample of $\mathcal{P}$ according to the sensitivity distribution, followed by a smart reweighting of the points. It's size is proportional to the sum of sensitivities $t = \sum_{Q \in \mathcal{P}} s(Q)$ and the combinatorial complexity d' of the problem at hand; see Definition A.2. The following theorem, which is a restatement of Theorem 5.5 in (Braverman et al., 2016), provides full details.

Theorem 3.1. *Let $\mathcal{P}$ be an (n,m)-set, and $(\tilde{D}, \mathcal{X}_k)$ be as in Definition 2.1. For every $P \in \mathcal{P}$ define the* sensitivity *of P as*

$$\sup_{C \in \mathcal{X}_k} \frac{\tilde{D}(P, C)}{\sum_{Q \in \mathcal{P}} \tilde{D}(Q, C)},$$

where the sup is over every $C \in \mathcal{X}_k$ such that the denominator is non-zero. Let $s : \mathcal{P} \to [0,1]$ be a function such that $s(P)$ is an upper bound on the sensitivity of P. Let $t = \sum_{P \in \mathcal{P}} s(P)$ and d' be a complexity measure of the set clustering problem; see Definition A.2. Let $c \geq 1$ be a sufficiently large constant, $\varepsilon, \delta \in (0,1)$, and let $\mathcal{S}$ be a random sample of $|S| \geq \frac{ct}{\varepsilon^2}\left(d' \log t + \log \frac{1}{\delta}\right)$ sets from $\mathcal{P}$, such that P is sampled with probability $s(P)/t$ for every $P \in \mathcal{P}$. Let $v(P) = \frac{t}{s(P)|C|}$ for every $P \in \mathcal{S}$. Then, with probability at least $1 - \delta$, (S, v) is an ε-coreset for $(\mathcal{P}, \mathcal{X}_k, \tilde{D})$.

4. Coreset for Sets Clustering

In this section we give our main algorithms that compute a coreset for the sets clustering problem, along with intuition, Full theoretical proofs can be found in the appendix.

4.1. Algorithms

Overview and intuition behind Algorithm 1. Given a set $\mathcal{P}$ of m-sets and an integer $k \geq 1$, Algorithm 1 aims to compute a set $\mathcal{P}^m \subset \mathcal{P}$ of "similar" m-sets, which are all equally important for the problem at hand; see Lemma 4.1. At the ith iteration we wish to find a $\frac{1}{4k}$ fraction of the remaining m-sets $\mathcal{P}^{i-1}$ which are similar in the sense that there is a dense ball of small radius that contains at least one point from each of those sets. To do so, we first compute at Line 5 $\hat{\mathcal{P}}^{i-1}$ which contains only the "non-projected" points of each m-sets in $\mathcal{P}^{i-1}$. We then compute a median b^i at Line 6 that satisfies at least $\frac{1}{4k}$ of $\hat{\mathcal{P}}^{i-1}$. b^i is the center of the desired dense ball. At Line 7 we pick the sets that indeed have a candidate inside this dense ball and continue to the next iteration (where again, we consider only the non-projected part of those sets); see Fig. 3. After m such iterations, the surviving m-sets in $\mathcal{P}^m$ have been "recursively similar" throughout all the iterations.

Overview and intuition behind Algorithm 2. Given an (n,m)-set $\mathcal{P}$ and an integer $k \geq 1$, Algorithm 2 aims to compute an ε-coreset (S, v) for $\mathcal{P}$; see Theorem 4.2. Algorithm 2 applies our onion sampling scheme; each while iteration at Line 6 corresponds to a pealing iteration.

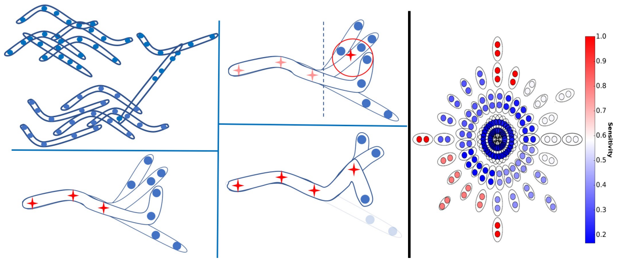


Figure 3. **Recursive similarity.** (Upper left): An input (n, m)-set $\mathcal{P}$ with $n = 14$ and $m = 5$. (Lower left): The set $\mathcal{B}^i$ (red stars) for $i = 3$ from the $3rd$ iteration of Algorithm 1, and the projection $\mathrm{T}(\mathcal{P}^3, \mathcal{B}^3)$ (blue snakes) of $\mathcal{P}^3$ onto $\mathcal{B}^3$. Therefore, all the sets in $\mathrm{T}(\mathcal{P}^3, \mathcal{B}^3)$ have $i = 3$ points in common, and $m - i = 2$ other points. (Upper mid): The set $\hat{\mathcal{P}}^3$ that contains four 2-sets (blue points right of the dashed line). The median b^4 (bold red star) considers only a fraction of $\hat{\mathcal{P}}^3$. The red circle contains the points that are closest to b^4. $\mathcal{P}^4$ contains the sets with a representative inside the red circle. (Lower mid): The projection $\mathrm{T}(\mathcal{P}^4, \mathcal{B}^4)$ (blue snakes) of the sets $\mathcal{P}^4$ onto the new $\mathcal{B}^4 = \mathcal{B}^3 \cup \{b^4\}$.
(Right): Onion sampling. A set $\mathcal{P}$ of pairs in the plane ($m = d = 2$) along with the sensitivity values $s(P)$ computed in Algorithm 2 via our onion sampling. First, the densest subset of pairs are assigned a low sensitivity value (dark blue). The densest subset of the remaining pairs is then assigned a higher sensitivity value (light blue), and so on. The scattered pairs that remain at the end are assigned the highest sensitivity (dark red). The size of the subset found decreases in each step.

At lines 6–14 Algorithm 2 first calls Algorithm 1 with the (n, m)-set $\mathcal{P}^0 = \mathcal{P}$ to obtain a set $\mathcal{P}^m \subseteq \mathcal{P}$ of "dense" and equally (un)important m-sets from the input. Second, it assigns all the sets in $\mathcal{P}^m$ the same sensitivity value as shown in Lemma 4.1. It then peals those sets off, and repeats this process with $\mathcal{P}^0 \setminus \mathcal{P}^m$. Those values increase in every step since the size of the dense set returned decreases, making every point more important. This process is illustrated in Fig. 3. We then randomly sample a sufficiently large set $S \subseteq \mathcal{P}$ at Line 17 according to the sensitivity values, and assign new weights $v(P)$ for every set $P \in \mathcal{S}$ in Line 19.

4.2. Main Theorems

The following lemma lies at the heart of our work. It proves that Algorithm 1 helps compute an upper bound for the sensitivity term of some of the input elements.

Lemma 4.1. *Let $\mathcal{P}$ be an (n, m)-set, $k \geq 1$ be an integer and $(\mathcal{X}, \tilde{D})$ be as in Definition 2.1. Let $(\mathcal{P}^m, \mathcal{B}^m)$ be the output of a call to* RECURSIVE-ROBUST-MEDIAN$(\mathcal{P}, k)$*; see Algorithm 1. Then, for every $P \in \mathcal{P}^m$ we have that*

$$\sup_{C \in \mathcal{X}_k} \frac{\tilde{D}(P, C)}{\sum\limits_{Q \in \mathcal{P}} \tilde{D}(Q, C)} \in O(1) \cdot \left(\frac{1}{|\mathcal{P}^m|} \right).$$

The following theorem is our main technical contribution.

Algorithm 1 RECURSIVE-ROBUST-MEDIAN$(\mathcal{P}, k)$

```
1:  Input: An $(n, m)$-set $\mathcal{P}$ and a positive integer $k$.
2:  Output: A pair $(\mathcal{P}^m, \mathcal{B}^m)$ where $\mathcal{P}^m \subseteq \mathcal{P}$ and
        $\mathcal{B}^m \in \mathcal{X}_m$; see Lemma 4.1.
3:  $\mathcal{P}^0 := \mathcal{P}$ and $\mathcal{B}^0 := \emptyset$
4:  for $i := 1$ to $m$ do
5:      $\hat{\mathcal{P}}^{i-1} := \left\{ \overline{\mathrm{proj}}(P, \mathcal{B}^{i-1}) \middle| P \in \mathcal{P}^{i-1} \right\}$ {see Definition 2.5}
6:      Compute a $\left(\frac{1}{2k}, \tau, 2\right)$-median $\left\{b^i\right\} \in \mathcal{X}_1$ for $\hat{\mathcal{P}}^{i-1}$
        for some $\tau \in (0, \frac{1}{20})$.
        {see Definition 2.4. Algorithm 3 provides a suggested implementation.}
7:      $\mathcal{P}^i := \left\{ \begin{array}{c} P \\ \in \mathcal{P}^{i-1} \end{array} \middle| \begin{array}{c} \overline{\mathrm{proj}}(P, \mathcal{B}^{i-1}) \in \\ \mathrm{closest}\left(\hat{\mathcal{P}}^{i-1}, \left\{b^i\right\}, \frac{(1-\tau)}{4k}\right) \end{array} \right\}$
        {$\mathcal{P}^i$ contains every $m$-set $P$ such that $\overline{\mathrm{proj}}(P, \mathcal{B}^{i-1})$
        is in the closest fraction of $(1 - \tau)/(4k)$ sets in
        $\hat{\mathcal{P}}^{i-1}$ to the center $b^i$; see Fig. 3.}
8:      $\mathcal{B}^i := \mathcal{B}^{i-1} \bigcup \left\{b^i\right\}$
9:  end for
10: Return$(\mathcal{P}^m, \mathcal{B}^m)$
```

Algorithm 2 CORESET$(\mathcal{P}, k, \varepsilon, \delta)$

```
1:  Input: An $(n, m)$-set $\mathcal{P}$, a positive integer $k$, an error
        parameter $\varepsilon \in (0, 1)$, and a probability of failure
        $\delta \in (0, 1)$.
2:  Output: An $\varepsilon$-coreset $(S, v)$; see Theorem 4.2.
3:  $b :=$ a constant determined by the proof of Theorem 4.2
4:  $d' := md^2k^2$ {the dimension of $(\mathcal{P}, \mathcal{X}_k, \tilde{D})$}
5:  $\mathcal{P}^0 := \mathcal{P}$
6:  while $|Q_0| > b$ do
7:      $(\mathcal{P}^m, \mathcal{B}^m) :=$ RECURSIVE-ROBUST-MEDIAN$(\mathcal{P}^0, k)$
8:      for every $P \in \mathcal{P}^m$ do
9:          $s(P) := \frac{b}{|\mathcal{P}^m|}$
10:     end for
11:     $\mathcal{P}^0 := \mathcal{P}^0 \setminus \mathcal{P}^m$
12: end while
13: for every $P \in \mathcal{P}^0$ do
14:     $s(P) = 1$
15: end for
16: $t := \sum_{P \in \mathcal{P}} s(P)$
17: Pick $S$ of $\frac{bt}{\varepsilon^2}\left(\log(t)d' + \log\left(\frac{1}{\delta}\right)\right)$ $m$-sets from $\mathcal{P}$ by
    repeatedly, i.i.d, selecting $P \in \mathcal{P}$ with probability $\frac{s(P)}{t}$
18: for each $P \in S$ do
19:     $v(P) := \frac{t}{|S| \cdot s(P)}$
20: end for
21: Return $(S, v)$
```

It proves that Algorithm 2 indeed computes an ε-coreset.

Theorem 4.2. *Let $\mathcal{P}$ be an (n, m)-set, $k \geq 1$ be an integer,*

$(\mathcal{X}, \tilde{D})$ be as in Definition 2.1, and $\varepsilon, \delta \in (0,1)$. Let $(\mathcal{S}, v)$ be the output of a call to CORESET$(\mathcal{P}, k, \varepsilon, \delta)$. *Then*

(i) $|\mathcal{S}| \in O\left(\left(\frac{md \log n}{\varepsilon}\right)^2 k^{m+4}\right)$.

(ii) With probability at least $1-\delta$, $(\mathcal{S}, v)$ is an ε-coreset for $(\mathcal{P}, \mathcal{X}_k, \tilde{D})$; see Section 1.4.

(iii) (S, v) can be computed in $O(n \log(n)(k)^m)$ time.

4.3. Sets Clustering Coreset Lower Bound

In this section we provide a lower bound for the coreset size of a sets clustering problem. This is by constructing an (n,m)-set $\mathcal{P}$ in $\mathbb{R}$ for which every $P \in \mathcal{P}$ has a sensitivity of 1; see Section 3 and Theorem 3.1.

Theorem 4.3. *Let $k, m \geq 1$ be integers. Let $\mathcal{P}$ be an (n,m)-set in $\mathbb{R}$ containing all the possible $n = \binom{m+k}{m}$ subsets of m points in $[m+k] = \{1, 2, ..m+k\}$, and let $(\mathcal{S}, v)$ be an ε-coreset for $\mathcal{P}$. Then $|S| \geq k^m$.*

Proof. By its construction, every $P \in \mathcal{P}$ contains $|P| = m$ reals among the $m+k$ reals $[m+k]$. Hence, every $P \in \mathcal{P}$ has a corresponding query $C = [m+k] \setminus P \subseteq \mathbb{R}$ (the remaining k reals in $[m+k]$) that yields non-zero distance $\tilde{D}(P, C) \neq 0$ to this set P, and zero distance $\tilde{D}(Q, C) = 0$ to all the other sets $Q \in \mathcal{P}$. Hence, any coreset must contain all the $\binom{m+k}{m}$ sets in $\mathcal{P}$, otherwise the cost of the coreset $(\mathcal{S}, v)$ might be 0 while the cost of the original data is non-zero, which yields an infinite multiplicative error for the coreset.

Therefore, the coreset size $|\mathcal{S}|$ for a sets clustering problem satisfies

$$k^m < |\mathcal{S}| = \binom{m+k}{m} < (2ek)^m,$$

where the last inequality is by the binomial inequality. □

4.4. Polynomial Time Approximation Scheme.

In the following theorem we present a reduction from an α-approximation for the sets clustering problem in $\mathbb{R}^d$ with $m, k \geq 1$, to an α-approximation for the simplest case where $m = k = 1$, for any $\alpha \geq 1$. We give a suggested implementation in Algorithm 4.

Theorem 4.4. *Let $\mathcal{P}$ be an (n,m)-set in $\mathbb{R}^d$, $w : \mathcal{P} \to [0, \infty)$ be a weights function, $k \geq 1$ be an integer, $\alpha \geq 1$ and $\delta \in [0,1)$. Let $\tilde{D}$ be a loss function as in Definition 2.1 for $\mathcal{X} = \mathbb{R}^d$. Let* ALG *be an algorithm that solves the case where $k = m = 1$, i.e., it takes as input a set $Q \subseteq \mathcal{X}$, a weights function $u : Q \to [0, \infty)$ and the failure probability δ, and in time $T(n)$ outputs $\hat{c} \in \mathcal{X}$ that with probability at least $1-\delta$ satisfies $\sum_{q \in Q} u(q) \cdot \tilde{D}(q, \hat{c}) \leq \alpha \cdot \min_{c \in \mathcal{X}} \sum_{q \in Q} u(q) \cdot \tilde{D}(q, c)$. Then in $T(n) \cdot (nmk)^{O(dk)}$ time we can compute $\hat{C} \in \mathcal{X}_k$ such that with probability at least $(1 - k \cdot \delta)$ we have*

$$\sum_{P \in \mathcal{P}} w(P) \cdot \tilde{D}(P, \hat{C}) \leq \alpha \cdot \min_{C \in \mathcal{X}_k} \sum_{P \in \mathcal{P}} w(P) \cdot \tilde{D}(P, C).$$

The previous theorem implies a polynomial time (optimal) solution for the sets-k-means, since it is trivial to compute an optimal solution for the case of $m = k = 1$.

Corollary 4.5 (PTAS for sets-k-means)**.** *Let $\mathcal{P}$ be an (n,m)-set, $k \geq 1$ be an integer, and put $\varepsilon \in \left(0, \frac{1}{2}\right]$ and $\delta \in (0,1)$. Let* OPT *be the cost of the sets-k-means. Then in $n \log(n)(k)^m + \left(\frac{\log n}{\varepsilon} dmk^m\right)^{O(dk)}$ time we can compute $\hat{C} \in \mathcal{X}_k$ such that with probability at least $1 - k \cdot \delta$,*

$$\sum_{P \in \mathcal{P}} \min_{p \in P, c \in \hat{C}} \|p - c\|^2 \leq (1 + 4\varepsilon) \cdot \text{OPT}.$$

5. Robust Median

In this section, we provide an algorithm that computes a robust approximation; see Definition 2.4 and its preceding paragraph. An overview is provided in Section D.

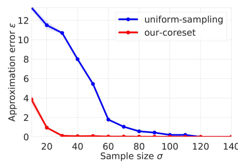
Algorithm 3 MEDIAN$(\mathcal{P}, k, \delta)$

```
1: Input: An $(n,m)$-set $\mathcal{P}$, a positive integer $k \geq 1$, and
          the probability of failure $\delta \in (0,1)$
2: Output: A point $q \in \mathcal{X}$ that satisfies Lemma 5.1
3: $b :=$ a universal constant that can be determined from
   the proof of Lemma 5.1
4: Pick a random sample $\mathcal{S}$ of $|S| = b \cdot k^2 \log\left(\frac{1}{\delta}\right)$ sets
   from $\mathcal{P}$
5: $q :=$ a point that minimizes
   $\sum_{p \in \text{closest}(\mathcal{S}, \{\hat{q}\}, (1-\tau)\gamma)} \tilde{D}(p, \hat{q})$ over $\hat{q} \in Q \in \mathcal{S}$
6: Return $q$
```

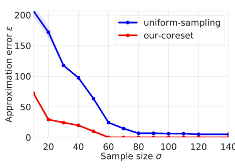
Lemma 5.1 (based on Lemma 9.6 in (Feldman & Langberg, 2011))**.** *Let $\mathcal{P}$ be an (n,m)-set, $k \geq 1$, $\delta \in (0,1)$ and $(\mathcal{X}, \tilde{D})$ be as in Definition 2.1. Let $q \in \mathcal{X}$ be the output of a call to* MEDIAN$(\mathcal{P}, k, \delta)$*; see Algorithm 3. Then with probability at least $1-\delta$, q is a $(1/(2k), 1/6, 2)$-median for P; see Definition 2.4. Furthermore, q can be computed in $O\left(tb^2k^4 \log^2\left(\frac{1}{\delta}\right)\right)$ time, where t is the time it takes to compute $\tilde{D}(P, Q)$ for $P, Q \in \mathcal{P}$.*

6. Experimental Results

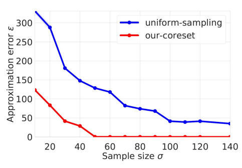
We implemented our coreset construction, as well as different sets-k-mean solvers. In this section we evaluate their empirical performance. Open source code for future research can be downloaded from (Jubran et al., 2020).



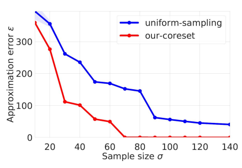
(a) dataset (i), $k = 2$



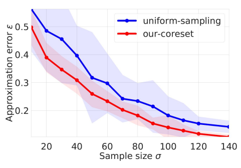
(b) dataset (i), $k = 4$



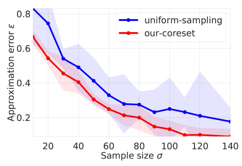
(c) dataset (i), $k = 6$



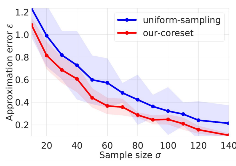
(d) dataset (i), $k = 8$



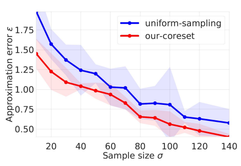
(e) dataset (ii), $k = 4$



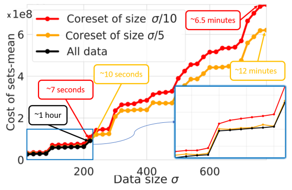
(f) dataset (ii), $k = 6$



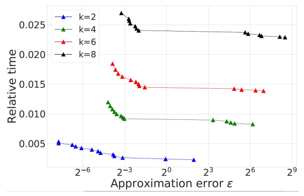
(g) dataset (ii), $k = 8$



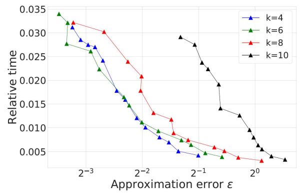
(h) dataset (ii), $k = 10$



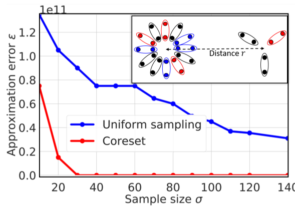
(i) Optimal sets-mean



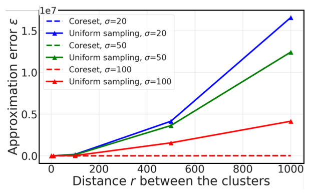
(j) $\mathcal{P}$:= dataset (i)



(k) $\mathcal{P}$:= dataset (ii)



(l) $r = 10^6$



(m)

Figure 4. **Experimental results.** Exact details are provided in Section 6.

Theory-implementation gaps. While Theorem 4.5 suggests a polynomial time solution for sets-k-means in $\mathbb{R}^d$, it is impractical even for $k = 1$, $d = 2$ and $n = 10$ sets. Moreover, its implementation seems extremely complicated and numerically unstable. Instead, we suggest a simple algorithm `exact-mean` for computing the sets-mean; see Fig. 6 in Section E. Its main components are Voronoi diagram (Aurenhammer, 1991) and hyperplanes arrangement that were implemented in Sage (The Sage Developers, 2020). For $k \geq 2$ we use Expectation-Maximization (EM) heuristic, which is a generalization of the well known Lloyd algorithm (Lloyd, 1982), as commonly used in k-means and its variants (Marom & Feldman, 2019; Lucic et al., 2017).

Implementations. Four algorithms were implemented:

(i) `our-coreset`$(\mathcal{P}, \sigma)$: the coreset construction from Algorithm 2 for a given (n, m)-set $\mathcal{P}$, an arbitrary given loss function $\tilde{D}$ that satisfies Definition 2.1 and a sample size of $|\mathcal{S}| = \sigma$ at Line 17 of Algorithm 2.

There is no epsilon and delta in the code. As common in the coreset literature, our actual code gets a number s of points and then samples s input elements (sets) based on the suggested novel distribution, i.e., sensitivities (that are independent of ε, $delta$, and k^{m+4} as in the theoretical bound). We then measure the empirical error epsilon and variance (kind of delta) over larger values of s.

Table 2. **Comparison between sets k-means and the regular k-means.**

Method	k	Data size	Cost
k-means	2	100	303
Sets-k-means	2	100	5
k-means	4	100	274
Sets-k-means	4	100	4
k-means	6	100	237
Sets-k-means	6	100	3
k-means	2	1000	299
Sets-k-means	2	1000	40
k-means	4	1000	268
Sets-k-means	4	1000	33
k-means	6	1000	220
Sets-k-means	6	1000	26

(ii) `uniform-sampling`$(\mathcal{P}, \sigma)$: outputs a uniform random sample $\mathcal{S} \subseteq \mathcal{P}$ of size $|\mathcal{S}| = \sigma$.

(iii) `exact-mean`$(\mathcal{P})$: returns the exact (optimal) sets-mean ($k = 1$) of a given set $\mathcal{P}$ of sets in $\mathbb{R}^d$ as in the previous paragraph.

(iv) `k-means`$(\mathcal{P}, k)$: generalization of the Lloyd k-means heuristic (Lloyd, 1982) that aims to compute the sets-k-mean of $\mathcal{P}$ via EM; see implementation details in Section E.

Software/Hardware. The algorithms were implemented in Python 3.7.3 using Sage 9.0 (The Sage Developers, 2020) as explained above on a Lenovo Z70 laptop with an Intel i7-5500U CPU @ 2.40GHZ and 16GB RAM.

Datasets. (i): The LEHD Origin-Destination Employment Statistics (LODES) (lod). It contains information about people that live and work at the united states. We pick a sample of $n = 10,000$ and their $m = 2$ home+work addresses, called Residence+Workplace Census Block Code. Each address is converted to a pair (x, y) of $d = 2$ doubles. As in Fig. 1, our goal was to compute the sets-k-mean (facilities) of these n pairs of addresses.

(ii): The Reuters-21578 benchmark corpus (Bird et al., 2009). It contains $n = 10,788$ records that corresponds to n Reuters newspapers. Each newspaper is represented as a "bag of bag of words" of its $m \in [3, 15]$ paragraphs in high dimensional-space; see Fig. 5. Handling sets of different sizes is also supported; see details in Section E. We reduce the dimension of the union of these ($3n$ to $15n$) vectors to $d = 15$ using LSA (Landauer et al., 2013). The goal was to cluster those n documents (sets of paragraphs) into k topics; see Fig 5.

(iii): Synthetic dataset. We drew a circle of radius 1, centered at the origin of $\mathbb{R}^2$ and then picked $n_1 = 9900$ points evenly (uniformly) distributed on this circle. For each of these n_1 points, we paired a point in the same direction but of distance 30 from the origin. This resulted in n_1 pairs of points. We repeat this for another circle of radius 1 that is centered at $(r, 0)$, for multiple values of r, and constructed $n_2 = 100$ points similarly; see top of Fig. 4(l).

Experiment (i) We ran $\mathcal{S}_1(\sigma) :=$ `our-coreset`$(\mathcal{P}, \sigma)$ and $\mathcal{S}_2(\sigma) :=$ `uniform-sampling`$(\mathcal{P}, \sigma)$ on each of the datasets for different values of sample size σ. Next, we computed the corresponding sets-k-means $C_1(\sigma), C_2(\sigma)$ and C_3 heuristically using Algorithm (iv). We denote the corresponding computation times in seconds by $t_1(\sigma)$, $t_2(\sigma)$ and t_3, respectively. The corresponding costs of $C_1(\sigma)$ and $C_2(\sigma)$ were evaluated by computing the *approximation error*, for $i \in \{1, 2\}$, as $\varepsilon_i(\sigma) = \frac{\left|\sum_{P\in\mathcal{P}} \tilde{D}(P,C_3) - \sum_{P\in\mathcal{P}} \tilde{D}(P,C_i(\sigma))\right|}{\sum_{P\in\mathcal{P}} \tilde{D}(P,C_3)}$.

Results (i). The approximation errors on the pair of real-world datasets are shown in Fig. 4(a)– 4(h). Fig 4(j)– 4(k) show relative time $t_1(\sigma)/t_3$ (y-axis) as a function of $\varepsilon := \varepsilon_1(\sigma)$ (x-axis), for $\sigma = 20, 30, \ldots, 140$. The approximation errors are shown for the synthetic dataset, either for different increasing σ in Fig. 4(l) or r values in Fig. 4(m).

Experiment (ii). We uniformly sampled 800 rows from the LEHD Dataset (i). Let $\mathcal{P}(\sigma)$ denote the first σ points in this sample, for $\sigma = 20, 40, 60, \ldots, 800$. For each such set $\mathcal{P}(\sigma)$ we computed two different size coresets

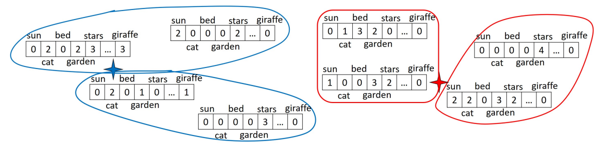


Figure 5. **bag of bag of words**: A set of $n = 4$ documents, each has $m = 2$ paragraphs. Each paragraph is represented by a bag of words and each document is represented by the set of m vectors of its paragraphs. The sets-2-means are presented (blue / red stars).

$\mathcal{S}_1(\sigma) :=$ `our-coreset`$(\mathcal{P}(\sigma), \sigma/10)$ and $\mathcal{S}_2(\sigma) :=$ `our-coreset`$(\mathcal{P}(\sigma), \sigma/5)$. We then applied Algorithm (iii) that computes the optimal sets-mean $C_1(\sigma)$, $C_2(\sigma)$ and $C_3(\sigma)$ on $\mathcal{S}_1(\sigma)$, $\mathcal{S}_2(\sigma)$ and the full data $\mathcal{P}(\sigma)$, respectively.

Results (ii). Fig 4(i) shows the cost of $C_i(\sigma)$ (y-axis) as a function of σ (x-axis), for $i \in \{1, 2, 3\}$.

Experiment (iii). This experiment aims to compare the sets-k-means to the known k-means. The input is a subset P from the Dataset (ii) (once of size 100, and other of size 1000). The goal is to compute a solution (k-centers) for the sets-k-means problem. We compare two methods: (i) computing the sets-k-means solution as explained in section 6, and (ii) computing the k-means solution on the set $P' = \left\{p \in P_i \middle| P_i \in P\right\}$. The cost is the cost of the sets-k-means problem on the computed k-centers and the original input data P.

Results (iii). The results are presented in Table 2.

Discussion. As common in the coreset literature, we see that the approximation errors are significantly smaller than the pessimistic worst-case bounds. In all the experiments the coreset yields smaller error compared to uniform sampling. When running exact algorithms on the coreset, the error is close to zero while the running time is reduced from hours to seconds as shown in Fig 4(i). The running time is faster by a factor of tens to hundreds using the coresets, in the price of an error between $1/64$ to $1/2$ as shown in Fig. 4(j)–4(k). Furthermore, Table 2 demonstrates the difference between the sets-k-means and the k-means problems for the task of clustering input sets.

7. Conclusions and Open Problems

This paper suggests coresets and near-linear time solutions for clustering of input sets such as the sets-k-means. Natural open problems include relaxation to convex optimization, handling other distance functions between sets e.g. max distance, handling infinite sets / shapes (triangles, circles, etc.) and continuous distributions (e.g. n Gaussians). We hope that this paper is only the first step toward a long line of research that include solutions to the above problems.

Acknowledgements

This work is an outcome of a joint work with Samsung Research Israel. We would like to thank Matan Weksler and Benjamin Lastmann for their support and contribution to this paper.

References

U.s. census bureau. longitudinal employer-household dynamics. URL https://lehd.ces.census.gov/data/lodes/LODES7/.

Abboud, A., Cohen-Addad, V., and Houdrougé, H. Subquadratic high-dimensional hierarchical clustering. In *Advances in Neural Information Processing Systems*, pp. 11576–11586, 2019.

Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

Ahmadian, S., Friggstad, Z., and Swamy, C. Local-search based approximation algorithms for mobile facility location problems. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1607–1621. SIAM, 2013.

Anthony, M. and Bartlett, P. L. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

Aurenhammer, F. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

Bachem, O., Lucic, M., and Krause, A. Coresets for nonparametric estimation-the case of dp-means. In *ICML*, pp. 209–217, 2015.

Bachem, O., Lucic, M., and Krause, A. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017a.

Bachem, O., Lucic, M., and Lattanzi, S. One-shot coresets: The case of k-clustering. *arXiv preprint arXiv:1711.09649*, 2017b.

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

Barger, A. and Feldman, D. k-means for streaming and distributed big sparse data. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 342–350. SIAM, 2016.

Berkhin, P. Survey of clustering data mining techniques: Technical report. In *Accrue software*. 2002.

Bird, S., Loper, E., and Klein, E. *Natural Language Processing with Python*. O Reilly Media Inc, 2009.

Blelloch, G. E. and Tangwongsan, K. Parallel approximation algorithms for facility-location problems. In *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*, pp. 315–324, 2010.

Braverman, V., Feldman, D., and Lang, H. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889*, 2016.

Chazelle, B., Edelsbrunner, H., Guibas, L. J., and Sharir, M. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoretical Computer Science*, 84(1):77–105, 1991.

Chen, K. On k-median clustering in high dimensions. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pp. 1177–1185, 2006.

Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

Cohen-Addad, V., Hjuler, N. O. D., Parotsidis, N., Saulpic, D., and Schwiegelshohn, C. Fully dynamic consistent facility location. In *Advances in Neural Information Processing Systems*, pp. 3250–3260, 2019.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

Dunia, R., Qin, S. J., Edgar, T. F., and McAvoy, T. J. Identification of faulty sensors using principal component analysis. *AIChE Journal*, 42(10):2797–2812, 1996.

Feldman, D. Core-sets: Updated survey. In *Sampling Techniques for Supervised or Unsupervised Tasks*, pp. 23–44. Springer, 2020.

Feldman, D. and Langberg, M. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 569–578. ACM, 2011.

Feldman, D. and Schulman, L. J. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pp. 1343–1354. SIAM, 2012.

Feldman, D., Xiang, C., Zhu, R., and Rus, D. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 3–16. IEEE, 2017.

Fichtenberger, H., Gillé, M., Schmidt, M., Schwiegelshohn, C., and Sohler, C. Bico: Birch meets coresets for k-means clustering. In *European Symposium on Algorithms*, pp. 481–492. Springer, 2013.

Frahling, G. and Sohler, C. A fast k-means implementation using coresets. *International Journal of Computational Geometry & Applications*, 18(06):605–625, 2008.

Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 291–300, 2004.

Hartigan, J. A. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

Huang, L., Jiang, S., Li, J., and Wu, X. Epsilon-coresets for clustering (with outliers) in doubling metrics. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 814–825. IEEE, 2018.

Huggins, J. H., Campbell, T., and Broderick, T. Coresets for scalable bayesian logistic regression. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 4087–4095, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Inaba, M., Katoh, N., and Imai, H. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the tenth annual symposium on Computational geometry*, pp. 332–339, 1994.

Jubran, I., Maalouf, A., and Feldman, D. Introduction to coresets: Accurate coresets. *arXiv preprint arXiv:1910.08707*, 2019.

Jubran, I., Tukan, M., Maalouf, A., and Feldman, D. Open source code for all the algorithms presented in this paper, 2020. Link for open-source code.

Landauer, T. K., McNamara, D. S., Dennis, S., and Kintsch, W. *Handbook of latent semantic analysis*. Psychology Press, 2013.

Langberg, M. and Schulman, L. J. Universal ε-approximators for integrals. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pp. 598–607. SIAM, 2010.

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

Li, J., Wu, W., Wang, T., and Zhang, Y. One step beyond histograms: Image representation using markov stationary features. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE, 2008.

Li, L.-J., Su, H., Fei-Fei, L., and Xing, E. P. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Advances in neural information processing systems*, pp. 1378–1386, 2010.

Liao, L., Fox, D., and Kautz, H. Location-based activity recognition. In *Advances in Neural Information Processing Systems*, pp. 787–794, 2006.

Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Lucic, M., Faulkner, M., Krause, A., and Feldman, D. Training gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

Maalouf, A., Jubran, I., and Feldman, D. Fast and accurate least-mean-squares solvers. In *Advances in Neural Information Processing Systems*, pp. 8305–8316, 2019a.

Maalouf, A., Statman, A., and Feldman, D. Tight sensitivity bounds for smaller coresets. *arXiv preprint arXiv:1907.01433*, 2019b.

Marom, Y. and Feldman, D. k-means clustering of lines for big data. In *Advances in Neural Information Processing Systems*, pp. 12797–12806, 2019.

Mladenic, D. Text-learning and related intelligent agents: a survey. *IEEE intelligent systems and their applications*, 14(4):44–54, 1999.

Munteanu, A., Sohler, C., and Feldman, D. Smallest enclosing ball for probabilistic data. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pp. 214–223, 2014.

Munteanu, A., Schwiegelshohn, C., Sohler, C., and Woodruff, D. On coresets for logistic regression. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6561–6570. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7891-on-coresets-for-logistic-regression.pdf`.

Murtagh, F. A survey of recent advances in hierarchical clustering algorithms. *The computer journal*, 26(4):354–359, 1983.

Nguyen, N. P., Dinh, T. N., Xuan, Y., and Thai, M. T. Adaptive algorithms for detecting community structure in dynamic social networks. In *2011 Proceedings IEEE INFOCOM*, pp. 2282–2290. IEEE, 2011.

Phillips, J. M. Coresets and sketches. *arXiv preprint arXiv:1601.00617*, 2016.

Régin, J.-C., Rezgui, M., and Malapert, A. Embarrassingly parallel search. In *International Conference on Principles and Practice of Constraint Programming*, pp. 596–610. Springer, 2013.

Schmidt, M., Schwiegelshohn, C., and Sohler, C. Fair coresets and streaming algorithms for fair k-means. In *International Workshop on Approximation and Online Algorithms*, pp. 232–251. Springer, 2019.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Spanakis, G., Siolas, G., and Stafylopatis, A. Exploiting wikipedia knowledge for conceptual hierarchical clustering of documents. *The Computer Journal*, 55(3):299–312, 2012.

Srivastava, A., Joshi, S. H., Mio, W., and Liu, X. Statistical shape analysis: Clustering, learning, and testing. *IEEE Transactions on pattern analysis and machine intelligence*, 27(4):590–602, 2005.

Suciu, D., Olteanu, D., Ré, C., and Koch, C. Probabilistic databases. *Synthesis lectures on data management*, 3(2): 1–180, 2011.

The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.0)*, 2020. `https://www.sagemath.org`.

Toth, C. D., O'Rourke, J., and Goodman, J. E. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.

Tuytelaars, T., Mikolajczyk, K., et al. Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280, 2008.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

Xiao, X.-Y., Peng, W.-C., Hung, C.-C., and Lee, W.-C. Using sensorranks for in-network detection of faulty readings in wireless sensor networks. In *Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access*, pp. 1–8, 2007.