algorithms

MDPI

Article

Deterministic Coresets for *k*-Means of Big Sparse Data †

Artem Barger ‡ and Dan Feldman *,‡

Department of Computer, Science University of Haifa, Haifa 32000, Israel; artem@bargr.net

* Correspondence: dfeldman@univ.haifa.ac.il

† This paper is an extended version of our paper published in the proceedings of Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, FL, USA, 5–7 May 2016.

‡ These authors contributed equally to this work.

Received: 3 January 2020; Accepted: 1 April 2020; Published: 14 April 2020

Abstract: Let P be a set of n points in $\mathbb{R}^d$, $k \geq 1$ be an integer and $\varepsilon \in (0,1)$ be a constant. An *ε-coreset* is a subset $C \subseteq P$ with appropriate non-negative weights (scalars), that approximates *any* given set $Q \subseteq \mathbb{R}^d$ of k centers. That is, the sum of squared distances over every point in P to its closest point in Q is the same, up to a factor of $1 \pm \varepsilon$ to the weighted sum of C to the same k centers. If the coreset is small, we can solve problems such as k-means clustering or its variants (e.g., discrete k-means, where the centers are restricted to be in P, or other restricted zones) on the small coreset to get faster provable approximations. Moreover, it is known that such coreset support streaming, dynamic and distributed data using the classic merge-reduce trees. The fact that the coreset is a subset implies that it preserves the sparsity of the data. However, existing such coresets are randomized and their size has at least linear dependency on the dimension d. We suggest the first such coreset of size *independent* of d. This is also the first deterministic coreset construction whose resulting size is not exponential in d. Extensive experimental results and benchmarks are provided on public datasets, including the first coreset of the English Wikipedia using Amazon's cloud.

Keywords: coreset; clustering; KMeans; big data; streaming

1. Background

Given a set of n points in $\mathbb{R}^d$, and an error parameter $\varepsilon > 0$, a coreset in this paper is a small set of weighted points in $\mathbb{R}^d$, such that the sum of squared distances from the original set of points to any set of k centers in $\mathbb{R}^d$ can be approximated by the sum of weighted squared distances from the points in the coreset. The output of running an existing clustering algorithm on the coreset would then yield approximation to the output of running the same algorithm on the original data, by the definition of the coreset.

Coresets were first suggested by [1] as a way to improve the theoretical running time of existing algorithms. Moreover, a coreset is a natural tool for handling Big Data using all the computation models that are mentioned in the previous section. This is mainly due to the merge-and-reduce tree approach that was suggested by [2,3] and is formalized by [4]: coresets can be computed independently for subsets of input points, e.g., on different computers, and then be merged and re-compressed again. Such a binary compression tree can also be computed using one pass over a possibly unbounded streaming set of points, where in every moment only $O(\log n)$ coresets exist in memory for the n points streamed so far. Here the coreset is computed only on small chunks of points, so a possibly inefficient coreset construction still yields efficient coreset constructions for large sets; see Figure 1. Note that the coreset guarantees are preserved while using this technique, while no assumptions are made on the order of the streaming input points. These coresets can be computed independently and in parallel via

M machines (e.g., on the cloud) and reduce the running time by a factor of M. The communication between the machines is also small, since each machine needs to communicate to a main server only the coreset of its data.

In practice, this technique can be implemented easily using the map-reduce approach of modern software for Big Data such as [5].

Coresets can also be used to support the dynamic model where points are deleted/inserted. Here the storage is linear in n since we need to save the tree in memory (practically, on the hard drive), however the update time is only logarithmic in n since we need to reconstruct only $O(\log n)$ coresets that correspond to the deleted/inserted point along the tree. First such coreset of size *independent* of d was introduced by [6]. See [1,2] for details.

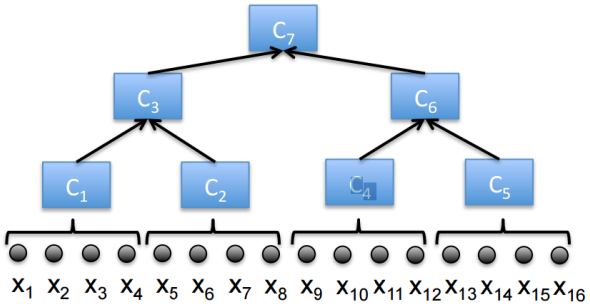


Figure 1. Coresets construction from data streams [2]. The black arrows indicate "merge-and-reduce" operations. The intermediate coresets $C_1, \ldots, C_7$ are numbered in the order in which they would be generated in the streaming case. In the parallel case, C_1, C_2, C_4 and C_5 would be constructed in parallel, followed by C_3 and C_6, finally resulting in C_7. The Figure is from [7].

Constrained k-Means and Determining k

Since the coreset approximates every set of k centers, it can also be used to solve the k-means problem under different constraints (e.g., allowed areas for placing the centers) or given a small set of candidate centers. In addition, the set of centers may contain duplicate points, which means that the coreset can approximate the sum of squared distances for $k' < k$ centers. Hence, coresets can be used to choose the right number k of centers up to k', by minimizing the sum of squared distances plus $f(k)$ for some function of k. Full opensource is available [8].

2. Related Work

We summarize existing coresets constructions for k-means queries, as will be formally defined in Section 4.

Importance Sampling

Following a decade of research, coreset of size polynomial in d were suggested by [9]. Ref [10] suggested an improved version of size $O(dk^2/\varepsilon^2)$ in which is a special case of the algorithms proposed by [11]. The construction is based on computing an approximation to the k-means of the input points (with no constraints on the centers) and then sample points proportionally to their distance to these centers. Each chosen point is then assigned a weight that is inverse proportional to this distance. The probability of failure in these algorithms reduces exponentially with the input size. Coresets of size $O(dk/\varepsilon^2)$, i.e., linear in k, were suggested in work of [12], however the weight of a point may be negative or a function of the given query. For the special case $k = 1$, Inaba et al. [13], provided constructions of coresets of size $O(1/\varepsilon^2)$ using uniform sampling.

Projection based coresets. Data summarization which are similar to coresets of size $O(k/\varepsilon)$ that are based on projections on low-dimensional subspaces that diminishes the sparsity of the input data were suggested by [14] by improving the analysis of [4]. Recently [15] improves both on [4,14] by applying Johnson-Lindenstrauss Lemma [16] on construction from [4]. However, due to the

projections, the resulting summarizations of all works mentioned above are not subset of the input points, unlike the coreset definition of this paper. In particular, for sparse data sets such as adjacency matrix of a graphs, documents-term matrix of Wikipedia, or images-objects matrix, the sparsity of the data diminishes and a single point in the summarization might be larger than the complete sparse input data.

Other type of weak coresets approximates only the optimal solution, and not every k centers. Such randomized coresets of size independent of d and only polynomial in k were suggested by [11] and simplified by [12].

Deterministic Constructions. The first coresets for k-means [2,17] were based on partitioning the data into cells, and take a representative point from each cell to the coreset, as is done in hashing or Hough transform [18]. However, these coresets are of size at least $k/\varepsilon^{O(d)}$, i.e., exponential in d, while still providing result which is a sub-set of the the input in contrast to previous work [17]. While, our technique is most related to the deterministic construction that was suggested in [4] by recursively computing k-means clustering of the input points. While the output set has size independent of d, it is not a coreset as defined in this paper, since it is not a subset of the input and thus cannot handle sparse data as explained above. Techniques such as uniform sampling for each cluster yields coresets with probability of failure that is linear in the input size, or whose size depends on d.

m-means is a coreset for k-means? A natural approach for coreset construction, that is strongly related to this paper, is to compute the m-means of the input set P, for a sufficiently large m, where the weight of each center is the number of point in its cluster. If the sum of squared distances to these m-centers is about ε factor from the k-means, we obtain a (k,ε)-coreset by (a weaker version of) the triangle inequality. Unfortunately, it was recently proved in [19] that there exists sets such that $m \in k^{\Omega(d)}$ centers are needed in order to obtain this small sum of squared distances.

3. Our Contribution

We suggest the following deterministic algorithms:

1. An algorithm that computes a $(1+\varepsilon)$ approximation for the k-means of a set P that is distributed (partitioned) among M machines, where each machine needs to send only $k^{O(1)}$ input points to the main server at the end of its computation.
2. A streaming algorithm that, after one pass over the data and using $k^{O(1)} \log n$ memory returns an $O(\log n)$-approximation to the k-means of P. The algorithm can run "embarrassingly in parallel [20] on data that is distributed among M machines, and support insertion/deletion of points as explained in the previous section.
3. Description of how to use our algorithm to boost both the running time and quality of any existing k-means heuristic using only the heuristic itself, even in the classic off-line setting.
4. Extensive experimental results on real world data-sets. This includes the first k-means clustering with provable guarantees for the English Wikipedia, via 16 EC2 instances on Amazon's cloud.
5. Open-code for for fully reproducing our results and for the benefit of the community. To our knowledge, this is the first coreset code that can run on the cloud without additional commercial packages.

3.1. Novel Approach: m-Means Is A Coreset for k-Means, for Smart Selection of m

One of our main technical result is that for every constant $\varepsilon > 0$, there exists an integer $m \in k^{O(1)}$ such that the m-means of the input set (or its approximation) is a (k,ε)-coreset; see Theorem 2. However, simply computing the m-means of the input set for a sufficiently large m might yield m that is exponential in d, as explained by [19] and the related work. Instead, Algorithm 1 carefully selects the right m between k and $k^{O(1)}$ after checking the appropriate conditions in each iteration.

3.2. Solving k-Means Using k-Means

It might be confusing that we suggest to solve the k-means problem by computing m-means for $m > k$. In fact, most of the coreset constructions actually solve the optimal problem in one of their first construction steps. The main observation is that we never run the coreset on the complete input of n (or unbounded stream of) points, but only on small subsets of size $\sim 2m$. This is since our coresets are composable and can be merged (to $2m$ points) and reduced back to m using the merge-and-reduce tree technique. The composable property follows from the fact that the coreset approximates the sum of squared distances to *every* k-centers, and not just the k-means for the subset at hand.

3.3. Running Time

Running time that is exponential in k is unavoidable for any $(1+\varepsilon)$-approximation algorithm that solves k-means, even in the planar case ($d = 2$) [21] and $\varepsilon < 0.1$ [4]. Our main contributions is a coreset construction that uses memory that is independent of d and running time that is *near-linear* in n. To our knowledge this is an open problem even for the case $k = 2$. Nevertheless, for large values of ε (e.g., $\varepsilon > 10$) we may use existing constant factor approximations that takes time polynomial in k to compute our coreset in time that is near-linear in n but also polynomial in k.

In practice, provable $(1+\varepsilon)$-approximations for k-means are rarely used due to the lower bounds on the running times above. Instead, heuristics are used. Indeed, in our experimental results we evaluate this approach instead of computing the optimal k-means during the coreset construction and on the resulting coreset itself.

4. Notation and Main Result

The input to our problem is a set P' of n points in $\mathbb{R}^d$, where each point $p \in P'$ includes a multiplicative weight $u(p) > 0$. In addition, there is an additive weight $\rho > 0$ for the set. Formally, a *weighted set* in $\mathbb{R}^d$ is a tuple $P = (P', u, \rho)$, where $P' \subseteq \mathbb{R}^d$, $u : P' \to [0, \infty)$. In particular, an *unweighted* set has a unit weight $u(p) = 1$ for each point, and a zero additive weight.

4.1. k-Means Clustering

For a given set $Q = \{q_1, \cdots, q_k\}$ of $k \geq 1$ centers (points) in $\mathbb{R}^d$, the Euclidean distance from a point $p \in \mathbb{R}^d$ to its closest center in Q is denoted by $1.7em(p, Q) = \min_{q \in Q} \|p - q\|_2$. The sum of these weighted squared distances over the points of P is denoted by

$$\text{cost}(P, Q) := \sum_{p \in P'} u(p) \cdot 1.7em^2(p, Q) + \rho.$$

If P is an unweighted set, this cost is just the sum of squared distances over each point in P' to its closest center in Q.

Let P'_i denote the subset of points in P' whose closest center in Q is q_i, for every $i \in [m] = \{1, \cdots, m\}$. Ties are broken arbitrarily. This yields a partition $\{P'_1, \cdots, P'_k\}$ of P' by Q. More generally, the *partition of P by Q* is the set $\{P_1, \cdots, P_k\}$ where $P_i = (P'_i, u_i, \rho/k)$, and $u_i(p) = u(p)$ for every $p \in P'_i$ and every $i \in [k]$.

A set Q_k that minimizes this weighted sum $\text{cost}(P, Q)$ over every set Q of k centers in $\mathbb{R}^d$ is called the *k-means* of P. The 1-means $\mu(P)$ of P is called the centroid, or the center of mass, since

$$\mu(P) = \frac{1}{\sum_{p' \in P'} u(p')} \sum_{p \in P'} u(p) \cdot p.$$

We denote the cost of the k-means of P by $\text{opt}(P, k) := \text{cost}(P, Q_k)$.

4.2. Coreset

Computing the k-means of a weighted set P is the main motivation of this paper. Formally, let $\varepsilon > 0$ be an error parameter. The weighted set $S = (S', w, \phi)$ is a (k, ε)-coreset for P, if for every set $Q \subset \mathbb{R}^d$ of $|Q| = k$ centers we have

$$(1-\varepsilon)\text{cost}(P,Q) \leq \text{cost}(S,Q) \leq (1+\varepsilon)\text{cost}(P,Q),$$

That is,

$$(1-\varepsilon)\sum_{p\in P} u(p)1.7em^2(p,Q) + \rho \leq \sum_{p\in S} w(p)\cdot 1.7em^2(p,Q) + \phi \leq (1+\varepsilon)\sum_{p\in P} u(p)\cdot 1.7em^2(p,Q) + \rho.$$

To handle streaming data we will need to compute "coresets for union of coresets", which is the reason that we assume that both the input P and its coreset S are weighted sets.

4.3. Sparse Coresets

Unlike previous work, we add the constraints that if each point in P is sparse, i.e., has few non-zeroes coordinates, then the set S will also be sparse. Formally, the *maximum sparsity* $s(P) := \max_{p\in P} \|p\|_0$ of P is the maximum number of non zeroes entries $\|p\|_0 = |\{i \mid p_i \neq 0, i \in [d]\}|$ over every point p in P.

In particular, if each point in S is a linear combination of at most α points in P, then $s(S) \leq \alpha \cdot s(P)$. In addition, we would like that the set S will be of size independent of both $n = |P|$ and d.

We can now state the main result of this paper.

Theorem 1 (Small sparse coreset)**.** *For every weighted set $P = (P', u, \rho)$ in $\mathbb{R}^d$, $\varepsilon > 0$ and an integer $k \geq 1$, there is a (k, ε)-coreset $S = (S', w, \phi)$ of size $|S| = k^{O(1/\varepsilon^2)}$ where each point in S' is a linear combination of $O(1/\varepsilon^2)$ points from P'. In particular, the maximum sparsity of S' is $s(P)/\varepsilon^2$.*

By plugging this result to the traditional merge-and-reduce tree in Figure 1, it is straight-forward to compute a coreset using one pass over a stream of points.

Corollary 1. *A $(k, \varepsilon \log n)$ coreset (S, w, ϕ) of size $|S| = \log(n)\cdot k^{O(1/\varepsilon^2)}$ and maximum sparsity $s(P)/\varepsilon^2$ can be computed for the set P of the n points seen so far in an unbounded stream, using $|S|\cdot s(P)/\varepsilon^2$ memory words. The insertion time per point in the stream is $\log(n)\cdot 2^{(k/\varepsilon)^{O(1)}}$. If the stream is distributed uniformly to M machines, then the amortized insertion time per point is reduced by a (multiplicative) factor of M to $(1/M)\log(n)\cdot 2^{(k/\varepsilon)^{O(1)}}$. The coreset for the union of streams can then be computed by communicating the M coresets to a main server.*

5. Coreset Construction

Our main coreset construction algorithm k-MEAN-CORESET(P, k, ε) gets a set P as input, and returns a (k, ε)-coreset (S, w); see Algorithm 1.

To obtain running time that is linear in the input size, without loss of generality, we assume that P has $|P| = k^{O(1/\varepsilon^2)}$ points, and that the cardinality of the output S is $|S| \leq |P|/2$. This is thanks to the traditional merge-and-reduce approach: given a stream of n points, we apply the coreset construction only on subsets of size $2\cdot|S|$ from P during the streaming and reduce them by half. See Figure 1 and e.g., [4,7] for details.

Algorithm Overview

In Line 1 we compute the smallest integer $m = k^t$ such that the cost $\text{opt}(P, m)$ of the m-means of P is close to the cost $\text{opt}(P, mk)$ of the (mk)-means of P. In Line 3 we compute the corresponding partition $\{P_1, \cdots, P_m\}$ of P by its m-means $Q_m = \{q_1, \cdots, q_m\}$. In Line 5 a $(1, \varepsilon)$-sparse coreset S_i of

size $O(1/\varepsilon^2)$ is computed for every P_i, $i \in [m]$. This can be done deterministically e.g., by taking the mean of P_i as explained in Lemma 1 or by using a gradient descent algorithm, namely Frank-Wolfe, as explained in [22] which also preserves the sparsity of the coreset as desired by our main theorem. The output of the algorithm is the union of the means of all these coresets, with appropriate weight, which is the size of the coreset.

The intuition behind the algorithm stems from the assumption that m-means clustering will have lower cost than k-means and this is actually supported by series of previous work [23,24]. In fact our experiments in Section 7 evidence that in practice it actually works better than anticipated by theoretical bounds.

Algorithm 1: k-MEAN-CORESET(P, k, ε)

Input: A weighted set P of points in $\mathbb{R}^d$, integer $k \geq 1$, and error parameter $\varepsilon \in (0, 1/4)$.
Output: A (k, ε)-coreset S for P that satisfies Theorem 1.

Compute the smallest integer $t \geq 0$ such that $\text{opt}(P, k^t) - \text{opt}(P, k^{t+1}) \leq \varepsilon^2 \cdot \text{opt}(P, k)$
/* $\text{opt}(P, m)$ is the m-mean cost of P. */
Set $m \leftarrow k^t$
Set $\{P_1, \ldots, P_m\} \leftarrow$ the partition of P by $\text{opt}(P, m)$
for $i := 1$ **to** m **do**
    Compute a $(1, \varepsilon)$-coreset $S_i = (S'_i, w_i, \phi_i)$ for P_i
    Set $w(\mu(S_i)) \leftarrow \sum_{p \in S'_i} w_i(p)$
Set $S' \leftarrow \bigcup_{i=1}^{m} \mu(S_i)$
Set $\phi \leftarrow \sum_{i=1}^{m} \text{cost}(S_i, \mu(S_i))$
Set $S \leftarrow (S', w, \phi)$
return S

6. Proof of Correctness

The first lemma states the common fact that the sum of squared distances of a set of point to a center is the sum of their squared distances to their center of mass, plus the squared distance to the center (the variance of the set).

Lemma 1. *For every* $x \in \mathbb{R}^d$

$$\text{cost}(P, x) = \text{cost}(P, \mu(P)) + \|\mu(P) - x\|^2 \sum_{p \in P} u(p).$$

Proof. We have

$$\begin{aligned}\text{cost}(P, x) - \rho &= \sum_{p \in P} u(p) \|p - x\|^2 = \sum_{p \in P} u(p) \|(p - \mu(P)) + (\mu(P) - x)\|^2 \\ &= \sum_{p \in P} u(p) \|p - \mu(P)\|^2 + \sum_{p \in P} u(p) \|\mu(P) - x\|^2 + 2(\mu(p) - x) \cdot \sum_{p \in P} u(p)(p - \mu(P)).\end{aligned}$$

The last term equals zero since $\mu(P) = \dfrac{1}{\sum_{p' \in P} u(p')} \cdot \sum_{p \in P} u(p) \cdot p$, and thus

$$\sum_{p \in P} u(p)(p - \mu(P)) = \sum_{p \in P} u(p) \cdot p - \sum_{p \in P} u(p)\mu(P) = \sum_{p \in P} u(p) \cdot p - \sum_{p \in P} u(p) \cdot p = 0.$$

Hence,

$$\text{cost}(P,x) = \rho + \sum_{p\in P} u(p)\,\|p-\mu(P)\|^2 + \sum_{p\in P} u(p)\,\|\mu(P)-x\|^2 = \text{cost}(P,\mu(P)) + \|\mu(P)-x\|^2 \sum_{p\in P} u(p).$$

□

The second lemma shows that assigning all the points of P to the closest center in Q yields a small multiplicative error if the 1-mean and the k-means of P has roughly the same cost. If $t = 0$, this means that we can approximate $\text{cost}(P,Q)$ using only one center in the query; see Line 1 of Algorithm 1. Note that $(1+2\varepsilon)/(1-2\varepsilon) \leq 1+4\varepsilon$ for $\varepsilon < 1/4$.

Lemma 2. *For every set $Q \subseteq \mathbb{R}^d$ of $|Q| = k$ centers we have*

$$\text{cost}(P,Q) \leq \min_{q\in Q} \text{cost}(P,\{q\}) \leq \text{cost}(P,Q)\cdot\frac{1+2\varepsilon}{1-2\varepsilon} + \frac{\text{opt}(P,1)-\text{opt}(P,k)}{(1-2\varepsilon)\varepsilon}. \tag{1}$$

Proof. Let q^* denote a center that minimizes $\text{cost}(P,\{q\})$ over $q \in Q$. The left inequality of (1) is then straight-forward since

$$\begin{aligned}\text{cost}(P,Q) - \min_{q\in Q}\text{cost}(P,\{q\}) &= \sum_{p\in P'} \min_{q\in Q} u(p)\,\|p-q\|^2 - \sum_{p\in P'} u(p)\,\|p-q^*\|^2 \\ &= \sum_{p\in P'} u(p)\left(\min_{q\in Q}\|p-q\|^2 - \|p-q^*\|^2\right) \leq 0.\end{aligned} \tag{2}$$

It is left to prove the right inequality of (1). Indeed, for every $p \in P'$, let $q_p \in Q$ denote the closest point to p in Q. Ties are broken arbitrarily. Hence,

$$\min_{q\in Q}\text{cost}(P,\{q\}) - \text{cost}(P,Q) = \sum_{p\in P'} u(p)\,\|p-q^*\|^2 - \sum_{p\in P'} u(p)\,\|p-q_p\|^2.$$

Let $\{P_1,\cdots,P_k\}$ denote the partition of P by $Q = \left\{q^1,\cdots,q^k\right\}$, where P_i are the closest points to q^i for every $i \in [k]$; see Section 4. For every $p \in P'_i$, let $q^*_p = \mu(P_i)$. Hence,

$$\sum_{p\in P'_i} u(p)\,\|p-q^*\|^2 - \sum_{p\in P'_i} u(p)\,\|p-q_p\|^2 \tag{3}$$

$$= \sum_{p\in P'_i} u(p)\,\|p-\mu(P_i)\|^2 + \|\mu(P_i)-q^*\|^2 \sum_{p\in P'_i} u(p) \tag{4}$$

$$- \left(\sum_{p\in P'_i} u(p)\,\|p-\mu(P_i)\|^2 + \left\|\mu(P_i)-q^i\right\|^2 \sum_{p\in P'_i} u(p)\right) \tag{5}$$

$$= \sum_{p\in P'_i} u(p)\left(\left\|q^*_p-q^*\right\|^2 - \left\|q^*_p-q_p\right\|^2\right), \tag{6}$$

where in (4) and (5) we substituted $x = q^*$ and $x = q_p$ respectively in Lemma 1, and in (6) we use the fact that $q^*_p = \mu(P_i)$ and $q_p = q^i$ for every $p \in P_i$. Summing (6) over $i \in [k]$ yields

$$\sum_{p\in P'} u(p)\left\|p-q^*\right\|^2-\sum_{p\in P'} u(p)\left\|p-q_p\right\|^2=\sum_{p\in P'} u(p)\left(\left\|q_p^*-q^*\right\|^2-\left\|q_p^*-q_p\right\|^2\right)$$

$$=\sum_{p\in P'} u(p)\left(\left\|(q_p^*-\mu(P))+(\mu(P)-q^*)\right\|^2-\left\|(q_p^*-\mu(P))+(\mu(P)-q_p)\right\|^2\right) \tag{7}$$

$$=\sum_{p\in P'} u(p)\left(\left\|\mu(P)-q^*\right\|^2-\left\|\mu(P)-q_p\right\|^2\right) \tag{8}$$

$$-2\sum_{p\in P'} u(p)(q_p^*-\mu(P))(q^*-q_p). \tag{9}$$

To bound (8), we substitute $x=q^*$ and then $x=q$ in Lemma 1, and obtain that for every $q\in Q$

$$\left(\left\|\mu(P)-q^*\right\|^2-\left\|\mu(P)-q\right\|^2\right)\sum_{p\in P'} u(p)=\operatorname{cost}(P,\{q^*\})-\operatorname{cost}(P,\mu(P))-(\operatorname{cost}(P,\{q\})-\operatorname{cost}(P,\mu(P)))$$

$$=\operatorname{cost}(P,\{q^*\})-\operatorname{cost}(P,\{q\})\le 0.$$

where the last inequality is by the definition of q^*. This implies that for every $p\in P'$,

$$\left\|\mu(P)-q^*\right\|^2-\left\|\mu(P)-q_p\right\|^2\le 0.$$

Plugging the last inequality in (8) yields

$$\sum_{p\in P'} u(p)\left\|p-q^*\right\|^2-\sum_{p\in P'} u(p)\left\|p-q_p\right\|^2\le -2\sum_{p\in P'} u(p)(q_p^*-\mu(P))(q^*-q_p) \tag{10}$$

$$\le 2\sum_{p\in P'} u(p)\left\|q_p^*-\mu(P)\right\|\left\|q^*-q_p\right\|=\sum_{p\in P'} u(p)\cdot 2\cdot\frac{\left\|q_p^*-\mu(P)\right\|}{\sqrt{\varepsilon}}\cdot\sqrt{\varepsilon}\left\|q^*-q_p\right\| \tag{11}$$

$$\le\sum_{p\in P'} u(p)\left(\frac{\left\|q_p^*-\mu(P)\right\|^2}{\varepsilon}+\varepsilon\left\|q^*-q_p\right\|^2\right) \tag{12}$$

$$=\frac{1}{\varepsilon}\sum_{p\in P'} u(p)\left\|q_p^*-\mu(P)\right\|^2+\varepsilon\sum_{p\in P'} u(p)\left\|q^*-q_p\right\|^2, \tag{13}$$

where (11) is by Cauchy-Schwartz inequality, and in (12) we use the fact that $2ab\le a^2+b^2$ for every $a,b\ge 0$.

To bound the left term of (13) we use the fact $q_p^*=\mu(P_i)$ and substitute $x=\mu(P)$, $P=P_i$ in Lemma 1 for every $i\in[k]$ as follows.

$$\sum_{p\in P'} u(p)\left\|q_p^*-\mu(P)\right\|^2=\sum_{i=1}^{k}\left\|\mu(P_i)-\mu(P)\right\|^2\sum_{p\in P_i'} u(p)$$

$$=\sum_{i=1}^{k}\left(\sum_{p\in P_i'} u(p)\left\|p-\mu(P)\right\|^2-\sum_{p\in P_i'} u(p)\left\|p-\mu(P_i)\right\|^2\right) \tag{14}$$

$$=\sum_{p\in P'} u(p)\left\|p-\mu(P)\right\|^2-\sum_{i=1}^{k}\sum_{p\in P_i'} u(p)\left\|p-\mu(P_i')\right\|^2\le\operatorname{opt}(P,1)-\operatorname{opt}(P,k).$$

To bound the right term of (13) we use $(a-b)^2 \leq a^2+b^2+2|ab| \leq 2a^2+2b^2$ to obtain

$$\begin{aligned}\sum_{p\in P'} u(p)\left\|q^*-q_p\right\|^2 &= \sum_{p\in P'} u(p)\left\|(q^*-p)+(p-q_p)\right\|^2 \\ &\leq \sum_{p\in P'} u(p)\left(2\left\|q^*-p\right\|^2+2\left\|p-q_p\right\|^2\right) = 2\cdot(\operatorname{cost}(P,\{q^*\})+\operatorname{cost}(P,Q))).\end{aligned}$$

Plugging (14) and the last inequality in (13) yields

$$\begin{aligned}\operatorname{cost}(P,\{q^*\})-\operatorname{cost}(P,Q) &= \sum_{p\in P'} u(p)\left\|p-q^*\right\|^2-\sum_{p\in P'} u(p)\left\|p-q_p\right\|^2 \\ &\leq \frac{\operatorname{opt}(P,1)-\operatorname{opt}(P,k)}{\varepsilon}+2\cdot\varepsilon\left(\operatorname{cost}(P,\{q^*\})+\operatorname{cost}(P,Q)\right).\end{aligned}$$

Rearranging,

$$\operatorname{cost}(P,\{q^*\}) \leq \operatorname{cost}(P,Q)\cdot\frac{1+2\varepsilon}{1-2\varepsilon}+\frac{\operatorname{opt}(P,1)-\operatorname{opt}(P,k)}{(1-2\varepsilon)\varepsilon}$$

Together with (2) this proves Lemma 2. □

Lemma 3. *Let S be a $(1,\varepsilon)$-coreset for a weighted set P in $\mathbb{R}^d$. Let $Q\subseteq\mathbb{R}^d$ be a finite set. Then*

$$(1-\varepsilon)\min_{q\in Q}\operatorname{cost}(P,\{q\}) \leq \min_{q\in Q}\operatorname{cost}(S,\{q\}) \leq (1+\varepsilon)\min_{q\in Q}\operatorname{cost}(P,\{q\}) \tag{15}$$

Proof. Let $q_P\in Q$ be a center such that $\operatorname{cost}(P,\{q_P\})=\min_{q\in Q}\operatorname{cost}(P,\{q\})$, and let $q_S\in Q$ be a center such that $\operatorname{cost}(S,\{q_S\})=\min_{q\in Q}\operatorname{cost}(S,\{q\})$. The right side of (15) is bounded by

$$\min_{q\in Q}\operatorname{cost}(S,\{q\})=\operatorname{cost}(S,\{q_S\}) \leq \operatorname{cost}(S,\{q_P\}) \leq (1+\varepsilon)\operatorname{cost}(P,\{q_P\}) = (1+\varepsilon)\min_{q\in Q}\operatorname{cost}(P,\{q\}),$$

where the first inequality is by the optimality of q_S, and the second inequality is since S is a coreset for P. Similarly, the left hand side of (15) is bounded by

$$\begin{aligned}(1-\varepsilon)\min_{q\in Q}\operatorname{cost}(P,\{q\}) = (1-\varepsilon)\operatorname{cost}(P,\{q_P\}) &\leq (1-\varepsilon)\operatorname{cost}(P,\{q_S\}) \leq (1-\varepsilon)(1+\varepsilon)\operatorname{cost}(S,\{q_S\}) \\ &= (1-\varepsilon^2)\min_{q\in Q}\operatorname{cost}(S,\{q\}) \leq \min_{q\in Q}\operatorname{cost}(S,\{q\}).\end{aligned}$$

where the last inequality follows from the assumption $\varepsilon<1$. □

Lemma 4. *Let S be the output of a call to k-MEAN-CORESET(P,k,ε). Then S is a $(k,15\varepsilon)$-coreset for P.*

Proof. By replacing P with P_i in Lemma 1 for each $i\in[m]$ it follows that

$$\operatorname{cost}(P_i,Q) \leq \min_{q\in Q}\operatorname{cost}(P_i,Q) \leq \operatorname{cost}(P_i,Q)\cdot\frac{1+2\varepsilon}{1-2\varepsilon}+\frac{\operatorname{opt}(P_i,1)-\operatorname{opt}(P_i,k)}{(1-2\varepsilon)\varepsilon}.$$

Summing the last inequality over each P_i yields

$$\operatorname{cost}(P,Q) \leq \sum_{i=1}^{m}\min_{q\in Q}\operatorname{cost}(P_i,Q) \leq \operatorname{cost}(P,Q)\cdot\frac{1+2\varepsilon}{1-2\varepsilon}+\frac{1}{(1-2\varepsilon)\varepsilon}\sum_{i=1}^{m}\left(\operatorname{opt}(P_i,1)-\operatorname{opt}(P_i,k)\right). \tag{16}$$

Since $\{P_1, \cdots, P_m\}$ is the partition of the m-means of P we have $\sum_{i=1}^{m} \text{opt}(P_i, 1) = \text{opt}(P, m)$. By letting Q_i be the m-means of P_i we have

$$\sum_{i=1}^{m} \text{opt}(P_i, k) = \sum_{i=1}^{m} \text{cost}(P_i, Q_i) \geq \sum_{i=1}^{m} \text{cost}(P_i, \cup_{j=1}^{m} Q_j) = \text{cost}(P, \cup_{j=1}^{m} Q_j) \geq \text{opt}(P, mk).$$

Hence,

$$\sum_{i=1}^{m} (\text{opt}(P_i, 1) - \text{opt}(P_i, k)) \leq \text{opt}(P, m) - \text{opt}(P, mk) \leq \varepsilon^2 \text{opt}(P, k) \leq \varepsilon^2 \text{cost}(P, Q),$$

where the second inequality is by Line 1 of the algorithm. Plugging the last inequality in (16) yields

$$\text{cost}(P, Q) \leq \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(P_i, Q) \leq \text{cost}(P, Q) \cdot \frac{1 + 3\varepsilon}{1 - 2\varepsilon}. \tag{17}$$

Using Lemma 3, for every $i \in [m]$

$$(1 - \varepsilon) \min_{q \in Q} \text{cost}(P_i, \{q\}) \leq \min_{q \in Q} \text{cost}(S_i, \{q\}) \leq (1 + \varepsilon) \min_{q \in Q} \text{cost}(P_i, \{q\})$$

By summing over $i \in [m]$ we obtain

$$(1 - \varepsilon) \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(P_i, \{q\}) \leq \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(S_i, \{q\}) \leq (1 + \varepsilon) \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(P_i, \{q\}).$$

By this and Lemma 1

$$(1 - \varepsilon) \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(P_i, \{q\}) \leq \text{cost}(S, Q) \leq (1 + \varepsilon) \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(P_i, \{q\}).$$

Plugging the last inequality in (17) yields

$$\begin{aligned} (1 - \varepsilon)\text{cost}(P, Q) &\leq (1 - \varepsilon) \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(P_i, Q) \\ &\leq \text{cost}(S, Q) \\ &\leq (1 + \varepsilon) \sum_{i=1}^{m} \min_{q \in Q} \text{cost}(P_i, \{q\}) \leq (1 + \varepsilon)\text{cost}(P, Q) \cdot \frac{1 + 3\varepsilon}{1 - 2\varepsilon} \leq (1 + 15\varepsilon)\text{cost}(P, Q). \end{aligned} \tag{18}$$

Hence, S is a $(k, 15\varepsilon)$ coreset for P. □

Lemma 5. *There is an integer $t < 1 + 1/\varepsilon^2$ such that*

$$\text{opt}(P, k^t) - \text{opt}(P, k^{t+1}) \leq \varepsilon^2 \cdot \text{opt}(P, k). \tag{19}$$

Proof. Contradictively assume that (19) does not hold for every integer $i < 1 + 1/\varepsilon^2$. Hence,

$$\text{opt}(P, k) - \text{opt}(P, k^{\lceil 1/\varepsilon^2 \rceil + 1}) = \sum_{i=1}^{\lceil 1/\varepsilon^2 \rceil} \left(\text{opt}(P, k^i) - \text{opt}(P, k^{i+1})\right) > \lceil 1/\varepsilon^2 \rceil \cdot \varepsilon^2 \text{opt}(P, k) \geq \text{opt}(P, k).$$

Contradiction, since $\text{opt}(P, k^{\lceil 1/\varepsilon^2 \rceil + 1}) \geq 0$. □

Using the mean of P_i in Line 5 of the algorithm yields a $(1, \varepsilon)$-coreset S_i as shown in Lemma 1. The resulting coreset is not sparse, but gives the following result.

Theorem 2. *There is $m \leq k^{1/\varepsilon^2}$ such that the m-means of P is a $(k, 15\varepsilon)$-coreset for P.*

Proof of Theorem 1: We compute S_i a $(1, \varepsilon)$ mean coreset for 1-mean of P_i at line 5 of Algorithm 1 by using variation of Frank-Wolfe [22] algorithm. It follows that $|S_i| = O(1/\varepsilon^2)$ for each i, therefore the overall sparsity of S is $s(P)/\varepsilon^2$. This and Lemma 4 concludes the proof. □

7. Comparison to Existing Approaches

In this section we provide experimental results of our main algorithm of coreset constructions. We compare the clustering with existing coresets and small/medium/large datasets. Unlike most of the coreset papers, we also run the algorithm on the distributed setting via a cloud as explained below.

7.1. Datasets

For our experimental results we use three well known datasets, and the English Wikipedia as follows.

MNIST handwritten digits [25]. The MNIST dataset consists of $n = 60,000$ grayscale images of handwritten digits. Each image is of size 28×28 pixels and was converted to a row vector row of $d = 784$ dimensions.

Pendigits [26]. This dataset was downloaded from the UCI repository. It consists of 250 written letters by 44 humans. These humans were asked to write 250 digits in a random order inside boxes of 500 by 500 tablet pixel resolution. The tablet sends x and y tablet coordinates and pressure level values of the pen at fixed time intervals (sampling rate) of 100 milliseconds. Digits are represented as constant length feature vectors of size $d = 16$ the number of digits in the dataset is $n = 10,992$.

NIPS dataset [27]. The OCR scanning of NIPS proceedings over 13 years. It has 15,000 pages and 1958 articles. For each author, there is a corresponding words counter vector, where the ith entry in the vector is the number of the times that the word used in one of the author's submissions. There are overall $n = 2865$ authors and $d = 14,036$ words in this corpus.

English Wikipedia [28]. Unlike previous datasets that were uploaded to memory and then compressed via streaming coresets, the English Wikipedia practically can not be uploaded completely to memory. The size of the dataset is 15GB after converting to a term-documents matrix via gensim [29]. It has 4M vectors, each of 10^5 dimensions and an average of 200 non-zero entries, i.e., words per document.

7.2. The Experiment

We applied our coreset construction to boost the performance of Lloyd's k-means heuristic as explained in Section 8 of previous work [6]. We compared the results with the current data summarization algorithms that can handle sparse data: uniform and importance sampling.

7.3. On the Small/Medium Datasets

we evaluate both the offline computation and the streaming data model. For offline computation we used the datasets above to produce coresets of size $100 \leq m \leq 1500$, then computed k-means for $k = 10, 15, 20, 25$ till convergence. For the streaming data model, we divided each dataset into small subsets and computed coresets via the merge-and-reduce technique to construct a coreset tree as in Figure 1. Here, the coresets are smaller, of size $10 \leq m \leq 500$, and the values for k-means are the same.

We computed the sum of squared distances to the original (full) set of points, from each resulting set of k centers that was computed from the coreset. These sets of centers are denoted by C_1, C_2 and C_3 for uniform, non uniform sampling and our coreset respectively. The "ground truth" or "optimal solution" C^k was computed using k-means on the entire dataset until convergence. The empirical estimated error ε was then defined to be $\epsilon = C_t/C_k - 1$ for coreset number $t = 1, 2, 3$. Note that, since Lloyd's k-means is a heuristic, its performance on the reduced data might be better, i.e., $\varepsilon < 0$.

These experiments were run on a single common laptop machine with 2.2GHz quad-core Intel Core i7 CPU and 16GB of 1600MHz DDR3L RAM with 256GB PCIe-based onboard SSD.

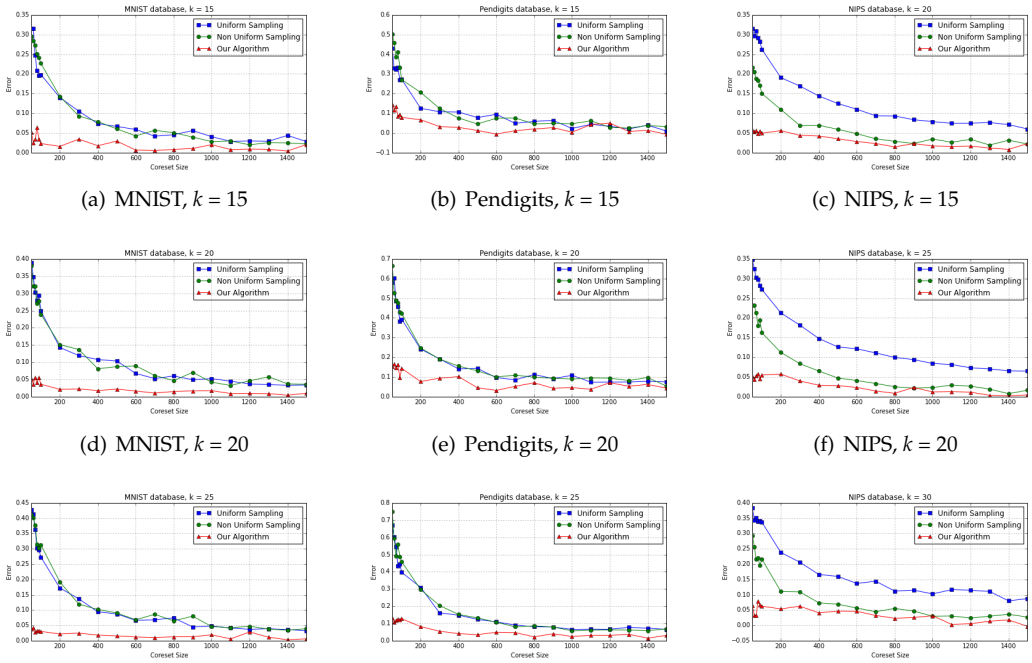
7.4. On the Wikipedia Dataset

We compared the three discussed data summarization techniques, while each one was computed in parallel and in a distributed fashion on 16 EC2 virtual servers. We repeated computation for $k = 16, 32, 64$ and 128, and coresets size in the range $[32, 1024]$.

This experiment was executed via Amazon's Web Services ("cloud"), using 16 EC2 virtual computation nodes of type c4.4xlarge, which 8 vCPU and 15GiB of RAM. We repeated distributed computation evaluating for coresets of sizes 256, 512, 1024 and 2048 points for k-means with $k = 16, 32, 64, 128$.

7.5. Results

The results of experiment for $k = 15, 20, 25$ on small datasets for offline computation are depicted on Figure 2, where it's evident that error of kmeans computation fed by our coreset algorithm results outperforms error of uniform and non-uniform sampling.



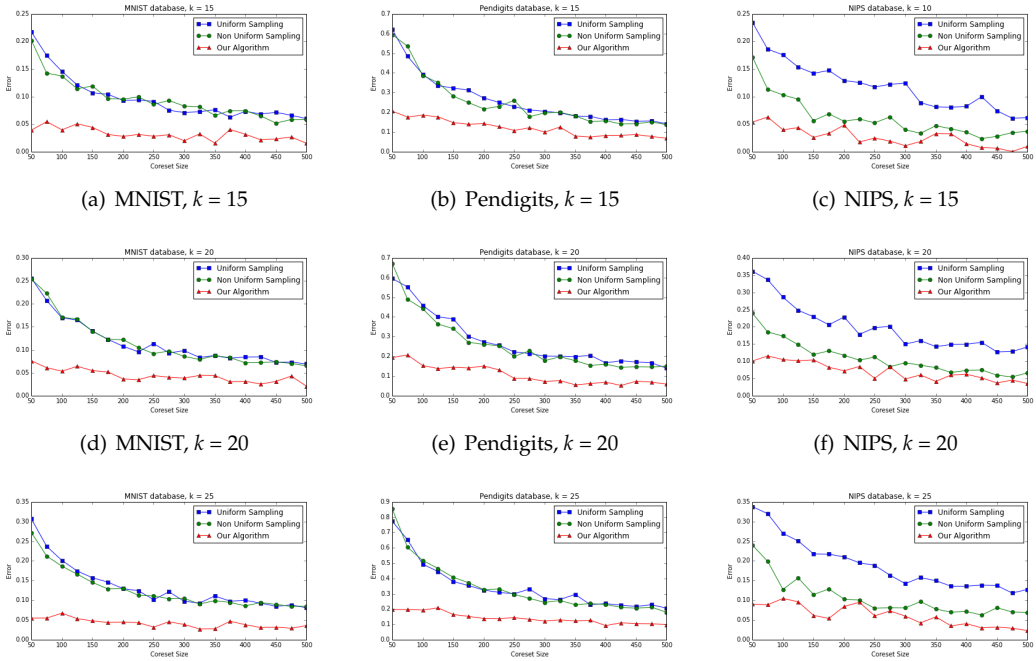
(a) MNIST, $k = 15$ (b) Pendigits, $k = 15$ (c) NIPS, $k = 15$

(d) MNIST, $k = 20$ (e) Pendigits, $k = 20$ (f) NIPS, $k = 20$

(g) MNIST, $k = 25$ (h) Pendigits, $k = 25$ (i) NIPS, $k = 25$

Figure 2. Offline coresets computation for small datasets (uniform sampling, non uniform sampling and our algorithm).

For streaming computation model our algorithm is able to provide results which are better than other two as could be explored in Figure 3. In addition, existing algorithms suffer from "cold start" as common in random sampling techniques: there is a slow convergence to the small error, compared to our deterministic algorithm that introduces small error already after a small sample size.



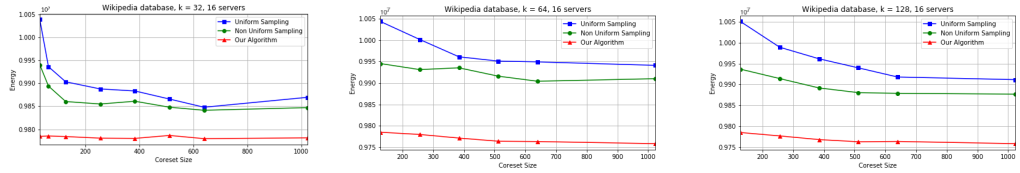
(a) MNIST, $k = 15$ (b) Pendigits, $k = 15$ (c) NIPS, $k = 15$

(d) MNIST, $k = 20$ (e) Pendigits, $k = 20$ (f) NIPS, $k = 20$

(g) MNIST, $k = 25$ (h) Pendigits, $k = 25$ (i) NIPS, $k = 25$

Figure 3. Streaming coresets computation for small datasets (uniform sampling, non uniform sampling and our algorithm).

At Figure 4 presented results of the experiment on Wikipedia dataset for different values of $k = 32, 64, 128$, as it could be easily observed proposed coreset algorithm provides good results of big sparse dataset and provides lower energy cost compared to uniform and non-uniform approaches.



(a) Wikipedia, $k = 32$ (b) Wikipedia, $k = 64$ (c) Wikipedia, $k = 128$

Figure 4. Comparison of uniform sampling, non uniform sampling and our algorithm based on Wikipedia in distributed setting with 16 servers.

Figures 5–7 show the box-plot of error distribution for all the three coresets in the offline and streaming settings. Our algorithm shows a little variance across all experiments, its mean error is very close to its median error, indicating that it produces stable results.

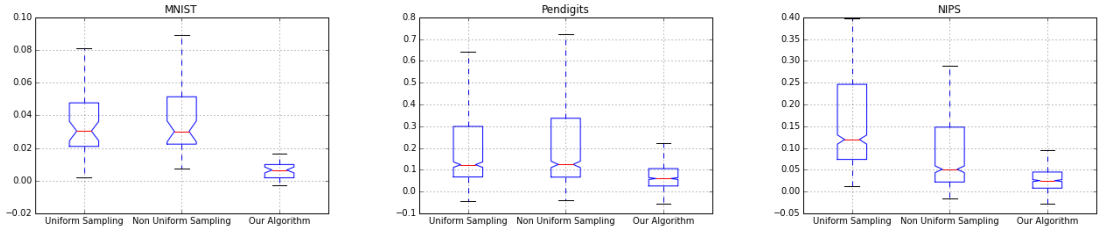


Figure 5. Error (y-axis) box-plots for real-data sets, ofline computation model.

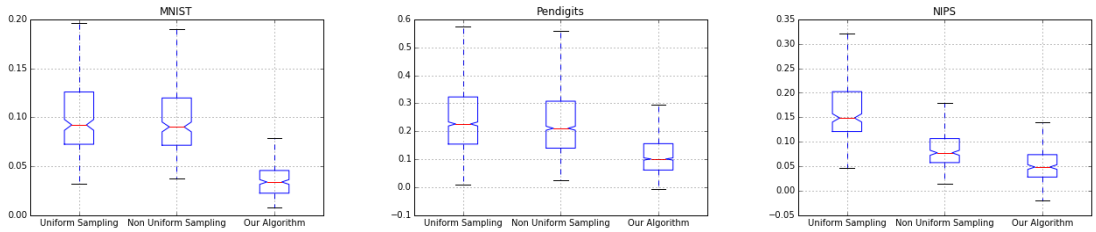


Figure 6. Error (y-axis) box-plots for real-data sets, streaming computation model.

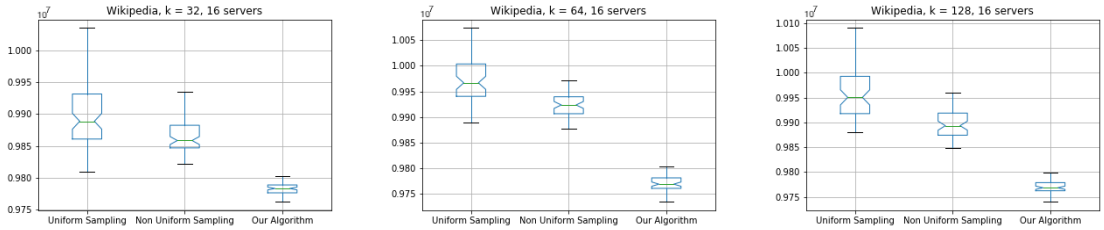


Figure 7. Error (y-axis) box-plots for wikipedia dataset, distributed computation for k = 32, 64 and 128.

Figure 8 shows the memory (RAM) footprint during the coreset construction based on synthetically generated random data. The oscillations corresponds to the number of coresets in the tree that each new subset needs to update. For example, the first point in a streaming tree is updated in $O(1)$, however the 2^ith point for some $i \geq 1$ climbs up through $O(\log i)$ levels in the tree, so $O(\log i)$ coresets are merged.

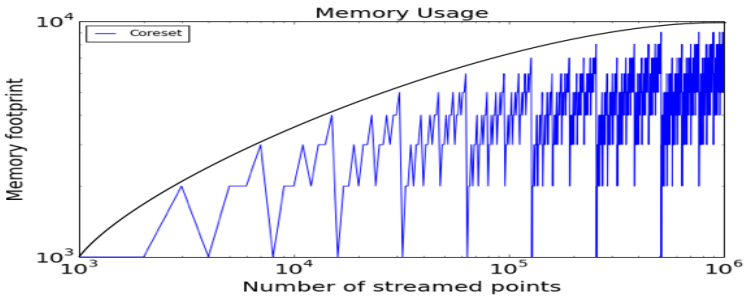


Figure 8. Allocated memory (y-axis) grows logarithmically during streaming coreset construction. The Zig-zag patterns caused by the binary merge-reduce tree in Figure 1.

8. Conclusions

We proved that any set of points in $\mathbb{R}^d$ has a (k, ε)-coreset which consists of a weighted subset of the input points whose size is independent of n and d, and polynomial in $1/\varepsilon$. Our algorithm carefully selects m such that the m-means of the input with appropriate weights (clusters' size) yields such a coreset.

This allows us to finally compute coreset for sparse high dimensional data, in both the streaming and the distributed setting. As a practical example, we computed the first coreset for the full English Wikipedia. We hope that our open source code will allow researchers in the industry and academia to run these coresets on more databases such as images, speech or tweets.

The reduction to k-means allows us to use popular k-means heuristics (such as Lloyd-Max) and provable constant factor approximations (such as k-means++) in practice. Our experimental results on both a single machine and on the cloud shows that our coreset construction significantly improves over existing techniques, especially for small coresets, due to its deterministic approach.

We hope that this paper will also help the community to answer the following three open problems:

(i) Can we simply compute the m-means for a specific value $m \in k^{O(1/\varepsilon)}$ and obtain a (k, ε)-coreset without using our algorithm?
(ii) can we compute such a coreset (subset of the input) whose size is $m \in (k/\varepsilon)^{O(1)}$?
(iii) Can we compute such a smaller coreset deterministically?

Author Contributions: Conceptualization and Formal Analysis D.F. and A.B.; funding acquisition, D.F.; Code writing A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by BSF/NSF Grant Number: 2014627 and by GIF 2408- 407.6 Young Scientists' Program Contract No.: I- 1186-407.9-2014.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Agarwal, P.K.; Har-Peled, S.; Varadarajan, K.R. Approximating extent measures of points. *J. ACM* **2004**, *51*, 606–635. [CrossRef]
2. 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, 13–15 June 2004*; ACM Press: New York, NY, USA, 2004.
3. Bentley, J.L.; Saxe, J.B. Decomposable Searching Problems I: Static-to-Dynamic Transformation. *J. Algorithms* **1980**, *1*, 301–358. [CrossRef]
4. Feldman, D.; Schmidt, M.; Sohler, C. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 6–8 January 2013; pp. 1434–1453.
5. Apache Hadoop. Available online: http://hadoop.apache.org (accessed on 10 March 2020).
6. 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, 5–7 May 2016; pp. 342–350.
7. Feldman, D.; Faulkner, M.; Krause, A. Scalable training of mixture models via coresets. In Proceedings of the NIPS 2011—Advances in Neural Information Processing Systems, Granada, Spain, 12–14 December 2011; pp. 2142–2150.
8. Barger, A.; Feldman, D. Source code for running streaming SparseKMeans coreset on the cloud 2017. (in process)
9. Chen, K. On k-median clustering in High Dimensions. In Proceedings of the 17th Annu. ACM-SIAM Symposium on Discrete Algorithms (SODA), Barcelona, Spain, 5–7 July 2006; pp. 1177–1185.
10. 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, 17–19 January 2010.
11. Feldman, D.; Monemizadeh, M.; Sohler, C. A PTAS for k-means clustering based on weak coresets. In Proceedings of the Twenty-Third Annual Symposium on Computational Geometry, Gyeongju, South Korea, 6–8 June 2007.
12. Feldman, D.; Langberg, M. A Unified Framework for Approximating and Clustering Data. *arXiv* **2016**, arXiv:1106.1379 STOC. 2011.
13. Inaba, M.; Katoh, N.; Imai, H. Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-Clustering. In Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, NY, USA, 6–8 June 1994; pp. 332–339.
14. Cohen, M.; 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 Symposium on Theory of Computing, Portland, OR, USA, 14–17 June 2015.
15. Becchetti, L.; Bury, M.; Cohen-Addad, V.; Grandoni, F.; Schwiegelshohn, C. Oblivious dimension reduction for k-means: Beyond subspaces and the Johnson-Lindenstrauss lemma. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, Phoenix, AZ, USA, 23–26 June 2019; pp. 1039–1050.
16. Lindenstrauss, W.J.J. Extensions of Lipschitz maps into a Hilbert space. *Contemp. Math.* **1984**, *26*, 189–206.

17. Har-Peled, S.; Kushal, A. Smaller coresets for *k*-median and *k*-means clustering. *Discret. Comput. Geom.* **2007**, *37*, 3–19. [CrossRef]
18. Ballard, D.H. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognit.* **1981**, *13*, 111–122. [CrossRef]
19. Bhattacharya, A.; Jaiswal, R. On the *k*-Means/Median Cost Function. *arXiv* **2017**, arXiv:1704.05232.
20. Wilkinson, B.; Allen, M. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*; Prentice-Hall: Upper Saddle River, NJ, USA, 1999.
21. Mahajan, M.; Nimbhorkar, P.; Varadarajan, K. The planar *k*-means problem is NP-hard. In *WALCOM*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 274–285.
22. Feldman, D.; Volkov, M.V.; Rus, D. Dimensionality Reduction of Massive Sparse Datasets Using Coresets. *arXiv* **2015**, arXiv:abs/1503.01663.
23. Fichtenberger, H.; Gillé, M.; Schmidt, M.; Schwiegelshohn, C.; Sohler, C. BICO: BIRCH meets coresets for *k*-means clustering. In *European Symposium on Algorithms*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 481–492.
24. Ackermann, M.R.; Märtens, M.; Raupach, C.; Swierkot, K.; Lammersen, C.; Sohler, C. StreamKM++ A clustering algorithm for data streams. *J. Exp. Algorithmics (JEA)* **2012**, *17*, 2.1–2.30.
25. LeCun, Y.; Cortes, C. The MNIST Database of Handwritten Digits. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 10 March 2020).
26. Alimoglu, F.; Doc, D.; Alpaydin, E.; Denizhan, Y. Combining Multiple Classifiers for Pen-Based Handwritten Digit Recognition. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.6299&rep=rep1&type=pdf (accessed on 10 March 2020).
27. LeCun, Y. Nips Online Web Site. 2001. Available online: http://nips.djvuzone.org (accessed on 10 March 2020).
28. The Free Wikipedia. Encyclopedia. 2004. Available online: https://dumps.wikimedia.org/enwiki/20170220/ (accessed on 1 February 2017).
29. Rehurek, R.; Sojka, P. Gensim—Statistical Semantics in Python. Available online: https://www.fi.muni.cz/usr/sojka/posters/rehurek-sojka-scipy2011.pdf (accessed on 10 March 2020).