Robust Coreset Construction for Distributed Machine Learning

Hanlin Lu*, Ming-Ju Li*, Ting He*, Shiqiang Wang†, Vijay Narayanan*, Kevin S. Chan‡
*Pennsylvania State University, University Park, PA, USA. Email: {hzl263,mxl592,tzh58,vxn9}@psu.edu
†IBM T. J. Watson Research Center, Yorktown, NY, USA. Email: wangshiq@us.ibm.com
‡Army Research Laboratory, Adelphi, MD, USA. Email: kevin.s.chan.civ@mail.mil

***Abstract*—Motivated by the need of solving machine learning problems over distributed datasets, we explore the use of *coreset* to reduce the communication overhead. Coreset is a summary of the original dataset in the form of a small weighted set in the same sample space. Compared to other data summaries, coreset has the advantage that it can be used as a proxy of the original dataset. However, existing coreset construction algorithms are each tailor-made for a specific machine learning problem. Thus, to solve different machine learning problems, one has to collect coresets of different types, defeating the purpose of saving communication overhead. We resolve this dilemma by developing robust coreset construction algorithms based on k-means/median clustering, that give a provably good approximation for a broad range of machine learning problems with sufficiently continuous cost functions. Through evaluations on diverse datasets and machine learning problems, we verify the robust performance of the proposed algorithms.**

***Index Terms*—Coreset, machine learning, k-means, k-median**

I. Introduction

The recent decade has observed a dramatic growth in distributed data generation, powered by various Internet of Things (IoT) applications and social networking applications. It has been predicted that the rate of such distributed data generation will exceed the current Internet capacity in the near future [1]. This phenomenon presents both opportunities and challenges for machine learning. On the one hand, the real-time and location-based nature of the distributed data enables novel applications based on machine learning, such as augmented reality and cognitive assistance. On the other hand, the distributed nature of the data sources, coupled with the difficulty of collecting the data to a central location due to network bandwidth, energy, and/or privacy constraints, calls for a fundamentally different way of applying machine learning that is suitable for distributed datasets.

Broadly speaking, three approaches have been proposed for distributed machine learning: *sharing the output*, *sharing the model*, and *sharing the data*. In the first approach [2], data sources independently compute and share outputs of local models, which are then aggregated into a global output (e.g., by majority vote). In the second approach [3], [4], data sources share models learned on local data, which are then aggregated into a global model (e.g., by taking weighted average). In the third approach [5]–[7], data sources share summaries of their local data, which are then used to compute a global model. Each approach has its pros and cons: the first approach usually has the smallest communication overhead, but only supports one query; the second approach builds a global model that can be used for multiple queries, but only supports one model; the third approach may incur a larger communication overhead, but can potentially support multiple models.

This research was partly sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. Narayanan and Hanlin were partly supported by NSF 1317560. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

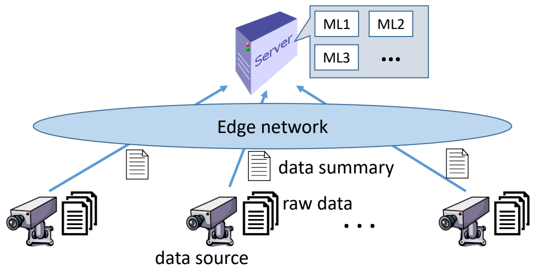


Fig. 1. Application scenario (ML i: machine learning model i).

In this work, we take the third approach, with a particular interest in supporting diverse machine learning models. Fig. 1 illustrates a typical application scenario in the context of *mobile edge computing* [4], where data sources report local summaries to an edge server, which then computes various models from these summaries.

In particular, we consider data summarization using *coreset* [8]. A coreset is a small weighted dataset as a proxy of the original dataset with *provable approximation guarantees*. Compared to other data summaries (e.g., sketches), a coreset preserves the sample space of the original dataset, and is hence more convenient to use, e.g., a classifier learned from the coreset can classify a new data point, the principle components learned from the coreset can be used for feature selection, both *in the original sample space*. Algorithms have been developed to construct coresets for various machine learning problems (see Section II-B), such that the model learned on the coreset approximates the model learned on the original dataset. However, existing coreset construction algorithms are *tailor-made* for specific machine learning problems, which means that we have to collect different coresets to solve different problems. The question we raise is: *Is there a coreset that is robust against different machine learning problems, i.e., having good performance in computing a variety of machine learning models?*

In this work, we answer the above question affirmatively by proving that a particular type of coreset, generated by k-means/median clustering, can give a good approximation for a broad set of machine learning problems.

A. Related Work

Distributed learning is one of the most promising lines of research for large-scale learning [2], particularly for naturally distributed data. The main challenge in distributed learning is to incorporate information from each distributed dataset, without the high overhead of collecting all the data.

Traditionally, this is achieved by collecting the outputs of learned models or the models themselves [9]. The first approach (i.e., collecting outputs) is more popular among earlier works. For example, [10] proposed various heuristic decision rules (e.g., majority vote) to combine outputs of local classifiers, and [11] proposed to train a global classifier using labeled outputs of local classifiers.

The second approach (i.e., collecting models) is more useful when we want to learn not just one answer, but the rule to give answers. For example, the distributed boosting framework in [12] requires nodes to share locally trained classifiers, and the federated learning framework in [3] requires nodes to report locally learned models to a single node, which then aggregates the models and broadcasts the result to others.

Meanwhile, research on data summarization has inspired a third approach: collecting data summaries. Data summaries, e.g., coresets, sketches, projections [13], [14], are derived datasets that are much smaller than the original dataset, and can hence be transferred to a central location with a low communication overhead. This approach has been adopted in recent works, e.g., [5]–[7]. We are particularly interested in a specific type of data summary, *coreset*, as it can be used as a proxy of the original dataset. See Section II-B for a detailed review of related works on coreset.

B. Summary of Contributions

We are the first to explore using coreset to support diverse machine learning problems on distributed data. Specifically:

1) We prove that the optimal k-clustering (including k-means/median) gives a coreset that provides a guaranteed approximation for any machine learning problem with a sufficiently continuous cost function (Theorem III.1).
2) We further prove that the same holds for the coreset given by a suboptimal k-clustering algorithm, as long as it satisfies certain assumptions (Theorem III.2).
3) We develop an algorithm based on k-clustering to construct a robust coreset with a very low communication overhead.
4) Our evaluations on diverse machine learning problems verify that k-clustering (especially k-means) provides a robust coreset.

Roadmap. Section II reviews the background on coreset. Section III presents our main theoretical results on the universal performance guarantee of k-clustering-based coreset. Section IV evaluates the proposed algorithm. Section V concludes the paper. Supporting proofs, analysis, and evaluations are provided in [15].

II. Background

A. Coreset and Machine Learning

Many machine learning problems can be cast as a cost (or loss) minimization problem. Given a dataset in d-dimensional space $P \subseteq \mathbb{R}^d$, a generic machine learning problem over P can be characterized by a solution space $\mathcal{X}$, a *per-point cost function* $\text{cost}(p,x)$ ($p \in P$, $x \in \mathcal{X}$), and an *overall cost function* $\text{cost}(P,x)$ ($x \in \mathcal{X}$) that aggregates the per-point costs over P. For generality, we consider P to be a weighted set, where each $p \in P$ has weight w_p. Let $w_{\min} := \min_{p\in P} w_p$ denote the minimum weight. For an unweighted dataset, we have $w_p \equiv 1$. The machine learning problem is then to solve

$$x^* = \arg\min_{x\in\mathcal{X}} \text{cost}(P,x) \tag{1}$$

for the optimal model parameter x^*.

Example: Let $\text{dist}(p,x) := \|p-x\|_2$ denote the Euclidean distance between points p and x. The *minimum enclosing ball (MEB)* problem [8] aims at minimizing the maximum distance between any data point and a center, i.e., $\text{cost}(p,x) = \text{dist}(p,x)$, $\text{cost}(P,x) = \max_{p\in P} \text{cost}(p,x)$, and $\mathcal{X} = \mathbb{R}^d$. The *$k$-means clustering* problem aims at minimizing the weighted sum of the squared distance between each data point and the nearest center in a set of k centers, i.e., $\text{cost}(p,x) = \min_{x_i\in x} \text{dist}(p,x_i)^2$, $\text{cost}(P,x) = \sum_{p\in P} w_p \text{cost}(p,x)$, and $\mathcal{X} = \{x := \{x_i\}_{i=1}^k : x_i \in \mathbb{R}^d\}$.

Typically, the overall cost is defined as: (i) *sum cost*, i.e., $\text{cost}(P,x) = \sum_{p\in P} w_p \text{cost}(p,x)$ (e.g., k-means), or (ii) *maximum cost*, i.e., $\text{cost}(P,x) = \max_{p\in P} \text{cost}(p,x)$ (e.g., MEB).

A coreset is a small weighted dataset in the same space as the original dataset that approximates the original dataset in terms of cost, formally defined below.

Definition II.1 ([16]). *A weighted set $S \subseteq \mathbb{R}^d$ with weights u_q ($q \in S$) is an ϵ-coreset for P with respect to (w.r.t.) cost(P,x) ($x \in \mathcal{X}$) if $\forall x \in \mathcal{X}$,*

$$(1-\epsilon)\mathit{cost}(P,x) \le \mathit{cost}(S,x) \le (1+\epsilon)\mathit{cost}(P,x), \tag{2}$$

where cost(S,x) is defined in the same way as cost(P,x), i.e., cost$(S,x) = \sum_{q\in S} u_q$cost$(q,x)$ for sum cost, and cost$(S,x) = \max_{q\in S}$ cost(q,x) for maximum cost.

From Definition II.1, it is clear that the quality of a coreset depends on the cost function it needs to approximate, and hence the machine learning problem it supports.

B. Coreset Construction Algorithms

Because of the dependence on the cost function (Definition II.1), existing coreset construction algorithms are tailor-made for specific machine learning problems. Here we briefly summarize common approaches for coreset construction and representative algorithms, and refer to [13], [14] for detailed surveys.

1) Gradient descent algorithms: Originally proposed for MEB [8], [17], these algorithms iteratively add to the coreset a point far enough or furthest from the current center, and stop when the enclosing ball of the coreset, expanded by $1+\epsilon$, includes all data points. This coreset has been used to compute ϵ-approximation to several clustering problems, including k-center clustering, 1-cylinder clustering, and k-flat clustering [8], [18].

2) Random sampling algorithms: These algorithms construct a coreset by sampling from the original dataset. The basic version, uniform sampling, usually requires a large coreset size to achieve a good approximation. Advanced versions use *sensitivity sampling* [19], where each data point is sampled with a probability proportional to its contribution to the overall cost. Proposed for numerical integration [19], the idea was extended into a framework supporting projective clustering problems that include k-median/means and *principle component analysis (PCA)* as special cases [16]. Although the framework can instantiate algorithms for different machine learning problems by plugging in different cost functions, the resulting coreset only guarantees approximation for the specific problem defined by the plugged-in cost function.

3) Geometric decomposition algorithms: These algorithms divide the sample space or input dataset into partitions, and then selecting points to represent each partition. An example machine learning problem supported by such algorithms is k-means/median [7].

Using a generic merge-and-reduce approach in [20], *all these algorithms can be used to construct coresets from distributed datasets*. Of course, the resulting coresets are still tailor-made for specific problems. In contrast, we seek coreset construction algorithms which can construct coresets that simultaneously support multiple machine learning problems.

III. Robust Coreset

Our main result is that selecting *representative points* using clustering techniques yields a robust coreset with a guaranteed approximation for a broad set of machine learning problems.

A. The k-clustering Problem

At the core, the proposed algorithm constructs a k-point coreset by partitioning the dataset into k clusters, and then using the cluster centers as the coreset points. To present this algorithm, we introduce a few definitions.

Given a weighted dataset $P \subseteq \mathbb{R}^d$ with weight w_p ($p \in P$), and a set $Q = \{q_1, ..., q_k\}$ of $k \geq 1$ points in $\mathbb{R}^d$ (referred to as *centers*), the cost of clustering P into Q is defined as

$$c(P,Q) = \sum_{p\in P} w_p (\min_{q\in Q} \operatorname{dist}(p,q))^z, \tag{3}$$

for a constant $z > 0$. The *k-clustering* problem is to find the set of k centers that minimizes (3). For $z = 1$, this is the k-median problem. For $z = 2$, this is the k-means problem. We will use the solution to the k-clustering problem to construct coresets, based on which we can solve general machine learning problems. We use $c(P,\cdot)$ to denote the cost function of the k-clustering problem and $\operatorname{cost}(P,\cdot)$ to denote the cost function of a general machine learning problem of interest.

We denote by $\mu(P)$ the optimal center for 1-clustering of P. It is known that for $z = 2$, $\mu(P)$ is the sample mean:

$$\mu(P) = \frac{1}{\sum_{p\in P} w_p} \sum_{p\in P} w_p \cdot p. \tag{4}$$

We denote by $\operatorname{opt}(P,k)$ the optimal cost for k-clustering of P. It is known that k-means and k-median are both NP-hard problems [21], [22], for which efficient heuristics exist (e.g., Lloyd's algorithm and variations) [23]. Let $\operatorname{approx}(P,k)$ denote the cost of a generic k-clustering algorithm, which always satisfies $\operatorname{approx}(P,k) \geq \operatorname{opt}(P,k)$.

Each set of k centers $Q = \{q_i\}_{i=1}^k$ induces a partition of P into $\{P_1, \ldots, P_k\}$, where P_i is the subset of points in P whose closest center in Q is q_i (ties broken arbitrarily). For ease of presentation, we use[1] $\{P_i\}_{i\in[k]}$ to denote the partition induced by the optimal k-clustering, and $\{\widetilde{P}_i\}_{i\in[k]}$ to denote the partition induced by a suboptimal k-clustering.

B. Coreset by Optimal k-clustering

We will show that the superior performance of the algorithm in [7] is not a coincidence; instead, it is a fundamental property of any coreset computed by k-clustering, as long as the cost function of the machine learning problem satisfies certain continuity conditions.

Sketch of analysis: At a high level, our analysis is based on the following observations:

1) If doubling the number of centers only reduces the optimal k-clustering cost by a little, then using two centers instead of one in any cluster gives little reduction to its clustering cost (Lemma III.1).
2) If selecting two centers in a cluster P_i gives little reduction to its clustering cost, then all the points in P_i must be close to its center $\mu(P_i)$ (Lemma III.2), as otherwise selecting an outlier as the second center would have reduced the cost substantially.
3) If each data point is represented by a coreset point with a similar per-point cost, then the coreset gives a good approximation of the overall cost (Lemmas III.3 and III.4).

Therefore, for any machine learning problem with a sufficiently continuous cost function, if the condition in item (1) is satisfied, then the per-point cost of each k-clustering center will closely approximate the per-point costs of all the points in its cluster, and hence the set of k-clustering centers will give a good coreset (Theorem III.1).

[1]Throughout the paper, for $k \in \mathbb{Z}^+$, $[k] := \{1, \ldots, k\}$.

Complete analysis: We now present the precise statements, supported by proofs in Appendix A in [15].

Lemma III.1. *For any $\epsilon' > 0$, if $opt(P,k) - opt(P,2k) \le \epsilon'$, then $opt(P_i,1) - opt(P_i,2) \le \epsilon'$ ($\forall i \in [k]$), where $\{P_i\}_{i=1}^k$ is the partition of P generated by the optimal k-clustering.*

Lemma III.2. *If $opt(P_i,1) - opt(P_i,2) \le \epsilon'$, then $dist(p,\mu(P_i)) \le (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$, $\forall p \in P_i$.*

Lemma III.3. *For any machine learning problem with cost function $cost(P,x) = \sum_{p\in P} w_p cost(p,x)$, if $\exists$ a partition $\{P_i\}_{i=1}^k$ of P such that $\forall x \in \mathcal{X}$, $i \in [k]$, and $p \in P_i$,*

$$(1-\epsilon)cost(p,x) \le cost(\mu(P_i),x) \le (1+\epsilon)cost(p,x), \quad (5)$$

then $S = \{\mu(P_i)\}_{i=1}^k$ with weight $u_{\mu(P_i)} = \sum_{p\in P_i} w_p$ is an ϵ-coreset for P w.r.t. $cost(P,x)$.

Lemma III.4. *For any machine learning problem with cost function $cost(P,x) = \max_{p\in P} cost(p,x)$, if $\exists$ a partition $\{P_i\}_{i=1}^k$ of P such that (5) holds for any $x \in \mathcal{X}$, $i \in [k]$, and $p \in P_i$, then $S = \{\mu(P_i)\}_{i=1}^k$ (with arbitrary weights) is an ϵ-coreset for P w.r.t. $cost(P,x)$.*

We now prove the main theorem based on Lemmas III.1–III.4.

Theorem III.1. *If $opt(P,k) - opt(P,2k) \le w_{\min}(\frac{\epsilon}{\rho})^z$, then the optimal k-clustering of P gives an ϵ-coreset for P w.r.t. both the sum cost and the maximum cost for any per-point cost function satisfying (i) $cost(p,x) \ge 1$, and (ii) $cost(p,x)$ is ρ-Lipschitz-continuous in p, $\forall x \in \mathcal{X}$.*

Proof. By Lemma III.1, $\text{opt}(P,k) - \text{opt}(P,2k) \le \epsilon'$ implies $\text{opt}(P_i,1) - \text{opt}(P_i,2) \le \epsilon'$, $\forall$ cluster P_i generated by the optimal k-clustering. By Lemma III.2, this in turn implies that $\text{dist}(p,\mu(P_i)) \le (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$, $\forall p \in P_i$. Because $\text{cost}(p,x)$ is ρ-Lipschitz-continuous in p for all $x \in \mathcal{X}$, we have

$$|\operatorname{cost}(p,x) - \operatorname{cost}(\mu(P_i),x)| \le \rho(\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}, \forall x \in \mathcal{X}, p \in P_i.$$

Moreover, as $\text{cost}(p,x) \ge 1$,

$$\frac{|\operatorname{cost}(p,x) - \operatorname{cost}(\mu(P_i),x)|}{\operatorname{cost}(p,x)} \le \rho(\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}} = \epsilon$$

for $\epsilon' = w_{\min}(\frac{\epsilon}{\rho})^z$. By Lemma III.3, k-clustering gives an ϵ-coreset for P w.r.t. the sum cost; by Lemma III.4, k-clustering gives an ϵ-coreset for P w.r.t. the maximum cost. □

Often in practice, the coreset size must satisfy some upper bound specified by the maximum communication overhead. In this case, we can rephrase Theorem III.1 to characterize the quality of approximation as a function of the coreset size.

Corollary III.1.1. *Given a maximum coreset size $k \in \mathbb{Z}^+$ (positive integers), for any cost function satisfying the conditions in Theorem III.1, the optimal k-clustering gives an ϵ-coreset for P w.r.t. this cost function, where*

$$\epsilon = \rho\left(\frac{opt(P,k) - opt(P,2k)}{w_{\min}}\right)^{\frac{1}{z}}. \quad (6)$$

Proof. This is a direct implication of Theorem III.1, as setting ϵ by (6) satisfies the condition in Theorem III.1. □

Remark: Condition (i) in Theorem III.1 is easily satisfied by any machine learning problem with nonnegative per-point costs, as we can add '+1' to the cost function without changing the optimal solution. Even without this condition, a similar proof will show that the coreset S given by k-clustering approximates the original dataset P in that $|\text{cost}(P,x) - \text{cost}(S,x)| \le \widetilde{\epsilon}$ ($\forall x \in \mathcal{X}$), where $\widetilde{\epsilon} = \epsilon\sum_{p\in P} w_p$ for the sum cost, and $\widetilde{\epsilon} = \epsilon$ for the maximum cost.

Condition (ii) is satisfied by many machine learning problems with distance-based cost functions. For example, for MEB, $\text{cost}(p,x) = \text{dist}(p,x)$, where $x \in \mathbb{R}^d$ denotes the center of the enclosing ball. For any data points $p, p' \in \mathbb{R}^d$, by the triangle inequality, we have:

$$|\text{dist}(p,x) - \text{dist}(p',x)| \le \text{dist}(p,p'). \quad (7)$$

Hence, its cost function is 1-Lipschitz-continuous (i.e., $\rho = 1$). See Appendix B in [15] for more examples. In Section IV, we will stress-test our coreset when this condition is violated.

C. Coreset by Suboptimal k-clustering

While Theorem III.1 and Corollary III.1.1 suggest that the optimal k-clustering gives a good coreset, the k-clustering problem is NP-hard [21], [22]. The question is: *does similar performance guarantee hold for the coreset computed by an efficient but suboptimal k-clustering algorithm?* To this end, we introduce a few assumptions on the k-clustering algorithm:

Assumption 1 (local optimality): Given the partition $\{\widetilde{P}_i\}_{i=1}^k$ generated by the algorithm, the center it selects in each $\widetilde{P}_i$ is $\mu(\widetilde{P}_i)$, i.e., the optimal 1-clustering center for $\widetilde{P}_i$.

Assumption 2 (self-consistency): For any P and any $k \ge 1$, the cost of the algorithm satisfies

$$\text{approx}(P,2k) \le \sum_{i=1}^k \text{approx}(\widetilde{P}_i, 2). \quad (8)$$

Assumption 3 (greedy dominance): For any P, the 2-clustering cost of the algorithm satisfies

$$\text{approx}(P,2) \le c(P, \{\mu(P), p^*\}), \quad (9)$$

where $c(P,Q)$ is defined in (3), and $p^* := \arg\max_{p\in P} w_p \cdot \text{dist}(p,\mu(P))^z$ is the point with the highest 1-clustering cost.

We argue that these are mild assumptions that should be satisfied or approximately satisfied by any good k-clustering algorithm. Assumption 1 is easy to satisfy, as computing the 1-mean is easy (i.e., sample mean), and there exists an algorithm [24] that can compute the 1-median to an arbitrary precision in nearly linear time. Assumption 2 means that applying the algorithm for $2k$-clustering of P should perform no worse than first using the algorithm to partition P into k clusters, and then computing 2-clustering of each cluster. Assumption 3 means that for $k = 2$, the algorithm should perform no worse than a greedy heuristic that starts with the 1-clustering center, and then adds the point with the highest clustering cost as the second center.

Algorithm 1: Robust Coreset Construction(P, ϵ, ρ)

```
input : A weighted set $P$ with minimum weight $w_{\min}$,
        approximation error $\epsilon > 0$, Lipschitz constant $\rho$
output: An $\epsilon$-coreset $S$ for $P$ w.r.t. a cost function satisfying
        Theorem III.2
1 foreach $k = 1, \ldots, |P|$ do
2     if $approx(P, k) - approx(P, 2k) \le w_{\min}(\frac{\epsilon}{\rho})^z$ then
3         break;
4 $(\{\mu(\widetilde{P}_i)\}_{i=1}^k, \{\widetilde{P}_i\}_{i=1}^k) \leftarrow k$-clustering$(P, k)$;
5 $S \leftarrow \{\mu(\widetilde{P}_i)\}_{i=1}^k$, where $\mu(\widetilde{P}_i)$ has weight $\sum_{p \in \widetilde{P}_i} w_p$;
6 return $S$;
```

We show that for any k-clustering algorithm satisfying these assumptions, statements analogous to Lemma III.1 and Lemma III.2 can be made. Let $\{\widetilde{P}_i\}_{i=1}^k$ denote the partition of P generated by the k-clustering algorithm.

Lemma III.5. *For any $\epsilon' > 0$, if approx$(P, k)-$approx$(P, 2k) \le \epsilon'$, then approx$(\widetilde{P}_i, 1) -$ approx$(\widetilde{P}_i, 2) \le \epsilon'$ for any $i \in [k]$.*

Lemma III.6. *If approx$(\widetilde{P}_i, 1) -$ approx$(\widetilde{P}_i, 2) \le \epsilon'$, then dist$(p, \mu(\widetilde{P}_i)) \le (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$, $\forall p \in \widetilde{P}_i$.*

Theorem III.2. *If approx$(P, k)-$approx$(P, 2k) \le w_{\min}(\frac{\epsilon}{\rho})^z$, where approx$(P, k)$ is the cost of a (possibly suboptimal) k-clustering algorithm satisfying Assumptions 1–3, then the centers computed by the algorithm for k-clustering of P give an ϵ-coreset for P w.r.t. both the sum cost and the maximum cost for any per-point cost function satisfying (i–ii) in Theorem III.1.*

Proof. The proof follows the same steps as that of Theorem III.1, except that Lemma III.1 is replaced by Lemma III.5, and Lemma III.2 is replaced by Lemma III.6. Note that Lemmas III.3 and III.4 hold for any partition of P, which in this case is $\{\widetilde{P}_i\}_{i=1}^k$ generated by the k-clustering algorithm. □

Similar to Corollary III.1.1, we can rephrase Theorem III.2 to characterize the quality of a coreset of a specified size.

Corollary III.2.1. *Given a maximum coreset size $k \in \mathbb{Z}^+$, for any cost function satisfying the conditions in Theorem III.1 and any k-clustering algorithm satisfying Assumptions 1–3, the centers computed by the algorithm for k-clustering of P give an ϵ-coreset for P w.r.t. the given cost function, where*

$$\epsilon = \rho \left(\frac{approx(P, k) - approx(P, 2k)}{w_{\min}} \right)^{\frac{1}{z}}. \qquad (10)$$

D. Coreset Construction Algorithm

Based on Theorem III.2, we propose a k-clustering-based coreset construction algorithm, called *Robust Coreset Construction (RCC)* (Algorithm 1), which uses a k-clustering algorithm as subroutine in lines 2 and 4. If the coreset size k is predetermined, we can directly start from line 4. The constant $z = 1$ if the adopted clustering algorithm is for k-median, or $z = 2$ if it is for k-means.

TABLE I
MACHINE LEARNING COST FUNCTIONS

problem	overall cost function[2]	ρ
MEB	$\max_{p \in P} \text{dist}(x, p)$	1
k-means	$\sum_{p \in P} w_p \cdot \min_{q_i \in x} \text{dist}(q_i, p)^2$	2Δ
PCA	$\sum_{p \in P} w_p \cdot \text{dist}(p, xp)^2$	$2\Delta(l + 1)$
SVM	$\sum w_p \max(0, 1 - p_d(p_{1:d-1}^T x_{1:d-1} + x_d))$	∞

IV. PERFORMANCE EVALUATION

We evaluate the proposed coreset construction algorithm and its benchmarks over a variety of machine learning problems, and compare the cost of each model learned from a coreset with the cost of the model learned from the original dataset.

Coreset construction algorithms: We evaluate RCC based on k-median clustering ('RCC-kmedian') and RCC based on k-means clustering ('RCC-kmeans'), together with benchmarks including the algorithm in [17] ('gradient descent'), the framework in [16] instantiated for k-means ('random sampling'), and uniform sampling ('baseline').

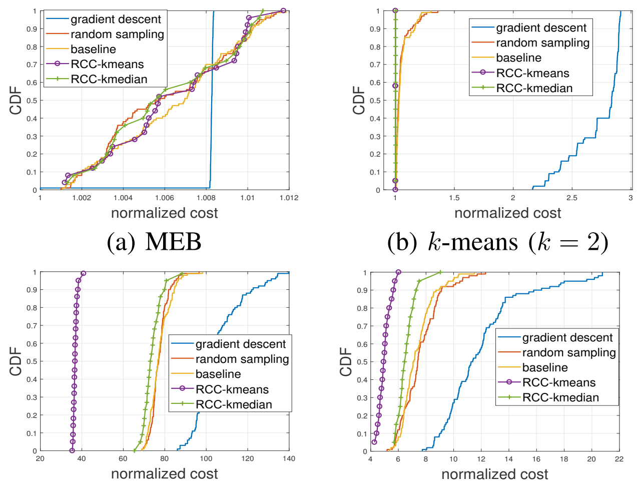
Datasets: We use MNIST data [25], which consists of $60,000$ images of handwritten digits, each trimmed to 20×20 pixels. We pre-process the data as explained in [15]. All our findings have been verified on three other datasets [15].

Machine learning problems: We evaluate three unsupervised learning problems—MEB, k-means, and PCA, and one supervised learning problem—SVM. Table I gives their cost functions, where for a data point $p \in \mathbb{R}^d$, $p_{1:d-1} \in \mathbb{R}^{d-1}$ denotes the numerical portion and $p_d \in \mathbb{R}$ denotes the label. The meaning of the model parameter x is problem-specific, as explained in the footnote. We also provide (upper bounds of) the Lipschitz constant ρ; see Appendix B in [15] for analysis. Here l is the number of principle components computed by PCA, and Δ is the diameter of the sample space. In our experiments, $\Delta = \sqrt{(d-1)(L^2 - 2L + 2)}$, which is 181.1 for MNIST. *Although SVM does not have a finite ρ, we still include it to stress-test our algorithm.*

Results: We evaluate the coreset construction algorithms by the normalized costs of the models learned from the coresets. This means the performance of a coreset S is evaluated by the *normalized cost*, defined as cost$(P, x_S)/$cost(P, x^*), where x^* is the model learned from the original dataset P, and x_S is the model learned from the coreset. The smaller the normalized cost, the better the performance. As these coreset construction algorithms are randomized, we plot the CDF of the normalized costs computed over 100 Monte Carlo runs for dataset MNIST in Figure 2. Due to page limit, additional results of the other three datasets are included in Appendix D in [15].

We see that the proposed algorithms ('RCC-kmeans' and 'RCC-kmedian') perform either the best or comparably to the best across all machine learning problems. The gradient descent algorithm in [17], designed for MEB, can perform very

[2]The model x denotes the center of enclosing ball for MEB and the set of centers for k-means. For PCA, $x = WW^T$, where W is a $d \times l$ matrix consisting of the first l ($l < d$) principle components as columns. For SVM, $x_{1:d-1} \in \mathbb{R}^{d-1}$ is the coefficient vector and $x_d \in \mathbb{R}$ is the offset.



(a) MEB (b) k-means ($k = 2$) (c) PCA (300 components) (d) SVM ('0': 1; others: -1)

Fig. 2. Evaluation on MNIST dataset (label: 'labels', coreset size: 50).

TABLE II
AVERAGE RUNNING TIME (SECONDS)

algorithm	MNIST
gradient descent	7.27
random sampling	4.34
baseline	0.01
RCC-kmeans	13.59
RCC-kmedian	123.82

poorly for other machine learning problems. The sampling-based algorithms ('random sampling' in [16] and 'baseline') perform relatively poorly for MEB and PCA. Note that all the algorithms are within 3% of the optimal for MEB.

Moreover, we see from the CDFs that the proposed algorithms ('RCC-kmeans' and 'RCC-kmedian') also have significantly less performance variation than the benchmarks, especially the sampling-based algorithms ('random sampling' and 'baseline'). This means that the quality of the coresets constructed by the proposed algorithms is more reliable, which is a desirable property.

Between the proposed algorithms, 'RCC-kmeans' sometimes outperforms 'RCC-kmedian', e.g., Fig. 2 (c–d). We note that 'RCC-kmeans' can be an order of magnitude faster than 'RCC-kmedian', as shown in Table II. Other than 'RCC-kmedian', all the algorithms can finish in a few seconds. In Appendix D in [15], we have also validated the approximation bound given by Corollary III.2.1.

V. CONCLUSION

We show, both theoretically and empirically, that the k-clustering centers form a coreset that provides a universal approximation for a broad set of machine learning problems with sufficiently continuous cost functions. As k-clustering (including k-means/median) is one of the most well-studied machine learning problems, this result allows us to leverage existing k-clustering algorithms for coreset construction. While our evaluations show that existing algorithms already give good performance, the assumptions in our proofs (see Section III-C) do suggest new requirements that can be investigated in future work. Our extensive evaluations verify the superior robustness of the proposed k-clustering-based coreset construction algorithm in *simultaneously* supporting a diverse set of machine learning problems.

REFERENCES

[1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, December 2016.

[2] D. Peteiro-Barral and B. Guijarro-Berdinas, "A survey of methods for distributed machine learning," in *Progress in Artificial Intelligence*, November 2012.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, May 2016.

[4] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM*, April 2018.

[5] M. F. Balcan, S. Ehrlich, and Y. Liang, "Distributed k-means and k-median clustering on general topologies," in *NIPS*, 2013.

[6] R. Kannan, S. Vempala, and D. Woodruff, "Principal component analysis and higher correlations for distributed data," in *COLT*, 2014.

[7] A. Barger and D. Feldman, "k-means for streaming and distributed big sparse data," in *SDM*, 2016.

[8] M. Bādoiu, S. Har-Peled, and P. Indyk, "Approximate clustering via core-sets," in *ACM STOC*, 2002.

[9] P. K. Chan and S. J. Stolfo, "Toward parallel and distributed learning by meta-learning," in *AAAI Workshop in Knowledge Discovery in Databases*, 1997.

[10] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, March 1998.

[11] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[12] A. Lazarevic and Z. Obradovic, "Boosting algorithms for parallel and distributed learning," *Distributed and Parallel Databases*, vol. 11, no. 2, pp. 203–229, March 2002.

[13] J. M. Phillips, "Coresets and sketches," *CoRR*, vol. abs/1601.00617, 2016.

[14] A. Munteanu and C. Schwiegelshohn, "Coresets-methods and history: A theoreticians design pattern for approximation and streaming algorithms," *KI - Künstliche Intelligenz*, vol. 32, no. 1, pp. 37–53, 2018.

[15] H. Lu, M.-J. Li, T. He, S. Wang, V. Narayanan, and K. S. Chan, "Robust coreset construction for distributed machine learning," 2019. [Online]. Available: http://arxiv.org/abs/1904.05961

[16] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data," in *STOC*, June 2011.

[17] M. Bādoiu and K. L. Clarkson, "Smaller core-sets for balls," in *SODA*, 2003.

[18] S. Har-Peled and K. R. Varadarajan, "Projective clustering in high dimensions using core-sets," in *SOCG*, 2002.

[19] M. Langberg and L. J. Schulman, "Universal ϵ approximators for integrals," in *SODA*, 2010.

[20] D. Feldman, A. Krause, and M. Faulkner, "Scalable training of mixture models via coresets," in *NIPS*, 2011.

[21] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering," *Machine Learning*, vol. 75, no. 2, pp. 245–248, May 2009.

[22] N. Megiddo and K. J. Supowit, "On the complexity of some common geometric location problems," *SIAM Journal of Computing*, vol. 13, no. 1, pp. 182–196, 1984.

[23] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *SODA*, January 2007.

[24] M. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford, "Geometric median in nearly linear time," in *STOC*, 2016.

[25] Y. LeCun, C. Cortes, and C. Burges, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/, 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/