sensors

MDPI

Article

Coresets for the Average Case Error for Finite Query Sets

Alaa Maalouf *, Ibrahim Jubran, Murad Tukan and Dan Feldman

Robotics & Big Data Labs, University of Haifa, Haifa 3498838, Israel; ijubran@staff.haifa.ac.il (I.J.); mtukan@campus.haifa.ac.il (M.T.); dfeldman@univ.haifa.ac.il (D.F.)

* Correspondence: amaalouf@campus.haifa.ac.il; Tel.: +972-52-5103-817

Abstract: Coreset is usually a small weighted subset of an input set of items, that provably approximates their loss function for a given set of queries (models, classifiers, hypothesis). That is, the maximum (worst-case) error over all queries is bounded. To obtain smaller coresets, we suggest a natural relaxation: coresets whose average error over the given set of queries is bounded. We provide both deterministic and randomized (generic) algorithms for computing such a coreset for any finite set of queries. Unlike most corresponding coresets for the worst-case error, the size of the coreset in this work is independent of both the input size and its Vapnik–Chervonenkis (VC) dimension. The main technique is to reduce the average-case coreset into the vector summarization problem, where the goal is to compute a weighted subset of the n input vectors which approximates their sum. We then suggest the first algorithm for computing this weighted subset in time that is linear in the input size, for $n \gg 1/\varepsilon$, where ε is the approximation error, improving, e.g., both [ICML'17] and applications for principal component analysis (PCA) [NIPS'16]. Experimental results show significant and consistent improvement also in practice. Open source code is provided.

Keywords: coreset; average case analysis; big data; sparsification; dimensionality reduction; approximation algorithms

Citation: Maalouf, A.; Jubran, I.; Tukan, M.; Feldman, D. Coresets for the Average Case Error for Finite Query Sets. *Sensors* **2021**, *21*, 6689. https://doi.org/10.3390/s21196689

Academic Editor: Rebeca P. Díaz Redondo

Received: 7 September 2021
Accepted: 30 September 2021
Published: 8 October 2021

1. Introduction

In this paper, we assume that the input is a set P of items, called points. Usually, P is simply a finite set of n points in $\mathbb{R}^d$ or other metric space. In the context of PAC (probably approximately correct) learning [1], or empirical risk minimization [2] it represents the training set. In supervised learning every point in P may also include its label or class. We also assume a given function $w : P \to (0, \infty)$ called weights function that assigns a "weight" $w(p) > 0$ for every point $p \in P$. The weights function represents a distribution of importance over the input points, where the natural choice is uniform distribution, i.e., $w(p) = 1/|P|$ for every $p \in P$. We are also given a (possibly infinite) set X that is the set of queries [3] which represents candidate models or hypothesis, e.g., neural networks [4], SVMs [5] or a set of vectors in $\mathbb{R}^d$ with tuning parameters as in linear/ridge/lasso regression [6–8].

In machine learning and PAC-learning in particular, we often seek to compute the query that best describes our input data P for either prediction, classification, or clustering tasks. To this end, we define a loss function $f : P \times X \to \mathbb{R}$ that assigns a fitting cost $f(p, x)$ to every point $p \in P$ with respect to a query $x \in X$. For example, it may be a kernel function [9], a convex function [10], or an inner product. The tuple (P, w, X, f) is called a query space and represents the input to our problem. In this paper, we wish to approximate the weighted sum of losses $f_w(P, x) = \sum_{p \in P} w(p) f(p, x)$.

Methodology.

Our main tool for approximating the sum of losses above is called a coreset, which is a (small) weighted subset of the (potentially huge) input data, from which the desired sum of losses can be recovered in a very fast time, with guarantees on the small induced error;

see Section 1.2. To compute such a coreset, we utilize the famous Frank–Wolfe algorithm. However, to compute a coreset in fast time, we first provide a scheme for provably boosting the running time of the Frank–Wolfe algorithm for our special case, without compromising its output accuracy; see Section 3.1. We then utilize the boosted version in order to compute a deterministic coreset in time faster than the state of the art.

1.1. Approximation Techniques for the Sum of Losses

Approximating the loss of a single query via uniform sampling. Suppose that we wish to approximate the mean $f(P,x) = \frac{1}{n}\sum_{p\in P} f(p,x)$ for a specific x in sub-linear time. Picking a random point p uniformly at random from P would give this result in expectation as $E[f(p,x)] = \sum_{p\in P} f(p,x)/n = f(P,x)$. By Hoeffding inequality, the mean $f(S,x) = \frac{1}{|S|}\sum_{p\in S} f(p,x)$ of a uniform sample $S \subseteq P$ would approximate $f(P,x)$ with high probability. More precisely, for a given $\varepsilon \in (0,1)$, if the size of the sample is $|S| \in O(M/\varepsilon^2)$ where $M = \max_{p\in P} |f(p,x)|$ is the maximum absolute value of f, then with constant probability our approximation error is

$$err_f(x) = |f(P,x) - f(S,x)| \leq \varepsilon.$$

ε-Sample. Generally, we are interested in such data summarization S of P that approximates every query $x \in X$. An ε-sample is a pair (S,u) where S is a subset of P (unlike, e.g., sketches [11]), and $u : S \to [0,\infty)$ is its weights function such that the weighted loss $f_u(S,x) = \sum_{p\in S} u(p)f(p,x)$ of the (hopefully small) weighted subset S approximates the original weighted loss $f_w(P,x)$ [12], i.e.,

$$\forall x \in X : |f_w(P,x) - f_u(S,x)| \leq \varepsilon. \tag{1}$$

We usually assume that the input is normalized in the sense that w is a distribution, and $f : P \times X \to [-1,1]$. By defining the pair of vectors $f_w(P,X) = (\sum_{p\in P} w(p)f(p,x))_{x\in X}$ and $f_u(S,X) = (\sum_{p\in S} u(p)f(p,x))_{x\in X}$, we can define the error for a single x by $err(x) = |f_w(P,x) - f_u(S,x)|$, and then the error vector for the coreset $err(X) = (err(x))_{x\in X}$. We can rewrite (1) by

$$\|err(X)\|_\infty = \|f_w(P,X) - f_u(S,X)\|_\infty \leq \varepsilon. \tag{2}$$

PAC/DAC learning for approximating the sum of losses for multiple queries. Probably approximately correct (PAC) randomized constructions generalizes the Hoeffding inequality above from a single to multiple (usually infinite) queries and returns an ε-sample for a given query space (P,w,X,f) and $\delta \in (0,1)$, with probability at least $1-\delta$. Here, δ corresponds to the "probably" part, while "approximately correct" corresponds to ε in (2); see [13,14]. Deterministic approximately correct (DAC) versions of PAC-learning suggest deterministic construction of ε-samples, i.e., the probability of failure of the construction is $\delta = 0$.

As common in machine learning and computer science in general, the main advantage of deterministic constructions is smaller bounds (in this case, on the size of the resulting ε-sample), and their disadvantage is usually the slower construction time that may be unavoidable. When the query set X if finite, the Caratheodory theorem [15,16] suggests a deterministic algorithm that returns a 0-sample (S,u) (i.e., $f_w(P,X) = f_u(S,X)$) of size $|S| \leq |X|+1$. Deterministic constructions of ε-sample are known for infinite sets of queries even when the VC-dimension is unbounded [17,18].

Sup-sampling: reducing the sample size via non-uniform sampling. As explained above, Hoeffding inequality implies an approximation of $f_w(P,x)$ by $f_u(S,x)$ where $u(p) = 1/|S|$ and S is a random sample according to w whose size depends on $M(f) = \max_{p\in P} |f(p,x)|$. To reduce the sample size we may thus define $g(p,x) = \frac{f(p,x)}{|f(p,x)|} \in \{-1,1\}$, and $s(p) = \frac{w(p)|f(p,x)|}{\sum_{q\in P} w(q)|f(q,x)|}$. Now, $M(g) = \max_{p\in P} |g(p,x)| = 1$, and by Hoeffding's inequality, the error of approximating $g_s(P,x)$ via non-uniform random sample of size $1/\varepsilon^2$

drawn from s, is ε. Define $T = \sum_{q \in P} w(q)|f(q,x)|$. Since $f_w(P,x) = T \cdot g_s(P,x)$, approximating $g_s(P,x)$ up to ε error yields an error of εT for $f_w(p,x)$. Therefore, the size is reduced from $M(f)/\varepsilon^2$ to T^2/ε^2 when $T^2 \leq M(f)$. Here, we sample $|S| = O(T^2/\varepsilon^2)$ points from P according to the distribution s, and re-weight the sampled points by $u'(p) = \frac{T}{|S||f(p,x)|}$.

Unlike traditional PAC-learning, the sample now is non-uniform, and is proportional to $s(p)$, rather than w, as implied by Hoeffding inequality for non-uniform distributions. For sets of queries we generalize the definition for every $p \in P$ to $s(p) = \sup_{x \in X} \frac{w(p)|f(p,x)|}{\sum_{q \in P} w(q)|f(p,x)|}$ as in [19], which is especially useful for coresets below.

1.2. Coresets: A Data Summarization Technique for Approximating the Sum of Losses

Coreset for a given query space (P, w, X, g), in this and many other papers, is a pair (C, u) that is similar to ε-sample in the sense that $C \subseteq P$ and $u : C \to [0, \infty)$ is a weights function. However, the additive error ε is now replaced with a multiplicative error $1 \pm \varepsilon$, i.e., for every $x \in X$ we require that, $|g_w(P,X) - g_u(C,X)| \leq \varepsilon \cdot g_w(P,X)$. Dividing by $g_w(P,X)$ and assuming $g_w(P,X) > 0$, yields

$$\forall x \in X : \left| 1 - \frac{g_u(C,X)}{g_w(P,x)} \right| \leq \varepsilon. \tag{3}$$

Coresets are especially useful for learning big data since an off-line and possibly inefficient coreset construction for "small data" implies constructions that maintains coreset for streaming, dynamic (including deletions) and distributed data in parallel. This is via a simple and easy to implement framework that is sometimes called merge–reduce trees; see [20,21]. The fact that a coreset approximates every query (and not just the optimal one for some criterion) implies that we may solve hard optimization problems with non-trivial and non-convex constraints by running a possibly inefficient algorithm such as exhaustive search on the coreset, or running existing heuristics numerous times on the small coreset instead of once on the original data. Similarly, parameter tuning or cross validation can be applied on a coreset that is computed once for the original data as explained in [22].

An ε-coreset for a query space (P, w, X, g) is simply an ε-sample for the query space (P, w, X, f), after defining $f(p,x) := \frac{g(p,x)}{g_w(P,x)}$, as explained, e.g., in [19]. By defining the error for a single x by $err'(x) = |1 - g_u(C,x)/g_w(P,x)| = |f_w(P,x) - f_u(C,x)| = err_f(x)$, we obtain an error vector for the coreset $err'(X) = (err'(x))_{x \in X}$. We can then rewrite (3) as in (2):

$$\left\| err'(X) \right\|_\infty \leq \varepsilon.$$

In the case of coresets, the $\sup_{x \in X} w(p) f(p,x) = \sup_{x \in X} \frac{w(p) g(p,x)}{g_w(P,x)}$ of a point $p \in P$ is called sensitivity [14], leverage score (in ℓ_2 approximations) [23], Lewis weights (in ℓ_p approximations), or simply importance [24].

1.3. Problem Statement: Average Case Analysis for Data Summarization.

Average case analysis (e.g., [25]) was suggested about a decade ago as an alternative to the (sometimes infamous) worst-case analysis of algorithms in theoretical computer science. The idea is to replace the analysis for the worst-case input by the average input (in some sense). Inspired by this idea, a natural variant of (2) and its above implications is an ε-sample that approximates well the average query. We suggest to define an $(\varepsilon, \|\cdot\|)$-sample as

$$\|err(X)\| = \|f_w(P,X) - f_u(S,X)\| \leq \varepsilon, \tag{4}$$

which generalizes (2) from $\|\cdot\|_\infty$ to any norm, such as the ℓ_z norm $\|err(X)\|_z$. For example, for the ℓ_2, MSE or Frobenius norm, we obtain

$$\sqrt{\sum_{x\in X}\left(f_w(P,x)-f_u(S,x)\right)^2}\leq\varepsilon. \tag{5}$$

A generalization of the Hoeffding Inequality from 1963 with tight bounds was suggested relatively recently for the ℓ_z norm for any $z\geq 2$ and many other norms [26,27]. Here we assume a single query ($|X|=1$), a distribution weights function, and a bound on $\sup_{p\in P}|f(p,X)|$ that determines the size of the sample, as in Hoeffding inequality.

A less obvious question, which is the subject of this paper, is how to compute deterministic ε-samples that satisfies (4), for norms other than the infinity norm. While the Caratheodory theorem suggests deterministic constructions of 0-samples (for any error norm) as explained above, our goal is to obtain coreset whose size is smaller or independent of $|X|$.

The next question is how to generalize the idea of sup-sampling, i.e., where the function f is unbounded, for the case of norms other than $\|\cdot\|_\infty$. Our main motivation for doing so is to obtain new and smaller coresets by combining the notion of ε-sample and sup-sampling or sensitivity as explained above for the $\|\cdot\|_\infty$ case. That is, we wish a coreset for a given query space, that would bound the non-ℓ_∞ norm error

$$\left\|\left(1-\frac{g_u(C,x)}{g_w(P,x)}\right)_{x\in X}\right\|=\|err'(X)\|\leq\varepsilon.$$

To summarize, our questions are: **How can we smooth the error function and approximate the "average" query via: (i) Deterministic ε-samples (for DAC-learning)? (ii) Coresets (via sensitivities/sup sampling for non-infinity norms)?**

1.4. Our Contribution

We answer affirmably these questions by suggesting ε-samples and coresets for the average query. We focus on the case $z=2$, i.e., the Frobenius norm, and finite query set X and hope that this would inspire the research and applications of other norms and general sets. For suggestions in this direction and future work see Section 5.2. The main results of this paper are the following constructions of an $(\varepsilon,\|\cdot\|_2)$-sample (S,u) for any given finite query space (P,w,X,f) as defined in (5):

(i) Deterministic construction that returns a coreset of size $|S|\in O(1/\varepsilon^2)$ in time $O\left(\min\left\{nd/\varepsilon^2, nd+d\log(n)^2/\varepsilon^4\right\}\right)$; see Theorem 2 and Corollary 4.

(ii) Randomized construction that returns such a coreset (of size $|S|\in O(1/\varepsilon^2)$) with probability at least $1-\delta$ in sub-linear time $O\left(d\left(\log\left(\frac{1}{\delta}\right)^2+\frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2}\right)\right)$; see Lemma 5.

Algorithm. This result is of independent interest for faster and sparser convex optimization. To our knowledge, this is also the first application of sensitivity outside the coreset regime.

1.5. Overview and Organization

The rest of the paper in organized as follows. First in Section 2, we list the applications of our proposed methods, such as a faster coreset construction algorithm for least mean squares solver. We also compare our results to the state of the art to justify our practical contribution.

In Section 3, we first give our notations and relevant mathematical definitions, we explain the relation between the problem of computing an $(\varepsilon,\|\cdot\|_2)$-sample (average-case coreset) to the problem of computing a vector summarization coreset, where the goal (of the vector summarization coreset problem) is to compute a weighted subset of the n input vectors which approximates their sum. Here, we suggest a coreset for this problem of size $O(1/\varepsilon)$ in $O(nd/\varepsilon)$ time; see Theorem 2 and Algorithm 2. Then, in Section 3.1 we show

how to improve the running time of this result and compute a coreset of the same size in $O(nd + d\log^2(n)/\varepsilon^2)$ time; see Corollary 4 and Algorithm 3. In addition, we suggest a non-deterministic coreset of the same size but in time that is independent of the number of points n; see Lemma 5 and Algorithm 4.

In Section 4, we explain how our vector summarization coreset results can be used to improve all the previously mentioned applications (from Section 2). In Section 5 we conduct various experiments on real world datasets, where we apply different coreset construction algorithms presented in this paper to a variety of applications, in order to boost their running time, or reduce their memory storage. We also compare our results to many competing methods. Finally, we conclude our paper and discuss future work at Section 5.2. Due to space limitations and simplicity of the reading, the proofs of the claims are placed in the Appendixes A–I.

2. On the Applications of Our Method and the Current State of the Art

In what follows, we will present some of the applications of our theoretical contributions as well as discussing the current state of the art coreset/sketch methods in terms of running time for each of application. Figure 1 summarizes the main applications of our result.

(i) **Vector summarization:** the goal is to maintain the sum of a (possibly infinite) stream of vectors in $\mathbb{R}^d$, up to an additive error of ε multiplied by their variance. This is a generalization of frequent items/directions [28].
As explained in [29], the main real-world application is extractions and compactly representing groups and activity summaries of users from underlying data exchanges. For example, GPS traces in mobile networks can be exploited to identify meetings, and exchanges of information in social networks sheds light on the formation of groups of friends. Our algorithm tackles these application by providing provable solution to the heavy hitters problem in proximity matrices. The heavy hitters problem can be used to extract and represent in a compact way friend groups and activity summaries of users from underlying data exchanges.
We propose a deterministic algorithm which reduces each subset of n vectors into $O(1/\varepsilon)$ weighted vectors in $O(nd + d\log(n)^2/\varepsilon^2)$ time, improving upon the nd/ε of [29] (which is the current state of the art in terms of running time), for a sufficiently large n; see Corollary 4, and Figures 2 and 3. We also provide a non-deterministic coreset construction in Lemma 5. The merge-and-reduce tree can then be used to support streaming, distributed or dynamic data.

(ii) **Kernel Density Estimates (KDE):** by replacing ε with ε^2 for the vector summarization, we obtain fast construction of an ε-coreset for KDE of Euclidean kernels [17]; see more details in Section 4. Kernel density estimate is a technique for estimating a probability density function (continuous distribution) from a finite set of points to better analyse the studied probability distribution than when using a traditional [30,31].

(iii) **1-mean problem:** a coreset for 1-mean which approximates the sum of squared distances over a set of n points to any given center (point) in $\mathbb{R}^d$. This problem arises in facility location problems (e.g., to compute the optimal location for placing an antenna such that all the customers are satisfied). Our deterministic construction computes such a weighted subset of size $O(1/\varepsilon^2)$ in $O(\min\{nd/\varepsilon^2, nd + d\log(n)^2/\varepsilon^4\})$ time. Previous results of [19,32–34] suggested coresets for such problem. Unlike our results, these works are either non-deterministic, the coreset is not a subset of the input, or the size of the coreset is linear in d.

(iv) **Coreset for LMS solvers and dimensionality reduction:** for example, a deterministic construction for singular value decomposition (SVD) that gets a matrix $A \in \mathbb{R}^{n\times d}$ and returns a weighted subset of k^2/ε^2 rows, such that their weighted distance to any k-dimensional non-affine (or affine in the case of PCA) subspace approximates the distance of the original points to this subspace. The SVD and PCA are very common algorithms (see [35]), and can be used for noise reduction, data visualization, cluster

analysis, or as an intermediate step to facilitate other analyses. Thus, improving them might be helpful for a wide range of real-world applications. In this paper, we propose a deterministic coreset construction that takes $O(nd^2 + d^2k^4\log(n)^2/\varepsilon^4)$ time, improving upon the state of the art result of [35] which requires $O(nd^2k^2/\varepsilon^2)$ time; see Table 1. Many non-deterministic coresets constructions were suggested for those problems, the construction techniques apply non-uniform sampling [36–38], Monte-Carlo sampling [39], and leverage score sampling [23,40–45].

Table 1. Known deterministic subset coresets for LMS solvers. Our result has the fastest running time for sufficiently large n and d.

Error	Size	Time	Citation	Notes
ε	$O(k^2/\varepsilon^2)$	$O(nd^2k^2/\varepsilon^2)$	[35]	N/A
ε	$O(d/\varepsilon^2)$	$poly(n,d,\varepsilon)$	[46]	inefficient for large n
0	$O(d^2)$	$O(nd^2 + \log(n)poly(d))$	[22]	inefficient for large d
ε	$O(k/\varepsilon^2)$	$poly(n,d,k,\varepsilon)$	[47]	inefficient for large n
ε	$O(k^2/\varepsilon^2)$	$O(nd^2 + \log(n)^2d^2k^4/\varepsilon^4)$	$\star$	N/A

3. Vector Summarization Coreset

Notation 1. *We denote by* $[n] = \{1,\cdots,n\}$. *For a vector* $v \in \mathbb{R}^d$, *the* 0*-norm is denoted by* $\|v\|_0$ *and is equal to the number of non-zero entries in* v. *We denote by* $e_{(i)}$ *the* i*th standard basis vector in* $\mathbb{R}^n$ *and by* $\mathbf{0}$ *the vector* $(0,\cdots,0)^T \in \mathbb{R}^n$. *A vector* $w \in [0,1]^n$ *is called a distribution vector if all its entries are non-negative and sums up to one. For a matrix* $A \in \mathbb{R}^{m\times n}$ *and* $i \in [m], j \in [n]$ *we denote by* $A_{i,j}$ *the* j*th entry of the* i*th row of* A. *A weighted set is a pair* (Q,m) *where* $Q = \{q_1,\cdots,q_n\} \subseteq \mathbb{R}^d$ *is a set of* n *points, and* $m = (m_1,\cdots,m_n)^T \in \mathbb{R}^n$ *is a weights vector that assigns every* $q_i \in Q$ *a weight* $m_i \in \mathbb{R}$. *A matrix* $A \in \mathbb{R}^{d'\times d}$ *is orthogonal if* $A^TA = I \in \mathbb{R}^{d\times d}$.

Adaptations. To adapt to the notation of the following sections and the query space (P,w,X,f) to the techniques that we use, we restate (4) as follows. Previously, we denote the queries $X = \{x_1,\cdots,x_d\}$, and the input set by $P = \{p_1,\cdots,p_n\}$. Now, each input point p_i in the input set P corresponds to a point $q_i = \big(f(p_i,x_1),\cdots,f(p_i,x_d)\big) \in \mathbb{R}^d$, i.e., each entry of q_i equals to $f(p_i,x)$ for a different query x. Throughout the rest of the paper, for technical reason and simplicity, we might alternate between the weights function notation and a weights vector notation. In such cases, the weights function $w : P \to [0,\infty)$ and weight $w(q_i)$ of q_i, $i \in [m]$ are replaced by a vector of weights $m = (m_1,\cdots,m_n) \in [0,\infty)^n$ and m_i, respectively, and vice versa. In such cases, the ε-sample is represented by a sparse vector $u \in [0,\infty)$ where $S = \{p_i \in P \mid u_i > 0, i \in [n]\}$ is the chosen subset of P.

Hence, $f_w(P,X) = \sum_{p\in P} w(p)\big(f(p,x_1),\cdots,f(p,x_d)\big) = \sum_{i=1}^n m_iq_i$, and $f_u(S,X) = \sum_{i=1}^n u_iq_i$.

From $(\varepsilon,\|\cdot\|_2)$-samples to ε-coresets. We now define an ε-coreset for vector summarization, which is a re-weighting of the input weighted set (Q,m) by a new weights vector u, such that the squared norm of the difference between the weighted means of (Q,u) and (Q,m) is small. This relates to Section 1.3, where an $(\sqrt{\varepsilon},\|\cdot\|_2)$-sample there (in Section 1.3) is an ε-coreset for the vector summarization here.

Definition 1 (vector summarization ε-coreset). *Let* (Q,m) *and* (Q,u) *be two weighted sets of* n *points in* $\mathbb{R}^d$, *and let* $\varepsilon \in [0,1)$. *Let* $\mu = \sum_{i=1}^n \frac{m_i}{\|m\|_1}q_i$, $\sigma^2 = \sum_{i=1}^n \frac{m_i}{\|m\|_1}\|q_i-\mu\|^2$, *and* $\tilde{\mu} = \sum_{i=1}^n \frac{u_i}{\|u\|_1}q_i$. *Then* (Q,u) *is a vector summarization* ε*-coreset for* (Q,m) *if* $\|\tilde{\mu}-\mu\|_2^2 \le \varepsilon\sigma^2$.

Analysis flow. In what follows we (first) assume that the points of our input set P lie inside the unit ball ($\forall_{p\in P} : \|p\| \le 1$). For such an input set, we present a construction of a

variant of a vector summarization coreset, where the error is ε and does not depend on the variance of the input. This construction is based on the Frank–Wolfe algorithm [48]; see Theorem 1 and Algorithm 1. This is by reducing the problem to the problem of maximizing a concave function $f(x)$ over every vector in the unit simplex. Such problems can be solved approximately by a simple greedy algorithm known as the Frank–Wolfe algorithm.

Algorithm 1: FRANK-WOLFE(f, K); Algorithm 1.1 of [48]

Input: A concave function $f : \mathbb{R}^n \to \mathbb{R}$, and the number of iterations K.
Output: A vector $x \in \mathbb{R}^n$ that satisfies Theorem 1.
$x_{(0)} :=$ the unit n-simplex vertex with largest f value.
for $k \in \{0, \cdots, K\}$ **do**
  $i' := \arg\max_i \nabla f(x_{(k)})_i$
  $\alpha' := \arg\max_{\alpha_{[0,1]}} f\left(x_{(k)} + \alpha(e_{(i')} - x_{(k)})\right)$
  $x_{(k+1)} := x_{(k)} + \alpha'(e_{(i')} - x_{(k)})$
end for
Return $x_{(k+1)}$

We then present a proper coreset construction in Algorithm 2 and Theorem 2 for a general input set Q in $\mathbb{R}^d$. This algorithm is based on a reduction to the simpler case of points inside the unit ball; see Figure 1 for illustration. This reduction is inspired by the sup-sampling (see Section 1), there (in Section 1) the functions are normalized (to obtain values in $[-1, 1]$) and reweighted (to obtain a non-biased estimator), then the bounds were easily obtained using the Hoeffding inequality. Here, we apply different normalizations and reweightings, and instead of the non-deterministic Hoeffding inequality, we suggest a deterministic version using the Frank–Wolfe algorithm. Our new suggested normalizations (and reweightings) allow us to generalize the result to many more applications as in Section 4.

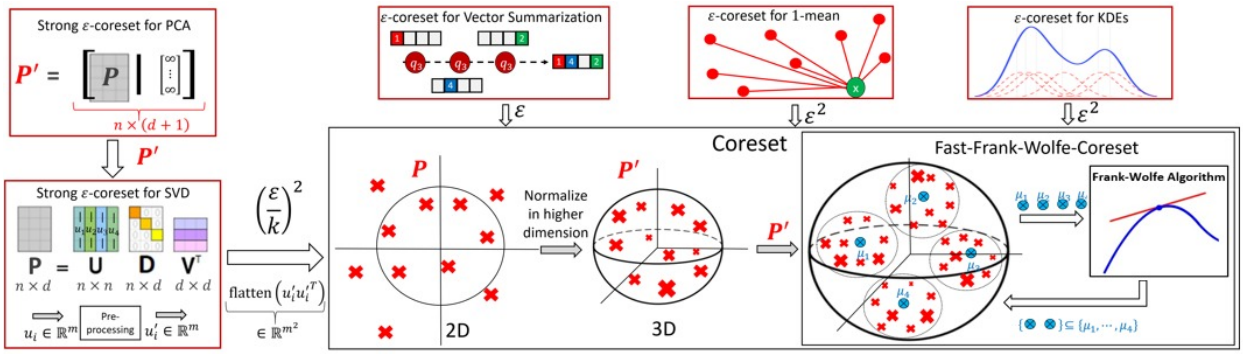


Figure 1. Illustration of Algorithm 2, its normalization of the input, its main applications (red boxes) and their plugged parameters. Algorithm 2 utilizes and boosts the run-time of the Frank–Wolfe algorithm for those applications; see Section 1.4.

For brevity purposes, all proofs of the technical results can be found at the Appendixes A–I.

Theorem 1 (Coreset for points in the unit ball). *Let $P = \{p_1, \cdots, p_n\}$ be a set of n points in $\mathbb{R}^d$ such that $\|p_i\| \leq 1$ for every $i \in [n]$. Let $\varepsilon \in (0, 1)$ and $w = (w_1, \cdots, w_n)^T \in [0, 1]^n$ be a distribution vector. For every $x = (x_1, \cdots, x_n)^T \in \mathbb{R}^n$, define $f(x) = -\|\sum_{i=1}^n (w_i - x_i)p_i\|^2$. Let $\tilde{u}$ be the output of a call to* FRANK-WOLFE$(f, \lceil \frac{8}{\varepsilon} \rceil)$*; see Algorithm 1. Then:*

(i) $\tilde{u}$ *is a distribution vector with* $\|\tilde{u}\|_0 \leq \lceil \frac{8}{\varepsilon} \rceil$,
(ii) $\|\sum_{i=1}^n (w_i - \tilde{u}_i)p_i\|^2 \leq \varepsilon$, *and*
(iii) $\tilde{u}_i$ *is computed in* $O\left(\frac{nd}{\varepsilon}\right)$ *time.*

We now show how to obtain a vector summarization ε-coreset of size $O(1/\varepsilon)$ in $O(\frac{nd}{\varepsilon})$ time for any set $P \subseteq \mathbb{R}^d$.

Theorem 2 (Vector summarization coreset)**.** *Let (Q, m) be a weighted set of n points in $\mathbb{R}^d$, $\varepsilon \in (0, 1)$, and let u be the output of a call to* CoreSet$(Q, m, \frac{\varepsilon}{16})$*; see Algorithm 2. Then, $u \in \mathbb{R}^n$ is a vector with $\|u\|_0 \leq \frac{128}{\varepsilon}$ non-zero entries that is computed in $O(\frac{nd}{\varepsilon})$ time, and (Q, u) is a vector summarization ε-coreset for (Q, m).*

Algorithm 2: CoreSet(Q, m, ε)

Input: A weigthed set (Q, m) of $n \geq 2$ points in $\mathbb{R}^d$ and an error parameter $\varepsilon \in (0, 1)$.
Output: A weight vector $u \in [0, \infty)^n$ with $O(1/\varepsilon)$ non-zero entries that satisfies Theorem 2.
$w := \frac{m}{\|m\|_1}$
$\mu_w := \sum_{i=1}^{n} w_i q_i$
$\sigma_w := \sqrt{\sum_{i=1}^{n} w_i \|q_i - \mu\|^2}$
for every $i \in \{1, \cdots, n\}$ **do**
    $p_i := \frac{q_i - \mu}{\sigma}$
    $p'_i := \frac{(p_i^T \mid 1)^T}{\|(p_i^T \mid 1)\|^2}$ {Notice: $\|p'_i\| \leq 1$.}
    $w'_i := \frac{w_i \|(p_i^T \mid 1)\|^2}{2}$
end for
Compute a sparse vector u' with $O(1/\varepsilon)$ non-zero entries, such that $\|\sum_{i=1}^{n}(w'_i - u'_i)p'_i\|^2 \leq \varepsilon$
{E.g., using Algorithm 1 (see Theorem 1).}
$u_i := \|m\|_1 \cdot \frac{2u'_i}{\|(p_i^T \mid 1)\|^2}$ for every $i \in \{1, \cdots, n\}$
Return u

3.1. Boosting the Coreset's Construction Running Time

In this section, we present Algorithm 3, which aims to boost the running time of Algorithm 1 from the previous section; see Theorem 3. The main idea behind this new boosted algorithm is as follows: instead of running the Frank–Wolfe algorithm on a (full) set of input data, it can be more efficient to partition the input into a constant number k of equal-sized chunks, pick some representative for each chunk (its mean), run the Frank–Wolfe algorithm only on the set of representatives (the set of means) to obtain back a subset of those representative, and then continue recursively only with the chunks whose representative was chosen by the algorithm. Although the Frank–Wolfe algorithm is now applied multiple times (rather than once), each of those runs is much more efficient since only the small set of representatives is considered.

This immediately implies a faster construction time of vector summarization ε-coresets for general input sets; see Corollary 4 and Figure 1 for illustration.

Theorem 3 (Faster coreset for points in the unit ball)**.** *Let P be a set of n points in $\mathbb{R}^d$ such that $\|p\| \leq 1$ for every $p \in P$. Let $w : P \to (0, 1)$ be a weights function such that $\sum_{p \in P} w(p) = 1$, $\varepsilon \in (0, 1)$, and let (C, u) be the output of a call to* Fast-FW-CoreSet(P, w, ε)*; see Algorithm 3. Then*

(i) $|C| \leq 8/\varepsilon$ *and* $\sum_{p \in C} u(p) = 1$,

(ii) $\left\|\sum_{p \in P} w(p)p - \sum_{p \in C} u(p)p\right\|^2 \leq 2\varepsilon$, *and*

(iii) (C, u) *is computed in* $O\left(nd + \frac{d \cdot \log(n)^2}{\varepsilon^2}\right)$ *time.*

Corollary 4 (Faster vector summarization coreset)**.** *Let* (Q, m) *be a weighted set of* n *points in* $\mathbb{R}^d$*, and let* $\varepsilon \in (0, 1)$*. Then in* $O\left(nd + d \cdot \frac{\log(n)^2}{\varepsilon^2}\right)$ *time, we can compute a vector* $u = (u_1, \cdots, u_n)^T \in \mathbb{R}^n$*, such that* u *has* $\|u\|_0 \leq 128/\varepsilon$ *non-zero entries and* (Q, u) *is a vector summarization* (2ε)*-coreset for* (Q, m)*.*

Algorithm 3: FAST-FW-CORESET(P, w, ε)

Input: A weighed set (P, w) of $n \geq 2$ points in $\mathbb{R}^d$ and an error parameter $\varepsilon \in (0, 1)$.
Output: A pair (C, u) that satisfies Theorem 3
$k := \frac{2\log(n)}{\varepsilon}$
if $|P| \leq k$ **then**
    Return: A vector summarization ε-corset for (P, w) using Theorem 1.
end if
$\{P_1, \cdots, P_k\} :=$ a partition of P into k disjoint subsets, each contains at most $\lceil n/k \rceil$ points.
for every $i \in \{1, \cdots, k\}$ **do**
    $\mu_i := \frac{1}{\sum_{q \in P_i} w(q)} \cdot \sum_{p \in P_i} w(p) \cdot p$ {The weighted mean of P_i}
    $w'(\mu_i) := \sum_{p \in P_i} w(p)$
end for
$(\tilde{\mu}, \tilde{u}) :=$ a vector summarization $\left(\frac{\varepsilon}{\log(n)}\right)$-corset for the weighted set $(\{\mu_1, \cdots, \mu_k\}, w')$ using Theorem 1.
$C := \bigcup_{\mu_i \in \tilde{\mu}} P_i$ {C is the union over all subsets P_i whose mean μ_i was chosen in $\tilde{\mu}$.}
for every $\mu_i \in \tilde{\mu}$ and $p \in P_i$ **do**
    $u(p) := \frac{\tilde{u}(\mu_i) w(p)}{\sum_{q \in P_i} w(q)}$
end for
$(C, u) :=$ FAST-FW-CORESET(C, u, ε)
Return: (C, u)

In what follows, we show how to compute a vector summarization coreset with high probability in a time that is sublinear in the input size $|Q| = n$. This is based on the geometric median trick, that suggests the following procedure: (i) sample $k > 1$ sets $\{S_1, \cdots, S_k\}$ of the same (small) size from the original input set Q, (ii) for each such sampled set S_i ($i \in [k]$), compute its mean $\bar{s}_i$, and finally, (iii) compute and return the geometric median of those means $\bar{s} = \{\bar{s}_1, \cdots, \bar{s}_k\}$. This geometric median is guaranteed to approximate the mean of the original input set Q.

We show that there is no need to compute this geometric median, as it is a difficult computational task. We prove that there exists a set S_{i^*} from the sampled subsets such that its mean $\bar{s}_{i^*}$ is very close to this geometric median, with high probability. Thus, $\bar{s}_{i^*}$ is a good approximation to the desired mean of the original input set. Furthermore, we show that $\bar{s}_{i^*}$ is simply the point in $\bar{s}$ that minimizes its sum of (non-squared) distances to this set $\bar{s}$, i.e., $i^* \in \arg\min_{j \in [k]} \sum_{i=1}^{k} \|\bar{s}_i - \bar{s}_j\|_2$. An exhaustive search over the points of $\bar{s}$ can thus recover $\bar{s}_{i^*}$. The corresponding set S_{i^*} is the resulted vector summarization coreset; see Lemma 5 and Algorithm 4.

Lemma 5 (Fast probabilistic vector summarization coreset)**.** *Let* Q *be a set of* n *points in* $\mathbb{R}^d$*,* $\mu = \frac{1}{n} \sum_{p \in P} q$*, and* $\sigma^2 = \frac{1}{n} \sum_{p \in P} \|q - \mu\|^2$*. Let* $\varepsilon \in (0, 1)$*,* $\delta \in (0, 0.9]$*, and let* $S \subseteq \mathbb{R}^d$ *be the output of a call to* PROB-WEAK-CORESET(Q, ε, δ)*; see Algorithm 4. Then:*

(i) $S \subseteq Q$ *and* $|S| = \frac{4}{\varepsilon}$,

(ii) *with probability at least* $1 - 3\delta$ *we have* $\left\| \frac{1}{|S|} \sum_{p \in S} p - \mu \right\|^2 \leq 33 \cdot \varepsilon\sigma^2$, *and*

(iii) S *is computed in* $O\left(d \log\left(\frac{1}{\delta}\right)^2 + \frac{d \log\left(\frac{1}{\delta}\right)}{\varepsilon} \right)$ *time.*

Algorithm 4: PROB-WEAK-CORESET(Q, ε, δ)

1: **Input:** A set Q of $n \geq 2$ points in $\mathbb{R}^d$, $\varepsilon \in (0,1)$, and $\delta \in (0,1)$.
2: **Output:** A subset $S \subseteq P$ that satisfies Lemma 5.
3: $k := \lfloor 3.5 \log\left(\frac{1}{\delta}\right) \rfloor + 1$.
4: $S :=$ an i.i.d sample of size $\frac{4k}{\varepsilon}$.
5: $\{S_1, \cdots, S_k\} :=$ a partition of S into k disjoint subsets, each contains $\frac{4}{\varepsilon}$ points .
6: $\bar{s}_i :=$ the mean of the i'th subset S_i for $i \in [k]$.
7: $i^* := \arg\min_{j \in [k]} \sum_{i=1}^{k} \left\| \bar{s}_i - \bar{s}_j \right\|_2$.
8: **Return** S_{i^*}

4. Applications

Coreset for 1-mean. A 1-mean ε-coreset for (Q, m) is a weighted set (Q, u) such that for every $x \in \mathbb{R}^d$, the sum of squared distances from x to either (Q, m) or (Q, u), is approximately the same. To maintain the above property, we prove that it suffices for (Q, u) to satisfy the following: the mean, the variance, and the sum of weights of (Q, u) should approximate the mean, the variance, and the sum of weights of (Q, m), respectively, up to an additive error that depends linearly on ε. Then note that when plugging ε^2 (rather than ε) as input to Algorithm 2, the output is guaranteed to satisfy the above 3 properties, by construction of u.

The following theorem computes a 1-mean ε-coreset.

Theorem 6. *Let* (Q, m) *be a weighted set of* n *points in* $\mathbb{R}^d$, $\varepsilon \in (0,1)$. *Then in*

$$O\left(\min\left\{ nd + d \cdot \frac{\log(n)^2}{\varepsilon^4}, \frac{nd}{\varepsilon^2} \right\} \right)$$

time we can compute a vector $u = (u_1, \cdots, u_n)^T \in \mathbb{R}^n$, *where* $\|u\|_0 \leq \frac{128}{\varepsilon^2}$, *such that:*

$$\forall x \in \mathbb{R}^d : \quad \left| \sum_{i=1}^{n} (m_i - u_i) \|q_i - x\|^2 \right| \leq \varepsilon \sum_{i=1}^{n} m_i \|q_i - x\|^2.$$

Coreset for KDE. Given two sets of points Q and Q', and a kernel $\mathbb{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that is defined by the kernel map ϕ, the maximal difference

$$\sup_{x \in \mathbb{R}^d} \left| \sum_{q \in Q} \frac{\mathbb{K}(x, q)}{|Q|} - \sum_{q' \in Q'} \frac{\mathbb{K}(x, q')}{|Q'|} \right|$$

between the kernel costs of Q and Q' is upper bounded by $\left\| \mu_{\hat{Q}} - \mu_{\hat{Q}'} \right\|_2$, where $\mu_{\hat{Q}}$ and $\mu_{\hat{Q}'}$ are the means of $\hat{Q} = \{\phi(q) \mid q \in Q\}$ and $\hat{Q}' = \{\phi(q) \mid q \in Q'\}$, respectively, [49]. Given $\hat{Q}$, we can compute a vector summarization ε^2-coreset $\hat{Q}'$, which satisfies that $\left\| \mu_{\hat{Q}} - \mu_{\hat{Q}'} \right\|_2^2 \leq \varepsilon^2$. By the above argument, this is also an ε-KDE coreset.

Coreset for dimensionality reduction and LMS solvers. An ε-coreset for the k-SVD (k-PCA) problem of Q is a small weighted subset of Q that approximates the sum of squared distances from the points in Q to every non-affine (affine) k-dimensional subspace of $\mathbb{R}^d$,

up to a multiplicative factor of $1 \pm \varepsilon$; see Corollary 7. Coreset for LMS solvers is the special case of $k = d - 1$.

In [35], it is shown how to leverage an $(\varepsilon/k)^2$-coreset for the vector summarization problem in order to compute an ε-coreset for k-SVD. In [45], it is shown how to compute a coreset for k-PCA via a coreset for k-SVD, by simply adding another entry with some value $r \in \mathbb{R}$ to each vector of the input. Algorithm 5 combines both the above reductions, along with a computation of a vector summarization $(\varepsilon/k)^2$-coreset to compute the desired coreset for dimensionality reduction (both k-SVD and k-PCA). To compute the vector summarization coreset we utilize our new algorithms from the previous sections, which are faster than the state of the art algorithms.

Algorithm 5: DIM-CORESET(A, k, ε)

Input: A matrix $A \in \mathbb{R}^{n \times d}$, an integer $k \in [d]$, and an error parameter $\varepsilon \in (0, 1)$.
Output: A diagonal matrix $W \in \mathbb{R}^{n \times n}$ that satisfies Corollary 7.
$r := 1 + \max_{i \in [n]} \frac{4\|a_i\|^2}{\varepsilon^4}$ {where a_i is the ith row of A}
$U, \Sigma, V :=$ the full SVD of $[A \mid (r, \cdots, r)^T] \in \mathbb{R}^{n \times (d+1)}$
$v_i := \left(U_{i,1}, \cdots, U_{i,k}, \frac{U_{i,k+1:d}\Sigma_{k+1:d,k+1:d}}{\left\|\Sigma_{k+1:d,k+1:d}\right\|_F} \right)$ for every $i \in [n]$
$\tilde{v_i} :=$ the row stacking of $v_i v_i^T \in \mathbb{R}^{d \times d}$ for every $i \in [n]$
$(\{\tilde{v_1}, \cdots, \tilde{v_n}\}, u) :=$ a vector summarization $(\frac{\varepsilon}{5k})^2$-coreset for $(\{\tilde{v_1}, \cdots, \tilde{v_n}\}, (1, \cdots, 1))$.
$W :=$ a diagonal matrix in $\mathbb{R}^{n \times n}$, where $W_{i,i} = \sqrt{u_i}, \forall i \in [n]$.
Return W

Corollary 7 (Coreset for dimensionality reduction)**.** *Let Q be a set of n points in $\mathbb{R}^d$, and let $A \in \mathbb{R}^{n \times d}$ be a corresponding matrix containing the points of Q in its rows. Let $\varepsilon \in (0, \frac{1}{2})$ be an error parameter, $k \in [d]$ be an integer, and W be the output of a call to* DIM-CORESET(A, k, ε). *Then:*

(i) *W is a diagonal matrix with $O\left(\frac{k^2}{\varepsilon^2}\right)$ non-zero entries,*

(ii) *W is computed in $O\left(\min\left\{nd^2 + \frac{d^2 \log(n)^2 k^4}{\varepsilon^4}, \frac{nd^2k^2}{\varepsilon^2}\right\}\right)$ time, and*

(iii) *there is a constant c, such that for every $\ell \in \mathbb{R}^d$ and an orthogonal $X \in \mathbb{R}^{d \times (d-k)}$ we have*

$$\left| 1 - \frac{\|W(A - \ell)X\|_F^2}{\|(A - \ell)X\|_F^2} \right| \leq c\varepsilon.$$

Here, $A - \ell$ is the subtraction of ℓ from every row of A.

Where do our methods fit in? Theoretically speaking, the 1-mean problem (also known as the arithmetic mean problem), is a widely used tool for reporting central tendencies in the field of statistics, as it is also used in machine learning. As for the practical aspect of such problem, it can be either used to obtain an estimation of the mathematical expectation of signal strength in a area [50], or as an imputation technique used to fill in missing values, e.g., in the context of filling in missing values of heart monitor sensor data [51]. Note that a variant of this problem is widely used in the context of deep learning, namely, the moving averages. Algorithms 3 and 4 can boost such methods when given large-scale datasets. In addition, our algorithms extend also to SVD, PCA, and LMS where these methods are known for their usages and efficiencies in discovering a low dimensional representation of high dimensional data. From a practical point of view, SVD showed promising results when dealing with on calibration of a star sensor on-orbit calibration [52], denoising a 4-dimensional computed tomography of the brain in stroke

patients [53], removal of cardiac interference from trunk electromyogram [54], among many other applications.

We propose a summarization technique (see Algorithm 5) that aims to compute an approximation towards the SVD factorization of large-scale datasets where applying the SVD factorization on the dataset is not possible due to insufficient memory or long computational time.

5. Experimental Results

We now apply different coreset construction algorithms presented in this paper to a variety of applications, in order to boost their running time, or reduce their memory storage. We note that a complete open source code is provided [55].

Software/Hardware. The algorithms were implemented in Python 3.6 [56] using "Numpy" [57]. Tests were conducted on a PC with Intel i9-7960X CPU @2.80 GHz x 32 and 128 Gb RAM.

We compare the following algorithms: (To simply distinguish between our algorithms and the competing ones in the graphs, observe that the labels of our algorithms starts with the prefix "Our-", while the competing methods do not.)

(i) `Uniform`: Uniform random sample of the input Q, which requires sublinear time to compute.

(ii) `Sensitivity-sum`: Random sampling based on the "sensitivity" for the vector summarization problem [58]. Sensitivity sampling is a widely known technique [19], which guarantees that a subsample of sufficient size approximates the input well. The sensitivity of a point $q \in Q$ is $\frac{1}{n} + \frac{\|q\|^2}{\sum_{q' \in Q} \|q'\|^2}$. This algorithm takes $O(nd)$ time.

(ii) `ICML17`: The vector summarization coreset construction algorithm from [29] (see Algorithm 2 there), which runs in $O(nd/\varepsilon)$ time.

(iv) `Our-rand-sum`: Our coreset construction from Lemma 5, which requires $O\left(d \log\left(\frac{1}{\delta}\right)^2 + \frac{d \log\left(\frac{1}{\delta}\right)}{\varepsilon}\right)$ time.

(v) `Our-slow-sum`: Our coreset construction from Corollary 2, which requires $O(nd/\varepsilon)$ time.

(vi) `Our-fast-sum`: Our coreset construction from Corollary 4, which requires $O(nd + d\log(n)^2/\varepsilon^2)$ time.

(vii) `Sensitivity-svd`: Similar to `Sensitivity-sum` above, however, now the sensitivity is computed by projecting the rows of the input matrix A on the optimal k-subspace (or an approximation of it) that minimizes its sum of squared distances to the rows of A, and then computing the sensitivity of each row i in the projected matrix A' as $\|u_i\|^2$, where u_i is the ith row the matrix U from the SVD of $A' = UDV^T$; see [37]. This takes $O(ndk)$ time.

(viii) `NIPS16`: The coreset construction algorithm from [35] (see Algorithm 2 there) which requires $O(nd^2k^2/\varepsilon^2)$ time.

(ix) `Our-slow-svd`: Corollary 7 offers a coreset construction for SVD using Algorithm 5, which utilizes Algorithm 2. However, Algorithm 2 either utilizes Algorithm 1 (see Theorem 2) or Algorithm 3 (see Theorem 4). `Our-slow-svd` applies the former option, which requires $O(nd^2k^2/\varepsilon^2))$ time.

(x) `Our-fast-svd`: Corollary 7 offers a coreset construction for SVD using Algorithm 5, which utilizes Algorithm 2. However, Algorithm 2 either utilizes Algorithm 1 (see Theorem 2) or Algorithm 3 (see Theorem 4). `Our-fast-svd` uses the latter option, which requires $O(nd^2 + d^2\log(n)^2k^4/\varepsilon^4)$ time.

Datasets. We used the following datasets from the UCI ML library [59]:

(i) New York City Taxi Data [60]. The data covers the taxi operations at New York City. We used the data describing $n = 14.7M$ trip fares at the year of 2013. We used the $d = 6$ numerical features (real numbers).

(ii) US Census Data (1990) [61]. The dataset contains $n = 2.4M$ entries. We used the entire $d = 68$ real-valued attributes of the dataset.

(iii) Buzz in social media Data Set [62]. It contains $n = 0.5M$ examples of buzz events from two different social networks: Twitter, and Tom's Hardware. We used the entire $d = 77$ real-valued attributes.

(iv) Gas Sensors for Home Activity Monitoring Data Set [63]. This dataset has $n = 919,438$ recordings of a gas sensor array composed of 8 MOX gas sensors, and a temperature and humidity sensor. We used the last $d = 10$ real-valued attributes of the dataset.

Discussion regarding the chosen datasets. The Buzz in social media data set is widely used in the context of Principal Component Regression (or, PCR in short), that is used for estimating the unknown regression coefficients in a standard linear regression model. The goal of PCR in the context of this dataset, is to predict popularity of a certain topic on Twitter over a period. It is known that the solution of the PCR problem can be approximated using the known SVD decomposition problem. Our techniques enable us to benefit from the coreset advantages, e.g., to boost the PCR approximated solution (PCA) while using low memory, and supporting the streaming model by maintaining a coreset for the data (tweets) seen so far; each time a new point (tweet) is received, it is added to current stored coreset in memory. Once the stored coreset is large enough, our compression (coreset construction algorithm) is applied. This procedure is repeated until the stream of points is empty.

The New York City taxi data contains information about the locations of passengers as well as the locations of their destinations. Thus, the goal is to find a location which is close to the most wanted destinations. This problem can be formulated as a facility location problem, which can be reduced to an instance of the 1-mean problem. Hence, since our methods admit faster solution as well as provable approximation for the facility location problem, we can leverage our coreset to speed up the computations using this dataset.

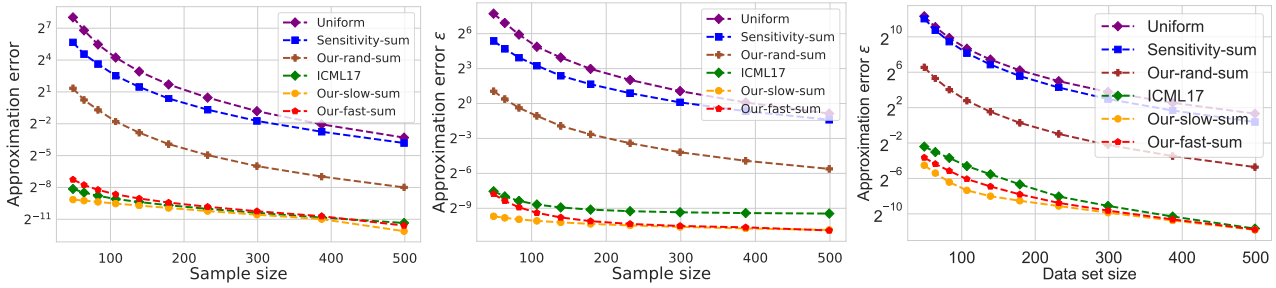
Finally, regarding the remaining datasets, PCA has been widely used either for low-dimensional embedding or, e.g., to compute the arithmetic mean. By using our methods, we can boost the PCA while admitting an approximated solution.

The experiments.

(i) **Vector summarization:** The goal is to approximate the mean of a huge input set, using only a small weighted subset of the input. The empirical approximation error is defined as $\|\mu - \mu_s\|^2$, where μ is the mean of the full data and μ_s is the mean of the weighted subset computed via each compared algorithm; see Figures 2 and 3.
In Figure 2, we report the empirical approximation error $\|\mu - \mu_s\|^2$ as a function of the subset (coreset) size, for each of the datasets (i)–(ii), while in Figure 3 we report the overall computational time for computing the subset (coreset) and for solving the 1-mean problem on the coreset, as a function of the subset size.

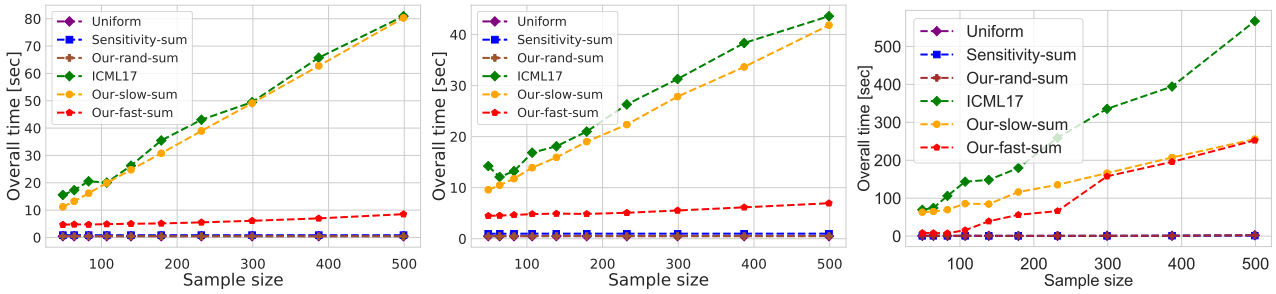
(ii) ***k*-SVD:** The goal is to compute the optimal k-dimensional non-affine subspaces of a given input set. We can either compute the optimal subspace using the original (full) input set, or using a weighted subset (coreset) of the input. We denote by S^* and S' the optimal subspace when computed either using the full data or using the subset at hand, respectively. The empirical approximation error is defined as the ratio $|(c^* - c')/c^*|$, where c^* and c' are the sum of squared distances between the points of original input set to S^* and S', respectively; see Figures 4–6. Intuitively, this ratio represents the relative SSD error of recovering an optimal k-dimensional non-affine subspace on the compression, rather than using the full data.
In Figure 4 we report the empirical error $|(c^* - c')/c^*|$ as a function of the coreset size. In Figure 5 we report the overall computational time in took to compute the coreset and to recover the optimal subspace using the coreset, as a function of the coreset size. In both figures we have three subfigures, each one for a different chosen value of k (the dimension of the subspace). Finally, in Figure 6 the x axis is the size of the dataset (which we compress to a subset of size 150), while the y-axis is the approximation error on the left hand side graph, and on the right hand side it is the

overall computational time it took to compute the coreset and to recover the optimal subspace using the coreset.



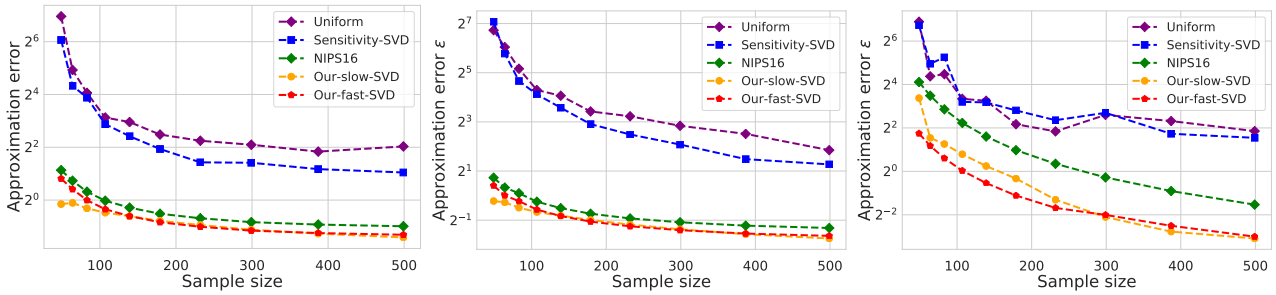
(**a**) Dataset (i). (**b**) Dataset (ii). (**c**) Dataset (iv).

Figure 2. Experimental results for vector summarization. The x axis is the size of the subset (coreset), while the y axis is the approximation error $\|\mu - \mu_s\|^2$. The difference between the two graphs is the chosen dataset.



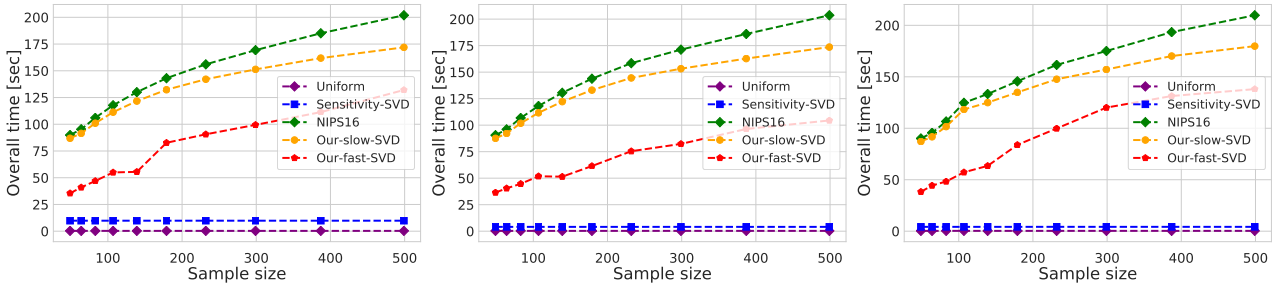
(**a**) Dataset (i). (**b**) Dataset (ii). (**c**) Dataset (iv).

Figure 3. Experimental results for vector summarization. The x axis is the size of the subset (coreset), while the y axis is the overall time took to compute the coreset and to solve the problem on it. The difference between the two graphs is the chosen dataset.



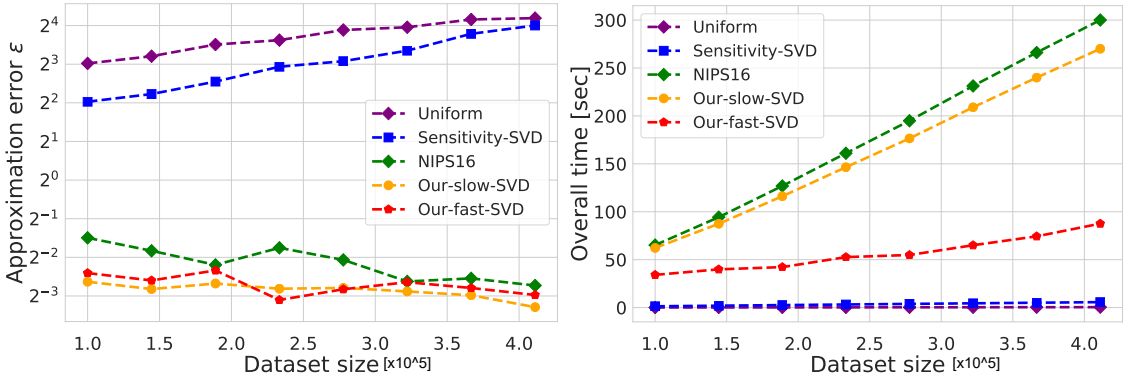
(**a**) $k = 40$, Dataset (iii). (**b**) $k = 50$, Dataset (iii). (**c**) $k = 60$, Dataset (iii).

Figure 4. Experimental results for k-SVD, we used Dataset (iii). The x axis is the size of the subset (coreset), while the y axis is the approximation error ε. The difference between the 3 graphs is the chosen low dimension k.



(**a**) $k = 40$, Dataset (iii). (**b**) $k = 50$, Dataset (iii). (**c**) $k = 60$, Dataset (iii).

Figure 5. Experimental results for k-SVD, we used Dataset (iii). The x axis is the size of the subset (coreset), while the y axis is the overall time took to compute the coreset and to solve the problem on it. The difference between the 3 graphs is the chosen low dimension k.



(**a**) $k = 40$, Dataset (iii). (**b**) $k = 40$, Dataset (iii).

Figure 6. Experimental results for k-SVD, we used Dataset (iii). The x axis is the size of the dataset which we compress to subsample of size 150, while the y-axis is the approximation error in the left hand side graph, and in the right hand side it is the overall time took to compute the coreset and to solve the problem on it.

5.1. Discussion

Vector summarization experiment: As predicted by the theory and as demonstrated in Figures 2 and 3, our fast and deterministic algorithm `Our-fast-sum` (the red line in the figures) achieves either the same or smaller approximation errors in most cases compared to the deterministic alternatives `Our-slow-sum` (orange line) and `ICML17` (green line), while being up to $\times 10$ times faster. Hence, when we seek a fast time deterministic solution for computing a coreset for the vector summarization problem, our algorithm `Our-fast-sum` is the favorable choice.

Compared to the randomized alternatives, `Our-fast-sum` is obviously slower, but achieves an error more than 3 orders of magnitude smaller. However, our fast and randomized algorithm `Our-rand-sum` (brown line) constantly achieves better results compared to the other randomized alternatives; It yields approximation error up to $\times 50$ smaller, while maintaining the same computational time. This is demonstrated on both datasets. Hence, our compression can be used to speed up tasks, e.g., computing the PCA or PCR, as described above.

k-SVD experiment: Here, in Figures 4–6 we witness a similar phenomena, where our fast and deterministic algorithm `Our-fast-svd` achieves the same or smaller approximation errors compared to the deterministic alternatives `Our-slow-svd` and `NIPS16`, respectively, while being up to $\times 4$ times faster. Compared to the randomized alternatives, `Our-fast-svd` is slower as predicted, but achieves an error up to 2 orders of magnitude smaller. This is demonstrated for increasing sample sizes (as in Figures 4 and 5), for increasing dataset size (as in Figure 6), and for various values of k (see Figures 4–6).

5.2. Conclusions and Future Work

This paper generalizes the definition of ε-sample and coreset from the worst case error over every query to average ℓ_2 error. We then showed a reduction from the problem of computing such coresets to the vector summarization coreset construction problem. Here, we suggest deterministic and randomized algorithms for computing such coresets, the deterministic version takes $O(\min\{nd/\varepsilon, nd + d\log(n)^2/\varepsilon^2\})$, and the randomized $O(d\log(\frac{1}{\delta})^2 + \frac{d\log(\frac{1}{\delta})}{\varepsilon})$. Finally, we showed how to leverage an (ε^2)-coreset for the vector summarization problem in order to compute an ε-coreset for the 1-mean problem, and similarly for k-SVD and k-PCA problem via computing an $(\varepsilon/k)^2$ vector summarization coreset after some reprocessing on the data.

Open problems include generalizing these results for other types of norms, or other functions such as M-estimators that are robust to outliers. We hope that the source code and the promising experimental results would encourage also practitioners to use these new types of approximations. Normalization via this new sensitivity type reduced the bounds on the number of iterations of the Frank–Wolfe algorithm by orders of magnitude. We believe that it can be used more generally for provably faster convex optimization, independently of coresets or ε-samples. We leave this for future research.

Author Contributions: Conceptualization, A.M., I.J., M.T. and D.F.; methodology, A.M. and I.J.; software, A.M. and M.T.; validation, A.M. and M.T.; formal analysis, A.M., I.J., M.T. and D.F.; investigation, A.M., I.J., M.T. and D.F.; resources, D.F.; data curation, A.M., I.J. and M.T.; writing—original draft preparation, A.M., I.J. and M.T.; writing—review and editing, A.M., I.J. and M.T.; visualization, I.J.; supervision, D.F.; project administration, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare that one of the authors is Prof. Dan Feldman who is a guest editor of special issue "Sensor Data Summarization: Theory, Applications, and Systems".

Appendix A. Problem Reduction for Vector Summarization ε-Coresets

First, we define a normalized weighted set, which is simply a set which satisfies three properties: weights sum to one, zero mean, and unit variance.

Definition A1 (Normalized weighted set)**.** *A normalized weighted set is a weighted set (P, w) where $P = \{p_1, \cdots, p_n\} \subseteq \mathbb{R}^d$ and $w = (w_1, \cdots, w_n)^T \in \mathbb{R}^n$ satisfy the following properties:*

(a) Weights sum to one: $\sum_{i=1}^n w_i = 1$,
(b) The weighted sum is the origin: $\sum_{i=1}^n w_i p_i = \mathbf{0}$, and
(c) Unit variance: $\sum_{i=1}^n w_i \|p_i\|^2 = 1$.

Appendix A.1. Reduction to Normalized Weighted Set

In this section, we argue that in order to compute a vector summarization ε-coreset for an input weighted set (Q, m), it suffices to compute a vector summarization ε-coreset for its corresponding normalized (and much simpler) weighted set (P, w) as in Definition A1; see Corollary A1. However, first, in Observation A1, we show how to compute a corresponding normalized weighted set (P, w) for any input weighted set (Q, m).

Observation A1. *Let $Q = \{q_1, \cdots, q_n\}$ be a set of $n \geq 2$ points in $\mathbb{R}^d$, $m \in (0, \infty)^n$, $w \in (0, 1]^n$ be a distribution vector such that $w = \frac{m}{\|m\|_1}$, $\mu = \sum_{i=1}^n w_i q_i$ and $\sigma = \sqrt{\sum_{i=1}^n w_i \|q_i - \mu\|^2}$. Let $P = \{p_1, \cdots, p_n\}$ be a set of n points in $\mathbb{R}^d$, such that for every $j \in [n]$ we have $p_j = \frac{q_j - \mu}{\sigma}$. Then, (P, w) is the corresponding normalized weighted set of (Q, m), i.e., (i)–(iii) hold as follows:*

(i) $\sum_{i=1}^{n} w_i = 1$,
(ii) $\sum_{i=1}^{n} w_i p_i = \mathbf{0}$, and
(iii) $\sum_{i=1}^{n} w_i \|p_i\|^2 = 1$.

Proof.

$$\sum_{i=1}^{n} w_i = 1, \text{ immediately holds by the definition of } w.$$

$$\begin{aligned}\sum_{i=1}^{n} w_i p_i &= \sum_{i=1}^{n} w_i \cdot \frac{q_i - \mu}{\sigma} = \frac{1}{\sigma}\left(\sum_{i=1}^{n} w_i q_i - \sum_{i=1}^{n} w_i \mu\right) \\ &= \frac{1}{\sigma}\left(\mu - \sum_{i=1}^{n} w_i \mu\right) = \frac{1}{\sigma}\mu\left(1 - \sum_{i=1}^{n} w_i\right) = 0,\end{aligned}$$

where the first equality holds by the definition of p_i, the third holds by the definition of μ, and the last is since w is a distribution vector.

$$\begin{aligned}\sum_{i=1}^{n} w_i \|p_i\|^2 &= \sum_{i=1}^{n} w_i \left\| \frac{q_i - \mu}{\sigma} \right\|^2 = \frac{1}{\sigma^2} \sum_{i=1}^{n} w_i \|q_i - \mu\|^2 \\ &= \frac{\sum_{i=1}^{n} w_i \|q_i - \mu\|^2}{\sum_{i=1}^{n} w_i \|q_i - \mu\|^2} = 1,\end{aligned}$$

where the first and third equality hold by the definition of p_i and σ, respectively. □

Corollary A1. *Let (Q, m) be a weighted set, and let (P, w) be its corresponding normalized weighted set as computed in Observation A1. Let (P, u) be a vector summarization ε-coreset for (P, w) and let $u' = \|m\|_1 \cdot u$. Then (Q, u') is a vector summarization ε-coreset for (Q, m).*

Proof. Put $x \in \mathbb{R}^d$ and let $y = \frac{x-\mu}{\sigma}$. Now, for every $j \in [n]$, we have that

$$\begin{aligned}\|q_j - x\|^2 &= \|\sigma p_j + \mu - (\sigma y + \mu)\|^2 \\ &= \|\sigma p_j - \sigma y\|^2 \\ &= \sigma^2 ||p_j - y||^2,\end{aligned} \tag{A1}$$

where the first equality is by the definition of y and p_j.

Let (P, u) be a vector summarization ε-coreset for (P, w). We prove that (Q, u') is a vector summarization ε-coreset for (Q, m). We observe the following

$$\begin{aligned}&\left\| \sum_{i=1}^{n} \frac{m_i}{\|m\|_1} q_i - \sum_{i=1}^{n} \frac{u'_i}{\|u'\|_1} q_i \right\|^2 = \left\| \sum_{i=1}^{n} w_i q_i - \sum_{i=1}^{n} \frac{u_i}{\|u\|_1} q_i \right\|^2 \\ &= \left\| \sum_{i=1}^{n} (w_i - \frac{u_i}{\|u\|_1})(p_i \sigma + \mu) \right\|^2 \\ &= \left\| \sum_{i=1}^{n} (w_i - \frac{u_i}{\|u\|_1}) p_i \sigma + \mu \sum_{i=1}^{n} (w_i - \frac{u_i}{\|u\|_1}) \right\|^2 \\ &= \left\| \sum_{i=1}^{n} (w_i - \frac{u_i}{\|u\|_1})(p_i \sigma) \right\|^2 \leq \varepsilon \sigma^2\end{aligned} \tag{A2}$$

where the first equality holds since $w = \frac{m}{\|m\|_1}$ and $\frac{u'}{\|u'\|_1} = \frac{mu}{m\|u\|_1} = \frac{u_i}{\|u\|_1}$, the second holds by (A1), and the last inequality holds since (P, u) is a vector summarization ε-coreset for (P, w). □

Appendix A.2. Vector Summarization Problem Reduction

Given a normalized weighted set (P, w) as in Definition A1, in the following lemma we prove that a weighted set (P, u) is a vector summarization ε-coreset for the normalized weighted set (P, w) if and only if the squared ℓ_2 norm of the weighted mean of (P, u) is smaller than ε.

Lemma A2. *Let (P, w) be a normalized weighted set of n points in $\mathbb{R}^d$, $\varepsilon \in (0, 1)$, and $u \in \mathbb{R}^n$ be a weight vector. Let $\overline{p} = \sum_{i=1}^{n} \frac{w_i}{\|w\|_1} p_i$, $\overline{s} = \sum_{i=1}^{n} \frac{u_i}{\|u\|_1} p_i$, and $\sigma^2 = \|p_i - \overline{p}\|^2$. Then, (P, u) is a vector summarization ε-coreset for (P, w), i.e., $\|\overline{p} - \overline{s}\|^2 \leq \varepsilon \sum_{i=1}^{n} w_i \sigma^2$ if and only if $\|\overline{s}\|^2 \leq \varepsilon$.*

Proof. The proof holds since (P, w) is a normalized wighted set, i.e., $\overline{p} = 0$, and $\sigma^2 = 1$. □

Appendix B. Frank–Wolfe Theorem

Here, for completeness we state the Frank–Wolfe Theorem [48]. This theorem will be used in the proof of Theorem 1 that shows how to compute a variant of vector summarization coreset for points inside the unit ball.

To do so, we consider the measure C_f defined in [48]; see equality (9) in Section 2.2. For a simplex S and concave function f, the quantity C_f is defined as

$$C_f := \sup \frac{1}{\alpha^2}(f(x) + (y - x)^T \nabla f(x) - f(y)), \tag{A3}$$

where the supremum is over every x and z in S, and over every α so that $y = x + \alpha(z - x)$ is also in S. The set of such α includes $[0, 1]$, but α can also be negative.

Theorem A3 (Theorem 2.2 from [48]). *For simplex S and concave function f, Algorithm 1 (Algorithm 1.1 from [48]) finds a point $x_{(k)}$ on a k-dimensional face of S such that*

$$\frac{f(x*) - f(x_{(k)})}{4C_f} \leq \frac{1}{k+3},$$

for $k > 0$, where $f(x*)$ is the optimal value of f.

Appendix C. Proof of Theorem 1

Proof of Theorem 1. Let C_f be defined for f and S as in (A3), and let $f(x^*)$ be the maximum value of f in S. Based on Theorem A3 we have:

1. $\tilde{u}$ is a point on a $\lceil \frac{8}{\varepsilon} \rceil$-dimensional face of S, i.e., $\|\tilde{u}\|_0 \leq \lceil \frac{8}{\varepsilon} \rceil$, $u \in S \subset [0, 1]^n$ and $\sum_{i=1}^{n} \tilde{u}_i = 1$. Hence, claim (i) of this theorem is satisfied.
2. $\frac{f(x^*) - f(x_{(k)})}{4C_f} \leq \frac{1}{k+3}$, for every $k \in \{0, \cdots, \lceil \frac{8}{\varepsilon} \rceil\}$.

   Since $f(x) \leq 0$ for every $x \in S$, we have that,

$$f(x^*) = f(w) = -\left\| \sum_{i=1}^{n} (w_i - w_i) p_i \right\|^2 = 0.$$

Define A to be the matrix of $d \times n$ such that the i-th column of A is the i-th point in P, and let $\mu = \sum_{i=1}^{n} w_i p_i$. We get that

$$\begin{aligned} f(x) &= -\left\| \sum_{i=1}^{n} (w_i - x_i) p_i \right\|^2 = -\left\| \mu - \sum_{i=1}^{n} x_i p_i \right\|^2 \\ &= -\|\mu\|^2 + 2\mu^T (\sum_{i=1}^{n} x_i p_i) - \left\| \sum_{i=1}^{n} x_i p_i \right\|^2 \\ &= -\|\mu\|^2 + 2\mu^T Ax - \|Ax\|^2 = -\|\mu\|^2 + 2x^T A^T \mu - x^T A^T Ax, \end{aligned} \tag{A4}$$

where the second equality holds by the definition of μ, and the fourth equality holds by since $\sum_{i=1}^{n} x_i p_i = Ax$ for every $x \in \mathbb{R}^n$.

At Section 2.2. in [48], it was shown that for any quadratic function $f' : \mathbb{R}^n \to \mathbb{R}$ that is defined as

$$f'(x) = a + x^T b + x^T Mx, \tag{A5}$$

where M is a negative semidefinite $n \times n$ matrix, $b \in \mathbb{R}^n$ is a vector, and $a \in \mathbb{R}$, we have that $C_{f'} \leq diam(A'S)^2$, where $A' \in \mathbb{R}^{d \times n}$ is a matrix that satisfies $M = A'^T A'$; see equality (12) at [48].

Hence, plugging $a = -\|\mu\|^2$, $b = 2A^T \mu$, and $M = A^T A$ in (A5) yields that for the function f we have $C_f \leq diam(AS)^2$, and

$$diam(AS)^2 = \sup_{a,b \in AS} \|a - b\|_2^2 = \sup_{x,y \in S} \|Ax - Ay\|_2^2.$$

Observe that x and y are distribution vectors, thus

$$\sup_{x,y \in S} \|Ax - Ay\|_2^2 = \sup_{i,j} \|p_i - p_j\|_2^2.$$

Since $\|p_i\| \leq 1$ for each $i \in [n]$, we have that

$$\sup_{i,j} \|p_i - p_j\|_2^2 \leq 2.$$

By substituting $C_f \leq 2$, $k = 8/\varepsilon$, $f(x_{(k)}) = f(\tilde{u}) = -\|\sum_{i=1}^{n} (w_i - \tilde{u}_i) p_i\|^2$, and $f(x^*) = 0$ in (2) we get that,

$$\frac{\|\sum_{i=1}^{n} (w_i - \tilde{u}_i) p_i\|^2}{8} \leq \frac{1}{8/\varepsilon + 3}. \tag{A6}$$

Multiplying both sides of the inequality by 8 and rearranging prove Theorem (ii) as

$$\left\| \sum_{i=1}^{n} (w_i - \tilde{u}_i) p_i \right\|^2 \leq \frac{8}{8/\varepsilon + 3} \leq \frac{8}{8/\varepsilon} = \varepsilon. \tag{A7}$$

Running time: We have $K = \lceil \frac{8}{\varepsilon} \rceil$ iterations in Algorithm 1, where each iteration takes $O(nd)$ time, since the gradient of f based on the vector $x = (x_1, \cdots, x_n)^T \in S$ is $-2A^T \sum_{i=1}^{n} (w_i - x_i) p_i$. This term is the multiplication between an a matrix in $\mathbb{R}^{n \times d}$ and a vector in $\mathbb{R}^d$, which takes $O(nd)$ time. Hence, the running time of the Algorithm is $O(\frac{nd}{\varepsilon})$. □

Appendix D. Proof of Theorem 2

Proof of Theorem 2. Let (P, w) be the normalized weighted set that is computed at Lines 3–5 of Algorithm 2 where $P = \{p_1, \cdots, p_n\}$, and let $\tilde{u} = \frac{u}{\|m\|_1}$. We show that $(P, \tilde{u})$ is a vector summarization ε-coreset for (P, w), then by Corrolary A1 we get that (Q, u) is a vector summarization ε-coreset for (Q, m). For every $i \in [n]$ let w'_i,u'_i,u_i and p'_i be defined as in Algorithm 2, and let $\varepsilon' = \frac{\varepsilon}{16}$. First, by the definition of u' we have that

$$\left\|u'\right\|_0 \leq \frac{8}{\varepsilon'} = \frac{128}{\varepsilon}, \tag{A8}$$

and since $u_i = \|m\|_1 \cdot \frac{2u'_i}{\left\|(p_i^T|1)\right\|^2}$ for every $i \in [n]$, we get that

$$\|u\|_0 \leq \frac{128}{\varepsilon}. \tag{A9}$$

We also have by Theorem 1 that

$$4\varepsilon' \geq 4\left\|\sum_{i=1}^{n}(w'_i - u'_i)p'\right\|^2 \tag{A10}$$

$$= 4\left\|\sum_{i=1}^{n}\frac{w_i\left\|(p_i^T \mid 1)\right\|^2 - \frac{u_i}{\|m\|_1}\left\|(p_i^T \mid 1)\right\|^2}{2} \cdot \frac{(p_i^T \mid 1)^T}{\left\|(p_i^T \mid 1)\right\|^2}\right\|^2 \tag{A11}$$

$$= \left\|\sum_{i=1}^{n}(w_i - \tilde{u}_i) \cdot (p_i^T \mid 1)^T\right\|^2$$

$$= \left\|\left(\sum_{i=1}^{n}(w_i - \tilde{u}_i) \cdot p_i^T \mid \sum_{i=1}^{n}(w_i - \tilde{u}_i)\right)^T\right\|^2 \tag{A12}$$

$$\geq \left\|\sum_{i=1}^{n}(w_i - \tilde{u}_i) \cdot p_i\right\|^2, \tag{A13}$$

where the first derivative is by the definition of u' in Algorithm 2 at line 11, the second holds by the definition of p', w' and u at Lines 8, 9, and 12 of the algorithm, the third holds since $\tilde{u} = \frac{u}{\|m\|_1}$, and the last inequality holds since $\|(x \mid y)\|^2 \geq x^2$ for every $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Combining the fact that $\sum_{i=1}^{n} w_i p_i = 0$ with (A13) yields that

$$4\varepsilon' \geq \left\|\sum_{i=1}^{n}\tilde{u}_i p_i\right\|^2. \tag{A14}$$

By (A12) and since w is a distribution vector we also have that

$$4\varepsilon' \geq \left|\sum_{i=1}^{n}(w_i - \tilde{u}_i)\right|^2 = \left|1 - \sum_{i=1}^{n}\tilde{u}_i\right|^2,$$

which implies

$$2\sqrt{\varepsilon'} \geq \left|1 - \sum_{i=1}^{n}\tilde{u}_i\right|. \tag{A15}$$

Combining (A15) and (A14) yields that:

$$\left\| \frac{\sum_{i=1}^{n} \tilde{u}_i p_i}{\sum_{i=1}^{n} \tilde{u}_i} \right\|^2 \leq \frac{4\varepsilon'}{(1-2\sqrt{\varepsilon'})^2} \leq 16\varepsilon' = \varepsilon, \tag{A16}$$

where that second inequality holds since $\varepsilon' = \varepsilon/16 \leq 1/16$.

By Lemma A2, Corollary A1, and (A16), Theorem 2 holds as

$$\left\| \sum_{i=1}^{n} \frac{u_i}{\|u\|_1} q_i - \sum_{i=1}^{n} \frac{m_i}{\|m\|_1} q_i \right\|_2^2 = \|\mu_u - \mu_m\|_2^2 \leq \varepsilon \sigma_m^2.$$

□

Appendix E. Proof of Theorem 3

Proof of Theorem 3. We use the notation and variable names as defined in Algorithm 3.

First, we assume that $w(p) > 0$ for every $p \in P$, otherwise we remove all the points in P which have zero weight, since they do not contribute to the weighted sum. Identify the input set $P = \{p_1, \cdots, p_n\}$ and the set C that is computed at Line 13 of Algorithm 3 as $C = \left\{c_1, \cdots, c_{|C|}\right\}$. We will first prove that the weighted set (C, u) that is computed in Lines 13–15 at an arbitrary iteration satisfies:

(a) $C \subseteq P$,

(b) $\sum_{p \in C} u(p) = 1$,

(c) $\left\| \sum_{p \in P} w(p) \cdot p - \sum_{p \in C} u(p) \cdot p \right\|^2 \leq \frac{\varepsilon}{\log(n)}$, and

(d) $|C| \leq \left\lceil \frac{|P|}{2} \right\rceil$.

Let $(\tilde{\mu}, \tilde{u})$ be the vector summarization $\frac{\varepsilon}{\log(n)}$-coreset of the weighted set $(\{\mu_1, \cdots, \mu_k\}, w')$ that is computed during the execution of the current iteration at Line 12. Hence, by Theorem 1

$$\begin{aligned} &\left\| \sum_{\mu_i \in \tilde{\mu}} \tilde{u}(\mu_i)\mu_i - \sum_{i=1}^{k} w'(\mu_i) \cdot \mu_i \right\|^2 \leq \frac{\varepsilon}{\log(n)}, \\ &\tilde{\mu} \subseteq \{\mu_1, \cdots, \mu_k\}, \text{ and} \\ &|\tilde{\mu}| \leq \frac{8 \cdot \log(n)}{\varepsilon}. \end{aligned} \tag{A17}$$

Proof of (a). Property (i) is satisfied by Line 13 as we have that $C \subseteq P$.

Proof of (b). Property (ii) is also satisfied since

$$\begin{aligned} \sum_{p \in C} u(p) &= \sum_{\mu_i \in \tilde{\mu}} \sum_{p \in P_i} \frac{\tilde{u}(\mu_i) w(p)}{w'(\mu_i)} = \sum_{\mu_i \in \tilde{\mu}} \frac{\tilde{u}(\mu_i)}{w'(\mu_i)} \sum_{p \in P_i} w(p) \\ &= \sum_{\mu_i \in \tilde{\mu}} \frac{\tilde{u}(\mu_i)}{\sum_{p \in P_i} w(p)} \sum_{p \in P_i} w(p) = \sum_{\mu_i \in \tilde{\mu}} \tilde{u}(\mu_i) = 1, \end{aligned} \tag{A18}$$

where the first equality holds by the definition of C at Line 13 and $w(p)$ for every $p \in C$ at Line 15, and the third equality holds by the definition of $u'(\mu_i)$ for every $\mu_i \in \tilde{\mu}$ as in Line 10.

Proof of (c). By the definition of w' and μ_i, for every $i \in \{1, \cdots, k\}$

$$\begin{aligned}\sum_{i=1}^{k} w'(\mu_i) \cdot \mu_i &= \sum_{i=1}^{k} w'(\mu_i) \cdot \left(\frac{1}{w'(\mu_i)} \cdot \sum_{p \in P_i} w(p) \cdot p \right) \\ &= \sum_{i=1}^{k} \sum_{p \in P_i} w(p)p = \sum_{p \in P} w(p)p.\end{aligned} \tag{A19}$$

The weighted sum of (C, u) is

$$\begin{aligned}\sum_{p \in C} u(p)p &= \sum_{\mu_i \in \tilde{\mu}} \sum_{p \in P_i} \frac{\tilde{u}(\mu_i)w(p)}{w'(\mu_i)} \cdot p \\ &= \sum_{\mu_i \in \tilde{\mu}} \tilde{u}(\mu_i) \sum_{p \in P_i} \frac{w(p)}{w'(\mu_i)} p = \sum_{\mu_i \in \tilde{\mu}} \tilde{u}(\mu_i)\mu_i,\end{aligned} \tag{A20}$$

where the first equality holds by the definitions of C and w, and the third equality holds by the definition of μ_i at Line 9. Plugging (A19) and (A20) in (A17) satisfies (iii) as

$$\left\| \sum_{p \in P} w(p) \cdot p - \sum_{p \in C} u(p) \cdot p \right\|^2 \leq \frac{\varepsilon}{\log(n)}. \tag{A21}$$

Proof of (d). By (A17) we have that C contains at most $\frac{\log(n)}{\varepsilon}$ clusters from P and at most $|C| \leq \frac{\log(n)}{\varepsilon} \cdot \lceil \frac{n}{k} \rceil$ points, and by plugging $k = \frac{2\log(n)}{\varepsilon}$ we obtain that $|C| \leq \left\lceil \frac{|P|}{2} \right\rceil$ as required.

We now prove (i)–(iii) from Theorem 3.

Proof of Theorem 3 (i). The first condition $|C| \leq 8/\varepsilon$ in (i) is satisfied since at each iteration we reduce the data size by a factor of 2, and we keep reducing until we reach the stopping condition, which is $O(\frac{\log(n)}{\varepsilon})$ by Theorem 1 (since we require a $\frac{\varepsilon}{\log(n)}$ error when we use Theorem 1, i.e., we need coreset of size $O(\frac{\log(n)}{\varepsilon})$). Then, at Line 5 when the if condition is satisfied (it should be, as explained) we finally use Theorem 1 again to obtain a coreset of size $\lceil 8/\varepsilon \rceil$ with ε-error on the small data (that was of size $\frac{O(\log(n))}{\varepsilon}$).

The second condition in (i) is satisfies since at each iteration we either return such a pair (C, u) at Line 18, we get by (b) that the sum of weight is always equal to 1.

Proof of Theorem 3 (ii). By (d) we also get that we have at most $\log(n)$ recursive calls. Hence, by induction on (2) we conclude that last computed set (C, u) at Line 18 satisfies (ii)

$$\left\| \sum_{p \in P} w(p) \cdot p - \sum_{p \in C} w(p) \cdot p \right\|^2 \leq \log(n) \cdot \frac{\varepsilon}{\log(n)} = \varepsilon.$$

At Line we return an ε coreset for the input weighted set (P, w) that have reached the size of $(\frac{\log(n)}{\varepsilon})$. Hence, the output of a the call satisfies $\left\| \sum_{p \in P} w(p) \cdot p - \sum_{p \in C} w(p) \cdot p \right\|^2 \leq 2\varepsilon$.

Proof of Theorem 3 (iii). As explained before, there are at most $log(n)$ recursive calls before the stopping condition at Line 4 is met. At each iteration we compute the set of means $\tilde{\mu}$, and compute a vector summarization $\left(\frac{\varepsilon}{\log n}\right)$-coreset for them. Hence, the time complexity of each iteration is $n'd + T(k, d, \frac{\varepsilon}{\log(n)})$ where n' is the number of points in the current iteration, and $T(k, d, \frac{\varepsilon}{\log(n)})$ is the running time of Algorithm 1 on k points in $\mathbb{R}^d$ to

obtain a $\frac{\varepsilon}{\log(n)}$-coreset. Thus, the total running of time the algorithm until the "If" condition at Line 4 is satisfied is

$$\sum_{i=1}^{\log(n)} \left(\frac{nd}{2^{i-1}} + T(k,d,\frac{\varepsilon}{\log(n)}) \right)$$
$$\leq 2nd + \log(n) \cdot T(k,d,\frac{\varepsilon}{\log(n)}) \in O\left(nd + \frac{kd}{\frac{\varepsilon}{\log(n)}} \right).$$

Plugging $k = \frac{2\log(n)}{\varepsilon}$ and observing the the last compression at Line 5 is done on a data of size $O(\frac{\log(n)}{\varepsilon})$ proves (iii) as the running time of Algorithm 3 is $O\left(nd + \frac{\log(n)^2 d}{\varepsilon^2}\right)$. □

Appendix F. Proof of Corollary 4

Proof. The corollary immediately holds by using Algorithm 2 with a small change. We change Line 11 in Algorithm 2 to use Algorithm 3 and Theorem 3, instead of Algorithm 1 and Theorem 1. □

Appendix G. Proof of Lemma 5

We first prove the following lemma:

Lemma A4. *Let P be a set of n points in $\mathbb{R}^d$, $\mu = \frac{1}{n}\sum_{p\in P} p$, and $\sigma^2 = \frac{1}{n}\sum_{p\in P}\|p-\mu\|^2$. Let $\varepsilon, \delta \in (0,1)$, and let S be a sample of $m = \frac{1}{\varepsilon\delta}$ points chosen i.i.d uniformly at random from P. Then, with probability at least $1-\delta$ we have that $\left\|\frac{1}{m}\sum_{p\in S} p - \mu\right\|^2 \leq \varepsilon\sigma^2$.*

Proof. For any random variable X, we denote by $E(X)$ and $\mathrm{var}(X)$ the expectation and variance of the random variable X, respectively. Let x_i denote the random variable that is the ith sample for every $i \in [m]$. Since the samples are drawn i.i.d, we have

$$\begin{aligned} \mathrm{var}\left(\frac{1}{m}\sum_{p\in S} p\right) &= \sum_{i=1}^{m} \mathrm{var}\left(\frac{x_i}{m}\right) = m \cdot \mathrm{var}\left(\frac{x_1}{m}\right) \\ &= m\left(\frac{\sigma^2}{m^2}\right) = \frac{\sigma^2}{m} = \varepsilon\delta\sigma^2. \end{aligned} \tag{A22}$$

For any random variable X and error parameter $\varepsilon' \in (0,1)$, the generalize Chebyshev's inequality [64] reads that

$$\Pr(\|X - E(X)\| \geq \varepsilon') \leq \frac{\mathrm{var}(X)}{(\varepsilon')^2}. \tag{A23}$$

Substituting $X = \frac{1}{m}\sum_{p\in S} p$, $E(X) = \mu$ and $\varepsilon' = \sqrt{\varepsilon}\sigma$ in (A23) yields that

$$\Pr\left(\left\|\frac{1}{m}\sum_{p\in S} p - \mu\right\| \geq \sqrt{\varepsilon}\sigma\right) \leq \frac{\mathrm{var}(\frac{1}{m}\sum_{p\in S} p)}{\sigma^2\varepsilon}. \tag{A24}$$

Combining (A22) with (A24) proves the lemma as:

$$\Pr\left(\left\|\frac{1}{m}\sum_{p\in S} p - \mu\right\|^2 \geq \varepsilon\sigma^2\right) \leq \frac{\varepsilon\delta\sigma^2}{\sigma^2\varepsilon} = \delta. \tag{A25}$$

□

Now we prove Lemma 5

Proof. Let $\{S_1, \cdots, S_k\}$ be a set of k i.i.d sampled subsets each of size $\frac{4}{\varepsilon}$ as defined at Line 5 of Algorithm 4, and let $\bar{s}_i$ be the mean of the ith subset S_i as define at Line 6. Let $\hat{s} := \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{k} \|\bar{s}_i - x\|_2$ be the geometric median of the set of means $\{\bar{s}_1, \cdots, \bar{s}_k\}$.

Using Corollary 4.1. from [65] we obtain that

$$\Pr\left(\|\hat{s} - \mu\| \geq 11 \sqrt{\frac{\sigma^2 \log(1.4/\delta)}{\frac{4k}{\varepsilon}}} \right) \leq \delta,$$

from the above we have that

$$\Pr\left(\|\hat{s} - \mu\|^2 \geq 121 \frac{\varepsilon \sigma^2 \log(1.4/\delta)}{4k} \right) \leq \delta. \tag{A26}$$

Note that

$$\Pr\left(\|\hat{s} - \mu\|^2 \geq 121 \frac{\varepsilon \sigma^2 \log(1.4/\delta)}{4k} \right) \tag{A27}$$

$$= \Pr\left(\|\hat{s} - \mu\|^2 \geq 30.25 \cdot \varepsilon \sigma^2 \frac{\log(1.4/\delta)}{\lfloor 3.5 \log\left(\frac{1}{\delta}\right) \rfloor + 1} \right) \tag{A28}$$

$$\geq \Pr\left(\|\hat{s} - \mu\|^2 \geq 31 \cdot \varepsilon \sigma^2 \right), \tag{A29}$$

where (A28) holds by substituting $k = \lfloor 3.5 \log\left(\frac{1}{\delta}\right) \rfloor + 1$ as in Line 3 of Algorithm 4, and (A29) holds since $\frac{\log(1.4/\delta)}{\lfloor 3.5 \log\left(\frac{1}{\delta}\right) \rfloor + 1} < 1$ for every $\delta \leq 0.9$ as we assumed. Combining (A29) with (A26) yields,

$$\Pr\left(\|\hat{s} - \mu\|^2 \geq 31 \cdot \varepsilon \sigma^2 \right) \leq \delta. \tag{A30}$$

For every $i \in [k]$, by substituting $S = S_i$, which is of size $\frac{4}{\varepsilon}$, in Lemma A4, we obtain that

$$\Pr(\|\bar{s}_i - \mu\|^2 \geq \varepsilon \sigma^2) \leq 1/4.$$

Hence, with probability at least $1 - (1/4)^k$ there is at least one set S_j such that

$$\|\bar{s}_j - \mu\|^2 \leq \varepsilon \sigma^2.$$

By the following inequalities:

$$(1/4)^k = (1/4)^{\lfloor 3.5 \log\left(\frac{1}{\delta}\right) \rfloor + 1} \leq (1/4)^{\log(1/\delta)} = 4^{\log(\delta)} \leq 2^{\log(\delta)} = \delta$$

we get that with probability at least $1 - \delta$ there is a set S_j such that

$$\|\bar{s}_j - \mu\|^2 \leq \varepsilon \sigma^2. \tag{A31}$$

Combining (A31) with (A30) yields that with probability at least $(1 - \delta)^2$ the set S_j satisfies that

$$\|\bar{s}_j - \hat{s}\|^2 \leq 32 \varepsilon \sigma^2. \tag{A32}$$

Let $f : \mathbb{R}^d \to [0, \infty)$ be a function such that $f(x) = \sum_{i=1}^{k} \|\bar{s}_i - x\|_2$ for every $x \in \mathbb{R}^d$. Therefore, by the definitions of f and $\hat{s}$,

$$\hat{s} := \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{k} \|\bar{s}_i - x\|_2 = \arg\min_{x \in \mathbb{R}^d} f(x).$$

Observe that f is a convex function since it is a sum over convex functions. By the convexity of f, we get that for every pair of points $p, q \in P$ it holds that:

$$\text{if } f(q) \leq f(p) \text{ then } \|q - \hat{s}\| \leq \|p - \hat{s}\|. \tag{A33}$$

Therefore, by the definition of i^* at in Algorithm 4 we get that

$$i^* \in \arg\min_{i \in [k]} \|\bar{s}_i - \hat{s}\|. \tag{A34}$$

Now by combining (A32) with (A34) we have that:

$$\Pr\left(\|\bar{s}_{i^*} - \hat{s}\|^2 \leq 32\varepsilon\sigma^2\right) \geq (1-\delta)^2. \tag{A35}$$

Combining (A35) with (A30) and noticing the following inequality

$$\begin{aligned}(1-\delta)^3 &= (1 - 2\delta + \delta^2)(1-\delta) \geq (1-2\delta)(1-\delta) \\ &= 1 - \delta - 2\delta + 2\delta^2 \geq 1 - 3\delta,\end{aligned}$$

satisfies Lemma 5 as,

$$\Pr\left(\|\bar{s}_{i^*} - \mu\|^2 \leq 33\varepsilon\sigma^2\right) \leq 1 - 3\delta.$$

Running time. It takes $O\left(\frac{d \log(\frac{1}{\delta})}{\varepsilon}\right)$ to compute the set of means at Line 6, and $O\left(d \log(\frac{1}{\delta})^2\right)$ time to compute Line 7 by simple exhaustive search over all the means. Hence, the total running time is $O\left(d\left(\log(\frac{1}{\delta})^2 + \frac{\log(\frac{1}{\delta})}{\varepsilon}\right)\right)$. □

Appendix H. Proof of Theorem 6

We first show a reduction to a normalized weighted set as follows:

Corollary A5. *Let (Q, m) be a weighted set, and let (P, w) be its corresponding normalized weighted set as computed in Observation A1. Let (P, u) be a 1-mean ε-coreset for (P, w) and let $u' = \|m\|_1 \cdot u$. Then (Q, u') is a 1-mean ε-coreset for (Q, m).*

Proof. Let (P, u) be a 1-mean ε-coreset for (P, w). We prove that (Q, u') is a 1-mean ε-coreset for (Q, m). Observe that

$$\left|\sum_{i=1}^{n} (m_i - u'_i)\|q_i - x\|^2\right| = \left|\sum_{i=1}^{n} (m_i - u'_i)\sigma^2\|p_i - y\|^2\right| \tag{A36}$$

$$= \left|\sum_{i=1}^{n} \|m\|_1 \sigma^2 (w_i - u_i)\|p_i - y\|^2\right|, \tag{A37}$$

where the first equality holds by (A1), and the second holds by the definition of w and u'. Since (P, u) is a 1-mean ε-coreset for (P, w)

$$\begin{aligned}&\left|\sum_{i=1}^{n}\|m\|_1\sigma^2(w_i-u_i)\|p_i-y\|^2\right| \\ &\leq \varepsilon\sum_{i=1}^{n}\|m\|_1\sigma^2 w_i\|p_i-y\|^2 \\ &= \varepsilon\sum_{i=1}^{n} m_i\|q_i-x\|^2,\end{aligned} \tag{A38}$$

where the equality holds by (A1) and since $w = \frac{m}{\|m\|_1}$. The proof concludes by combining (A36) and (A38) as $\left|\sum_{i=1}^{n}(m_i-u'_i)\|q_i-x\|^2\right| \leq \varepsilon\sum_{i=1}^{n} m_i\|q_i-x\|^2$. □

1-*Mean Problem Reduction*

Given a normalized weighted set (P, w) as in Definition A1, in the following lemma we prove that a weighted set (P, u) is a 1-mean ε-coreset for (P, w) if some three properties related to the mean, variance, and weights of (P, u) hold.

Lemma A6. *Let (P, w) be a normalized weighted set of n points in $\mathbb{R}^d$, $\varepsilon \in (0, 1)$, and $u \in \mathbb{R}^n$ such that,*

1. $\left\|\sum_{i=1}^{n} u_i p_i\right\| \leq \varepsilon$,
2. $\left|1-\sum_{i=1}^{n} u_i\right| \leq \varepsilon$, *and*
3. $\left|1-\sum_{i=1}^{n} u_i\cdot\|p_i\|^2\right| \leq \varepsilon$.

Then, (P, u) is a 1-mean ε-coreset for (P, w), i.e., for every $x \in \mathbb{R}^d$ we have that

$$\left|\sum_{i=1}^{n}(w_i-u_i)\|p_i-x\|^2\right| \leq 2\varepsilon\sum_{i=1}^{n} w_i\|p_i-x\|^2. \tag{A39}$$

Proof. First we have that,

$$\begin{aligned}\sum_{i=1}^{n} w_i\|p_i-x\|^2 &= \sum_{i=1}^{n} w_i\|p_i\|^2 - 2x^T\sum_{i=1}^{n} w_i p_i + \|x\|^2\sum_{i=1}^{n} w_i &\text{(A40)}\\ &= 1+\|x\|^2, &\text{(A41)}\end{aligned}$$

where the last equality holds by the attributes (a)–(c) of the normalized weighted set (P, w). By rearranging the left hand side of (A39) we get,

$$\begin{aligned}&\left|\sum_{i=1}^{n}(w_i-u_i)\|p_i-x\|^2\right| = \left|\sum_{i=1}^{n}(w_i-u_i)(\|p_i\|^2-2p_i^Tx+\|x\|^2)\right| &\text{(A42)}\\ &\leq \left|\sum_{i=1}^{n}(w_i-u_i)\|p_i\|^2\right| + \left|\|x\|^2\sum_{i=1}^{n}(w_i-u_i)\right| + \left|2x^T\sum_{i=1}^{n}(w_i-u_i)p_i\right| &\text{(A43)}\\ &= \left|1-\sum_{i=1}^{n} u_i\|p_i\|^2\right| + \|x\|^2\left|1-\sum_{i=1}^{n} u_i\right| + \left|2x^T\sum_{i=1}^{n} u_i p_i\right| &\text{(A44)}\\ &\leq \varepsilon+\varepsilon\|x\|^2+2\|x\|\left\|\sum_{i=1}^{n} u_i p_i\right\|, &\text{(A45)}\end{aligned}$$

where (A43) holds by the triangle inequality, (A44) holds by attributes (a)–(c), and (A45) holds by combining assumptions (2), (3), and the Cauchy-Schwarz inequality, respectively. We also have for every $a, b \geq 0$ that $2ab \leq a^2 + b^2$, hence,

$$2ab = 2\sqrt{\varepsilon}a\frac{b}{\sqrt{\varepsilon}} \leq \varepsilon a^2 + \frac{b^2}{\varepsilon}. \tag{A46}$$

By (A46) and assumption (1) we get that,

$$\begin{aligned} 2\|x\|\left\|\sum_{i=1}^{n} u_i p_i\right\| &\leq \varepsilon\|x\|^2 + \frac{\|\sum_{i=1}^{n} u_i p_i\|^2}{\varepsilon} \\ &\leq \varepsilon\|x\|^2 + \frac{\varepsilon^2}{\varepsilon} \\ &= \varepsilon\|x\|^2 + \varepsilon. \end{aligned} \tag{A47}$$

Lemma A6 now holds by plugging (A47) in (A45) as,

$$\begin{aligned} \left|\sum_{i=1}^{n}(w_i - u_i)\|p_i - x\|^2\right| &\leq \varepsilon + \varepsilon\|x\|^2 + \varepsilon\|x\|^2 + \varepsilon \\ &= 2\varepsilon + 2\varepsilon\|x\|^2 \\ &= 2\varepsilon(1 + \|x\|^2) \\ &= 2\varepsilon\sum_{i=1}^{n} w_i\|p_i - x\|^2, \end{aligned} \tag{A48}$$

where the last equality holds by (A41).

Observe that if assumptions (1), (2) and (3) hold, then (A48) hold. We therefore obtain an ε-coreset. □

To Proof Theorem 6, we split it into 2 claims:

Claim A7. *Let (Q, m) be a weighted set of n points in $\mathbb{R}^d$, $\varepsilon \in (0, 1)$, and let u be the output of a call to* CoreSet$(Q, m, (\frac{\varepsilon}{4})^2)$*; see Algorithm 2. Then $u = (u_1, \cdots, u_n) \in \mathbb{R}^n$ is a vector with $\|u\|_0 \leq \frac{128}{\varepsilon^2}$ non-zero entries that is computed in $O(\frac{nd}{\varepsilon^2})$ time, and (Q, u) is a 1-mean ε-coreset for (Q, m).*

Proof. Let (P, w) be the normalized weighted set that is computed at Lines 3–5 of Algorithm 2 where $P = \{p_1, \cdots, p_n\}$, and let $\tilde{u} = \frac{u}{\|m\|_1}$. We show that $(P, \tilde{u})$ is a 1-mean ε-coreset for (P, w), then by Corollary A5 we get that (Q, u) is a 1-mean coreset for (Q, m).

Let $\varepsilon' = \frac{\varepsilon}{4}$, let $p'_i := \frac{(p_i^T|1)^T}{\|(p_i^T|1)\|^2}$ and $w'_i := \frac{w_i\|(p_i^T|1)\|^2}{2}$ for every $i \in [n]$. By the definition of u' at line 11 in Algorithm 2, and since the algorithm gets ε'^2 as input, we have that

$$\|u'\|_0 \leq 8/\varepsilon'^2 = \frac{128}{\varepsilon^2}, \tag{A49}$$

and

$$\left\|\sum_{i=1}^{n}(w'_i - u'_i)p'_i\right\|^2 \leq \varepsilon'^2. \tag{A50}$$

For every $i \in [n]$ let $u_i = \|m\|_1 \cdot \frac{2u'_i}{\left\|(p_i^T|1)\right\|^2}$ be defined as at Line 12 of the algorithm. It immediately follows by the definition of $u = (u_1, \cdots, u_n)$ and (A49) that

$$\|u\|_0 \leq 128/\varepsilon'^2. \tag{A51}$$

We now prove that Properties (1)– (3) in Lemma A6 hold for $(P, \tilde{u})$. We have that

$$\begin{aligned}
2\varepsilon' &\geq 2\left\|\sum_{i=1}^{n}(w'_i - u'_i)p'_i\right\| && \text{(A52)}\\
&= 2\left\|\sum_{i=1}^{n}\frac{w_i\left\|(p_i^T \mid 1)\right\|^2 - \frac{u_i}{\|m\|_1}\left\|(p_i^T \mid 1)\right\|^2}{2}\cdot\frac{(p_i^T \mid 1)^T}{\left\|(p_i^T \mid 1)\right\|^2}\right\| \\
&= \left\|\sum_{i=1}^{n}(w_i - \tilde{u}_i)\cdot(p_i^T \mid 1)^T\right\| && \text{(A53)}\\
&= \left\|\left(\sum_{i=1}^{n}(w_i - \tilde{u}_i)\cdot p_i^T \mid \sum_{i=1}^{n}(w_i - \tilde{u}_i)\right)^T\right\| && \text{(A54)}\\
&\geq \left\|\sum_{i=1}^{n}(w_i - \tilde{u}_i)\cdot p_i\right\|, && \text{(A55)}
\end{aligned}$$

where the first derivation follows from (A50), the second holds by the definition of w'_i, u'_i, u_i and p'_i for every $i \in [n]$, the third holds since $\tilde{u} = \frac{u}{\|m\|_1}$, and the last holds since $\|(x \mid y)\| \geq \|x\|$ for every x, y such that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

By (A54) and since w is a distribution vector we also have that

$$2\varepsilon' \geq \left|\sum_{i=1}^{n}(w_i - \tilde{u}_i)\right| = \left|1 - \sum_{i=1}^{n}\tilde{u}_i\right|. \tag{A56}$$

By Theorem 1, we have that u' is a distribution vector, which yields,

$$2 = 2\sum_{i=1}^{n}u'_i = \sum_{i=1}^{n}\tilde{u}_i\left\|(p_i^T \mid 1)^T\right\|^2 = \sum_{i=1}^{n}\tilde{u}_i\|p_i\|^2 + \sum_{i=1}^{n}\tilde{u}_i,$$

By the above we get that $2 - \sum_{i=1}^{n}\tilde{u}_i = \sum_{i=1}^{n}\tilde{u}_i\|p_i\|^2$. Hence,

$$\begin{aligned}
\left|\sum_{i=1}^{n}(w_i - \tilde{u}_i)\|p_i\|^2\right| &= \left|\sum_{i=1}^{n}w_i\|p_i\|^2 - (2 - \sum_{i=1}^{n}\tilde{u}_i)\right| \\
= \left|1 - (2 - \sum_{i=1}^{n}\tilde{u}_i)\right| &= \left|\sum_{i=1}^{n}\tilde{u}_i - 1\right| \leq 2\varepsilon'
\end{aligned} \tag{A57}$$

where the first equality holds since $\sum_{i=1}^{n}\tilde{u}_i\|p_i\|^2 = 2 - \sum_{i=1}^{n}\tilde{u}_i$, the second holds since w is a distribution and the last is by (A56). Now by (A57), (A56) and (A55) we obtain that $(P, \tilde{u}_i)$ satisfies Properties (1)–(3) in Lemma A6. Hence, by Lemma A6 and Corollary A5 we get that

$$\begin{aligned}
\left|\sum_{i=1}^{n}(w_i - u_i)\|p_i - x\|^2\right| &\leq 4\varepsilon'\sum_{i=1}^{n}w_i\|p_i - x\|^2 \\
&= \varepsilon\sum_{i=1}^{n}w_i\|p_i - x\|^2.
\end{aligned}$$

The running time is the running time of Algorithm 1 with ε^2 instead of ε, i.e., $O(nd/\varepsilon^2)$. □

Now we proof the following claim:

Claim A8. *Let (Q, m) be a weighted set of n points in $\mathbb{R}^d$, $\varepsilon \in (0,1)$. Then in $O(nd + d \cdot \frac{\log(n)^2}{\varepsilon^4})$ we can compute a vector $u = (u_1, \cdots, u_n)^T \in \mathbb{R}^n$, such that u has $\|u\|_0 \leq \frac{128}{\varepsilon^2}$ non-zero entries, and (Q, u) is a 1-mean (2ε)-coreset for (Q, m).*

Proof. The Claim immediately holds by using Algorithm 2 with a small change. We change Line 11 in Algorithm 2 to use Algorithm 3 and Theorem 3, instead of Algorithm 1 and Theorem 1. □

Combining both Claim A7 with Claim A8 proves Theorem 6.

Appendix I. Proof of Corollary 7

Proof. We consider the variables defined in Algorithm 5. Let $X \in \mathbb{R}^{d \times (d-k)}$ such that $X^T X = I$, and let $A' = [A|(r, \cdots, r)^T]$. Plugging $A = A'$ into Theorem 3 at [35]

$$\left| 1 - \frac{\|WA'X\|^2}{\|A'X\|^2} \right| \leq 5 \left\| \sum_{i=1}^{n} \tilde{v}_i - W_{i,i}^2 \tilde{v}_i \right\|. \tag{A58}$$

We also have by the definition of W and Theorem 2

$$\left\| \sum_{i=1}^{n} \tilde{v}_i - W_{i,i}^2 \tilde{v}_i \right\| \leq (\varepsilon/k) \sqrt{\sum_{i=1}^{n} \|\tilde{v}_i\|^2} \leq (\varepsilon/k) \sum_{i=1}^{n} \|\tilde{v}_i\|, \tag{A59}$$

where the first inequality holds since $W_{i,i} = u_i^2$ for every $i \in [n]$, and the vector $u \in \mathbb{R}^n$ is a vector summarization $(\varepsilon/5k)^2$-coreset for $(\{\tilde{v}_1, \cdots, \tilde{v}_n\}, (1, \cdots, 1))$.

Finally, at [35] they show that $(\varepsilon/5k)\sum_{i=1}^{n} \|\tilde{v}_i\| \leq \varepsilon$. Hence, combining this fact with (A58), and (A59) yields

$$\left| 1 - \frac{\|WA'X\|^2}{\|A'X\|^2} \right| \leq \varepsilon. \tag{A60}$$

Finally, the corollary holds by combing Lemma 4.1 at [45] with (A60). □

References

1. Valiant, L.G. A theory of the learnable. *Commun. ACM* **1984**, *27*, 1134–1142. [CrossRef]
2. Vapnik, V. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*; Morgan-Kaufmann: Denver, CO, USA, 1992; pp. 831–838.
3. Feldman, D.; Langberg, M. A unified framework for approximating and clustering data. In Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, San Jose, CA, USA, 6–8 June 2011; pp. 569–578.
4. Nielsen, M.A. *Neural Networks and Deep Learning*; Determination Press: San Francisco, CA, USA, 2015; Volume 2018.
5. Steinwart, I.; Christmann, A. *Support Vector Machines*; Springer Science & Business Media: Berlin, Germany, 2008.
6. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B (Methodol.)* **1996**, *58*, 267–288. [CrossRef]
7. Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009.
8. Hoerl, A.E.; Kennard, R.W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* **1970**, *12*, 55–67. [CrossRef]
9. Bergman, S. *The Kernel Function and Conformal Mapping*; American Mathematical Soc.: Providence, RI, USA, 1970; Volume 5.
10. Eggleston, H.G. Convexity. *J. Lond. Math. Soc.* **1966**, *1*, 183–186. [CrossRef]
11. Phillips, J.M. Coresets and sketches. *arXiv* **2016**, arXiv:1601.00617.
12. Har-Peled, S. *Geometric Approximation Algorithms*; Number 173; American Mathematical Soc.: Providence, RI, USA, 2011.
13. Vapnik, V. *The Nature of Statistical Learning Theory*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.

14. Langberg, M.; Schulman, L.J. Universal ε-approximators for integrals. In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, Austin, TX, USA, 17 January 2010; pp. 598–607.
15. Carathéodory, C. Über den Variabilitätsbereich der Koeffizienten von Potenzreihen, die gegebene Werte nicht annehmen. *Math. Ann.* **1907**, *64*, 95–115. [CrossRef]
16. Cook, W.; Webster, R. Caratheodory's theorem. *Can. Math. Bull.* **1972**, *15*, 293. [CrossRef]
17. Phillips, J.M.; Tai, W.M. Near-optimal coresets of kernel density estimates. *Discret. Comput. Geom.* **2020**, *63*, 867–887. [CrossRef]
18. Matousek, J. Approximations and optimal geometric divide-and-conquer. *J. Comput. Syst. Sci.* **1995**, *50*, 203–208. [CrossRef]
19. Braverman, V.; Feldman, D.; Lang, H. New frameworks for offline and streaming coreset constructions. *arXiv* **2016**, arXiv:1612.00889.
20. Bentley, J.L.; Saxe, J.B. Decomposable searching problems I: Static-to-dynamic transformation. *J. Algorithms* **1980**, *1*, 301–358. [CrossRef]
21. Har-Peled, S.; Mazumdar, S. On coresets for k-means and k-median clustering. In Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, 13 June 2004; pp. 291–300.
22. Maalouf, A.; Jubran, I.; Feldman, D. Fast and accurate least-mean-squares solvers. *arXiv* **2019**, arXiv:1906.04705.
23. Drineas, P.; Magdon-Ismail, M.; Mahoney, M.W.; Woodruff, D.P. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res.* **2012**, *13*, 3475–3506.
24. Cohen, M.B.; Peng, R. Lp row sampling by lewis weights. In Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, Portland, OR, USA, 4 June 2015; pp. 183–192.
25. Ritter, K. *Average-Case Analysis of Numerical Problems*; Springer: Berlin/Heidelberg, Germany, 2007.
26. Juditsky, A.; Nemirovski, A.S. Large deviations of vector-valued martingales in 2-smooth normed spaces. *arXiv* **2008**, arXiv:0809.0813.
27. Tropp, J.A. An introduction to matrix concentration inequalities. *arXiv* **2015**, arXiv:1501.01571.
28. Charikar, M.; Chen, K.; Farach-Colton, M. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 693–703.
29. Feldman, D.; Ozer, S.; Rus, D. Coresets for vector summarization with applications to network graphs. In Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 17 July 2017; Volume 70, pp. 1117–1125.
30. Węglarczyk, S. Kernel density estimation and its application. In *ITM Web of Conferences*; EDP Sciences: Les Ulis, France, 2018; Volume 23.
31. Zheng, Y.; Jestes, J.; Phillips, J.M.; Li, F. Quality and efficiency for kernel density estimates in large data. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22 June 2013; pp. 433–444.
32. Bachem, O.; Lucic, M.; Krause, A. Scalable k-means clustering via lightweight coresets. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19 July 2018; pp. 1119–1127.
33. Barger, A.; Feldman, D. k-Means for Streaming and Distributed Big Sparse Data. In Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, FL, USA, 30 June 2016; pp. 342–350.
34. Feldman, D.; Schmidt, M.; Sohler, C. Turning Big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. *arXiv* **2018**, arXiv:1807.04518.
35. Feldman, D.; Volkov, M.; Rus, D. Dimensionality reduction of massive sparse datasets using coresets. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 2766–2774.
36. Cohen, M.B.; Elder, S.; Musco, C.; Musco, C.; Persu, M. Dimensionality reduction for k-means clustering and low rank approximation. In Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, Portland, OR, USA, 14 June 2015; pp. 163–172.
37. Varadarajan, K.; Xiao, X. On the sensitivity of shape fitting problems. *arXiv* **2012**, arXiv:1209.4893.
38. Feldman, D.; Tassa, T. More constraints, smaller coresets: Constrained matrix approximation of sparse big data. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10 August 2015; pp. 249–258.
39. Frieze, A.; Kannan, R.; Vempala, S. Fast Monte-Carlo algorithms for finding low-rank approximations. *J. ACM (JACM)* **2004**, *51*, 1025–1041. [CrossRef]
40. Yang, J.; Chow, Y.L.; Ré, C.; Mahoney, M.W. Weighted SGD for ℓ_p regression with randomized preconditioning. *J. Mach. Learn. Res.* **2017**, *18*, 7811–7853.
41. Cohen, M.B.; Lee, Y.T.; Musco, C.; Musco, C.; Peng, R.; Sidford, A. Uniform sampling for matrix approximation. In Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, Rehovot, Israel, 11 January 2015; pp. 181–190.
42. Papailiopoulos, D.; Kyrillidis, A.; Boutsidis, C. Provable deterministic leverage score sampling. In Proceedings of the 20th ACM SIGKDD iInternational Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24 August 2014; pp. 997–1006.
43. Drineas, P.; Mahoney, M.W.; Muthukrishnan, S. Relative-error CUR matrix decompositions. *SIAM J. Matrix Anal. Appl.* **2008**, *30*, 844–881. [CrossRef]
44. Cohen, M.B.; Musco, C.; Musco, C. Input sparsity time low-rank approximation via ridge leverage score sampling. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, Barcelona, Spain, 16 January 2017; pp. 1758–1777.

45. Maalouf, A.; Statman, A.; Feldman, D. Tight sensitivity bounds for smaller coresets. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 23 August 2020; pp. 2051–2061.
46. Batson, J.; Spielman, D.A.; Srivastava, N. Twice-ramanujan sparsifiers. *SIAM J. Comput.* **2012**, *41*, 1704–1721. [CrossRef]
47. Cohen, M.B.; Nelson, J.; Woodruff, D.P. Optimal approximate matrix product in terms of stable rank. *arXiv* **2015**, arXiv:1507.02268.
48. Clarkson, K.L. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms (TALG)* **2010**, *6*, 63. [CrossRef]
49. Desai, A.; Ghashami, M.; Phillips, J.M. Improved practical matrix sketching with guarantees. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 1678–1690. [CrossRef]
50. Madariaga, D.; Madariaga, J.; Bustos-Jiménez, J.; Bustos, B. Improving Signal-Strength Aggregation for Mobile Crowdsourcing Scenarios. *Sensors* **2021**, *21*, 1084. [CrossRef]
51. Mahendran, N.; Vincent, D.R.; Srinivasan, K.; Chang, C.Y.; Garg, A.; Gao, L.; Reina, D.G. Sensor-assisted weighted average ensemble model for detecting major depressive disorder. *Sensors* **2019**, *19*, 4822. [CrossRef]
52. Wu, L.; Xu, Q.; Heikkilä, J.; Zhao, Z.; Liu, L.; Niu, Y. A star sensor on-orbit calibration method based on singular value decomposition. *Sensors* **2019**, *19*, 3301. [CrossRef]
53. Yang, W.; Hong, J.Y.; Kim, J.Y.; Paik, S.h.; Lee, S.H.; Park, J.S.; Lee, G.; Kim, B.M.; Jung, Y.J. A novel singular value decomposition-based denoising method in 4-dimensional computed tomography of the brain in stroke patients with statistical evaluation. *Sensors* **2020**, *20*, 3063. [CrossRef]
54. Peri, E.; Xu, L.; Ciccarelli, C.; Vandenbussche, N.L.; Xu, H.; Long, X.; Overeem, S.; van Dijk, J.P.; Mischi, M. Singular value decomposition for removal of cardiac interference from trunk electromyogram. *Sensors* **2021**, *21*, 573. [CrossRef]
55. Code. Open Source Code for All the Algorithms Presented in This Paper. 2021. Available online: https://github.com/alaamaalouf/vector-summarization-coreset (accessed on 29 September 2021).
56. Van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
57. Oliphant, T.E. *A Guide to NumPy*; Trelgol Publishing USA: 2006; Volume 1. Available online: https://ecs.wgtn.ac.nz/foswiki/pub/Support/ManualPagesAndDocumentation/numpybook.pdf (accessed on 29 September 2021).
58. Tremblay, N.; Barthelmé, S.; Amblard, P.O. Determinantal Point Processes for Coresets. *J. Mach. Learn. Res.* **2019**, *20*, 1–70.
59. Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available online: http://archive.ics.uci.edu/ml (accessed on 29 September 2021)
60. Donovan, B.; Work, D. Using Coarse GPS Data to Quantify City-Scale Transportation System Resilience to Extreme Events. 2015. Available online: http://vis.cs.kent.edu/DL/Data/ (accessed on 29 September 2021).
61. US Census Data (1990) Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990) (accessed on 10 June 2021).
62. Kawala, F.; Douzal-Chouakria, A.; Gaussier, E.; Dimert, E. Prédictions D'activité dans les Réseaux Sociaux en Ligne. 2013. Available online: https://archive.ics.uci.edu/ml/datasets/Buzz+in+social+media+ (accessed on 29 September 2021).
63. Huerta, R.; Mosqueiro, T.; Fonollosa, J.; Rulkov, N.F.; Rodriguez-Lujan, I. Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring. *Chemom. Intell. Lab. Syst.* **2016**, *157*, 169–176. [CrossRef]
64. Chen, X. A new generalization of Chebyshev inequality for random vectors. *arXiv* **2007**, arXiv:0707.0805.
65. Minsker, S. Geometric median and robust estimation in Banach spaces. *Bernoulli* **2015**, *21*, 2308–2335. [CrossRef]