The Pennsylvania State University
The Graduate School

DATA REDUCTION FOR COMMUNICATION-EFFICIENT MACHINE LEARNING

A Dissertation in
Computer Science and Engineering
by
Hanlin Lu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2021

The dissertation of Hanlin Lu was reviewed and approved by the following:

Ting He
Associate Professor of the School of Electrical Engineering and Computer Science
Dissertation Advisor
Chair of Committee

Vijaykrishnan Narayanan
A. Robert Noll Chair Professor of the School of Electrical Engineering and Computer Science

Mehrdad Mahdavi
Assistant Professor of the School of Electrical Engineering and Computer Science

Lingzhou Xue
Associate Professor of the Department of Statistics

Chita R. Das
Professor of Computer Science and Engineering
Department Head of Computer Science and Engineering

Abstract

In recent years, we have observed a dramatic growth of data generation in edge-based machine learning applications. Motivated by the need of solving machine learning problem over distributed datasets, we would like to reduce the size of datasets as well as minimizing the machine learning performance degradation. Suppose we are given a dataset P, it could be represented by a data cube with three dimensions: cardinality n, number of features d and number of precision bits b. In this dissertation, we will explore different data reduction techniques to reduce these three dimensions and make three steps toward reducing the total size of the dataset.

In our first step, we consider using coreset to reduce the cardinality of the collected dataset. Coreset is a small weighted dataset, functioning as a proxy of the original dataset. However, existing coreset construction algorithms are each tailor-made for a specific machine learning problem. That is, we are required to construct different coresets to support different machine learning models. In our first step, we resolve this dilemma by developing robust coreset construction algorithms based on k-clustering algorithms. Our solution is proved to give a guaranteed approximation for a broad range of machine learning problems with sufficiently continuous cost functions.

In our second step, we propose the first framework to incorporate quantization techniques into the process of coreset construction. Specifically, we theoretically analyze the ML error caused by a combination of coreset construction techniques and quantization techniques. Based on that, we formulate an optimization problem to minimize the ML error under a fixed budget of communication cost. To improve the scalability for large datasets, we identify two proxies of the original objective function, for which efficient algorithms are developed. For the case of data on multiple nodes, we further design a novel algorithm to allocate the communication budgets to different nodes while minimizing the overall ML error.

As our third step, we consider the problem of solving edge-based k-means on a large dataset in high dimensional space. In this application scenario, data sources offload machine learning computation to nearby edge servers under limited communication budget and computation power. To solve this problem, we propose to construct small data summaries with fewer data samples (by techniques for Cardinality Reduction (CR)), fewer features (by techniques for Dimensionality Reduction (DR)) and fewer precision bits (by techniques for Quantization (QT)). By analyzing the complexity, the communication cost, and the approximation error of k-means algorithms based on state-of-the-art data reduction methods, we show that: (i) it is possible to achieve a near-optimal approximation at a near-linear complexity and a constant communication

cost, (ii) the order of applying DR and CR leads to a tradeoff between the complexity and the communication cost, (iii) combining DR/CR methods with a properly selected quantizer can further reduce the communication cost without compromising the other performance metrics.

At last, in each step, the effectiveness of our analysis is verified through extensive experiments on multiple real datasets and different machine learning problems.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor Professor Ting He for her kind help during my PhD study at Penn State. Prof. He shows great passion about research and rigorous attitudes towards difficult problems. I also want to thank my committee members for your availabilities. I know all my committee members are very busy and it is very appreciated you can help review my dissertation and give valuable comments. At last, I would like to thank my labmates and my friends for their support and friendship during the past several years. I am very glad I can make it step by step with all support from my advisor, committee and family. And I hope I can keep going and solve more problems in research as well as in my life.

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, either expressed or implied, 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.

Chapter 1
Introduction

Sensor-driven distributed intelligent systems are becoming ubiquitous in a variety of applications such as precision agriculture, machine health monitoring, environmental tracking, traffic management, and infrastructure security. The rapid development of Wearables, Internet of Things (IoT) and other technologies are generating more and more data everyday and enables more data-driven applications. It has been predicted that the rate of such distributed data generation will exceed the current Internet capacity in the near future [1]. While the surge in distributed data generation powered by various sensors has enabled novel features based on machine learning techniques, there are many challenges towards collecting such distributed data. Various resource and application constraints make it challenging to gather voluminous data from distributed data sources, including limitation on network connectivity and bandwidth, limitation on power consumption, and the need to preserve the privacy of raw data. Consequently, there is an increasing need for techniques to efficiently apply machine learning on distributed datasets. In this dissertation, we are particularly interested in reducing the size of data while we can still preserve good machine learning performances under the constraints of communication and computation.

The current distributed machine learning approaches can be broadly classified as follows: (1) those that globally aggregate the outputs of local models; (2) those that construct global models from individual models derived from local data, and (3) those that share representative local data with a global aggregator. An example of the first approach involves independent computations at individual nodes and sharing the outputs of local models [2]. These independent outputs are aggregated using methods such as majority voting at a global aggregator. In contrast to the first approach, the second approach shares the models created from local data [3–5]. The individual models are combined to create a global model using techniques such as weighted average. The third

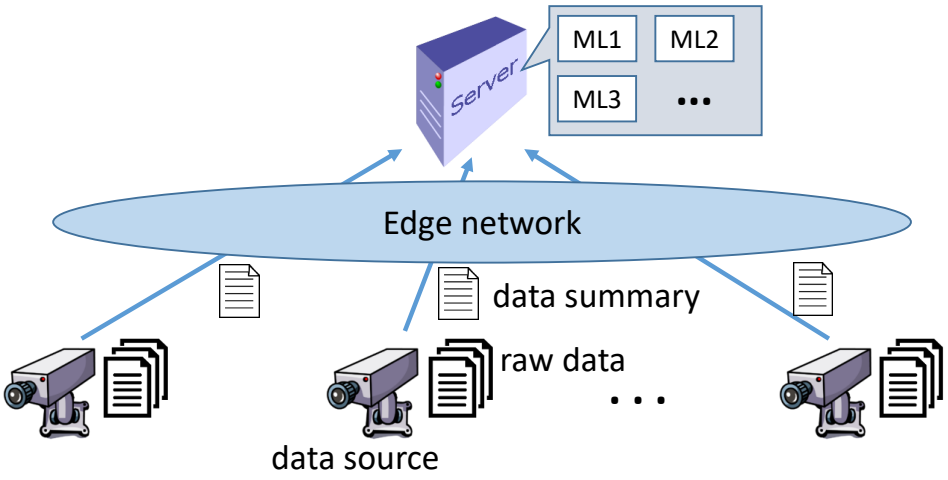


Figure 1.1. Application scenario (ML i: machine learning model i).

approach [6–8], which is the focus of this dissertation, is to share summaries of local data towards the creation of a shared 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, hence amortizing the overhead. Our focus on third approach is driven by its promise to train multiple models using the same data summary, amortizing the communication overhead between the edge nodes and the global aggregator across machine learning problems. Compared to alternative approaches, e.g., transmitting locally learned model parameters as in federated learning, transmitting data summaries has the advantages that: (i) only one round of communication is required,[1] (ii) the transmitted data can potentially be used to compute other machine learning models, and (iii) the edge server can usually solve the machine learning problem closer to the optimality than the data-collecting devices (*data sources*) within the same time. It is also promising to consider combining the strengths of different approaches in a hybrid way such that we could maximize the advantages from them and minimize the inherited negative impacts. But this research problem still largely remains open, and [9] presented the first comparison between sharing models vs sharing data, which made the first step towards exploiting the connections and differences between different approaches.

In this dissertation, we are particularly interested in the edge-based learning scenario

[1]In cases that the raw data are spread over multiple nodes, another round of communications is needed to decide the sizes of data summaries to collect from each node [6]. However, each node only sends one scalar in this round and hence the communication cost is negligible.

illustrated in Figure 1.1 [4], where data sources report local summaries to an edge server, which then computes various machine learning models from these summaries. Under this scenario, machine learning applications over large distributed datasets propose challenges to unleash the valuable information hidden in the data. It is easy to observe the following: the large amount of raw data are located at different data sources with limited computation power, while the server is powerful to train multiple machine learning models but has no direct access to the datasets. One key challenge for such applications is the high communication cost in training ML models. In the edge-based learning scenario, it is impossible to transmit all local datasets located at each client to the server due to limited network bandwidth, limited power assumption, privacy issue and other potential reasons. It is intuitive to consider two straightforward approaches: a straightforward solution that first computes the machine learning results at the data sources and then transmits these results will incur a high complexity at the source; meanwhile, another straightforward solution that first sends all the data to the server and then solves the machine learning problem at the server will incur a high communication cost at the source. Consequently, efficient machine learning model constructions over large distributed data is worth exploring. Utilizing modern machine learning techniques, we are able to do personal navigation utilizing smart phone sensors (accelerometer, GPS, temperature, etc) in the field of IoT, real-time facial recognition based on the images captured by surveillance cameras, driving behavior analysis based on trajectories from autonomous vehicles and so on. Huge amount of high-dimensional data are collected on all edge devices while these devices are usually under the constraints of limited computation resources, network bandwidth, etc, which makes communication-efficient machine learning calculations a more imperative task than ever.

One good example of possible application utilizing our methods discussed in this dissertation would be Internet of Things (IoT), where countless physical devices are collecting data and connected with each other through the internet. These devices include surveillance cameras, motion sensors, smart phones, smart cities, autonomous vehicles, etc. A large amount of data is collected and shared between all devices and eventually the whole ecosystem will act in smart ways to save energy consumption, network bandwidth, computation resources and so on. "The Internet of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service" [10]. Today IoT has been growing rapidly due to recent progress in network communications, computation resources, sensor technologies and other research fields. While machine learning technologies are developed

rapidly in the past several decades, they are proved to provide excellent prediction performances, classification accuracy and make better decisions. Thus it is promising to apply modern machine learning methods on real-world IoT problems utilizing the huge amount of distributed high-dimensional data collected and transmitted in IoT systems. Supervised learning, unsupervised learning and reinforcement learning are three popular categories of machine learning models, and they have been proved to bring benefits to IoT. Deep learning is one of the most popular and powerful supervised learning methods, which consists of multiple layers of neurons and tries to minimize the differences between true labels and model predictions. To support deep learning in the field of IoT, [11] presented the performance comparison study of common deep learning techniques including Convolutional Neural Networks and Deep Neural Networks on distributed data collected from wearables and mobile phones. Taking clustering as an example for unsupervised learning model in the IoT world, we know one of the most frequently used clustering models is k-means, which minimizes the sum of distances between data points and corresponding nearest clustering centers. [12] proposed an adaptive approach that augmented the k-means clustering method to help classify the incoming sensor data. Reinforcement learning (RL) is another learning method which falls between supervised learning and unsupervised learning. RL does not rely on the data labels but utilizes rewards and feedbacks. [13] advocated the application of RL algorithms in wireless networks to achieve context awareness and intelligence, and improve the network performances. All of the machine learning applications discussed above requires us to solve the problem of communication-efficient machine learning in edge-based learning scenario, which will be the main topic of this dissertation.

1.1 Proposed Approach

Suppose we are given a dataset with multiple data instances and many different features, we also suppose we use fixed number of binary bits to represent one value in this dataset. Then this dataset will have three dimensions: cardinality n, number of features d and number of precision bits b_0. This data cube is illustrated in Figure 1.2. Taking the popular MNIST dataset as an example, in the training set of MNIST, there are 60000 images with $28 * 28 = 784$ pixels in each. The gray level of each pixel is an integer in $[0, 255]$, which can be denoted by 8 binary bits. Therefore, for MNIST training set, $n = 60000$, $d = 784$ and $b_0 = 8$. If we use k, d' and b to denote the reduced dimensions, then in our problem, since we would like to reduce the three dimensions simultaneously,

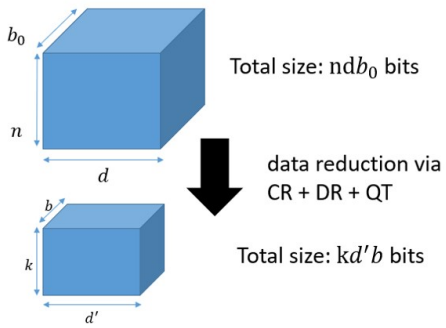


Figure 1.2. The data cube

we will want $kd'b$ to be much smaller than ndb_0.

Regarding three dimensions, we need different data reduction techniques: coreset construction, projection, quantization and so on. To reduce the cardinality, the main technique we used is coreset [14]. Coreset is a small, weighted set of data points which approximates the original dataset in fitting a machine learning model. More precisely, for a set of candidate queries X, and a measure function $\text{cost}(P, x), x \in X$, the set D is an ϵ-coreset for the original dataset P if $\text{cost}(D, x)$ approximates $\text{cost}(P, x)$ up to a multiplicative factor of $1 \pm \epsilon$ for every $x \in X$, i.e.,

$$(1-\epsilon)\text{cost}(P,x) \leq \sum_{p \in D} w(p)\text{cost}(p,x) \leq (1+\epsilon)\text{cost}(P,x) \tag{1.1}$$

The advantage of coreset is that it is quite small such that we can apply complex machine learning algorithm on this smaller dataset. Coreset could also be applicable to streaming or distributed setting. However, one limitation is that we need to construct different coresets to support different machine learning problems.

To reduce the number of precision bits, we asked quantization [15] for help. Quantization utilizes less bits and could give us the approximation error of machine learning performance degradation. There are many quantizers eligible to use, one of which is rounding. We can just truncate each value and we will know the distance between original data point and quantized data point. Our solution will utilize the maximum quantization error, defined as the maximum Euclidean distance between any data point and its quantized version. Thus in our analysis, the approximation error is established based on this maximum quantization error.

To reduce the number of features, we are interested in different dimensionality reduction techniques, including SVD decomposition [16], sparse PCA [17,18], JL projection [19]

and so on. If we only consider cardinality and feature dimension, then the data cube is a data matrix, on which we can do SVD decomposition. Setting several diagonal elements to be zeros, our new data set is now on a low-dimensional plane in original space.

1.2 Roadmap and problems

This thesis has three main chapters addressing three different problems, all focused on data reduction for communication-efficient machine learning problems. We take three steps towards reducing the size of dataset in all three dimensions at last.

We know that the limitation of coreset is that it is tailor-made for specific machine learning problem. Therefore, the question is that is there a universally good coreset construction algorithm to support various machine learning algorithms? This question is answered affirmatively in Chapter 2. An efficient coreset construction algorithm based on k-clustering is proposed. Our solution is also extended to edge-based distributed setting.

Following the same direction, we continue to further reduce the number of precision bits. Given the communication budget as the number of bits to bound the size of the constructed data summary, Chapter 3 proposes the first framework to incorporate quantization techniques into the process of coreset construction. Verified by extensive experiments, we are able to show our proposed algorithm works well to configure the cardinality and precision.

At last, in order to jointly reduce cardinality, feature dimension and precision, we limit our scope to k-means problem since it is one of the fundamental research fields in computer science. We show in Chapter 4 the upper bound of approximation error after we reduce the cardinality and feature dimension. Based on these analysis, we answer the questions of applying these techniques repeatedly by following different orders. At last, we give the method to configure the dimensionality reduction, cardinality reduction and quantization, and validate our findings through various experiments.

Chapter 2
Robust Coreset Construction for Distributed Machine Learning

2.1 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. The rate of such distributed data generation is predicted to 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, as discussed in our Chapter 1, 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 [6–8], data sources share summaries of their local data, which are then used to compute a global model.

In this chapter, we take the third approach, with a particular interest in supporting diverse machine learning models. Fig. 2.1 illustrates a typical application scenario in the context of *mobile edge computing* [4], where data sources report local summaries to an

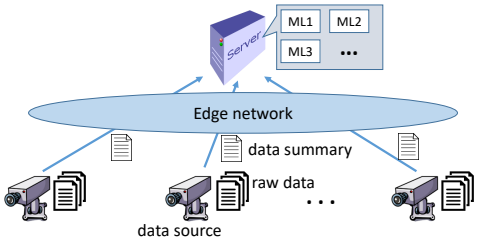


Figure 2.1. Application scenario (ML i: machine learning model i).

edge server, which then computes various models from these summaries.

In particular, we consider data summarization using *coreset* [14]. 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 2.2.2), 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 good for a broad set of machine learning problems?*

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.

2.1.1 Related Work

Distributed learning is considered 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 [20]. The first approach (i.e., collecting outputs) is more popular among earlier works. For example, [21] proposed various heuristic decision rules (e.g., majority vote) to combine outputs of local classifiers, and [22] proposed to train a global classifier using labeled outputs of local classifiers. The solution in [22] was modified

in [23] to improve efficiency for large-scale distributed data, and extended in [20] to include various ways of composing the global training set. The idea was later used to build a descriptive model from distributed data [24]. To further improve the accuracy, a distributed-pasting-votes framework was proposed in [25] to learn sets of classifiers (ensembles).

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 [26] 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 [27,28], 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., [6–8]. 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 2.2.2 for a detailed review of related works on coreset.

2.2 Background

2.2.1 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* $\mathrm{cost}(p, x)$ ($p \in P$, $x \in \mathcal{X}$), and an *overall cost function* $\mathrm{cost}(P, x)$ ($x \in \mathcal{X}$) 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 $w_{\min} \coloneqq \min_{p \in P} w_p$ denote the minimum weight. For an unweighted dataset, we have $w_p \equiv 1$. The machine learning problem is then to solve

$$x^* = \arg\min_{x \in \mathcal{X}} \mathrm{cost}(P, x) \tag{2.1}$$

for the optimal model parameter x^*.

Example: Let $\mathrm{dist}(p, x) := \|p - x\|_2$ denote the Euclidean distance between points p and x. The *minimum enclosing ball (MEB)* problem [14] 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 2.2.1 ([29]). *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 $\forall x \in \mathcal{X}$,*

$$(1-\epsilon)\,cost(P,x) \leq cost(S,x) \leq (1+\epsilon)\,cost(P,x), \tag{2.2}$$

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 2.2.1, 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.

2.2.2 Coreset Construction Algorithms

Because of the dependence on the cost function (Definition 2.2.1), 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 [27, 28] for detailed surveys.

1) Gradient descent algorithms: Originally proposed for MEB [14,30], 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 [14,31]. As *support vector machine (SVM)* training can be formulated as MEB problems [32], similar algorithms have been used to support SVM [32, 33]. Variations have been used for dimensionality reduction [34] and probabilistic MEB [35]. These algorithms are considered as variations of the Frank-Wolfe algorithm [36].

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* [37], where each data point is sampled with a probability proportional to its contribution to the overall cost. Proposed for numerical integration [37], the idea was extended into a framework supporting projective clustering problems that include k-median/means and *principle component analysis (PCA)* as special cases [29]. The framework has been used to generate coresets for other problems, e.g., dictionary learning [38] and dependency networks [39], and is further generalized in [40]. 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. Specific instances have been developed for weighted facility problems [41], Euclidean graph problems [42], k-means/median [8, 43].

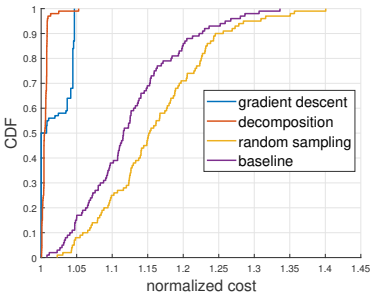
While there are a few works not fully covered by the above approaches, e.g., SVD-based algorithms in [16, 44], the above represents the key approaches used by existing coreset construction algorithms. Using a generic merge-and-reduce approach in [45], all these algorithms can be used to construct coresets of 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.

2.3 Robust Coreset Construction

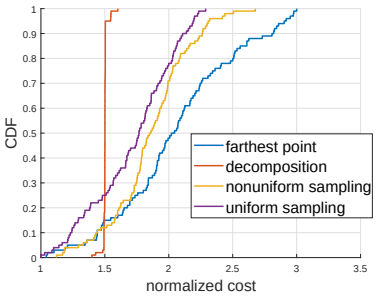
Our main result is that selecting *representative points* using clustering techniques yields a coreset that achieves a good approximation for a broad set of machine learning problems.

2.3.1 Motivating Experiment

We start with an initial experiment that compares three algorithms representing the three approaches in Section 2.2.2: the algorithm in [30] representing 'gradient descent', the framework in [29] instantiated for k-means representing 'random sampling', and the algorithm in [8] representing 'decomposition'. We add uniform sampling as a baseline. As



(b) MEB



(c) k-means ($k = 2$)

Figure 2.2. Comparison of coreset construction algorithms (coreset size: 8).

the algorithm in [30] was designed for MEB and the others were designed for k-means, we evaluate both MEB and k-means.

The evaluation is based on a synthetic dataset containing 4000 points uniformly distributed in $[1, 50]^3$; evaluations on real datasets will be presented later (Section 2.5). All the algorithms are tuned to have the same average coreset size. We evaluate the performance of a coreset S by the *normalized cost*, defined as $\text{cost}(P, x_S)/\text{cost}(P, x^*)$, 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 in Fig. 2.2.

Not surprisingly, a coreset construction algorithm designed for one problem can perform poorly for another, e.g., the gradient descent algorithm [30] designed for MEB performs poorly for k-means. Interestingly, the decomposition algorithm [8], although designed for k-means, also performs well for MEB. This observation triggers a question: Is the superior performance of the decomposition algorithm [8] just a coincidence, or is there something fundamental?

2.3.2 The k-clustering Problem

At the core, the decomposition algorithm in [8] constructs a k-point coreset by partitioning the dataset into k clusters using k-means clustering, and then using the cluster centers as the coreset points. To analyze its performance in supporting a general machine learning problem, 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

$$c(P, Q) = \sum_{p \in P} w_p (\min_{q \in Q} \text{dist}(p, q))^z, \tag{2.3}$$

for a constant $z > 0$. The *k-clustering* problem is to find the set of k centers that minimizes (2.3). 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 this auxiliary problem and $\text{cost}(P, \cdot)$ 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:

$$\mu(P) = \frac{1}{\sum_{p \in P} w_p} \sum_{p \in P} w_p \cdot p. \tag{2.4}$$

We denote by $\text{opt}(P, k)$ the optimal cost for k-clustering of P. It is known that k-means and k-median are both NP-hard problems [46, 47], for which efficient heuristics exist (e.g., Lloyd's algorithm and variations) [48]. Let $\text{approx}(P, k)$ denote the cost of a generic k-clustering algorithm, which always satisfies $\text{approx}(P, k) \geq \text{opt}(P, k)$.

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.

2.3.3 Coreset by Optimal k-clustering

We will show that the superior performance of the algorithm in [8] observed in Section 2.3.1 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 2.3.1).

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 2.3.2), as otherwise selecting an outlier as the second center would have reduced the cost substantially.

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

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 2.3.3 and 2.3.4).

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 2.3.1).

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

Lemma 2.3.1. *For any $\epsilon' > 0$, if $opt(P,k) - opt(P,2k) \leq \epsilon'$, then $opt(P_i,1) - opt(P_i,2) \leq \epsilon'$ $(\forall i \in [k])$, where $\{P_i\}_{i=1}^k$ is the partition of P generated by the optimal k-clustering.*

Lemma 2.3.2. *If $opt(P_i,1) - opt(P_i,2) \leq \epsilon'$, then $dist(p,\mu(P_i)) \leq (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$, $\forall p \in P_i$.*

Lemma 2.3.3. *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 $p \in P_i$,*

$$(1-\epsilon)cost(p,x) \leq cost(\mu(P_i),x) \leq (1+\epsilon)cost(p,x), \tag{2.5}$$

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 2.3.4. *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 (2.5) 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 2.3.1–2.3.4.

Theorem 2.3.1. *If $opt(P,k) - opt(P,2k) \leq w_{\min}(\frac{\epsilon}{\rho})^z$, 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) $cost(p,x) \geq 1$, and (ii) $cost(p,x)$ is ρ-Lipschitz-continuous in p, $\forall x \in \mathcal{X}$.*

Proof. By Lemma 2.3.1, $\text{opt}(P,k) - \text{opt}(P,2k) \leq \epsilon'$ implies $\text{opt}(P_i,1) - \text{opt}(P_i,2) \leq \epsilon'$, $\forall$ cluster P_i generated by the optimal k-clustering. By Lemma 2.3.2, this in turn implies that $\text{dist}(p,\mu(P_i)) \leq (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$, $\forall p \in P_i$. Because $\text{cost}(p,x)$ is ρ-Lipschitz-continuous in p for all $x \in \mathcal{X}$, we have

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

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

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

for $\epsilon' = w_{\min}(\frac{\epsilon}{\rho})^z$. By Lemma 2.3.3, k-clustering gives an ϵ-coreset for P w.r.t. the sum cost; by Lemma 2.3.4, 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 2.3.1 to characterize the quality of approximation as a function of the coreset size.

Corollary 2.3.1.1. *Given a maximum coreset size $k \in \mathbb{Z}^+$ (positive integers), for any cost function satisfying the conditions in Theorem 2.3.1, the optimal k-clustering gives an ϵ-coreset for P w.r.t. this cost function, where*

$$\epsilon = \rho \left(\frac{opt(P, k) - opt(P, 2k)}{w_{\min}} \right)^{\frac{1}{z}}. \tag{2.6}$$

Proof. This is a direct implication of Theorem 2.3.1, as setting ϵ by (2.6) satisfies the condition in Theorem 2.3.1. □

Remark: Condition (i) in Theorem 2.3.1 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 $|\text{cost}(P, x) - \text{cost}(S, x)| \leq \widetilde{\epsilon}$ $(\forall x \in \mathcal{X})$, 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:

$$|\text{dist}(p, x) - \text{dist}(p', x)| \leq \text{dist}(p, p'). \tag{2.7}$$

Hence, its cost function is 1-Lipschitz-continuous (i.e., $\rho = 1$). See Appendix B for more examples. In Section 2.5, we will stress-test our coreset when this condition is violated.

2.3.4 Coreset by Suboptimal k-clustering

While Theorem 2.3.1 and Corollary 2.3.1.1 suggest that the optimal k-clustering gives a good coreset, the k-clustering problem is NP-hard [46,47]. The question is whether similar performance guarantee holds 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 $k \geq 1$, the cost of the algorithm satisfies

$$\text{approx}(P, 2k) \leq \sum_{i=1}^{k} \text{approx}(\widetilde{P}_i, 2). \tag{2.8}$$

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

$$\text{approx}(P, 2) \leq c(P, \{\mu(P), p^*\}), \tag{2.9}$$

where $c(P, Q)$ is defined in (2.3), and $p^* \coloneqq \arg\max_{p \in P} w_p \cdot \text{dist}(p, \mu(P))^z$ is the point with the highest 1-clustering cost.

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 [49] that can compute the 1-median to 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. We will discuss how to ensure these assumptions for the proposed algorithm in Section 2.3.5.

We show that for any k-clustering algorithm satisfying these assumptions, statements analogous to Lemma 2.3.1 and Lemma 2.3.2 can be made (proofs in Appendix A). Let $\{\widetilde{P}_i\}_{i=1}^k$ denote the partition of P generated by the k-clustering algorithm.

Lemma 2.3.5. *For any $\epsilon' > 0$, if $approx(P,k) - approx(P,2k) \leq \epsilon'$, then $approx(\widetilde{P}_i, 1) - approx(\widetilde{P}_i, 2) \leq \epsilon'$ for any $i \in [k]$.*

Lemma 2.3.6. *If $approx(\widetilde{P}_i, 1) - approx(\widetilde{P}_i, 2) \leq \epsilon'$, then $dist(p, \mu(\widetilde{P}_i)) \leq (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$, $\forall p \in \widetilde{P}_i$.*

Theorem 2.3.2. *If $approx(P,k) - approx(P,2k) \leq w_{\min}(\frac{\epsilon}{\rho})^z$, 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 2.3.1.*

Proof. The proof follows the same steps as that of Theorem 2.3.1, except that Lemma 2.3.1 is replaced by Lemma 2.3.5, and Lemma 2.3.2 is replaced by Lemma 2.3.6. Note that Lemmas 2.3.3 and 2.3.4 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 2.3.1.1, we can rephrase Theorem 2.3.2 to characterize the quality of a coreset of a specified size.

Corollary 2.3.2.1. *Given a maximum coreset size $k \in \mathbb{Z}^+$, for any cost function satisfying the conditions in Theorem 2.3.1 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*

$$\epsilon = \rho \left(\frac{approx(P,k) - approx(P,2k)}{w_{\min}} \right)^{\frac{1}{z}}. \tag{2.10}$$

2.3.5 Coreset Construction Algorithm

Based on Theorem 2.3.2, we propose a centralized 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.

The k-clustering subroutine: Algorithm 1 can use any k-clustering algorithm as subroutine, although our performance guarantee holds only if the algorithm satisfies Assumptions 1–3. We note that these assumptions are easy to satisfy. Consider the standard k-means algorithm (Lloyd's algorithm), which iteratively assigns each point to

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

input : A weighted set P with minimum weight $w_{\min}$, approximation error $\epsilon > 0$, Lipschitz constant ρ

output : An ϵ-coreset S for P w.r.t. a cost function satisfying Theorem 2.3.2

foreach $k = 1, \ldots, |P|$ **do**
  if $\mathit{approx}(P, k) - \mathit{approx}(P, 2k) \leq w_{\min}(\frac{\epsilon}{\rho})^z$ **then**
    break;
$(\{\mu(\widetilde{P}_i)\}_{i=1}^k, \{\widetilde{P}_i\}_{i=1}^k) \leftarrow k$-clustering$(P, k)$;
$S \leftarrow \{\mu(\widetilde{P}_i)\}_{i=1}^k$, where $\mu(\widetilde{P}_i)$ has weight $\sum_{p \in \widetilde{P}_i} w_p$;
return S;

the nearest center and updates the centers to the means of the clusters. Clearly, this algorithm satisfies Assumption 1. Moreover, with the following initialization, it also satisfies Assumptions 2 and 3. For $2k$-clustering of P:

1. if $k = 1$, then use the mean $\mu(P)$ and the point p^* with the highest clustering cost as defined in (2.9) as the initial centers;

2. if $k > 1$, then first compute k-clustering of P, then compute 2-clustering of each of the k clusters, and finally use the union of the 2-clustering centers as the initial centers.

Any odd number of initial centers are chosen randomly. Since iterations can only reduce the cost, Lloyd's algorithm with this initialization satisfies Assumptions 1–3.

2.4 Distributed Coreset Construction

In a distributed setting, the entire dataset P is distributed across n ($n > 1$) nodes $v_1, \ldots, v_n$, where each v_j has a subset $P_j \subseteq P$. We have shown in Section 2.3 that the k-clustering centers of P form a robust coreset. However, computing the global k-clustering centers of a distributed dataset is highly non-trivial. Note that a naive solution that only includes local centers in the global coreset may select nearly identical points at different nodes if the local datasets are similar, which is non-optimal and inefficient.

An existing algorithm: The state-of-the-art solution to the distributed k-clustering problem is based on an algorithm called *Communication-aware Distributed Coreset Construction (CDCC)* [6]. In our context, this solution works as follows:

1. the server allocates a given sample size t among the nodes, such that the sample size t_j allocated to node v_j is proportional to the local k-clustering cost $c(P_j, B_j)$ reported by v_j;

2. each node v_j generates and reports a local coreset D_j, consisting of the local centers B_j and t_j points sampled i.i.d. from P_j, where each $p \in P_j$ has a sampling probability proportional to the cost of clustering p to the nearest center in B_j;

3. the server computes a set of k-clustering centers Q_D from the coreset $D = \bigcup_{j=1}^{n} D_i$.

It is shown in [6] that if $t = O(\frac{1}{\epsilon^2}(kd + \log \frac{1}{\delta}))$ for k-median and $t = O(\frac{1}{\epsilon^4}(kd + \log \frac{1}{\delta}) + nk \log \frac{nk}{\delta})$ for k-means, then with probability at least $1 - \delta$, D is an ϵ-coreset for P w.r.t. the cost function of k-median/means. According to Definition 2.2.1, this implies that if Q_P is the set of optimal k-clustering centers for P, then $c(P, Q_D)/c(P, Q_P) \leq (1 + \epsilon)/(1 - \epsilon)$.

Adaptation for coreset construction: First, we skip step (3) (i.e., computation of Q_D) and directly use $D = \bigcup_{j=1}^{n} D_j$ as the coreset. This is because the coreset of a coreset cannot have a better quality than the original coreset [45].

Moreover, in CDCC, the number of local centers k is a given parameter as it is only designed to support k-clustering. Since our goal is to support a variety of machine learning problems, the number of local centers k_j at each v_j becomes a design parameter that can vary across nodes. Given a global coreset size N, we will show that the approximation error of the constructed coreset depends on $(k_j)_{j=1}^{n}$ through $\frac{1}{\sqrt{N - \sum_{j=1}^{n} k_j}} \sum_{j=1}^{n} \text{approx}(P_j, k_j)$ (see Theorem 2.4.1). Thus, we set $(k_j)_{j=1}^{n}$ to minimize this error, and obtain the remaining $t = N - \sum_{j=1}^{n} k_j$ points by sampling.

Combining these ideas yields a distributed coreset construction algorithm called *Distributed Robust Coreset Construction (DRCC)* (Algorithm 2). The algorithm works in three steps: (1) each node reports its local k-clustering cost for a range of k (lines 2-3), (2) the server uses the reported costs to configure the number of local centers k_j and the number of random samples t_j at each node v_j (lines 5-7), and (3) each node independently constructs a local coreset using a combination of samples and local centers (lines 9-12). DRCC generalizes CDCC in that: (i) it allows the input dataset to be weighted (w_p: weight of input point p; u_q: weight of coreset point q); (ii) it allows the number of local centers to be different for different nodes. In the special case of $k_j \equiv k$ for all $j \in [n]$ and $w_p \equiv 1$ for all $p \in P$, DRCC is reduced to CDCC.

Communication overhead: DRCC has a communication overhead of $O(Kn)$, measured by the total number of scalars reported by the nodes besides the coreset

Algorithm 2: Distributed Robust Coreset Construction$((P_j)_{j=1}^n, N, K)$

input : A distributed dataset $(P_j)_{j=1}^n$, global coreset size N, maximum number of local centers K

output : A coreset $D = \bigcup_{j=1}^n (S_j \cup B_j^{k_j})$ for $P = \bigcup_{j=1}^n P_j$

each v_j $(j \in [n])$**:**

compute local approximate k-clustering centers B_j^k on P_j for $k = 1, \ldots, K$;

report $(c(P_j, B_j^k))_{k=1}^K$ to the server;

the server:

find $(k_j)_{j=1}^n$ that minimizes $\frac{1}{\sqrt{N - \sum_{j=1}^n k_j}} \sum_{j=1}^n c(P_j, B_j^{k_j})$ s.t. $k_j \in [K]$ and $\sum_{j=1}^n k_j \leq N$;

randomly allocate $t = N - \sum_{j=1}^n k_j$ points i.i.d. among $v_1, \ldots, v_n$, where each point belongs to v_j with probability $\frac{c(P_j, B_j^{k_j})}{\sum_{j=1}^n c(P_j, B_j^{k_j})}$;

communicate $(k_j, t_j, \frac{C}{t})$ to each v_j $(j \in [n])$, where t_j is the number of points allocated to v_j and $C = \sum_{l=1}^n c(P_l, B_l^{k_l})$;

each v_j $(j \in [n])$**:**

sample a set S_j of t_j points i.i.d. from P_j, where each sample equals $p \in P_j$ with probability $\frac{m_p}{c(P_j, B_j^{k_j})}$ for $m_p = c(\{p\}, B_j^{k_j})$;

set the weight of each $q \in S_j$ to $u_q = \frac{C w_q}{t m_q}$;

set the weight of each $b \in B_j^{k_j}$ to $u_b = \sum_{p \in P_b} w_p - \sum_{q \in P_b \cap S_j} u_q$, where P_b is the set of points in P_j whose closest center in $B_j^{k_j}$ is b;

report each point $q \in S_j \cup B_j^{k_j}$ and its weight u_q to the server;

itself. In practice, K should be a small constant to allow efficient computation of local k-clustering for $k \in [K]$. This overhead is much smaller than the $O((n-1)Nd)$ overhead of the merge-and-reduce approach in [45].

Quality of coreset: Regarding the quality of the coreset, we have proved the following result in Appendix A. Given a general per-point cost function $\text{cost}(p, x)$, define $f_x(p) := w_p(\text{cost}(p, x) - \text{cost}(b_p, x) + \rho \text{dist}(p, b_p))$, where b_p is the center in $B_j^{k_j}$ closest to $p \in P_j$. Let $\dim(F, P)$ denote the *dimension of the function space* $F := \{f_x(p) : x \in \mathcal{X}\}$ [6].

Theorem 2.4.1. *If cost(p, x) is ρ-Lipschitz-continuous in p for any $x \in \mathcal{X}$, then $\exists t = O(\frac{1}{\epsilon^2}(\dim(F, P) + \log \frac{1}{\delta})$ such that with probability at least $1 - \delta$, the coreset D constructed by DRCC based on local k-median clustering, which contains k_j local centers from v_j*

($j \in [n]$) and t random samples, satisfies

$$\left|\sum_{p\in P} w_p \, cost(p,x) - \sum_{q\in D} u_q \, cost(q,x)\right| \leq 2\epsilon\rho \sum_{j=1}^{n} c(P_j, B_j^{k_j}) \tag{2.11}$$

for all $x \in \mathcal{X}$.

The parameter $\dim(F,P)$ is a property of the machine learning problem under consideration, which intuitively measures the degree of freedom in the solution $x \in \mathcal{X}$, e.g., $\dim(F,P) = O(kd)$ for k-means/median in d-dimensional space [29]. See Appendix C for more discussions.

Due to the relationship between t and ϵ given in Theorem 2.4.1, the bound on the right-hand side of (2.11) depends on parameters t and $(k_j)_{j=1}^n$ through $\frac{1}{\sqrt{t}} \sum_{j=1}^n c(P_j, B_j^{k_j})$. Thus, given a total coreset size $N = t + \sum_{j=1}^n k_j$, we select these parameters to minimize the error bound (line 5 in Algorithm 2).

Remark: The performance bound in Theorem 2.4.1 is on the absolute error, instead of the relative error as guaranteed by an ϵ-coreset. Nevertheless, if $\exists \beta > 0$ and $(k_j)_{j=1}^n$ such that $\text{cost}(P,x) = \sum_{p\in P} w_p \text{cost}(p,x) \geq \beta \sum_{j=1}^n c(P_j, B_j^{k_j})$ for all $x \in \mathcal{X}$, then (2.11) implies that D is an ϵ-coreset for P w.r.t. $\text{cost}(P,x)$ with probability at least $1-\delta$ if $t = O(\frac{\rho^2}{\epsilon^2}(\dim(F,P)+\log\frac{1}{\delta}))$, i.e., the total coreset size $N = O(\frac{\rho^2}{\epsilon^2}(\dim(F,P)+\log\frac{1}{\delta})+\sum_{j=1}^n k_j)$. In the special case where $\text{cost}(P,x)$ is the k-median clustering cost and $k_j \equiv k$, we have $\rho = 1$ (Appendix B) and $\dim(F,P) = O(kd)$ [29], and thus the size of an ϵ-coreset is $O(\frac{1}{\epsilon^2}(kd + \log\frac{1}{\delta}) + kn)$, which generalizes the result in [6] to weighted datasets.

2.5 Performance Evaluation

We evaluate the proposed coreset construction algorithms and their benchmarks on a variety of machine learning problems and datasets, and compare the cost of each model learned on a coreset with the cost of the model learned on the original dataset. We first perform evaluations in a centralized setting to compare different approaches to construct coresets, and then evaluate different ways of applying the most promising approach in a distributed setting.

Coreset construction algorithms: In the centralized setting, we evaluate RCC based on k-median clustering ('RCC-kmedian') and RCC based on k-means clustering

[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.

Table 2.1. Parameters of Datasets

dataset	size ($\|P\|$)	dimension (d)	#different labels (L)
Fisher's iris	150	5	3
Facebook	500	19	4
Pendigits	7494	17	10
MNIST	60000	401	10

Table 2.2. Machine Learning Cost Functions

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

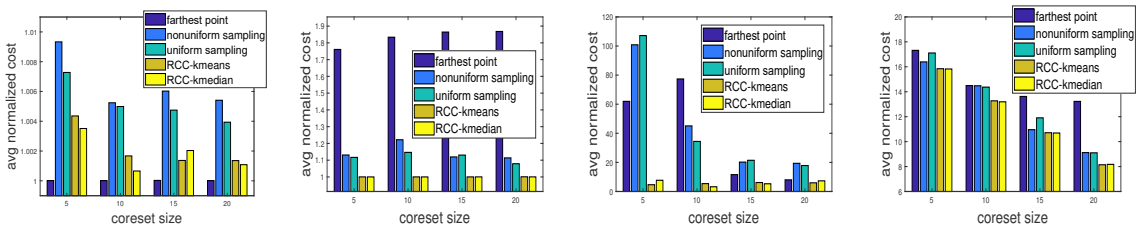
('RCC-kmeans'), together with benchmarks including the algorithm in [30] ('gradient descent'), the framework in [29] instantiated for k-means ('random sampling'), and uniform sampling ('baseline'). We note that the algorithm in [8] ('decomposition' in Fig. 2.2) is essentially RCC based on k-means clustering, hence omitted.

In the distributed setting, we take the best-performing algorithm in the centralized setting ('RCC-kmeans') and evaluate its distributed extensions, including CDCC [6], a straightforward extension where each node runs local (N/n)-means to compute its portion of the coreset (N: total coreset size, n: number of nodes), and DRCC.

Datasets: We use four different datasets: (1) Fisher's iris data [50], which is a 5-dimensional dataset consisting of measurements of 150 iris specimens and their species, (2) Facebook metrics [51], which is a 19-dimensional dataset consisting of features of 500 posts published in a popular Facebook page, (3) Pendigits data [52], which is a 17-dimensional dataset consisting of feature vectors of 7494 handwritten digits, and (4) MNIST data [53], which consists of $60,000$ images of handwritten digits, each trimmed to 20×20 pixels.

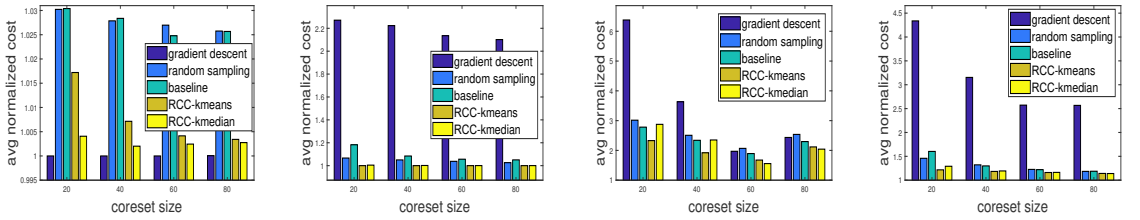
We normalize each numerical dimension to $[0, 1]$. We map labels to numbers such that the distance between two points with different labels is no smaller than the distance between points with the same label. Given a d-dimensional dataset (including labels) with L types of labels, we map type-l label to $(l-1)\tau$ ($l \in [L]$) for $\tau = \lceil\sqrt{d-1}\rceil$. See Table 2.1 for a summary. In testing SVM, we map one label to '1' and the rest to '-1'. Each data point has a unit weight.

To generate distributed datasets, we use three schemes: (i) *uniform*, where the points are uniformly distributed across n nodes, (ii) *specialized*, where each node is associated



(a) MEB (b) k-means ($k = 2$) (c) PCA (3 components) (d) SVM ('setosa': 1; others: -1)

Figure 2.3. Evaluation on Fisher's iris dataset with varying coreset size (label: 'species').



(a) MEB (b) k-means ($k = 2$) (c) PCA (5 components) (d) SVM ('photo': 1; others: -1)

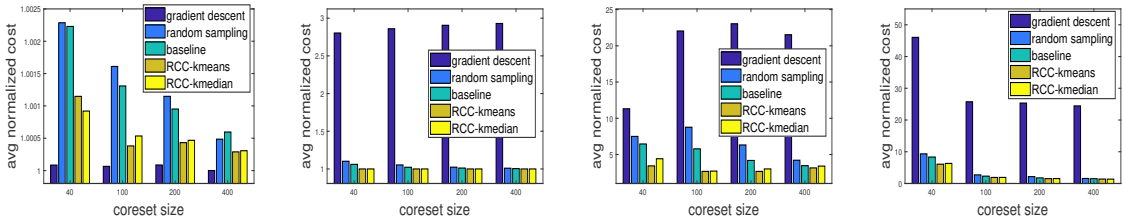
Figure 2.4. Evaluation on Facebook metrics dataset with varying coreset size (label: 'type').

with one label and contains all the data points with this label, and (iii) *hybrid*, where the first n_0 nodes are "specialized" as in (ii), and the remaining data are randomly partitioned among the remaining nodes.

Machine learning problems: We evaluate three unsupervised learning problems—MEB, k-means, and PCA, and one supervised learning problem—SVM. Table 2.2 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 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 4.5 for Fisher's iris, 13.4 for Facebook, 36.2 for Pendigits, and 181.1 for MNIST. While SVM does not have a finite ρ, we still include it to stress-test our algorithm.

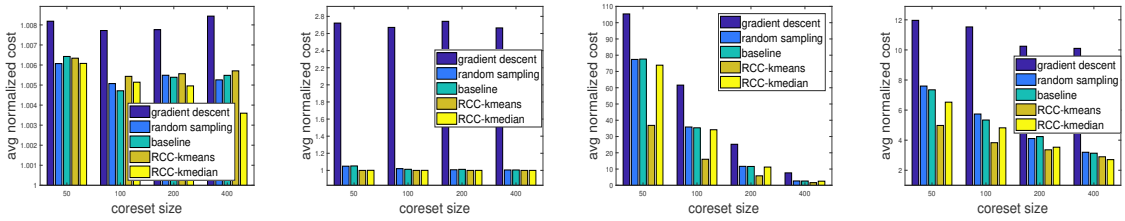
Results in centralized setting: We evaluate the coreset construction algorithms by the normalized costs of the models learned from the coresets, as explained in Section 2.3.1. Figures 2.3–2.6 show the average normalized costs over 100 Monte Carlo runs.

We see that the proposed algorithms ('RCC-kmeans' and 'RCC-kmedian') perform either the best or comparably to the best across all the datasets and all the machine learning problems. The gradient descent algorithm in [30], designed for MEB, can perform very poorly for other machine learning problems. The sampling-based algorithms



(a) MEB (b) k-means ($k = 2$) (c) PCA (11 components) (d) SVM ('0': 1; others: -1)

Figure 2.5. Evaluation on Pendigits with varying coreset size (label: 'digit').



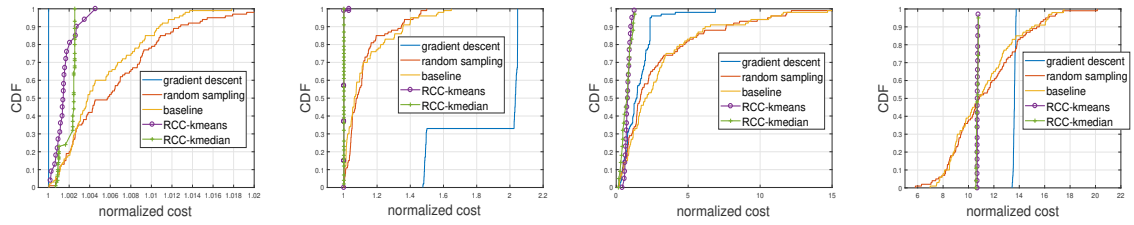
(a) MEB (b) k-means ($k = 2$) (c) PCA (300 components) (d) SVM ('0': 1; others: -1)

Figure 2.6. Evaluation on MNIST with varying coreset size (label: 'labels').

('random sampling' in [29] and 'baseline') perform relatively poorly for MEB and PCA. Note that all the algorithms are within 3% of the optimal for MEB.

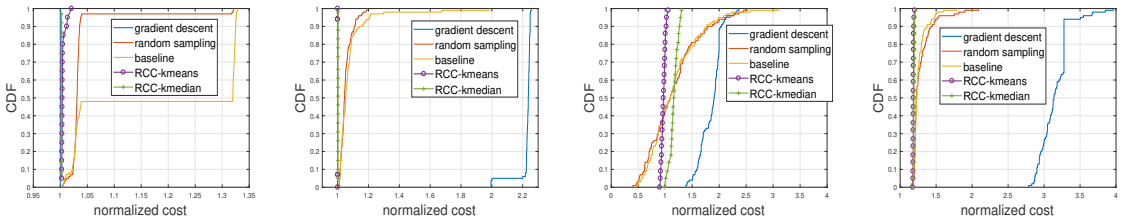
Besides the average normalized costs, we also evaluated the CDFs of the normalized costs. The results are shown in Figures 2.7–2.10. The results show similar comparisons between the algorithms as observed in Figures 2.3–2.6. Moreover, we see from the CDFs that the proposed algorithms ('RCC-kmeans' and '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. 2.6 (c–d). Moreover, we note that 'RCC-kmeans' can be an order of magnitude faster than 'RCC-kmedian', as shown in Table 2.3. Other than 'RCC-kmedian',



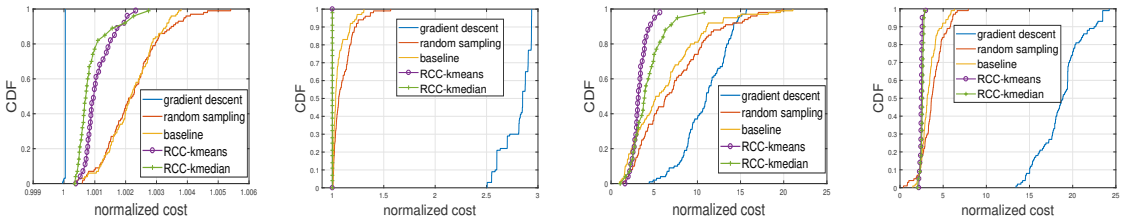
(a) MEB (b) k-means ($k = 2$) (c) PCA (3 components) (d) SVM ('setosa': 1; others: -1)

Figure 2.7. Detailed evaluation on Fisher's iris dataset (label: 'species', coreset size: 15).



(a) MEB (b) k-means ($k = 2$) (c) PCA (5 components) (d) SVM ('photo': 1; others: -1)

Figure 2.8. Detailed evaluation on Facebook metrics dataset (label: 'type', coreset size: 40).



(a) MEB (b) k-means ($k = 2$) (c) PCA (11 components) (d) SVM ('0': 1; others: -1)

Figure 2.9. Detailed evaluation on Pendigits dataset (label: 'digit', coreset size: 40).

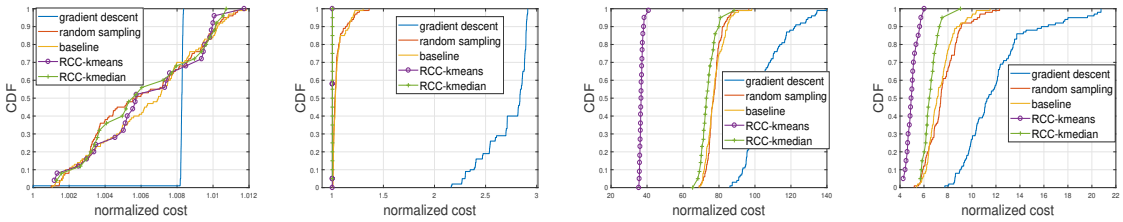
all the algorithms can finish in a few seconds. In Appendix D, we have also validated the approximation bound given by Corollary 2.3.2.1.

Results in distributed setting: We use the two larger datasets, Pendigits and MNIST, to generate distributed datasets at $n = 10$ nodes according to the three aforementioned schemes ('uniform', 'specialized', 'hybrid'), where node j ($j \in [n]$) is associated with label '$j - 1$', and $n_0 = 5$. We parameterize CDCC with $k = 2$ according to the evaluated k-means problem. For DRCC (Algorithm 2), we solve line 5 by a greedy heuristic. As a benchmark, we also show performance of 'RCC-kmeans', which is the best-performing algorithm *in the centralized setting*. We generate m_1 distributed datasets using each scheme and repeat coreset construction for m_2 times for each dataset. For Pendigits, $m_1 = m_2 = 10$. For MNIST, $m_1 = m_2 = 5$. Figures 2.11–2.12 show the average normalized costs.

As CDCC blindly designates k coreset points to be local centers at each node, it suffers when the local datasets are highly heterogeneous (under the 'hybrid' scheme).

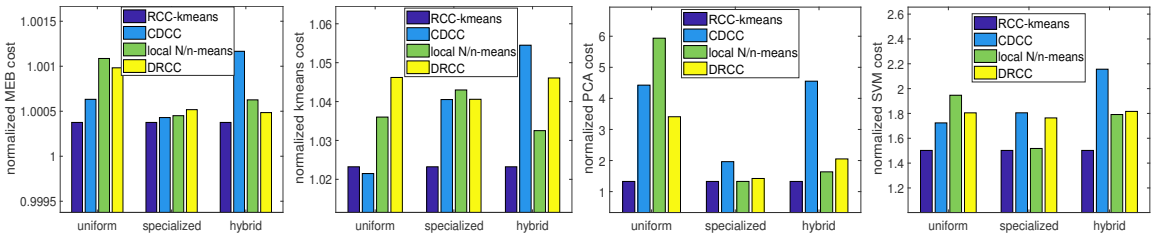
Table 2.3. Average Running Time (sec)

algorithm	Fisher's iris	Facebook	Pendigits	MNIST
gradient descent	0.02	0.04	0.06	7.27
random sampling	0.06	0.03	0.19	4.34
baseline	0.001	0.001	0.003	0.01
RCC-kmeans	0.04	0.02	0.13	13.59
RCC-kmedian	0.05	0.48	1.28	123.82



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

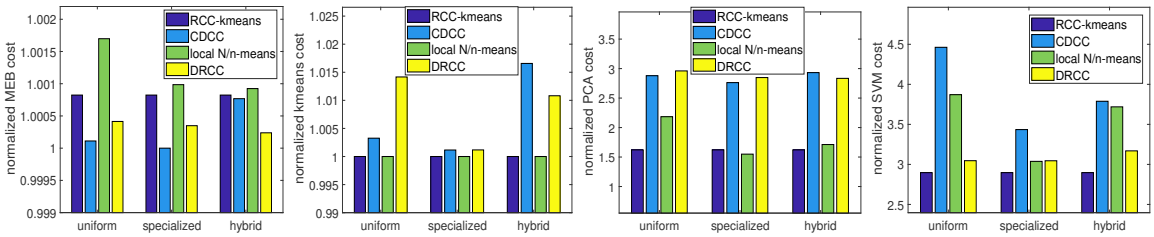
Figure 2.10. Detailed evaluation on MNIST dataset (label: 'labels', coreset size: 50).



(a) MEB (b) k-means ($k = 2$) (c) PCA (11 components) (d) SVM ('0': 1; others: -1)

Figure 2.11. Evaluation on Pendigits in distributed settings (label: 'digit', coreset size: 200, $K = 20$).

Meanwhile, a solution that only includes local centers into the coreset ('local N/n-means') may select nearly identical points at different nodes if the local datasets follow the same distribution (under the 'uniform' scheme), thus not fully utilizing the given coreset size. By automatically tuning the numbers of local centers according to the local clustering costs, DRCC is able to customize its configuration to the dataset at hand and thus achieves a more robust performance, i.e., providing better performance *on the average* across different models and datasets. When DRCC performs worse than CDCC and 'local N/n-means' (for k-means under the 'uniform' scheme), the difference is very small ($< 3\%$). Note that 'RCC-kmeans' is a centralized algorithm and is only used to provide a lower bound for the other three algorithms that run in the distributed setting.



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

Figure 2.12. Evaluation on MNIST in distributed settings (label: 'labels', coreset size: 400, $K = 10$).

2.6 Conclusion

We show, both theoretically and empirically, that the k-clustering centers form a coreset that provides a guaranteed 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 one to leverage existing k-clustering algorithms for coreset construction. In particular, we have adapted an existing distributed k-clustering algorithm to construct a robust coreset over distributed data with a very low communication overhead. Our extensive evaluations verify the superior performance of the proposed algorithms in supporting diverse machine learning problems.

Chapter 3
Joint Coreset Construction and Quantization for Distributed Machine Learning

3.1 Introduction

The rapid development of data capturing technologies, e.g., wearables and Internet of Things (IoT), has fueled the explosive growth of data-driven applications that employ various *machine learning (ML)* models to utilize the valuable information hidden in the data. One key challenge for such applications is the high communication cost in training ML models over large distributed datasets. One approach to address this challenge is federated learning [5, 54, 55], where distributed agents iteratively exchange model parameters to collectively train a global model. The exchanged parameters, however, are only useful for a single model, and separate communications are needed to train different models, limiting the efficiency in simultaneously training multiple ML models. When the goal is to train multiple models, which is the focus of our work in this paper, the alternative approach of collecting data summaries at a central location (e.g., a server) is often more efficient, as the summaries can be used to train multiple ML models, amortizing the communication cost.

To reduce the original dataset into small summaries, several techniques have been proposed, which can be generally classified into: 1) sketching techniques for reducing the feature dimension [56–58]; and 2) coreset construction techniques for reducing the sample dimension [6, 14, 16, 29, 34, 37, 59–61]. However, sketches change the feature space and thus require adaptations of the ML tasks, e.g., the feature space of a classifier needs

to be modified to be applicable to the sketching results. In comparison, coresets only reduce the cardinality of the datasets and preserve the feature space, making them directly applicable to the original ML tasks. Therefore, we focus on coreset-based data summarization.

Coresets [14, 29, 59–61] are small, weighted versions of the original dataset, lying in the same feature space. Existing coreset construction algorithms focus on maximally reducing the cardinality with provable guarantees on the ML error. However, most of these algorithms are model-specific, i.e., constructing different coresets for training different ML models, which seriously limits their capability in reducing the communication cost when training multiple ML models. Recently, a *robust coreset construction (RCC)* algorithm was proposed to address this issue [62], where a clustering-based coreset was proved to be applicable for training a variety of ML models with provable error bounds.

However, existing coreset construction algorithms only reduce the number of data points, but not the number of bits required to represent each data point. The latter is the goal of quantization [15, 63], where various techniques, from simple rounding-based quantizers to sophisticated vector quantizers, have been proposed to transform the data points from arbitrary values in the sample space to a set of discrete values that can be encoded by a smaller number of bits [64–67].

In this chapter, we propose the first framework to optimally integrate coreset construction and quantization. Intuitively, under a given communication budget specifying the total number of bits to collect, there is a trade-off between collecting more data points at a lower precision and collecting fewer data points at a higher precision. *Jointly* configuring the quantizer and the coreset construction algorithm to achieve the best trade-off can potentially achieve a smaller ML error than using quantization or coreset construction alone. Our goal is to realize this potential by developing efficient algorithms to compute the optimal configuration parameters explicitly.

3.2 Related work

Coresets have been widely applied in shape fitting and clustering problems [34]. Previous coreset construction algorithms can be classified into four categories: sampling-based algorithms [37, 68–70], SVD-based algorithms [16, 44, 71], space-decomposition algorithms [43, 62], and iterative algorithms [14, 36]. Sampling-based algorithms [37, 68–70] leverage importance/sensitivity sampling and other advanced sampling techniques to select a subset of the original data points to form a coreset. SVD-based algorithms [16, 44, 71] use

the singular value decomposition (SVD) to build coresets for ML tasks including dictionary design and sparse representation. Space-decomposition algorithms [43, 62] partition the original sample space based on different criteria and then select representative points from these partitions. Iterative algorithms [14, 36] select points according to pre-defined criteria to form a coreset in an iterative manner.

However, most existing coreset construction algorithms are model-specific [29]. That is, different coresets will be constructed for training different ML models, increasing the communication cost in collecting the coresets when training multiple ML models. To address this issue, *robust coreset construction (RCC)* has been recently proposed in [62], where a single coreset can support a variety of ML models with provable error bounds. Therefore, in this chapter, we focus on RCC as our choice of coreset construction algorithm.

Quantization techniques [15, 63–67, 72–74] aim to quantize the data points to a set of discrete values so that each quantized value can be encoded by a smaller number of bits. Different quantizers have been proposed to support different applications [64–67]. Recently, quantization has been leveraged to reduce the size of ML models without seriously degrading the model accuracy [72–74]. Existing quantizers can be classified into *scalar quantizers* and *vector quantizers*, where scaler quantizers apply quantization operations to each attribute of a data point, and vector quantizers [75, 76] apply quantization to each data point as a whole. In this work, we focus on a simple rounding-based scalar quantizer due to its simplicity and broad applicability. However, we note that our analysis can be easily extended to any given quantizer.

Despite extensive studies of coreset construction and quantization separately, to our knowledge, how to optimally combine them remains an open question. To this end, we propose the first framework to integrate coreset construction and quantization, by formulating and solving optimization problems to jointly configure the coreset construction algorithm and the quantizer at hand to achieve the optimal tradeoff between the ML error and the communication cost.

3.3 Preliminaries

In this section, we briefly review several definitions and algorithms that will be used in subsequent sections. Frequently used notations in this paper are listed in Table 3.1.

Table 3.1. Main notations

Variable	Definition
CS	The operation of coreset construction
QT	The operation of quantization
ϵ, ϵ_i	Overall/local ML error
B, B_i	Global/local communication budget
$\mathcal{Y}$, $\mathcal{Y}_i$	Total/local original dataset
$\mathcal{S}$	Coreset
n, k	Cardinalities of $\mathcal{Y}$ and $\mathcal{S}$
$\mathbf{y}_i$, y_{ij}	One data point and one attribute of the data point
b_0, b	#bits for representing each attribute in $\mathcal{Y}$ and $\mathcal{S}$
Δ	Maximum quantization error
$\mathbf{x}$, $\mathcal{X}$	One solution and solution space for the ML task
$\text{cost}(\mathcal{Y}, \mathbf{x})$	cost function of the ML task
ρ	Lipschitz constant for the ML cost function
$\text{opt}(k)$	Optimal k-means clustering cost for $\mathcal{Y}$
$\text{opt}_\infty(k)$	Optimal k-center clustering cost for $\mathcal{Y}$
m_e	# of exponent bits in the floating point representation of an attribute
N	Number of nodes in distributed setting

3.3.1 Data Representation

Let $\mathcal{Y}$ denote the original dataset with cardinality $n := |\mathcal{Y}|$, dimension d, and precision b_0. Each data point $\mathbf{y}_i \in \mathcal{Y}$ is a column vector in d-dimensional space, and each attribute y_{ij} is represented as a floating point number with a sign bit, an m_e-bit exponent, and a $(b_0 - 1 - m_e)$-bit significand. Let $\mathbf{Y} := [\mathbf{y}_1, ..., \mathbf{y}_i, ..., \mathbf{y}_n]$ denote the matrix with column vectors $\mathbf{y}_i$. For simplicity of analysis, we assume that y_{ij}'s have been normalized to $[-1, 1]$ with zero mean (i.e., $\frac{1}{n}\sum_i y_{ij} = 0$ for each j). Let $\mu(\mathcal{Y}) := \frac{1}{n}\sum_{\mathbf{y}_i \in \mathcal{Y}} \mathbf{y}_i$ denote the sample mean of $\mathcal{Y}$.

3.3.2 Coreset Construction

A generic ML task can be considered as a cost minimization problem. Using $\mathcal{X}$ to denote the set of possible models, and $\text{cost}(\mathcal{Y}, \mathbf{x})$ to denote the mismatch between the dataset $\mathcal{Y}$ and a candidate model $\mathbf{x}$, the problem seeks to find the model that minimizes $\text{cost}(\mathcal{Y}, \mathbf{x})$. The cost function $\text{cost}(\mathcal{Y}, \mathbf{x})$ is usually in the form of a summation $\text{cost}(\mathcal{Y}, \mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \text{cost}(\mathbf{y}, \mathbf{x})$ or a maximization $\text{cost}(\mathcal{Y}, \mathbf{x}) = \max_{\mathbf{y} \in \mathcal{Y}} \text{cost}(\mathbf{y}, \mathbf{x})$, where $\text{cost}(\mathbf{y}, \mathbf{x})$ is the per-point cost that is model-specific. For example, minimum enclosing ball (MEB) [60] minimizes a maximum cost and k-means minimizes a sum cost.

A coreset $\mathcal{S}$ is a weighted (and often smaller) dataset that approximates $\mathcal{Y}$ in terms of costs.

Definition 3.3.1 (ϵ_{CS}-coreset [29])**.** *A set $\mathcal{S} \subseteq \mathbb{R}^d$ with weights $u_{\mathbf{q}}$ ($\forall \mathbf{q} \in \mathcal{S}$) is an ϵ_{CS}-coreset for $\mathcal{Y}$ with respect to (w.r.t.) cost$(\mathcal{Y}, \mathbf{x})$ ($\mathbf{x} \in \mathcal{X}$) if $\forall \mathbf{x} \in \mathcal{X}$,*

$$(1 - \epsilon_{CS})\,cost(\mathcal{Y}, \mathbf{x}) \leq cost(\mathcal{S}, \mathbf{x}) \leq (1 + \epsilon_{CS})\,cost(\mathcal{Y}, \mathbf{x}), \tag{3.1}$$

where cost$(\mathcal{S}, \mathbf{x})$ is defined as cost$(\mathcal{S}, \mathbf{x}) = \sum_{\mathbf{q} \in \mathcal{S}} u_{\mathbf{q}}\, cost(\mathbf{q}, \mathbf{x})$ if cost$(\mathcal{Y}, \mathbf{x})$ is a sum cost, and cost$(\mathcal{S}, \mathbf{x}) = \max_{\mathbf{q} \in \mathcal{S}} cost(\mathbf{q}, \mathbf{x})$ if cost$(\mathcal{Y}, \mathbf{x})$ is a maximum cost.

Definition 3.3.1 also provides a performance measure for coresets: denoted by $\epsilon_{\text{CS},\mathcal{S}} := \sup_{\mathbf{x} \in \mathcal{X}} |\text{cost}(\mathcal{Y}, \mathbf{x}) - \text{cost}(\mathcal{S}, \mathbf{x})|/\text{cost}(\mathcal{Y}, \mathbf{x})$, it measures the maximum relative error in approximating the ML cost function by coreset $\mathcal{S}$, called the *ML error* of $\mathcal{S}$. The smaller $\epsilon_{\text{CS},\mathcal{S}}$, the better $\mathcal{S}$ is in supporting the ML task.

Although most coreset construction algorithms only provide guaranteed performance for specific ML tasks, a recent work [62] showed that using clustering centers, especially k-means clustering centers, as the coreset achieves guaranteed performances for a broad class of ML tasks with Lipschitz-continuous cost functions. In the sequel, we denote the optimal k-means clustering cost for $\mathcal{Y}$ by opt(k). It is known that the optimal 1-means center of $\mathcal{Y}$ is the sample mean $\mu(\mathcal{Y})$.

3.3.3 Quantization

Quantization reduces the number of bits required to encode each data point by transforming it to the nearest point in a set of discrete points, the selection of which largely defines the quantizer. Our solution will utilize the maximum quantization error, defined as $\Delta := \max_{\mathbf{y} \in \mathcal{Y}} \text{dist}(\mathbf{y}, \mathbf{y}')$, where $\mathbf{y}'$ denotes the quantized version of data point $\mathbf{y}$ and $\text{dist}(\mathbf{y}, \mathbf{y}')$ is their Euclidean distance. Given a quantizer, Δ depends on the number of bits used to represent each quantized value. Below we analyze Δ for a simple but practical *rounding-based quantizer* as a concrete example, but our framework also allows other quantizers.

Let y_{ij} denote the j-th attribute of the i-th data point. The b_0-bit binary floating point representation of y_{ij} is given by $(-1)^{\text{sign}(y_{ij})} \times 2^{e_{ij}} \times (a_{ij}(0) + a_{ij}(1) \times 2^{-1} + \ldots + a_{ij}(b_0 - 1 - m_e) \times 2^{-(b_0 - 1 - m_e)})$ [77]. Here, sign(y_{ij}) is the sign of y_{ij} (0: nonnegative, 1: negative), e_{ij} is an m_e-bit exponent, and $a_{ij}(\cdot) \in \{0, 1\}$ are the significant digits, where $a_{ij}(0) \equiv 1$ and does not need to be stored explicitly.

Consider a scalar quantizer that rounds each y_{ij} to s significant digits. The quantized value equals $y'_{ij} = (-1)^{\text{sign}(y_{ij})} \times 2^{e_{ij}} \times (a_{ij}(0) + a'_{ij}(1) \times 2^{-1} + \ldots + a'_{ij}(s) \times 2^{-s})$, where $a'_{ij}(s) \in \{0, 1\}$ is the result of rounding the remaining digits (0: round down, 1: round up). As $|y_{ij} - y'_{ij}| \leq 2^{e_{ij}-s}$ and $|y_{ij}| \geq 2^{e_{ij}}$, we have $|y_{ij} - y'_{ij}|/|y_{ij}| \leq 2^{-s}$. Hence, for $\mathcal{Y}$ in $\mathbb{R}^d$ where each attribute y_{ij} is normalized to $[-1, 1]$, the maximum quantization error of this quantizer is bounded by

$$\Delta \leq 2^{-s} \cdot \max_{\mathbf{y}_i \in \mathcal{Y}} \|\mathbf{y}_i\|. \tag{3.2}$$

3.4 Optimal Combination of Coreset Construction and Quantization

In this section, we first analyze the ML error bounds based on the data summary computed by a combination of coreset construction and quantization, and then formulate an optimization problem to minimize the ML error under a given budget of communication cost.

3.4.1 Workflow Design

The first question in the integration of quantization (QT) into coreset construction (CS) is to determine the order of these two operations. Intuitively, QT is needed after CS since the CS algorithm can result in arbitrary values that cannot be represented using b bits as specified for the quantizer. Therefore, we consider a pipeline where CS is followed by QT.

3.4.2 Error Bound Analysis

The error bound for CS + QT is stated as follows.

Theorem 3.4.1. *After applying a Δ-maximum-error quantizer to an ϵ_{CS}-coreset $\mathcal{S}$ of the original dataset $\mathcal{Y}$, the quantized coreset $\mathcal{S}'$ is an $(\epsilon_{CS} + \rho\Delta + \epsilon_{CS} \cdot \rho\Delta)$-coreset for $\mathcal{Y}$ w.r.t. any cost function satisfying:*

1. *cost$(\mathbf{y}, \mathbf{x}) \geq 1$*
2. *cost$(\mathbf{y}, \mathbf{x})$ is ρ-Lipschitz-continuous in $\mathbf{y} \in \mathcal{Y}$, $\forall \mathbf{x} \in \mathcal{X}$.*

Theorem 3.4.1 is directly implied by the following Lemma 3.4.1, which gives the ML error after one single quantization.

Lemma 3.4.1. *Given a set of points $\mathcal{Y} \subseteq \mathbb{R}^d$, let $\mathcal{Y}'$ be the corresponding set of quantized points with a maximum quantization error of Δ. Then, $\mathcal{Y}'$ is a $\rho\Delta$-coreset of $\mathcal{Y}$ w.r.t. any cost function satisfying the conditions in Theorem 3.4.1.*

Proof of Lemma 3.4.1. For each $\mathbf{y} \in \mathcal{Y}$, we know $\text{dist}(\mathbf{y}, \mathbf{y}') \leq \Delta$. By the ρ-lipschitz-continuity of $\text{cost}(\cdot, \mathbf{x})$, we have

$$|\text{cost}(\mathbf{y}, \mathbf{x}) - \text{cost}(\mathbf{y}', \mathbf{x})| \leq \rho\Delta. \tag{3.3}$$

Moreover, since $\text{cost}(\mathbf{y}, \mathbf{x}) \geq 1$, we have

$$\frac{|\text{cost}(\mathbf{y}, \mathbf{x}) - \text{cost}(\mathbf{y}', \mathbf{x})|}{\text{cost}(\mathbf{y}, \mathbf{x})} \leq \rho\Delta, \tag{3.4}$$

and thus

$$(1 - \rho\Delta)\text{cost}(\mathbf{y}, \mathbf{x}) \leq \text{cost}(\mathbf{y}', \mathbf{x}) \leq (1 + \rho\Delta)\text{cost}(\mathbf{y}, \mathbf{x}). \tag{3.5}$$

If $\text{cost}(\mathcal{Y}, \mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \text{cost}(\mathbf{y}, \mathbf{x})$, then treating $\mathcal{Y}'$ as a coreset with unit weights, its cost is $\text{cost}(\mathcal{Y}', \mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}'} \text{cost}(\mathbf{y}', \mathbf{x})$. Summing (3.5) over all $\mathbf{y} \in \mathcal{Y}$ (or $\mathbf{y}' \in \mathcal{Y}'$), we have

$$(1 - \rho\Delta)\text{cost}(\mathcal{Y}, \mathbf{x}) \leq \text{cost}(\mathcal{Y}', \mathbf{x}) \leq (1 + \rho\Delta)\text{cost}(\mathcal{Y}, \mathbf{x}). \tag{3.6}$$

If $\text{cost}(\mathcal{Y}, \mathbf{x}) = \max_{\mathbf{y} \in \mathcal{Y}} \text{cost}(\mathbf{y}, \mathbf{x})$, then the cost of $\mathcal{Y}'$ is $\text{cost}(\mathcal{Y}', \mathbf{x}) = \max_{\mathbf{y}' \in \mathcal{Y}'} \text{cost}(\mathbf{y}', \mathbf{x})$. Suppose that the maximum is achieved at $\mathbf{y}_1$ for $\text{cost}(\mathcal{Y}, \mathbf{x})$, and $\mathbf{y}'_2$ for $\text{cost}(\mathcal{Y}', \mathbf{x})$. Based on (3.5), we have

$$(1 - \rho\Delta)\text{cost}(\mathbf{y}_1, \mathbf{x}) \leq \text{cost}(\mathbf{y}'_1, \mathbf{x}) \leq \text{cost}(\mathbf{y}'_2, \mathbf{x}) \tag{3.7a}$$

$$\leq (1 + \rho\Delta)\text{cost}(\mathbf{y}_2, \mathbf{x}) \leq (1 + \rho\Delta)\text{cost}(\mathbf{y}_1, \mathbf{x}) \tag{3.7b}$$

which again leads to (3.6) as $\text{cost}(\mathcal{Y}, \mathbf{x}) = \text{cost}(\mathbf{y}_1, \mathbf{x})$ and $\text{cost}(\mathcal{Y}', \mathbf{x}) = \text{cost}(\mathbf{y}'_2, \mathbf{x})$. □

Proof of Theorem 3.4.1. By Lemma 3.4.1 and Definition 3.3.1, we have that $\forall \mathbf{x} \in \mathcal{X}$,

$$\begin{aligned}
&(1-\epsilon_{\text{CS}})(1-\rho\Delta)\text{cost}(\mathcal{Y}, \mathbf{x}) \leq (1-\rho\Delta)\text{cost}(\mathcal{S}, \mathbf{x}) \\
&\leq \text{cost}(\mathcal{S}', \mathbf{x}) \leq (1 + \rho\Delta)\text{cost}(\mathcal{S}, \mathbf{x}) \\
&\leq (1 + \epsilon_{\text{CS}})(1 + \rho\Delta)\text{cost}(\mathcal{Y}, \mathbf{x}),
\end{aligned} \tag{3.8}$$

which yields the result. □

3.4.3 Configuration Optimization

3.4.3.1 Abstract Formulation

Our objective is to minimize the ML error under bounded communication costs, through the joint configuration of coreset construction and quantization. Given a n-point dataset in $\mathbb{R}^d$ and a communication budget of B, we aim to find a quantized coreset $\mathcal{S}$ with k points and a precision of b bits per attribute, that can be represented by no more than B bits. Our goal is to *Minimize the Error under a given Communication Budget (MECB)*, formulated as

$$\min_{b,k} \quad \epsilon_{\text{CS}}(k) + \rho\Delta(b) + \epsilon_{\text{CS}}(k) \cdot \rho\Delta(b) \tag{3.9a}$$

$$\text{s.t.} \quad b \cdot k \cdot d \leq B, \tag{3.9b}$$

$$b,\ k \in \mathbb{Z}^+, \tag{3.9c}$$

where $\epsilon_{\text{CS}}(k)$ represents the ML error of a k-point coreset constructed by the given coreset construction algorithm, and $\Delta(b)$ is the maximum quantization error of b-bit quantization by the given quantizer. We want to find the optimal values of k and b to minimize the error bound (3.9a) according to Theorem 3.4.1, under the given budget B. Note that our focus is on finding the optimal configuration of known CS/QT algorithms instead of developing new algorithms.

3.4.3.2 Concrete Formulation

We now concretely formulate and solve an instance of MECB for two practical CS/QT algorithms. Suppose that the CS operation is by the k-means based *robust coreset construction (RCC)* algorithm in [62], which is proven to yield a $\rho\sqrt{f(k)}$-coreset for all ML tasks with ρ-Lipschitz-continuous cost functions, where $f(k) := \text{opt}(k) - \text{opt}(2k)$ is the difference between the k-means and the $2k$-means costs. Moreover, suppose that the QT operation is by the rounding-based quantizer defined in Section 3.3.3, which has a maximum quantization error of $\Delta(b) := 2^{-(b-1-m_e)} \max_{\mathbf{y}_i \in \mathcal{Y}} \|\mathbf{y}_i\|$ to generate a b-bit quantization with $s = b - 1 - m_e$ significant digits according to (3.2). Then, by Theorem 3.4.1, the MECB problem in this case becomes:

$$\min_{b,k} \quad \rho\sqrt{f(k)} + \rho\Delta(b) + \rho^2\Delta(b)\sqrt{f(k)} \tag{3.10a}$$

$$\text{s.t.} \quad b \cdot k \cdot d \leq B, \tag{3.10b}$$

$$b, k \in \mathbb{Z}^+. \tag{3.10c}$$

3.4.3.3 Straightforward Solution

In (3.10), only b (or k) is the free decision variable. Thus, a straightforward way to solve (3.10) is to evaluate the objective function[1] (3.10a) for each possible value of b and then select b^* that minimizes the objective value. We refer to this solution as the *EMpirical approach (EM)* later in the paper. However, this approach is computationally expensive for large datasets, as evaluating $f(k)$ requires solving k-means problems for large values of k. To address this challenge, we will develop efficient heuristic algorithms for approximately solving (3.10) in the following section by identifying proxies of the objective function that can be evaluated efficiently.

3.5 Efficient Algorithms for MECB

In this section, we propose two algorithms to effectively and efficiently solve the concrete MECB problem given in (3.10).

3.5.1 Eigenvalue Decomposition Based Algorithm for MECB (EVD-MECB)

3.5.1.1 Re-formulating the Optimization Problem

This algorithm is motivated by the following bound derived in [78].

Theorem 3.5.1 (Bound for k-means costs [78])**.** *The optimal k-means cost for $\mathcal{Y}$ is bounded by*

$$opt(k) \geq n\overline{\mathbf{y}^2} - \sum_{i=1}^{k-1} \lambda_i, \tag{3.11}$$

where $n\overline{\mathbf{y}^2} := \sum_{i=1}^{n} \mathbf{y}_i^T \mathbf{y}_i$ is the total variance and λ_i is the i-th principal eigenvalue of the covariance matrix $\mathbf{Y}\mathbf{Y}^T$.

We use the bound in (3.11) as an approximation of k-means cost that is much faster to evaluate than the exact k-means cost. Replacing opt(k) by this bound, we obtain an

[1]We note that it is NP-hard to compute the k-means costs opt(k) and opt($2k$) in evaluating $f(k)$. Nevertheless, one can compute an approximation using existing k-means heuristics, e.g., [48].

Algorithm 3: EVD-MECB

input : A dataset $\mathcal{Y}$, Lipschitz constant ρ for the targeted ML task, communication budget B.

output : Optimal (k^*, b^*) to configure a quantized ϵ-coreset $\mathcal{S}'$ for $\mathcal{Y}$ within budget B.

Calculate eigenvalues $\{\lambda_i\}_{i=1}^d$ for $\mathbf{Y}\mathbf{Y}^T$;
$\Lambda_j \leftarrow \sum_{i=1}^j \lambda_i, \forall 1 \leq j \leq d$;
foreach $b = [1 + m_e,\ 2 + m_e, \ldots, b_0]$ **do**
    $k \leftarrow \left\lfloor B/d/b \right\rfloor$;
    $f(k) \leftarrow \Lambda_{2k-1} - \Lambda_{k-1}$;
    $\Delta(b) \leftarrow 2^{-(b-1-m_e)} \max_{\mathbf{y}_i \in \mathcal{Y}} \|\mathbf{y}_i\|$;
    $\epsilon(k, b) \leftarrow \rho \cdot \sqrt{f(k)} + \rho \cdot \Delta(b) + \rho^2 \cdot \Delta(b) \cdot \sqrt{f(k)}$;
$(k^*, b^*) \leftarrow \arg\min \epsilon(k, b)$;
return (k^*, b^*);

approximation of (3.10), where $f(k)$ is approximated by

$$f(k) \approx n\overline{\mathbf{y}^2} - \sum_{i=1}^{k-1} \lambda_i - (n\overline{\mathbf{y}^2} - \sum_{i=1}^{2k-1} \lambda_i) = \sum_{i=k}^{2k-1} \lambda_i. \tag{3.12}$$

3.5.1.2 EVD-MECB Algorithm

The righthand side of (3.12) is easier to calculate than the exact value of $f(k)$, as we can compute the eigenvalue decomposition once [79], and use the results to evaluate $\sum_{i=k}^{2k-1} \lambda_i$ for all possible values of k. As each number in $\mathcal{Y}$ has $b_0 - 1 - m_e$ significant digits, the number of feasible values for b (and hence k) is $b_0 - 1 - m_e$. By enumerating all the feasible values, we can easily find the optimal solution (k^*, b^*) to this approximation of (3.10). We summarize the algorithm in Algorithm 3.

3.5.2 Max-distance Based Algorithm for MECB (MD-MECB)

3.5.2.1 Re-formulating the Optimization Problem

Alternatively, we can bound the ML error based on the maximum distance between each data point and its corresponding point in the coreset. We demote the maximum distance between any data point and its nearest k-means center by $f_2(k) := \max_{i=1,\ldots,k} \max_{\mathbf{y} \in \mathcal{Y}_i} \text{dist}(\mathbf{y}, \mu(\mathcal{Y}_i))$, where $\{\mu(\mathcal{Y}_i)\}_{i=1}^k$ are the k-means clusters. Then, a similar proof as that of Lemma 3.4.1 implies the following.

Lemma 3.5.1. *The centers of the optimal k-means clustering of $\mathcal{Y}$, each weighted by the number of points in its cluster, provide a $\rho f_2(k)$-coreset for $\mathcal{Y}$ w.r.t. any cost function*

Algorithm 4: k-center cost computation

input : A dataset $\mathcal{Y}$, the maximum number of centers K.
output : The costs $(g(k))_{k=1}^{K}$ for greedy k-center clustering for $k = 1, \ldots, K$.

$\mathcal{G} \leftarrow \emptyset$;
foreach $\mathbf{y} \in \mathcal{Y}$ **do**
  $d(y) \leftarrow \infty$;
while $|\mathcal{G}| < K$ **do**
  find $\mathbf{y} \leftarrow \arg\max_{\mathbf{q} \in \mathcal{Y}} d(\mathbf{q})$;
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathbf{y}\}$;
  foreach $j = 1, \ldots, |\mathcal{Y}|$ **do**
    $d(\mathbf{y}_j) \leftarrow \min(d(\mathbf{y}_j), \operatorname{dist}(\mathbf{y}_j, \mathbf{y}))$;
  $g(|\mathcal{G}|) \leftarrow \max_{\mathbf{y}_j \in \mathcal{Y}} d(\mathbf{y}_j)$;
return $(g(k))_{k=1}^{K}$;

satisfying the conditions in Theorem 3.4.1.

This lemma provides an alternative error bound for the RCC algorithm in [62], which constructs the coreset as in Lemma 3.5.1. Using $\epsilon_{\mathrm{CS}} = \rho f_2(k)$, if we apply the rounding-based quantization after RCC, we can apply Theorem 3.4.1 to obtain an alternative error bound for the resulting quantized coreset, which is $\rho f_2(k) + \rho\Delta(b) + \rho^2 f_2(k)\Delta(b)$. We note that minimizing $f_2(k)$ is exactly the objective of *k-center clustering* [80–82]. Hence, we would like to use the optimal k-center cost, denoted by $\mathrm{opt}_\infty(k)$, as a heuristic to calculate $f_2(k)$. By using this alternative error bound and approximating $f_2(k) \approx \mathrm{opt}_\infty(k)$, we can reformulate the MECB problem as follows:

$$\min_{b,k} \quad \rho \cdot \mathrm{opt}_\infty(k) + \rho\Delta(b) + \rho^2 \cdot \mathrm{opt}_\infty(k)\Delta(b) \tag{3.13a}$$

$$\text{s.t.} \quad b \cdot k \cdot d \leq B, \tag{3.13b}$$

$$b, k \in \mathbb{Z}^+, \tag{3.13c}$$

where $\Delta(b)$ is defined as in (3.10).

3.5.2.2 MD-MECB Algorithm

The re-formulation (3.13) allows us to leverage algorithms for k-center clustering to efficiently evaluate $\mathrm{opt}_\infty(k)$. Although k-center clustering is a NP-hard problem [83], a number of efficient heuristics have been proposed. In particular, it has been proved [83] that the best possible approximation for k-center clustering is 2-approximation, achieved by a simple greedy algorithm [84] that keeps adding the point farthest from the existing

Algorithm 5: MD-MECB

input : A dataset $\mathcal{Y}$, Lipschitz constant ρ for the targeted ML task; communication budget B.

output : Optimal (k^*, b^*) to configure a quantized ϵ-coreset S' for Y within budget B.

1 Run Algorithm 4 with input $\mathcal{Y}$ and $K = \min\{\lfloor \frac{B}{d\cdot(1+m_e)} \rfloor, n\}$;

2 **foreach** $b = [1 + m_e,\ 2 + m_e, \ldots, b_0]$ **do**

3 $\quad k \leftarrow \left\lfloor B/d/b \right\rfloor$;

4 $\quad \mathrm{opt}_\infty(k) \leftarrow g(k)$ by the output of Algorithm 4;

5 $\quad \Delta(b) \leftarrow 2^{-(b-1-m_e)} \max_{\mathbf{y}_i \in \mathcal{Y}} \|\mathbf{y}_i\|$;

6 $\quad \epsilon(k, b) \leftarrow \rho \cdot \mathrm{opt}_\infty(k) + \rho\Delta(b) + \rho^2 \cdot \mathrm{opt}_\infty(k)\Delta(b)$;

7 $(k^*, b^*) \leftarrow \arg\min \epsilon(k, b)$;

8 **return** (k^*, b^*);

centers to the set of centers until k centers are selected. The beauty of this algorithm is that we can modify it to compute the k-center clustering costs for all possible values of k in one pass, as shown in Algorithm 4. Specifically, after adding each center (lines 5–6) and updating the distance from each point to the nearest center (line 8), we record the clustering cost (line 9). As the greedy algorithm achieves 2-approximation [83], the returned costs satisfy $\mathrm{opt}_\infty(k) \le g(k) \le 2\mathrm{opt}_\infty(k)$, where $g(\cdot)$ is defined in line 9. Based on this algorithm, the MD-MECB algorithm, shown in Algorithm 5, solves an approximation of (3.13) with $\mathrm{opt}_\infty(k)$ approximated by $g(k)$.

3.5.3 Discussions

3.5.3.1 Performance Comparison

The straightforward solution EM (Section 3.4.3.3) directly minimizes the error bound (3.10a) and is thus expected to find the best configuration for CS + QT. In comparison, the two proposed algorithms (EVD-MECB and MD-MECB) only optimize approximations of the error bound. It is difficult to theoretically analyze or compare the ML errors of these algorithms since the bound may be loose and the approximations may be smaller than the bound. Instead, we will use empirical evaluations to compare the actual ML errors achieved by these algorithms (see Section 3.7).

3.5.3.2 Complexity Comparison

In terms of complexity, EM involves executions of the k-means algorithm for all (k, b) pairs, which is thus computationally complicated. In comparison, EVD-MECB only requires one eigenvalue decomposition (EVD) and one matrix multiplication, while MD-MECB

only needs to invoke Algorithm 4 once. Therefore, both of them can be implemented efficiently. As EVD can be computed with complexity $O(n^3)$ [85], EVD-MECB has a complexity of $O(n^3 + d + b_0)$. Since the computational complexity of Algorithm 4 is $O(n^2)$ (achieved at $K = n$), MD-MECB has a complexity of $O(n^2 + b_0)$.

Hence, MD-MECB is expected to be more efficient than EVD-MECB, which will be further validated in Section 3.7.

3.6 Distributed Setting

We now describe how to construct a quantized coreset under a global communication budget in distributed settings. Suppose that the data are distributed over N nodes as $\{\mathcal{Y}_1, \ldots, \mathcal{Y}_N\}$. Our goal is to configure the construction of local coresets $\{\mathcal{S}_1, \ldots, \mathcal{S}_N\}$ such that $\bigcup_{i=1}^N \mathcal{S}_i$ can be represented by no more than B bits and is an ϵ-coreset for $\bigcup_{i=1}^N \mathcal{Y}_i$ for the smallest ϵ. Given a distribution of the budget to each node, we can use the algorithms in Section 3.5 to make each $\mathcal{S}_i$ an ϵ_i-coreset of the local dataset $\mathcal{Y}_i$ for the smallest ϵ_i. However, the following questions remain: (1) How is ϵ related to ϵ_i's? (2) How can we distribute the global budget B to minimize ϵ? In this section, we answer these questions by formulating and solving the distributed version of the MECB problem.

3.6.1 Problem Formulation in Distributed Setting

In the following, we first show that $\epsilon = \max_i \epsilon_i$, and then formulate the MECB problem in the distributed setting.

Lemma 3.6.1. *If $\mathcal{C}_1$ and $\mathcal{C}_2$ are ϵ_1-coreset and ϵ_2-coreset for datasets $\mathcal{Y}_1$ and $\mathcal{Y}_2$, respectively, w.r.t. a cost function, then $\mathcal{C}_1 \bigcup \mathcal{C}_2$ is a $\max\{\epsilon_1, \epsilon_2\}$-coreset for $\mathcal{Y}_1 \bigcup \mathcal{Y}_2$ w.r.t. the same cost function.*

Proof. We consider both sum and maximum cost functions. Without loss of generality, we assume $\epsilon_2 \geq \epsilon_1$.

Sum cost: Given a feasible solution $\mathbf{x}$, we consider sum cost as $\text{cost}(\mathcal{Y}, \mathbf{x}) = \sum_{\mathbf{y}\in\mathcal{Y}} \text{cost}(\mathbf{y}, \mathbf{x})$. According to Definition 3.3.1, we have $(1 - \epsilon_1) \sum_{\mathbf{y}\in\mathcal{Y}_1} \text{cost}(\mathbf{y}, \mathbf{x}) \leq \sum_{\mathbf{c}\in\mathcal{C}_1} \text{cost}(\mathbf{c}, \mathbf{x}) \leq (1+\epsilon_1) \sum_{\mathbf{y}\in\mathcal{Y}_1} \text{cost}(\mathbf{y}, \mathbf{x})$ and $(1-\epsilon_2) \sum_{\mathbf{y}\in\mathcal{Y}_2} \text{cost}(\mathbf{y}, \mathbf{x}) \leq \sum_{\mathbf{c}\in\mathcal{C}_2} \text{cost}(\mathbf{c}, \mathbf{x}) \leq (1 + \epsilon_2) \sum_{\mathbf{y}\in\mathcal{Y}_2} \text{cost}(\mathbf{y}, \mathbf{x})$. Summing up these two equations and noting that $\epsilon_2 \geq \epsilon_1$, we can conclude that $\mathcal{C}_1 \bigcup \mathcal{C}_2$ is an ϵ_2-coreset for $\mathcal{Y}_1 \bigcup \mathcal{Y}_2$.

Maximum cost: The proof for maximum cost function is similar as above but taking the maximum instead. □

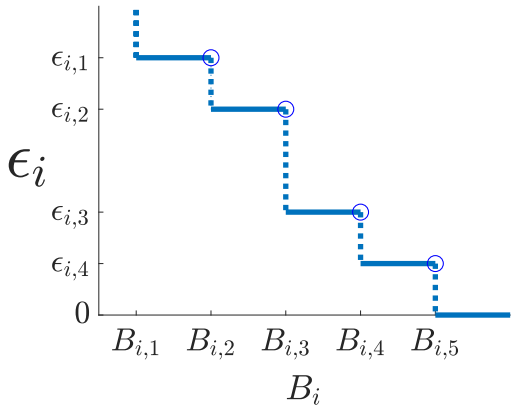


Figure 3.1. Illustration of step objective functions.

We can easily extend Lemma VI.1 to multiple nodes to compute the global ϵ error as: $\epsilon = \max_i \epsilon_i$. Thus the objective of minimizing ϵ is equivalent to minimizing the largest ϵ_i for $i \in \{1, \ldots, N\}$.

Let B_i denote the local budget for the i-th node. Intuitively, the larger the local budget B_i, the smaller ϵ_i. Therefore, we model ϵ_i as a non-increasing function w.r.t. the local budget B_i, denoted by $\epsilon_i(B_i)$.

Then, we can formulate the MECB problem in the distributed setting (MECBD) as follows:

$$\min \quad \max_{i \in \{1,\ldots,N\}} \epsilon_i(B_i) \tag{3.14a}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} B_i \leq B. \tag{3.14b}$$

Note that to compute $\epsilon_i(B_i)$ for a given B_i, we need to solve an instance of the MECB problem in (3.10) for dataset $\mathcal{Y}_i$ and budget B_i.

3.6.2 Optimal Budget Allocation Algorithm for MECBD

The MECBD problem in (3.14) is a *minimax knapsack problem* [86, 87] with a nonlinear non-increasing objective function. Special cases of this problem with strictly decreasing objective functions have been solved in [87]. However, the objective function of MECBD is a step function as shown below, which is not strictly decreasing.

Below we will develop a polynomial-time algorithm to solve our instance of the minimax knapsack problem using the following property of $\epsilon_i(B_i)$.

We note that $\epsilon_i(B_i)$ is a non-increasing step function of B_i (see Figure 3.1). This

Algorithm 6: OBA-MECBD

input : Distributed datasets $\{\mathcal{Y}_i\}_{i=1}^N$, Lipschitz constant ρ, communication budget B.
output : Optimal $\{(k_i^*, b_i^*)\}_{i=1}^N$ to configure the construction of local quantized coresets within a global budget B.

each node $i = 1, ..., N$**:**

1 $B_0 \leftarrow 1 \cdot (1 + m_e) \cdot d$;
2 compute $\epsilon_i(B_0)$ by MD-MECB or EVD-MECB;
3 $\mathcal{E}_i \leftarrow \{(B_0, \epsilon_i(B_0))\}$;
4 **foreach** *integer* $B_i \in [B_0 + 1, |\mathcal{Y}_i| \cdot b_0 \cdot d]$ **do**
5 compute $\epsilon_i(B_i)$ by MD-MECB or EVD-MECB;
6 **if** $\epsilon_i(B_i) < \min\{\epsilon_{i,j} : (B_{i,j}, \epsilon_{i,j}) \in \mathcal{E}_i\}$ **then**
7 $\mathcal{E}_i \leftarrow \mathcal{E}_i \cup \{(B_i, \epsilon_i(B_i))\}$;
8 report $\mathcal{E}_i$ to the server;

the server:

9 $\mathbf{E}$ is an ordered list of ϵ-values in $\bigcup_i \mathcal{E}_i$, sorted in descending order;
10 $I_{max} \leftarrow$ first index in $\mathbf{E}$;
11 $I_{min} \leftarrow$ last index in $\mathbf{E}$;
12 **while** *true* **do**
13 $I \leftarrow \left\lfloor \frac{I_{max} + I_{min}}{2} \right\rfloor$;
14 $\epsilon_I \leftarrow$ the I-th element in $\mathbf{E}$;
15 **for** $i = 1, \ldots, N$ **do**
16 $B_i(\epsilon_I) \leftarrow \min\{B_{i,j} : (B_{i,j}, \epsilon_{i,j}) \in \mathcal{E}_i,\ \epsilon_{i,j} \le \epsilon_I\}$;
17 **if** $I_{min} = I_{max} + 1$ **then**
18 send $B_i(\epsilon_I)$ to node i for each $i = 1, \ldots, N$;
19 break **while** loop;
20 **else**
21 **if** $\sum_{i=1}^N B_i(\epsilon_I) > B$ **then**
22 $I_{min} = I$;
23 **else**
24 $I_{max} = I$;

each node $i = 1, ..., N$**:**

25 find local (k_i^*, b_i^*) under budget $B_i(\epsilon_I)$ given by the server by MD-MECB or EVD-MECB;

26 **return** $\{(k_i^*, b_i^*)\}_{i=1}^N$;

is because the configuration parameters k and b in the CS + QT procedure are integers. Therefore, there exist intervals $[B_{i,j}, B_{i,j+1})$ $(j = 0, 1, 2, ...)$ such that for any $B_i, B_i' \in [B_{i,j}, B_{i,j+1})$, we have $\epsilon_i(B_i) = \epsilon_i(B_i')$, as shown in Figure 3.1. Given a target value of ϵ_i, the minimum B_i for reaching this target is thus always within the set $\{B_{i,j}\}$.

Our algorithm, shown in Algorithm 6, has three main steps. First, in lines 1–8, each

node computes the set $\mathcal{E}_i$ of all pairs of $B_{i,j}$ and the corresponding $\epsilon_i(B_{i,j})$. This is achieved by evaluating $\epsilon_i(B_i)$ according to MD-MECB or EVD-MECB for gradually increasing B_i and recording all the points where $\epsilon_i(\cdot)$ decreases. After that, the set $\mathcal{E}_i$ is sent to a server.

Second, the server allocates the global budget to the nodes according to lines 9–24. To this end, it computes an ordered list $\mathbf{E}$ of all possible values of the global $\epsilon := \max_i \epsilon_i$. Let $B_i(\epsilon)$ denote the smallest value of B_i such that $\epsilon_i(B_i) \leq \epsilon$. The main idea is to perform a binary search for the target value of $\epsilon \in \mathbf{E}$ (lines 12–24). For each candidate value of ϵ, we compute $B_i(\epsilon)$ for all i. If $\sum_i B_i(\epsilon) < B$ (i.e., we are below the budget when targeting at the current choice of ϵ), we will eliminate all $\epsilon' \in \mathbf{E}$ such that $\epsilon' > \epsilon$; otherwise, we will eliminate all $\epsilon' \in \mathbf{E}$ such that $\epsilon' < \epsilon$. After finding the target value of ϵ such that $\sum_i B_i(\epsilon)$ achieves the largest value within B, the server sends the corresponding local budget $B_i(\epsilon)$ to each node.

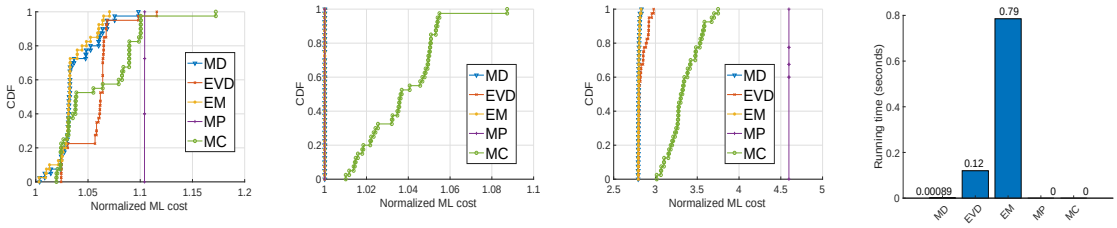
Finally, each node uses MD-MECB or EVD-MECB to compute its local configuration (k_i^*, b_i^*) under the given budget.

Complexity: We analyze the complexity step by step. First, computing $\mathcal{E}_i$ at each node (lines 1–8) has a complexity of $O((n^2+b_0)db_0n)$ if using MD-MECB and $O((n^3+d+b_0)db_0n)$ if using EVD-MECB, dominated by line 5. Second, computing the budget allocation at the server (lines 9–24) has a complexity of $O(db_0n\log(db_0n))$. Specifically, as $\mathbf{E}$ has $O(db_0n)$ elements, sorting it takes $O(db_0n\log(db_0n))$. The **while** loop is repeated $O(\log(db_0n))$ times, as each loop eliminates half of the candidate ϵ values in $\mathbf{E}$, and each loop takes $O(db_0n)$, dominated by lines 15–16. Finally, computing the local configuration at each node (line 25) takes $O(n^2 + b_0)$ using MD-MECB and $O(n^3 + d + b_0)$ using EVD-MECB.

Optimality: Next, we prove the optimality of OBA-MECBD in budget allocation. Let $\epsilon_i^\pi(B_i)$ be the error bound for a given solution π of the MECB problem for dataset $\mathcal{Y}_i$ and budget B_i. We show that OBA-MECBD is optimal in the following sense.

Theorem 3.6.1. *Using a given MECB algorithm π as the subroutine called in lines 2, 5, and 25, OBA-MECBD solves MECBD optimally w.r.t. π, i.e., its budget allocation $(B_i)_{i=1}^N$ is the optimal solution to (3.14) with $\epsilon_i(B_i)$ replaced by $\epsilon_i^\pi(B_i)$.*

Proof. Let $B_i^\pi(\epsilon)$ denote the smallest value of B_i such that $\epsilon_i^\pi(B_i) \leq \epsilon$. By lines 21–24 in Algorithm 6, I_{min} and I_{max} should always satisfy $\sum_{i=1}^N B_i^\pi(\epsilon_{I_{min}}) > B$ and $\sum_{i=1}^N B_i^\pi(\epsilon_{I_{max}}) \leq B$ for all nontrivial values of B. Let ϵ^* denote the value of ϵ_I at the end of budget allocation, which is the value of (3.14a) achieved by OBA-MECBD. As $I = I_{max}$ and



MEB k-means ($k = 2$) PCA (3 components) Running time

Figure 3.2. Evaluation on Fisher's Iris dataset (centralized setting).

$I_{min} = I_{max} + 1$ at this time, ϵ^* must be the smallest value of ϵ such that $\sum_{i=1}^{N} B_i^{\pi}(\epsilon) \leq B$. Therefore, for any other budget allocation $(B_i')_{i=1}^{N}$ such that $\sum_{i=1}^{N} B_i' \leq B$, we must have $\max_i \epsilon_i^{\pi}(B_i') \geq \epsilon^*$. □

3.7 Performance Evaluation

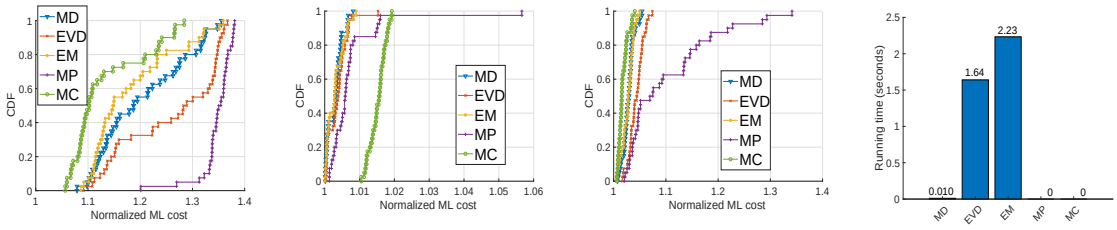
In this section, we evaluate our proposed algorithms using multiple real-world datasets for various ML tasks. Our objective is to validate the effectiveness and efficiency of our proposed algorithms (EVD-MECB, MD-MECB, OBA-MECBD) against benchmarks.

3.7.1 Datasets

In our experiments, we use four real-world datasets to evaluate our algorithms: (1) Fisher's Iris dataset [50], with 3 classes, 50 data points in each class, 5 attributes for each data point; (2) Facebook metric dataset [51], which has 494 data points with 19 attributes; (3) Pendigits dataset [52], which has $7,494$ data points and 17 attributes; (4) MNIST handwritten digits dataset in a 784-dimensional space [53], where we use $60,000$ data points for training and $10,000$ data points for testing. We leverage the approach in [62] to pre-process the labels, i.e., each label is mapped to a number such that distance between points with the same label is smaller than distance between points with different labels. All the original data are represented in the standard IEEE 754 double-precision binary floating-point format [77].

3.7.2 ML Tasks

We consider four ML tasks: (1) minimum enclosing ball (MEB) [60]; (2) k-means ($k = 2$ in our experiments); (3) principal component analysis (PCA); and (4) neural network (NN) (with three layers, 100 neurons in the hidden layer). Tasks (1–3) are unsupervised, and task (4) is supervised.



MEB k-means ($k = 2$) PCA (3 components) Running time

Figure 3.3. Evaluation on Facebook metric dataset (centralized setting).

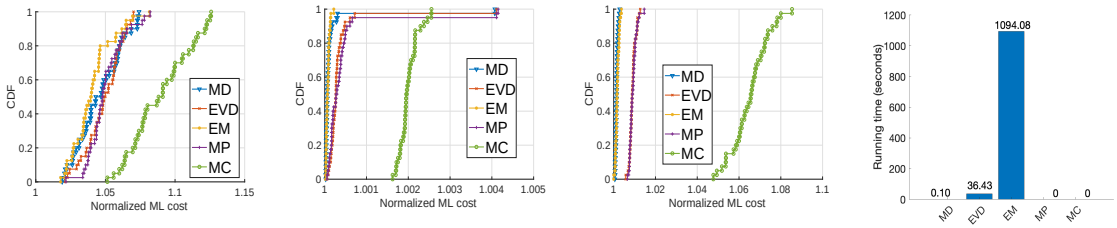
3.7.3 Algorithms

For the centralized setting, we consider five different algorithms for comparison. The first two are the proposed algorithms, i.e., EVD-MECB in Algorithm 3 (denoted by *EVD*), MD-MECB in Algorithm 5 (denoted by *MD*). The third algorithm is the straightforward solution *EM* (see Section 3.4.3.3). The fourth algorithm aims to Maximize the Precision (*MP*), i.e., using the configuration $k = \left\lfloor \frac{B}{d \cdot b_0} \right\rfloor$ and $b = b_0$ to construct coresets. The fifth algorithm aims to Maximize the Cardinality (*MC*), i.e., using $k = \min(n, \left\lfloor \frac{B}{d \cdot (1+m_e)} \right\rfloor)$ and $b = \max(1 + m_e, \left\lfloor \frac{B}{d \cdot n} \right\rfloor)$ to construct coresets, where $1 + m_e$ is the minimum number of bits required to represent a number by the rounding-based quantizer (Section 3.3.3).

For the distributed setting, we consider six algorithms for comparison. The first five algorithms correspond to instances of OBA-MECBD in Algorithm 6 that use EVD-MECB, MD-MECB, EM, MP, and MC as their subroutines, respectively. We denote these algorithms by OBA-EVD, OBA-MD, OBA-EM, OBA-MP and OBA-MC, respectively. The sixth algorithm is DRCC as proposed in [62] that optimizes the allocation of a given coreset cardinality to individual nodes.

3.7.4 Performance Metrics

We use the *normalized ML cost* to measure the performance of unsupervised ML tasks. The normalized ML cost is defined as $\text{cost}(\mathcal{Y}, \mathbf{x}_\mathcal{S})/\text{cost}(\mathcal{Y}, \mathbf{x}^*)$, where $\mathbf{x}_\mathcal{S}$ is the model learned from coreset $\mathcal{S}$ and $\mathbf{x}^*$ is the model learned from the original dataset $\mathcal{Y}$. For supervised ML tasks, we use *classification accuracy* to measure the performance. Furthermore, we report the running time of each algorithm. All metrics are computed over 40 Monte Carlo runs unless stated otherwise.



MEB k-means ($k = 2$) PCA (3 components) Running time

Figure 3.4. Evaluation on Facebook metric dataset (centralized setting).

3.7.5 Results in Centralized Setting

3.7.5.1 Unsupervised Learning

We first evaluate the unsupervised learning tasks: MEB, k-means, and PCA. We perform this evaluation on three datasets: Fisher's Iris, Facebook metric, and Pendigits. Figures 3.2–3.4 show the cumulative distribution function (CDF) of normalized ML costs as well as the average running time of each algorithm, when the budget is set to 2% of the size of the original dataset, i.e., $B = 960$, 12014 and 163069, respectively. We also list the b^* values over the Monte Carlo runs for EVD-MECB (*EVD*), MD-MECB (*MD*), and EM in Tables 3.2–3.4. We have the following observations: 1) In most cases, our proposed algorithms EVD-MECB and MD-MECB yield coresets that are much smaller (98% smaller) than the original dataset but support these ML tasks with less than 10% degradation in performance. 2) Compared to the proposed algorithms, EM achieves a slightly better ML performance, but has a much higher running time. 3) Compared to EVD-MECB, MD-MECB is not only faster, but also more closely approximates EM. 4) Compared with MP and MC that relies on a single operation, the algorithms jointly optimizing coreset constrution and quantization (EM, EVD-MECB, MD-MECB) achieve much better ML performance over all.

Table 3.2. Returned b^* for Fisher's Iris

Algorithm	b^*	# of occurrences
EVD	[31]	[40]
MD	[16, 18, 20, 23]	[1, 22, 16, 1]
EM	[18, 20]	[21, 19]

Table 3.3. Returned b^* for Facebook Metric

Algorithm	b^*	# of occurrences
EVD	[20]	[40]
MD	[18, 19, 20, 21, 22, 23]	[3, 5, 20, 6, 4, 2]
EM	[18, 19, 20, 21, 22, 23, 24, 27]	[1, 6, 9, 6, 10, 5, 2, 1]

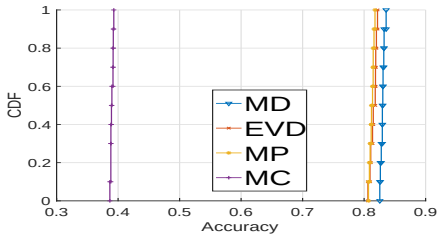


Figure 3.5. Overall CDFs

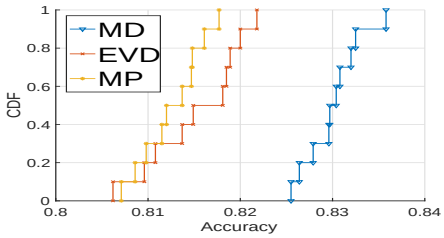


Figure 3.6. Zoomed-in CDFs

Figure 3.7. Evaluation on MNIST (Neural Net Accuracies)

3.7.5.2 Supervised Learning

For supervised learning, we evaluate neural network based classification on the MNIST dataset. We do not evaluate EM here because its running time for this dataset is prohibitively high.

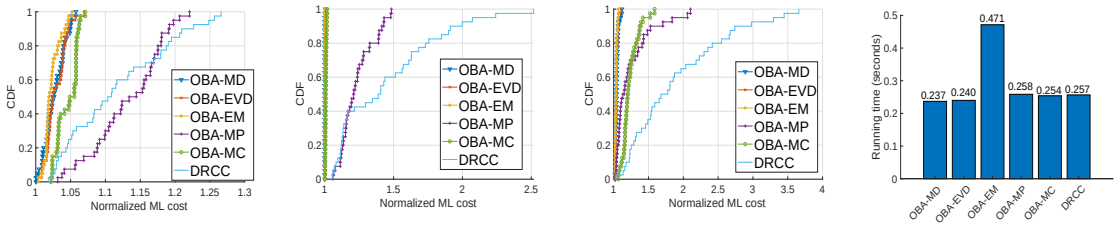
Same as unsupervised learning, we only use 2% of the original data, i.e., $B = 60,211,200$. Figure 3.7 shows the CDFs of classification accuracy over 10 Monte Carlo runs. Note that in contrast to costs, a higher accuracy means better performance. MD-MECB, EVD-MECB, and MP all achieve over 80% accuracy, while MC only achieves less than 40% accuracy, because it changes the value of each attribute too much. As we zoom in, we see that MD-MECB performs the best. Moreover, MD-MECB is also relatively fast, with a running time of approximately 15 minutes per run, whereas EVD-MECB takes up to 4.5 hours for each run due to computing eigenvalue decomposition for a large matrix. After evaluating different budgets, we note that although MP happens to perform well with 2% data, its performance is highly sensitive to the budget B, while the proposed algorithms (MD-MECB, EVD-MECB) adapt well to a wide range of budgets.

3.7.6 Results in Distributed Setting

In this experiment, we use Fisher's Iris dataset and Pendigits dataset to evaluate our proposed distributed algorithm (Algorithm 6). The original data points are randomly distributed across 10 nodes. The global communication budgets are set to $4,875$ bits for Fisher's Iris and $828,087$ bits for Pendigits, which correspond to 10% of the original

Table 3.4. Returned b^* for Pendigits

Algorithm	b^*	# of occurrences
EVD	[52]	[40]
MD	[19, 20, 21]	[4, 35, 1]
EM	[19, 20, 21, 22, 23, 24]	[5, 15, 6, 7, 6, 1]



MEB k-means ($k = 2$) PCA (3 components) Running time

Figure 3.8. Evaluation on Fisher's Iris dataset (distributed setting).

data size.

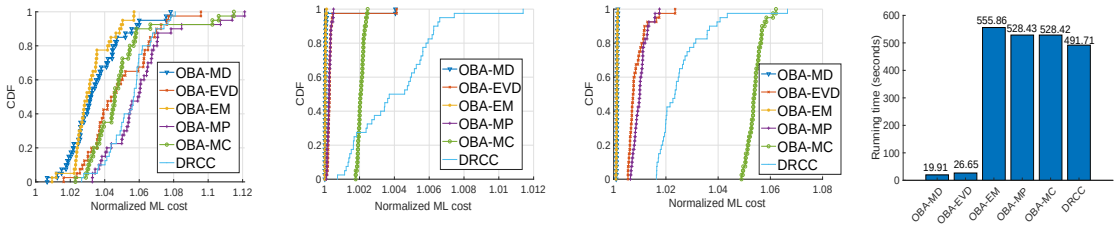
We present the results for distributed setting in Figures 3.8 and 3.9, from which we have the following observations: 1) With only 10% data in the distributed setting, most of the algorithms equipped with OBA outperform DRCC with a small degradation in the ML performance. 2) Compared with OBA-MP, OBA-MC, and DRCC that only rely on one operation to compress the data, our proposed OBA-EVD and OBA-MD which jointly optimize the operations of coreset construction and quantization perform significantly better. 3) OBA-MD is the most efficient over all these algorithms.

3.7.7 Summary of Experimental Results

- We demonstrate via real ML tasks and datasets that it is possible to achieve reasonable ML performance (less than 10% of degradation in most cases) and substantial data reduction (90–98% smaller than the original dataset) by combining coreset construction with quantization.
- The proposed algorithms approximate the performance of EM, with a significantly lower running time.
- Jointly optimizing coreset construction and quantization achieves much better ML performance than relying on only one of these operations.
- MD-MECB and its distributed variant (OBA-MD) achieve the best performance-efficiency tradeoff among all the evaluated algorithms, making them the most suitable for large datasets.

3.8 Conclusion

In this chapter, we have proposed the first framework, MECB, to jointly configure coreset construction algorithms and quantizers in order to minimize the ML error under



MEB k-means ($k = 2$) PCA (3 components) Running time

Figure 3.9. Evaluation on Fisher's Iris dataset (distributed setting).

a given communication budget. We have proposed two algorithms to efficiently compute approximate solutions to the MECB problem, whose effectiveness and efficiency have been demonstrated through experiments based on multiple real-world datasets. We have further proposed an algorithm to extend our solutions to the distributed setting by carefully allocating the communication budget across multiple nodes to minimize the overall ML error, which has shown significant improvements over alternatives when combined with our proposed solutions to MECB. Our solutions only depend on a smoothness parameter of the ML cost function, and can thus serve as a key enabler in reducing the communication cost for a broad range of ML tasks.

Chapter 4
Communication-efficient k-Means for Edge-based Machine Learning

4.1 Introduction

Given a dataset $P \subset \mathbb{R}^d$ with cardinality n, where both $n \gg 1$ and $d \gg 1$, consider the problem of finding k points $X = \{x_i\}_{i=1}^k$ to minimize the following cost function[1]:

$$\text{cost}(P, X) := \sum_{p \in P} \min_{x_i \in X} \|p - x_i\|^2. \tag{4.1}$$

This is the *k-means clustering* problem, and the points in X are called *centers*. Equivalently, the k-means clustering problem can be considered as the problem of finding the partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of P into k clusters that minimizes the following cost function:

$$\text{cost}(\mathcal{P}) := \sum_{i=1}^{k} \min_{x_i \in \mathbb{R}^d} \sum_{p \in P_i} \|p - x_i\|^2. \tag{4.2}$$

k-Means clustering is one of the most widely-used machine learning techniques. Algorithms for k-means are used in many areas of data science, e.g., for data compression, quantization, and transmission over noisy channels [88], and for hashing, sketching, and similarity search [89]; see the survey in [90] for more details. Recently, it was shown in [62, 91] that the centers of k-means can be used as a proxy (called a *coreset*) of the original dataset in computing a broader set of machine learning models with sufficiently continuous cost functions. Thus, efficient and accurate computation of k-means can bring

[1]The norms in (4.1) and (4.2) refer to the ℓ-2 norm.

broad benefits to machine learning applications.

Meanwhile, solving k-means is non-trivial. The problem is known to be NP-hard, even for two centers [92] or in the plane [93]. When either the number of centers k is a constant or the dimension d is a constant, a $(1+\epsilon)$-approximation for fixed $\epsilon > 0$ can be can be computed in polynomial time [29, 94–97]. In general, the problem is APX-hard and cannot be approximated to a factor better than 1.0013 in polynomial time [98, 99]. On the positive side, a number of efficient heuristic algorithms have been developed (see [48] and references therein), mostly based on the Lloyd's algorithm that iteratively computes a set of centers and improves these centers by clustering the points around these centers and updating the centers to the means of the clusters.

Due to its fundamental importance, how to speed up the k-means computation for large datasets has received significant attention in the machine learning community. Existing solutions can be classified into two approaches: *dimensionality reduction (DR)* techniques that aim at obtaining fast approximate solutions by running k-means on a "thinner" dataset with a reduced dimension (see [100] and references therein), and *cardinality reduction (CR)* techniques that aim at achieving the same by running k-means on a "smaller" dataset with a reduced number of points (see [101] and references therein). However, these solutions only considered the computation cost, while implicitly assuming that the data is stored at the same location where the k-means computation is performed.

To our knowledge, we are the first to explicitly address the communication cost in computing k-means over remote (and possibly distributed) data. The need of communications arises in the emerging application scenario of *edge-based machine learning* [4], where mobile/wireless devices (e.g., sensors, IoT devices, wearables) collect the raw data and transmit them (or their summaries) to nearby edge servers for processing. Recall in Chapter 1, compared to alternative approaches, e.g., transmitting the locally learned model parameters as in federated learning [4], transmitting data (summaries) has the advantage that: (i) we only need one round of communications[2], (ii) the transmitted data summary can potentially support various machine learning models [62, 91], and (iii) the edge server can usually solve the machine learning problem much more efficiently (within the same time) due to its greater computation power.

In this chapter, we consider the problem of solving k-means for a large high-dimensional dataset that is distributed across the edge, i.e., $n, d \gg 1$ (n: cardinality, d: dimension). For example, the problem arises when we want to generate large training datasets by

[2]In cases that the raw data are spread over multiple nodes, another round of communications is needed to decide the sizes of data summaries to collect from each node [6]. However, each node only sends one scalar in this round and hence the communication cost is negligible.

clustering unlabeled images and labeling a representative image in each cluster. An obvious solution of solving k-means at the data source and reporting the centers to the server will incur a high complexity at the data source, while the other obvious solution of sending the raw data to the server and solving k-means there will incur a high communication cost. We seek to achieve a guaranteed approximation under both a low complexity and a low communication cost by jointly applying DR and CR.

4.1.1 Related Work

The k-means problem has been studied since 1950s with both negative results (e.g., NP-hardness [92, 93]) and positive results (e.g., efficient heuristic algorithms [48]). This work belongs to a line of studies on data reduction for approximate k-means computation. Existing solutions can be classified into (i) *dimensionality reduction* and (ii) *cardinality reduction*.

Dimensionality reduction (DR) for k-means, initiated by [102], aims at speeding up k-means by reducing the number of features (i.e., the dimension) of the input data. To achieve this, two approaches have been proposed: 1) *feature selection* that selects a subset of the original features, and 2) *feature extraction* that constructs a smaller set of new features from the original features. For feature selection, the best known algorithms are from [103], including a random sampling algorithm that achieves a $(1+\epsilon)$-approximation using $O(k \log k/\epsilon^2)$ features, and a deterministic algorithm that achieves a $(1+\epsilon)$-approximation using $O(k/\epsilon^2)$ features. For feature extraction, there are two methods with guaranteed approximation, both based on linear projections. The first method is based on *singular value decomposition (SVD)*, where exact SVD gives 2-approximation using k features [104] or $(1+\epsilon)$-approximation using $\lceil k/\epsilon \rceil$ features [103], and approximate SVD gives $(2+\epsilon)$-approximation using k features [105] or $(1+\epsilon)$-approximation using $\lceil k/\epsilon \rceil$ features [103]. The second method is based on *random projections* that preserve vector ℓ-2 norms with an arbitrarily high probability, whose existence is guaranteed by the *Johnson-Lindenstrauss (JL) lemma* [19]. The best known algorithm there is given by [100], which achieves a $(1+\epsilon)$-approximation using $O(log(k/\epsilon)/\epsilon^2)$ features.

Cardinality reduction (CR) for k-means, initiated by [43], aims at using a small weighted set of points in the same space as the original data points, referred to as a *coreset*, to replace the original dataset as the input to k-means computation. A coreset is called an ϵ-coreset (for k-means) if it can approximate the k-means cost of the original dataset for every candidate set of centers up to a factor of $1 \pm \epsilon$. Many coreset construction algorithms have been proposed for k-means. Early algorithms use geometric partitions

to merge each group of nearby points into a single coreset point [42, 43, 106], which cause the cardinality of the coreset to be exponential in the dimension d. Later, [107] showed that sampling can be used to reduce the coreset cardinality to a polynomial in $k, \epsilon, \log n$, and d. Most state-of-the-art coreset construction algorithms are based on the *sensitivity sampling* framework that was first proposed in [37] and then formalized in [29]. To generate an ϵ-coreset, the solution in [29] needs a coreset cardinality of[3] $\tilde{O}(kd\epsilon^{-4})$, and its followup in [40] needs $\tilde{O}(k^2 d\epsilon^{-2})$. The best known solution[4] is the one in [101], which showed that by reducing the intrinsic dimension of the dataset and adding a constant term to the coreset-based cost, the cardinality of an ϵ-coreset can be reduced to $\tilde{O}(k^3\epsilon^{-4})$.

Another line of related work is on computing k-means over a distributed dataset. To this end, [6] proposed a distributed version of sensitivity sampling to construct a distributed ϵ-coreset, and [108] further combined this algorithm with a distributed PCA algorithm from [101] to reduce the communication cost. Under a limiting condition called $(1+\alpha, \epsilon)$-approximation stability, [109] proposed an algorithm with communication cost $\tilde{O}(mkd)$ (m: number of data sources) to compute a partition with bounded Hamming distance to the optimal partition. Besides the above theoretical studies, there are also efforts on adapting centralized k-means algorithms for distributed settings, e.g., MapReduce [110–112], sensor networks [113, 114], and Peer-to-Peer networks [115], or developing new k-means algorithms suitable for distributed execution, e.g., [116]. These algorithms are only heuristics, and differ fundamentally from our proposed algorithms that achieve a guaranteed approximation error *with respect to (wrt)* the optimal k-means cost.

Only [101, 108] considered joint DR and CR for k-means. However, they blindly assumed that DR should be applied before CR, leaving open several important problems: 1) Are there other algorithms that can achieve the same approximation error at a lower complexity or communication cost? 2) Does the order of applying DR and CR matter? 3) Will repeated application of DR and CR help? We will address all these questions in this chapter.

[3] We use $\tilde{O}(x)$ to denote a value that is at most linear in x times a factor that is polylogarithmic in x.

[4] There is another algorithm with a coreset cardinality independent of n and d (precisely, $k^{O(\epsilon^{-2})}$) in [8]. However, the algorithm has a high complexity and the coreset cardinality is larger than that in [101].

4.2 Background and Formulation

For completeness, we briefly review the state-of-the-art results on DR and CR in support of k-means computation.

4.2.1 Notations

We will use $\|x\|$ to denote the ℓ-2 norm if x is a vector, or the Frobenius norm if x is a matrix. We will use $A_P \in \mathbb{R}^{n\times d}$ to denote the matrix representation of a dataset $P \subset \mathbb{R}^d$, where each row corresponds to a point. Let $\mu(P)$ denote the optimal 1-means center of P. It is well-known that $\mu(P)$ is the sample mean, i.e., $\mu(P) = \frac{1}{|P|}\sum_{p\in P} p$. Let $\mathcal{P}_{P,X}$ denote the partition of dataset P induced by centers X, i.e., $\mathcal{P}_{P,X} = \{P_1, \ldots, P_{|X|}\}$ for $P_i := \{p \in P : \|p - x_i\| \leq \|p - x_j\|, \forall x_j \in X \setminus \{x_i\}\}$ (ties broken arbitrarily). Given scalars x, y, and ϵ ($\epsilon > 0$), $x \approx_{1+\epsilon} y$ denotes $\frac{1}{1+\epsilon}x \leq y \leq (1+\epsilon)x$.

Given a dimensionality reduction map $\pi : \mathbb{R}^d \to \mathbb{R}^{d'}$ ($d' < d$), we use $\pi(P) := \{\pi(p) : p \in P\}$ to denote the output dataset for an input dataset P, and $\pi(\mathcal{P}) := \{\pi(P_1), \ldots, \pi(P_k)\}$ to denote the partition of $\pi(P)$ corresponding to a partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of P. Moreover, given a partition $\mathcal{P}' = \pi(\mathcal{P})$, we use $\pi^{-1}(\mathcal{P}')$ to denote the corresponding partition of P, which puts $p, q \in P$ into the same cluster if and only if $\pi(p), \pi(q) \in P'$ belong to the same cluster under $\mathcal{P}'$. Finally, given $P' = \pi(P)$, we use $\pi^{-1}(P') := \{\pi^{-1}(p') : p' \in P'\}$ to denote a set of points in $\mathbb{R}^d$ that is mapped to P' by π. Note that there is no guarantee that $\pi^{-1}(P') = P$. However, solutions to $\pi(\tilde{P}) = P'$ must exist (P is a feasible solution) and $\pi^{-1}(P')$ denotes an arbitrary solution. If π is a linear map, i.e., $\pi(P) := A_P\Pi$ for a matrix $\Pi \in \mathbb{R}^{d\times d'}$, then the *Moore-Penrose inverse* Π^+ [117] of Π gives a feasible solution $\pi^{-1}(P') := A_{P'}\Pi^+$.

4.2.2 Dimensionality Reduction for k-Means

Definition 4.2.1. *We say that a DR map $\pi : \mathbb{R}^d \to \mathbb{R}^{d'}$ ($d' < d$) is an* ϵ-projection *if it preserves the cost of any partition up to a factor of $1+\epsilon$, i.e., cost$(\mathcal{P}) \approx_{1+\epsilon}$ cost$(\pi(\mathcal{P}))$ for every partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ of a finite set $P \subset \mathbb{R}^d$.*

For a sufficiently large d', it is known how to construct an ϵ-projection for an arbitrarily small ϵ with an arbitrarily high probability. One commonly used method is by computing the SVD of the data matrix: $A_P = U\Sigma V^T$, and then projecting the dataset onto a d'-dimensional subspace spanned by the first d' columns of V (i.e., the first d' right singular vectors of A_P). It was known [101] that if $d' = k + \lceil 72k\epsilon^{-2}\rceil - 1$, the optimal

k-means solution for the projected dataset gives an $(1+\epsilon)$-approximation for the original dataset.

Alternatively, one can construct π by random projection, where the cornerstone result is the JL Lemma:

Lemma 4.2.1 ([19]). *There exists a family of random linear maps $\pi : \mathbb{R}^d \to \mathbb{R}^{d'}$ with the following properties: for every $\epsilon, \delta \in (0, 1/2)$, there exists $d' = O(\frac{\log(1/\delta)}{\epsilon^2})$ such that for every $d \geq 1$ and all $x \in \mathbb{R}^d$, we have*

$$\Pr\{\|\pi(x)\| \approx_{1+\epsilon} \|x\|\} \geq 1 - \delta. \tag{4.3}$$

Based on this lemma, the following has been shown.

Theorem 4.2.1 ([100]). *Consider any family of random linear maps $\pi : \mathbb{R}^d \to \mathbb{R}^{d'}$ that (i) satisfies Lemma 4.2.1, and (ii) is sub-Gaussian-tailed (i.e., the probability for the norm after mapping to be larger than the norm before mapping by a factor of at least $1 + t$ is bounded by $e^{-\Omega(d't^2)}$). Then for every $\epsilon, \delta \in (0, 1/4)$, there exists*

$$d' = O(\frac{1}{\epsilon^2} \log \frac{k}{\epsilon\delta}), \tag{4.4}$$

such that π is an ϵ-projection with probability at least $1 - \delta$.

There are known methods to efficiently construct a random linear map that satisfies the conditions (i–ii) in Theorem 4.2.1: projections onto a random subspace [19, 118] and maps defined by matrices with i.i.d. Gaussian and sub-Gaussian entries [119–121].

Remark: Compared with SVD-based DR, DR based on random projection has the advantage that the projection matrix is *data-oblivious*, and can hence be pre-computed and distributed. As is shown later, this can lead to significant savings in communication cost.

4.2.3 Cardinality Reduction for k-Means

CR techniques are formally referred to as *coreset construction* algorithms, which aim at constructing the smallest coreset with a given approximation error defined as follows.

Definition 4.2.2 ([101]). *We say that a tuple (S, Δ, w), where $S \subset \mathbb{R}^d$, $w : S \to \mathbb{R}$, and $\Delta \in \mathbb{R}$, is an ϵ-coreset of $P \subset \mathbb{R}^d$ if it preserves the cost for every set of k centers*

up to a factor of $1 \pm \epsilon$*, i.e.,*

$$(1-\epsilon)cost(P,X) \leq cost(\mathbf{S},X) \leq (1+\epsilon)cost(P,X) \tag{4.5}$$

for any $X \subset \mathbb{R}^d$ *with* $|X| = k$*, where* $cost(\mathbf{S},X) := \sum_{q\in S} w(q) \cdot \min_{x_i \in X} \|q - x_i\|^2 + \Delta$ *for* $\mathbf{S} := (S, \Delta, w)$*.*

The above definition generalizes most of the existing definitions of ϵ-coreset, which typically ignore Δ.

For a sufficiently large $|S|$, it is known how to construct an ϵ-coreset for an arbitrarily small ϵ with an arbitrarily high probability. Using the sensitivity sampling framework proposed in [29], [40] proposed an algorithm that achieves the following performance.

Theorem 4.2.2 ([40])**.** *For any* $\epsilon, \delta \in (0,1)$*, with probability at least* $1-\delta$*, an* ϵ*-coreset* $(S, 0, w)$ *of size*

$$|S| = O\left(\frac{k^2 \log k}{\epsilon^2}(d \log k + \log(\frac{1}{\delta}))\right) \tag{4.6}$$

can be computed in time $O(ndk \log(1/\delta))$*.*

Using the idea of first performing DR by a SVD-based projection, and then applying sensitivity sampling to the dimension-reduced dataset, [101] reduced the coreset size to be constant in n and d.

Theorem 4.2.3 ([101])**.** *For any* $\epsilon, \delta \in (0,1)$*, with probability at least* $1-\delta$*, an* ϵ*-coreset* (S, Δ, w) *of size*

$$|S| = O\left(\frac{k^3 \log^2 k}{\epsilon^4} \log(\frac{1}{\delta})\right) \tag{4.7}$$

can be computed in time $O(\min(nd^2, n^2 d) + nk\epsilon^{-2}(d + k \log(1/\delta)))$*.*

Although giving the best known CR method for k-means, [101] only focused on the cardinality of coreset, ignoring the communication cost. It's possible this method is suboptimal in terms of communication cost.

4.2.4 Problem Statement

The motivation of existing DR/CR methods is to speed up k-means computation in the centralized setting where the node holding the data is also the node computing k-means.

In contrast, we want to develop efficient k-means algorithms based on joint DR and CR in scenarios where the data generation and the k-means computation occur at different nodes, as in the case of edge-based learning. We will refer to the node(s) holding the original dataset as the *data source(s)*, and the node running k-means computation as the *server*.

We will evaluate the performance of the proposed algorithms by the following metrics:

- *Approximation error:* We say that a set of k-means centers X is an α-approximation for k-means clustering of P if $\text{cost}(P, X) \leq \alpha \cdot \text{cost}(P, X^*)$, where X^* is the optimal set of k-means centers for P.
- *Communication cost:* We say that an algorithm incurs a communication cost of y if a data source employing the algorithm needs to send y scalars to the server.
- *Complexity:* We say that an algorithm incurs a complexity of z at the data source if a data source employing the algorithm needs to perform z elementary operations.

4.3 Joint DR and CR for k-Means

In this section, we will focus on the scenario of a single data source, and we will show that: using suitably selected DR/CR methods and a sufficiently powerful server, it is possible to solve k-means arbitrarily close to the optimal at the server, while incurring at most logarithmic communication cost and near-linear complexity at the data source.

4.3.1 DR+CR

We now discuss the approach of applying DR and then CR.

4.3.1.1 A Black-box Approach

We first treat the given DR/CR methods as black boxes, which allows us to derive bounds that are valid for arbitrary DR/CR methods. Since the DR quality (Definition 4.2.1) and the CR quality (Definition 4.2.2) are specified *with respect to (wrt)* different definitions of the k-means cost function in (4.1, 4.2), we first analyze the relationship between these definitions.

Lemma 4.3.1. *The two cost functions defined in (4.1, 4.2) are related by:*

1. *for* $\mathcal{P} = \{P_i\}_{i=1}^k$, $cost(\mathcal{P}) \geq cost(P, X)$ *if* $X = \{\mu(P_i)\}_{i=1}^k$;

2. $cost(P, X) \geq cost(\mathcal{P}_{P,X})$.

Proof. For 1), we note that since $x_i := \mu(P_i)$ is the optimal center of cluster P_i, we have

$$\begin{aligned} \text{cost}(\mathcal{P}) &= \sum_{i=1}^{k} \sum_{p \in P_i} \|p - x_i\|^2 \\ &= \sum_{p \in P} \|p - x_{i_p}\|^2 \qquad (4.8) \\ &\geq \sum_{p \in P} \min_{x_i \in X} \|p - x_i\|^2 = \text{cost}(P, X). \end{aligned}$$

where i_p in (4.8) is the index of the cluster P_i such that $p \in P_i$.

For 2), we note that by the definition of $\mathcal{P}_{P,X} = (P_1, \ldots, P_k)$ ($k := |X|$),

$$\begin{aligned} \text{cost}(P, X) &= \sum_{i=1}^{k} \sum_{p \in P_i} \|p - x_i\|^2 \\ &\geq \sum_{i=1}^{k} \min_{q \in \mathbb{R}^d} \sum_{p \in P_i} \|p - q\|^2 = \text{cost}(\mathcal{P}_{P,X}), \qquad (4.9) \end{aligned}$$

where (4.9) is by the definition in (4.2). □

Suppose that we first apply a DR method π_1 and then apply a CR method π_2. The quality of the k-means solution computed from the reduced dataset is bounded as follows.

Theorem 4.3.1. *Let X' be the optimal k-means centers[5] of $\mathbf{S}' := \pi_2(\pi_1(P))$, where π_1 is an ϵ_1-projection and π_2 generates an ϵ_2-coreset ($\epsilon_1, \epsilon_2 \in (0, 1)$). Then $X := \{\mu(P_i)\}_{i=1}^{k}$, where $\{P_1, \ldots, P_k\} = \pi_1^{-1}(\mathcal{P}_{\pi_1(P),X'})$, is an $(1+\epsilon_1)^2(1+\epsilon_2)/(1-\epsilon_2)$-approximation for k-means clustering of P, i.e.,*

$$cost(P, X) \leq \frac{(1+\epsilon_1)^2(1+\epsilon_2)}{1-\epsilon_2} cost(P, X^*), \qquad (4.10)$$

where X^* is the optimal set of k-means centers for P.

Proof. Let $\tilde{X}^*$ denote the set of means for each cluster under the partition $\pi_1(\mathcal{P}_{P,X^*})$ ($\tilde{X}^* = \pi_1(X^*)$ if π_1 is a linear map). Let P' denote $\pi_1(P)$ and $\mathbf{S}' := (S', \Delta, w)$ denote $\pi_2(P')$ (see Definition 4.2.2). Then

$$(1+\epsilon_1)\text{cost}(P, X^*) = (1+\epsilon_1)\text{cost}(\mathcal{P}_{P,X^*}) \qquad (4.11)$$

[5]Specifically, for $\mathbf{S}' = (S', \Delta, w)$, one can ignore Δ, and apply a weighted k-means algorithm to minimize $\sum_{q \in S'} w(q) \cdot \min_{x_i \in X} \|q - x_i\|^2$, or convert it into an unweighted dataset by duplicating each $q \in S'$ for $w(q)$ times and apply an unweighted k-means algorithm.

Algorithm 7: k-Means under Generic DR+CR

input : Original dataset P, number of centers k, DR method π_1, CR method π_2
output : Centers for k-means clustering of P

1 $P' \leftarrow \pi_1(P)$;
2 $(S', \Delta, w) \leftarrow \pi_2(P')$;
3 $X' \leftarrow \text{kmeans}(S', w, k)$;
4 $\mathcal{P}' \leftarrow \mathcal{P}_{P',X'}$;
5 $\mathcal{P} \leftarrow \pi_1^{-1}(\mathcal{P}')$ $(\mathcal{P} = \{P_1, \ldots, P_k\})$;
6 **foreach** $i = 1, \ldots, k$ **do**
7 $\quad x_i \leftarrow \mu(P_i)$;
8 return $\{x_i\}_{i=1}^k$;

$$\geq \text{cost}(\pi_1(\mathcal{P}_{P,X^*})) \tag{4.12}$$
$$\geq \text{cost}(P', \tilde{X}^*) \tag{4.13}$$
$$\geq \frac{1}{1+\epsilon_2}\text{cost}(\mathbf{S}', \tilde{X}^*) \tag{4.14}$$
$$\geq \frac{1}{1+\epsilon_2}\text{cost}(\mathbf{S}', X') \tag{4.15}$$
$$\geq \frac{1-\epsilon_2}{1+\epsilon_2}\text{cost}(P', X') \tag{4.16}$$
$$\geq \frac{1-\epsilon_2}{1+\epsilon_2}\text{cost}(\mathcal{P}_{P',X'}) \tag{4.17}$$
$$\geq \frac{1-\epsilon_2}{(1+\epsilon_1)(1+\epsilon_2)}\text{cost}(\pi_1^{-1}(\mathcal{P}_{P',X'})) \tag{4.18}$$
$$\geq \frac{1-\epsilon_2}{(1+\epsilon_1)(1+\epsilon_2)}\text{cost}(P, X), \tag{4.19}$$

where (4.11) is due to the optimality of X^*, (4.12) is because π_1 is an ϵ_1-projection, (4.13) is by Lemma 4.3.1.1), (4.14) is because $\mathbf{S}'$ is an ϵ_2-coreset of P', (4.15) is because X' is optimal for $\mathbf{S}'$, (4.16) is because $\mathbf{S}'$ is an ϵ_2-coreset of P', (4.17) is by Lemma 4.3.1.2), (4.18) is because π_1 is an ϵ_1-projection, and (4.19) is by Lemma 4.3.1.1). The last inequality (4.19) gives the desired upper bound on $\text{cost}(P, X)$. □

The proof of Theorem 4.3.1 implies an approximate k-means algorithm shown in Algorithm 7, for which Theorem 4.3.1 bounds the approximation error if line 3 computes the optimal k-means centers of the dimension and cardinality reduced dataset S' with weights w.

Solving k-means optimally is known to be NP-hard [46], but there exist efficient heuristics [48] and $(1+\epsilon)$-approximation algorithms [29, 94–97]. If line 3 in Algorithm 7 is only solved to an $(1+\epsilon_3)$-approximation, then the overall approximation error will

increase by a factor of $1 + \epsilon_3$.

Corollary 4.3.1.1. *If X' is an $(1 + \epsilon_3)$-approximation for the optimal k-means centers of $\mathbf{S}' := \pi_2(\pi_1(P))$, then under the conditions in Theorem 4.3.1, X given by Algorithm 7 is an $(1 + \epsilon_1)^2(1 + \epsilon_2)(1 + \epsilon_3)/(1 - \epsilon_2)$-approximation for k-means clustering of P.*

Proof. The proof is similar to that of Theorem 4.3.1, except that from (4.15) onwards, there is an extra factor of $1/(1 + \epsilon_3)$ because X' only guarantees that $\text{cost}(\mathbf{S}', \tilde{X}^*) \geq \text{cost}(\mathbf{S}', X')/(1 + \epsilon_3)$. □

Communication cost: If P resides on a data source and the k-means computation is performed by a server, then we have to transmit the entire dataset P in order to implement Algorithm 7, as its solution is an explicit function of a partition of P. Thus, Algorithm 7 is only useful for reducing the complexity in a centralized setting when everything is done at the same node, but is not useful for reducing the communication cost in edge-based learning. In contrast, for certain DR/CR methods, it is possible to achieve guaranteed approximation at a reduced communication cost as shown below.

4.3.1.2 An Existing DR+CR Algorithm

To our knowledge, the only existing work that considered joint DR and CR for k-means is [101]. The algorithm, referred to as *FSS* following the authors' last names, first applies a SVD-based DR by computing $A_P^{(d')} = A_P V^{(d')}(V^{(d')})^T$, where $d' = O(k/\epsilon^2)$, and $V^{(d')} \in \mathbb{R}^{d \times d'}$ contains the first d' right singular vectors of the data matrix A_P as its columns. After this step, the mapped dataset, represented by the rows of $A_P^{(d')}$, is still in $\mathbb{R}^d$, but all its points lie in a d'-dimensional subspace spanned by the columns of $V^{(d')}$. Then FSS applies a sensitivity-based coreset construction algorithm [40], whose performance is stated in Theorem 4.2.2, to construct an $(\epsilon/8)$-coreset $(S, 0, w)$ for $A_P^{(d')}$. It was shown in [101] that after adding a constant $\Delta := \|A_P - A_P^{(d')}\|^2$, the tuple (S, Δ, w) is an ϵ-coreset for A_P (and hence P), with a cardinality given in Theorem 4.2.3.

The approximation error and the communication cost of FSS (for transmitting its output) were not given in [101]. Thus, we provide them to facilitate comparison.

Corollary 4.3.1.2. *Suppose that the data source reports the coreset $\mathbf{S} := (S, \Delta, w)$ computed by FSS [101] and the server computes the optimal k-means centers X of $\mathbf{S}$. Then:*

1. *X is an $(1 + \epsilon)/(1 - \epsilon)$-approximation for k-means clustering of P with probability at least $1 - \delta$;*

2. the communication cost is $\tilde{O}\left(\frac{k}{\epsilon^2}(\frac{k^3}{\epsilon^4}+d)\right)$.

Proof. For 1), let X^* denote the optimal k-means centers of P. Since $\mathbf{S}$ is an ϵ-coreset with probability $\geq 1-\delta$ (Theorem 4.2.3), by Definition 4.2.2, the following holds with probability $\geq 1-\delta$:

$$\begin{aligned}\text{cost}(P,X) \leq \frac{1}{1-\epsilon}\text{cost}(\mathbf{S},X) &\leq \frac{1}{1-\epsilon}\text{cost}(\mathbf{S},X^*)\\ &\leq \frac{1+\epsilon}{1-\epsilon}\text{cost}(P,X^*). \end{aligned} \tag{4.20}$$

For 2), the cost of transferring (S,Δ,w) is dominated by the cost of transferring S. Since S lies in a d'-dimensional subspace spanned by the columns of $V^{(d')}$, it suffices to transmit the coordinates of each point in S in this subspace together with $V^{(d')}$. The former incurs a cost of $O(|S|\cdot d')$, and the latter incurs a cost of $O(dd')$. Plugging $d'=O(k/\epsilon^2)$ and $|S|=\tilde{O}(k^3/\epsilon^4)$ from Theorem 4.2.3 yields the overall communication cost. □

4.3.1.3 Communication-efficient DR+CR

As stated in Corollary 4.3.1.2, FSS incurs a communication cost that is linear in the dimension d, which is still rather high for high-dimensional data. The question is whether there exists a data reduction method with comparable complexity, that can achieve the same approximation error at a lower communication cost. Below we will show one such method.

Our key observation is that the linear communication cost in d is due to the cost of transmitting a basis of the projected subspace (i.e., $V^{(d')}$). This is required by SVD-based DR methods as the subspace is data-dependent. In contrast, random projections are *data-oblivious*, and can thus be pre-generated and shared among data sources and servers, or generated independently by data sources and servers using a shared random number generation seed, both incurring negligible (runtime) communication cost. Thus, we can circumvent the linear communication cost by employing a random projection that satisfies the JL Lemma (Lemma 4.2.1) as the DR method. We will refer to such a projection as a *JL projection*. The following is directly implied by the JL Lemma.

Lemma 4.3.2. *Let* $\pi:\mathbb{R}^d\to\mathbb{R}^{d'}$ *be a JL projection. There exists* $d'=O(\epsilon^{-2}\log(nk/\delta))$ *such that for any* $P\subset\mathbb{R}^d$ *with* $|P|=n$ *and* $X,X^*\subset\mathbb{R}^d$ *with* $|X|=|X^*|=k$, *the*

following holds with probability at least $1-\delta$*:*

$$cost(P,X) \approx_{(1+\epsilon)^2} cost(\pi(P),\pi(X)), \tag{4.21}$$

$$cost(P,X^*) \approx_{(1+\epsilon)^2} cost(\pi(P),\pi(X^*)). \tag{4.22}$$

Proof. Let $\delta' = \delta/(2nk)$. By the JL Lemma (Lemma 4.2.1), we could know that there exists $d' = O(\epsilon^{-2}\log(1/\delta')) = O(\epsilon^{-2}\log(nk/\delta))$, such that every $x \in \mathbb{R}^d$ satisfies $\|\pi(x)\| \approx_{1+\epsilon} \|x\|$ with probability $\geq 1-\delta'$. By the union bound, this implies that with probability $\geq 1-\delta$, every $p - x_i$ for $p \in P$ and $x_i \in X \cup X^*$ satisfies $\|\pi(p)-\pi(x_i)\| \approx_{1+\epsilon} \|p-x_i\|$. Therefore, with probability $\geq 1-\delta$,

$$\begin{aligned}\text{cost}(\pi(P),\pi(X)) &= \sum_{p\in P}\min_{x_i\in X}\|\pi(p)-\pi(x_i)\|^2 &(4.23)\\ &\leq \sum_{p\in P}\min_{x_i\in X}(1+\epsilon)^2\|p-x_i\|^2 \\ &= (1+\epsilon)^2\text{cost}(P,X), &(4.24)\end{aligned}$$

$$\begin{aligned}(4.23) &\geq \sum_{p\in P}\min_{x_i\in X}\frac{1}{(1+\epsilon)^2}\|p-x_i\|^2 \\ &= \frac{1}{(1+\epsilon)^2}\text{cost}(P,X). &(4.25)\end{aligned}$$

Combining (4.24) and (4.25) proves (4.21). Similar argument will prove (4.22).

□

Using a JL projection for DR and FSS for CR, we propose a new k-means algorithm with a lower communication cost, shown in Algorithm 8. Algorithm 8 differs from the generic algorithm Algorithm 7 in that it directly computes the centers in the original space from the optimal centers in the low-dimensional space (line 8), which helps to reduce the communication cost by only transmitting (S', Δ, w).

In the following, we will analyze Algorithm 8 in terms of approximation error, communication cost, and complexity.

Theorem 4.3.2. *For any* $\epsilon, \delta \in (0,1)$*, if* π_1 *satisfies Lemma 4.3.2 and* π_2 *generates an* ϵ*-coreset with probability at least* $1-\delta$*, then the output* X *of Algorithm 8 is an* $(1+\epsilon)^5/(1-\epsilon)$*-approximation for* k*-means clustering of* P *with probability at least* $(1-\delta)^2$*.*

Algorithm 8: k-Means under Communication-efficient DR+CR

input : Original dataset P, number of centers k, JL projection π_1, FSS-based CR method π_2

output : Centers for k-means clustering of P

1 **data source:**
2 $P' \leftarrow \pi_1(P)$;
3 $(S', \Delta, w) \leftarrow \pi_2(P')$;
4 report (S', Δ, w) to the server;
5 **server:**
6 $X' \leftarrow \text{kmeans}(S', w, k)$ $(X' = \{x'_1, \ldots, x'_k\})$;
7 **foreach** $i = 1, \ldots, k$ **do**
8 $x_i \leftarrow \pi_1^{-1}(x'_i)$;
9 return $\{x_i\}_{i=1}^k$;

Proof. Let X^* be the optimal k-means centers of P and $\mathbf{S}' := (S', \Delta, w)$ generated in line 3 of Algorithm 8. With probability at least $(1-\delta)^2$, π_1 satisfies (4.21, 4.22) and π_2 generates an ϵ-coreset. Thus, with probability at least $(1-\delta)^2$,

$$\begin{aligned}
\text{cost}(P, X) &\leq (1+\epsilon)^2 \text{cost}(\pi_1(P), X') && (4.26)\\
&\leq \frac{(1+\epsilon)^2}{1-\epsilon} \text{cost}(\mathbf{S}', X') && (4.27)\\
&\leq \frac{(1+\epsilon)^2}{1-\epsilon} \text{cost}(\mathbf{S}', \pi_1(X^*)) && (4.28)\\
&\leq \frac{(1+\epsilon)^3}{1-\epsilon} \text{cost}(\pi_1(P), \pi_1(X^*)) && (4.29)\\
&\leq \frac{(1+\epsilon)^5}{1-\epsilon} \text{cost}(P, X^*), && (4.30)
\end{aligned}$$

where (4.26) is by (4.21) and that $\pi_1(X) = X'$, (4.27) is because $\mathbf{S}'$ is an ϵ-coreset of $\pi_1(P)$, (4.28) is because X' minimizes $\text{cost}(\mathbf{S}', Q)$, (4.29) is again because $\mathbf{S}'$ is an ϵ-coreset of $\pi_1(P)$, and (4.30) is by (4.22). □

Corollary 4.3.2.1. *For Algorithm 8 to achieve the approximation error in Theorem 4.3.2,*

1. *the communication cost is* $\tilde{O}\left(k^4\epsilon^{-6} + k\epsilon^{-4}\log n\right)$*, and*
2. *the complexity at the data source is* $\tilde{O}\left(n\epsilon^{-2}(d + k\epsilon^{-2} + k^2)\right)$*,*

assuming $n \gg k, 1/\delta$.

Proof. The cost of transmitting (S', Δ, w) is dominated by the cost of transmitting S'. By Lemma 4.3.2, the dimension of P' needs to satisfy $d' = O(\epsilon^{-2}\log(nk/\delta)) =$

$O(\epsilon^{-2}\log n)$ (since $n \gg k, 1/\delta$). By Theorem 4.2.3, the cardinality of S' needs to satisfy $|S'| = O(k^3\epsilon^{-4}\log^2(k)\log(1/\delta))$. Moreover, points in S' lie in a $\tilde{d}$-dimensional subspace for $\tilde{d} = O(k/\epsilon^2)$. Thus, it suffices to transmit the coordinates of points in S' in the $\tilde{d}$-dimensional subspace and a basis of the subspace. The coordinates have size $O(|S'|\tilde{d})$, and the basis has size $O(d'\tilde{d})$. The total communication cost is

$$\begin{aligned} O((|S'| + d')\tilde{d}) &= O\left(\frac{k^4}{\epsilon^6}\log^2(k)\log(\frac{1}{\delta}) + \frac{k}{\epsilon^4}\log n\right) \\ &= \tilde{O}\left(\frac{k^4}{\epsilon^6} + \frac{k\log n}{\epsilon^4}\right). \end{aligned} \tag{4.31}$$

In terms of complexity, for a given projection matrix $\Pi \in \mathbb{R}^{d\times d'}$ such that $\pi_1(P) := A_P\Pi$, line 2 takes $O(ndd') = O(nd\epsilon^{-2}\log n)$ time, where we have plugged in $d' = O(\epsilon^{-2}\log n)$. By Theorem 4.2.3, line 3 takes time

$$\begin{aligned} &O\left(\min(nd'^2, n^2d') + \frac{nk}{\epsilon^2}\Big(d' + k\log(\frac{1}{\delta})\Big)\right) \\ &= O\left(\frac{n}{\epsilon^2}\Big(\frac{\log^2 n}{\epsilon^2} + \frac{k\log n}{\epsilon^2} + k^2\log(\frac{1}{\delta})\Big)\right). \end{aligned} \tag{4.32}$$

Thus, the total complexity at the data source is:

$$\begin{aligned} &O\left(\frac{n}{\epsilon^2}\Big(\frac{\log^2 n}{\epsilon^2} + \frac{k\log n}{\epsilon^2} + d\log n + k^2\log(\frac{1}{\delta})\Big)\right) \\ &= \tilde{O}\left(\frac{n}{\epsilon^2}\Big(d + \frac{k}{\epsilon^2} + k^2\Big)\right). \end{aligned} \tag{4.33}$$

□

Remark: We only focus on the complexity at the data source as the server is usually much more powerful. Together, Theorem 4.3.2 and Corollary 4.3.2.1 show that Algorithm 8 can solve k-means arbitrarily close to the optimal with an arbitrarily high probability, while incurring a complexity that is roughly linear in the data size (i.e., nd) and a communication cost that is roughly logarithmic in the data cardinality (i.e., n).

4.3.2 CR+DR

While Algorithm 8 can drastically reduce the communication cost without incurring much computation, it remains unclear whether its order of applying DR and CR is optimal.

To this end, we consider the approach of applying CR first.

4.3.2.1 A Black-box Approach

Now suppose that we first apply a generic CR method π_2 and then apply a generic DR method π_1. It turns out that the quality of the resulting k-means solution is the same as that in Theorem 4.3.1.

Theorem 4.3.3. *Let X' be the optimal k-means centers of $\mathbf{S}' := (\pi_1(S), \Delta, w)$ for $(S, \Delta, w) := \pi_2(P)$, where π_1 is an ϵ-projection and π_2 generates an ϵ-coreset for $\epsilon \in (0, 1)$. Then $X := \{\sum_{q \in S_i} \frac{w(q)q}{\sum_{p \in S_i} w(p)}\}_{i=1}^k$, where $\{S_1, \ldots, S_k\} = \pi_1^{-1}(\mathcal{P}_{\pi_1(S), X'})$, is a $(1+\epsilon)^3/(1-\epsilon)$-approximation for k-means clustering of P.*

Proof. Let X^* be the set of optimal k-means centers for P. Let $\mathbf{S} = (S, \Delta, w) := \pi_2(P)$. As the cost functions (4.1) and (4.2) are defined for unweighted datasets, we use a trick in [101] to convert the weighted dataset S into an unweighted dataset $\tilde{S}$ by duplicating each $q \in S$ for $w(q)$ times[6]. For any set of centers Q and $\text{cost}(\mathbf{S}, Q)$ defined in Definition 4.2.2, we have

$$\text{cost}(\mathbf{S}, Q) = \text{cost}(\tilde{S}, Q) + \Delta, \tag{4.34}$$

where $\text{cost}(\tilde{S}, Q)$ is defined as in (4.1). Similarly,

$$\text{cost}(\mathbf{S}', Q) = \text{cost}(\pi_1(\tilde{S}), Q) + \Delta. \tag{4.35}$$

We then have

$$\begin{aligned}
(1+\epsilon)\text{cost}(P, X^*) &\geq \text{cost}(\mathbf{S}, X^*) && (4.36)\\
&\geq \text{cost}(\mathcal{P}_{\tilde{S}, X^*}) + \Delta && (4.37)\\
&\geq \frac{1}{1+\epsilon}(\text{cost}(\pi_1(\mathcal{P}_{\tilde{S}, X^*})) + \Delta) && (4.38)\\
&\geq \frac{1}{1+\epsilon}(\text{cost}(\mathcal{P}_{\pi_1(\tilde{S}), X'}) + \Delta) && (4.39)\\
&\geq \frac{1}{(1+\epsilon)^2}(\text{cost}(\pi_1^{-1}(\mathcal{P}_{\pi_1(\tilde{S}), X'})) + \Delta) && (4.40)\\
&\geq \frac{1}{(1+\epsilon)^2}(\text{cost}(\tilde{S}, X) + \Delta) && (4.41)
\end{aligned}$$

[6]Strictly speaking, we need to replace each $q \in S$ by $\lceil w(q)/\delta \rceil$ identical points with weight δ each. The total weight of these points approximates $w(q)$ as $\delta \to 0$.

Algorithm 9: k-Means under Generic CR+DR

input : Original dataset P, number of centers k, DR method π_1, CR method π_2
output : Centers for k-means clustering of P

$(S, \Delta, w) \leftarrow \pi_2(P)$;
$S' \leftarrow \pi_1(S)$;
$X' \leftarrow \text{kmeans}(S', w, k)$;
$\mathcal{S}' \leftarrow \mathcal{P}_{S',X'}$;
$\mathcal{S} \leftarrow \pi_1^{-1}(\mathcal{S}')$ $(\mathcal{S} = \{S_1, \ldots, S_k\})$;
foreach $i = 1, \ldots, k$ **do**
    $x_i \leftarrow \frac{1}{\sum_{p \in S_i} w(p)} \sum_{q \in S_i} w(q) q$;
return $\{x_i\}_{i=1}^k$;

$$\geq \frac{1-\epsilon}{(1+\epsilon)^2} \text{cost}(P, X), \tag{4.42}$$

where (4.36) is because $\mathbf{S}$ is an ϵ-coreset of P, (4.37) is because of (4.34) and Lemma 4.3.1.2), (4.38) is because π_1 is an ϵ-projection, (4.39) is because X' minimizes $\text{cost}(\mathbf{S}', Q)$ and by (4.35) it minimizes $\text{cost}(\pi_1(\tilde{S}), Q)$ (and hence $\text{cost}(\mathcal{P}_{\pi_1(\tilde{S}),Q})$), (4.40) is because π_1 is an ϵ-projection, (4.41) is by noticing that the X defined in the theorem is the set of means for the clusters in $\pi_1^{-1}(\mathcal{P}_{\pi_1(\tilde{S}),X'})$ and then applying Lemma 4.3.1.1), and (4.42) is because of (4.34) and that $\mathbf{S}$ is an ϵ-coreset of P. The last inequality (4.42) gives the desired upper bound on $\text{cost}(P, X)$. □

Discussion: If P resides on a data source and k-means is performed by a server, then we only need to transmit the coreset (S, Δ, w) to implement Algorithm 9. Compared to the approach of applying DR first as in Algorithm 7, applying CR first allows us to substantially reduce the communication cost for datasets with large cardinalities. However, as shown below, utilizing suitably selected CR/DR methods can further reduce the communication cost.

4.3.2.2 Communication-efficient CR+DR

Recall that when applying DR first can reduce the communication cost to be logarithmic in the size of the original dataset (Theorem 4.3.2). Below we will show that when applying CR first, this cost can be reduced to a constant, i.e., independent of both the cardinality n and the dimension d of the original dataset.

We again choose JL projection as the DR method to avoid transmitting the projection matrix at runtime and FSS as the CR method, which generates an ϵ-coreset with the minimum cardinality among the existing CR methods for k-means. The algorithm, shown

Algorithm 10: k-Means under Communication-efficient CR+DR

input : Original dataset P, number of centers k, JL projection π_1, FSS-based CR method π_2

output : Centers for k-means clustering of P

data source:

$(S, \Delta, w) \leftarrow \pi_2(P)$;

$S' \leftarrow \pi_1(S)$;

report (S', Δ, w) to the server;

server:

$X' \leftarrow \text{kmeans}(S', w, k)$;

$X \leftarrow \pi_1^{-1}(X')$;

return X;

in Algorithm 10, differs from Algorithm 8 in that the order of applying DR and CR is reversed.

We now analyze the performance of Algorithm 10, starting with a counterpart of Lemma 4.3.2.

Lemma 4.3.3. *Let $\pi : \mathbb{R}^d \to \mathbb{R}^{d'}$ be a JL projection. There exists $d' = O(\epsilon^{-2} \log(n'k/\delta))$ such that for any coreset $\mathbf{S} := (S, \Delta, w)$, where $S \subset \mathbb{R}^d$ with $|S| = n'$, $w : S \to \mathbb{R}$, and $\Delta \in \mathbb{R}$, and any $X, X^* \subset \mathbb{R}^d$ with $|X| = |X^*| = k$, the following holds with probability at least $1 - \delta$:*

$$cost(\mathbf{S}, X) \approx_{(1+\epsilon)^2} cost((\pi(S), \Delta, w), \pi(X)), \tag{4.43}$$

$$cost(\mathbf{S}, X^*) \approx_{(1+\epsilon)^2} cost((\pi(S), \Delta, w), \pi(X^*)). \tag{4.44}$$

Proof. The proof is analogous to that of Lemma 4.3.2. Let $\delta' = \delta/(2n'k)$. Then there exists $d' = O(\epsilon^{-2} \log(1/\delta')) = O(\epsilon^{-2} \log(n'k/\delta))$, such that every $x \in \mathbb{R}^d$ satisfies $\|\pi(x)\| \approx_{1+\epsilon} \|x\|$ with probability $\geq 1 - \delta'$. By the union bound, this implies that with probability $\geq 1 - \delta$, every $p \in S$ and $x_i \in X \cup X^*$ satisfy $\|\pi(p) - \pi(x_i)\| \approx_{1+\epsilon} \|p - x_i\|$. Therefore,

$$\begin{aligned}
&\text{cost}((\pi(S), \Delta, w), \pi(X)) \\
&= \sum_{p \in S} w(p) \cdot \min_{x_i \in X} \|\pi(p) - \pi(x_i)\|^2 + \Delta \qquad (4.45) \\
&\leq (1+\epsilon)^2 \left(\sum_{p \in S} w(p) \cdot \min_{x_i \in X} \|p - x_i\|^2 + \Delta \right) \\
&= (1+\epsilon)^2 \text{cost}(\mathbf{S}, X), \qquad (4.46)
\end{aligned}$$

$$(4.45) \geq \frac{1}{(1+\epsilon)^2}\left(\sum_{p\in S} w(p)\cdot \min_{x_i\in X}\|p-x_i\|^2 + \Delta\right)$$
$$= \frac{1}{(1+\epsilon)^2}\text{cost}(\mathbf{S}, X), \tag{4.47}$$

which prove (4.43). Similar argument proves (4.44). □

Below, we will show that Algorithm 10 achieves the same approximation as Algorithm 8, but at a different communication cost and a different complexity.

Theorem 4.3.4. *For any $\epsilon, \delta \in (0,1)$, if in Algorithm 10, π_1 satisfies Lemma 4.3.3, π_2 generates an ϵ-coreset with probability at least $1-\delta$, and kmeans(S', w, k) returns the optimal k-means centers of the dataset S' with weights w, then*

1. *the output X is an $(1+\epsilon)^5/(1-\epsilon)$-approximation for k-means clustering of P with probability $\geq (1-\delta)^2$,*
2. *the communication cost is $\tilde{O}(k^3/\epsilon^6)$, and*
3. *the complexity at the data source is $O\left(nd\cdot\min(n,d)\right)$,*

assuming $\min(n,d) \gg k, 1/\epsilon,$ *and* $1/\delta$.

Proof. For 1), let X^* be the optimal k-means centers of P and $\mathbf{S} := (S, \Delta, w)$ generated in line 2 of Algorithm 10. With probability at least $(1-\delta)^2$, $\mathbf{S}$ is an ϵ-coreset of P, and π_1 satisfies (4.43, 4.44). Thus, with this probability,

$$\text{cost}(P, X) \leq \frac{1}{1-\epsilon}\text{cost}(\mathbf{S}, X) \tag{4.48}$$
$$\leq \frac{(1+\epsilon)^2}{1-\epsilon}\text{cost}((S', \Delta, w), X') \tag{4.49}$$
$$\leq \frac{(1+\epsilon)^2}{1-\epsilon}\text{cost}((S', \Delta, w), \pi_1(X^*)) \tag{4.50}$$
$$\leq \frac{(1+\epsilon)^4}{1-\epsilon}\text{cost}(\mathbf{S}, X^*) \tag{4.51}$$
$$\leq \frac{(1+\epsilon)^5}{1-\epsilon}\text{cost}(P, X^*), \tag{4.52}$$

where (4.48) is because $\mathbf{S}$ is an ϵ-coreset of P, (4.49) is due to (4.43) (note that $\pi_1(S) = S'$ and $\pi_1(X) = X'$), (4.50) is because X' minimizes $\text{cost}((S', \Delta, w), \cdot)$, (4.51) is due to (4.44), and (4.52) is again because $\mathbf{S}$ is an ϵ-coreset of P.

For 2), note that according to Theorem 4.2.3, we can have that the cardinality of S needs to be $n' = O(k^3\epsilon^{-4}\log^2(k)\log(1/\delta))$. By Lemma 4.3.3, the dimension of S' needs to be $d' = O(\epsilon^{-2}\log(n'k/\delta))$. Thus, the cost of transmitting (S', Δ, w), dominated by the cost of transmitting S', is

$$\begin{aligned} O(n'd') &= O\left(\frac{k^3\log^2 k}{\epsilon^6}\log(\frac{1}{\delta})\Big(\log k + \log(\frac{1}{\epsilon}) + \log(\frac{1}{\delta})\Big)\right) \\ &= \tilde{O}\left(\frac{k^3}{\epsilon^6}\right). \end{aligned} \tag{4.53}$$

For 3), line 2 of Algorithm 10 takes time $O(\min(nd^2, n^2d) + nk\epsilon^{-2}(d + k\log(1/\delta)))$ according to Theorem 4.2.3. Given a projection matrix $\Pi \in \mathbb{R}^{d\times d'}$ such that $\pi_1(S) := A_S\Pi$, line 3 takes time $O(n'dd')$. Thus, the total complexity at the data source is

$$\begin{aligned} &O\Big(\min(nd^2, n^2d) + \frac{k}{\epsilon^2}nd + \frac{k^2\log k}{\epsilon^2}n + \frac{k^3\log^3 k(\log k + \log(\frac{1}{\epsilon}))}{\epsilon^6}d\Big) \\ &= O\left(nd\cdot\min(n,d)\right). \end{aligned} \tag{4.54}$$

□

4.3.3 Comparison

We now summarize the above results via a comparison between the approach of applying DR first (i.e., DR+CR) and the approach of applying CR first (i.e., CR+DR).

Viewing the given DR/CR methods as black boxes, we have shown in Theorems 4.3.1 and 4.3.3 that the two approaches lead to the same approximation error. However, the approach of CR+DR has the advantage that the data source only needs to report coreset, resulting in a lower communication cost.

Utilizing the unique properties of carefully selected DR/CR methods, i.e., JL projection and FSS, we have shown in Theorems 4.3.2 and 4.3.4 that we can achieve an arbitrarily small approximation error with an arbitrarily high probability, while drastically reducing the communication cost and the complexity at the data source. Specifically, to achieve the same approximation error with the same probability, DR+CR (Algorithm 8) incurs a communication cost of $O(k\epsilon^{-4}\log n)$ and a complexity of $\tilde{O}(\epsilon^{-2}nd)$, while CR+DR (Algorithm 10) incurs a communication cost of $\tilde{O}(k^3\epsilon^{-6})$ and a complexity of $O(nd\cdot\min(n,d))$. This shows a *communication-computation tradeoff*: the approach of DR+CR, incurs a linear complexity and a logarithmic communication cost, whereas the

approach of CR+DR incurs a super-linear complexity (which is still less than quadratic) and a constant communication cost.

Meanwhile, we note that by Theorem 4.2.3 and Corollary 4.3.1.2, the best existing solution FSS [101] requires[7] a communication cost of $O(k\epsilon^{-2}d)$ and a complexity of $O(nd \cdot \min(n, d))$, worse than either of the proposed algorithms.

4.4 Multiple data sources and repeated DR/CR

So far we have assumed that all the data reside on a single data source and each DR/CR method is only applied once. In this section, we will investigate more general scenarios without these limiting assumptions.

4.4.1 Multiple Data Sources

Consider the scenario where the entire dataset P is split across m data sources ($m \geq 2$). Let P_i denote the dataset at data source i and n_i be its cardinality. As shown below, the previous algorithms can be adapted to the distributed setting.

4.4.1.1 Distributed Version of FSS

It turns out that the state-of-the-art distributed DR and CR algorithm, proposed in [108] (Algorithm 1), is exactly a distributed version of FSS, referred to as *BKLW* following the authors' last names. As in FSS, BKLW first uses PCA to reduce the intrinsic dimension of the dataset and then applies sensitivity sampling. However, it uses two distributed algorithms to perform these steps.

For distributed PCA, BKLW applies an algorithm *disPCA* from [101] (formalized in Algorithm 1 in [122]), where:

1. each data source i ($i = 1, \ldots, m$) computes local SVD $A_{P_i} = U_i\Sigma_i V_i^T$, and sends $\Sigma_i^{(t_1)}$ and $V_i^{(t_1)}$ to the server ($\Sigma_i^{(t_1)}$ and $V_i^{(t_1)}$ contain the first t_1 columns of Σ_i and V_i, respectively);

2. the server constructs $Y^T = [Y_1^T, \ldots, Y_m^T]$, with $Y_i = \Sigma_i^{(t_1)}(V_i^{(t_1)})^T$, computes a global SVD $Y = U\Sigma V^T$;

[7] To achieve the same approximation error with the same probability as the proposed algorithms, we need to plug in different values for ϵ and δ for FSS. However, this will not affect the order of the required complexity and communication cost wrt n, d, and k.

3. the first t_2 columns of V are returned as an approximate solution to the PCA of $\bigcup_{i=1}^m P_i$.

For distributed sensitivity sampling, BKLW applies an algorithm *disSS* from [6] (Algorithm 1), where:

1. each data source i $(i = 1, \ldots, m)$ computes a bicriteria approximation X_i for P_i and reports $\text{cost}(P_i, X_i)$;

2. the server allocates a global sample size s to each data source proportionally to its cost, i.e., $s_i = s \cdot \text{cost}(P_i, X_i) / \Big(\sum_{j=1}^m \text{cost}(P_j, X_j) \Big)$;

3. each data source i draws s_i i.i.d. samples S_i from P_i with probability proportional to $\text{cost}(\{p\}, X_i)$, and reports $S_i \cup X_i$ with their weights $w : S_i \cup X_i \to \mathbb{R}$, that are set to match the number of points per cluster;

4. the union of the reported sets $(\bigcup_{i=1}^m (S_i \cup X_i), 0, w)$ is returned as a coreset of $\bigcup_{i=1}^m P_i$.

BKLW first applies disPCA, followed by disSS with $s = O(\epsilon^{-4}(k^2/\epsilon^2 + \log(1/\delta)) + mk \log(mk/\delta))$ to the dimension-reduced dataset $\{A_{P_i} V^{(t_2)} (V^{(t_2)})^T\}_{i=1}^m$ to compute a coreset $(S, 0, w)$ at the server. Finally, the server computes the optimal k-means centers X on $(S, 0, w)$ and returns it as an approximation to the optimal k-means centers of $\bigcup_{i=1}^m P_i$.

Although a theorem was given in [108] without proof on the performance of BKLW, the result is imprecise and incomplete. Hence, we provide the complete analysis below to facilitate later comparison. We will leverage the following results.

Theorem 4.4.1 ([122]). *For any $\epsilon \in (0, 1/3)$, let $t_1 = t_2 \geq k + \lceil 4k/\epsilon^2 \rceil - 1$ in disPCA and $\tilde{P}_i$ be the projected dataset at data source i (i.e., the set of rows of $A_{P_i} V^{(t_2)} (V^{(t_2)})^T$). Then there exists a constant $\Delta \geq 0$ such that for any set $X \subset \mathbb{R}^d$ with $|X| = k$,*

$$(1-\epsilon)\, cost(P, X) \leq cost(\tilde{P}, X) + \Delta \leq (1+\epsilon)\, cost(P, X), \tag{4.55}$$

where $P := \bigcup_{i=1}^m P_i$ and $\tilde{P} := \bigcup_{i=1}^m \tilde{P}_i$.

Theorem 4.4.2 ([6]). *For a distributed dataset $\{P_i\}_{i=1}^m$ with $P_i \subset \mathbb{R}^d$ and any $\epsilon, \delta \in (0, 1)$, with probability at least $1 - \delta$, the output $(S, 0, w)$ of disSS is an ϵ-coreset of $\bigcup_{i=1}^m P_i$ of size*

$$|S| = O\left(\frac{1}{\epsilon^4} \Big(kd + \log(\frac{1}{\delta}) \Big) + mk \log(\frac{mk}{\delta}) \right). \tag{4.56}$$

Theorems 4.4.1 and 4.4.2 bound the performance of disPCA and disSS, respectively, based on which we have the following results for BKLW.

Theorem 4.4.3. *For any $\epsilon \in (0, 1/3)$ and $\delta \in (0, 1)$, suppose that in BKLW, disPCA satisfies Theorem 4.4.1 for the input dataset $\{P_i\}_{i=1}^m$ and disSS satisfies Theorem 4.4.2 for the input dataset $\{\tilde{P}_i\}_{i=1}^m$. Then*

1. *the output X is a $(1+\epsilon)^2/(1-\epsilon)^2$-approximation for k-means clustering of $\bigcup_{i=1}^m P_i$ with probability $\geq 1-\delta$,*
2. *the total communication cost over all the data sources is $O(mkd/\epsilon^2)$, and*
3. *the complexity at each data source is $O(nd \cdot \min(n, d))$,*

assuming $\min(n, d) \gg m, k, 1/\epsilon$, and $1/\delta$.

Proof. For 1), let $P := \bigcup_{i=1}^m P_i$, $\tilde{P} := \bigcup_{i=1}^m \tilde{P}_i$, and $\mathbf{S} := (S, 0, w)$ be the output of disSS. Let X^* be the optimal k-means centers of P. By Theorem 4.4.2, we know that with probability $\geq 1-\delta$,

$$\begin{aligned} \text{cost}(\tilde{P}, X) \leq \frac{1}{1-\epsilon}\text{cost}(\mathbf{S}, X) &\leq \frac{1}{1-\epsilon}\text{cost}(\mathbf{S}, X^*) \\ &\leq \frac{1+\epsilon}{1-\epsilon}\text{cost}(\tilde{P}, X^*), \end{aligned} \tag{4.57}$$

where the second inequality is because X is optimal for $\mathbf{S}$. Moreover, by Theorem 4.4.1, we have

$$(1-\epsilon)\text{cost}(P, X) - \Delta \leq \text{cost}(\tilde{P}, X), \tag{4.58}$$

$$\text{cost}(\tilde{P}, X^*) \leq (1+\epsilon)\text{cost}(P, X^*) - \Delta. \tag{4.59}$$

Combining (4.57, 4.58, 4.59) yields

$$\begin{aligned} (1-\epsilon)\text{cost}(P, X) - \Delta &\leq \frac{1+\epsilon}{1-\epsilon} \cdot ((1+\epsilon)\text{cost}(P, X^*) - \Delta) \\ &\leq \frac{(1+\epsilon)^2}{1-\epsilon}\text{cost}(P, X^*) - \Delta, \end{aligned} \tag{4.60}$$

which gives the desired approximation factor.

For 2), note that disPCA incurs a cost of $O(m \cdot (k/\epsilon^2) \cdot d)$ for transmitting $O(k/\epsilon^2)$ vectors from m data sources, and disSS incurs a cost of $O\left(\frac{k}{\epsilon^2} \cdot (\epsilon^{-4}(\frac{k^2}{\epsilon^2} + \log\frac{1}{\delta}) + mk\log\frac{mk}{\delta})\right)$

Algorithm 11: Distributed k-Means under DR+CR

input : Distributed dataset $\{P_i\}_{i=1}^m$, number of centers k, JL projection π_1, BKLW-based CR method π_2

output : Centers for k-means clustering of P

each data source i **(**$i = 1, \ldots, m$**):**

  $P_i' \leftarrow \pi_1(P_i)$;

run π_2 on the distributed dataset $\{P_i'\}_{i=1}^m$, which results in each data source i reporting a local coreset $(S_i', 0, w)$ to the server;

server:

  $X' \leftarrow \text{kmeans}(\bigcup_{i=1}^m S_i', w, k)$;

  $X \leftarrow \pi_1^{-1}(X')$;

  return X;

for transmitting $O(\epsilon^{-4}(\frac{k^2}{\epsilon^2} + \log\frac{1}{\delta}) + mk\log\frac{mk}{\delta})$ vectors in $\mathbb{R}^{O(k/\epsilon^2)}$. For $d \gg m, k, 1/\epsilon$, and $1/\delta$, the total communication cost is dominated by the cost of disPCA.

For 3), as computing the local SVD at data source i takes $O(n_i d \cdot \min(n_i, d))$ time, the complexity of disPCA at the data sources is $O(nd \cdot \min(n, d))$. The complexity of disSS at data source i is dominated by the computation of bicriteria approximation of $\tilde{P}_i$, which takes $O(n_i t_2 k \log\frac{1}{\delta}) = O(nk^2\epsilon^{-2}\log\frac{1}{\delta})$ according to [123]. For $\min(n, d) \gg m, k, 1/\epsilon$, and $1/\delta$, the overall complexity is dominated by that of disPCA. □

4.4.1.2 Enhancements

It is easy to see that each data source can apply JL projection independently at no additional communication cost at runtime. Following the ideas in Algorithms 8 and 10, we wonder: (i) Can we improve BKLW by combining it with JL projection? (ii) Is there an optimal order of applying BKLW and JL projection?

To this end, we first consider applying JL projection before invoking BKLW. For consistency with Algorithm 8, we only use the first two steps of BKLW, i.e., disPCA and disSS, that construct a coreset, which we refer to as a *BKLW-based CR method*. The algorithm, shown in Algorithm 11, is essentially the distributed counterpart of Algorithm 8.

We now analyze the performance of Algorithm 11, starting from a coreset-like property of the output of π_2.

Lemma 4.4.1. *Let $P := \bigcup_{i=1}^m P_i$ be the union of the input datasets for the BKLW-based CR method π_2 and $\mathbf{S} := (S, 0, w)$ be the resulting coreset reported to the server. For any $\epsilon \in (0, 1/3)$ and $\delta \in (0, 1)$, $\exists t_1 = t_2 = O(k/\epsilon^2)$, $s = O(\epsilon^{-4}(k^2/\epsilon^2 + \log(1/\delta)) + mk\log(mk/\delta))$, and $\Delta \geq 0$, such that with probability at least $1 - \delta$, π_2 with parameters t_1, t_2, and s*

satisfies

$$(1-\epsilon)^2 cost(P,X) \leq cost(\mathbf{S},X)+\Delta \leq (1+\epsilon)^2 cost(P,X) \tag{4.61}$$

for any set X of k points in the same space as P.

Proof. Let $\tilde{P}$ be the projection of P using the principal components computed by disPCA. Then by Theorem 4.4.1, there exists $\Delta \geq 0$ such that

$$(1-\epsilon)\text{cost}(P,X) \leq \text{cost}(\tilde{P},X)+\Delta \leq (1+\epsilon)\text{cost}(P,X). \tag{4.62}$$

Moreover, by Theorem 4.4.2, $\mathbf{S}$ is an ϵ-coreset of $\tilde{P}$ with probability at least $1-\delta$. Multiplying (4.62) by $1-\epsilon$, we have

$$\begin{aligned}(1-\epsilon)^2\text{cost}(P,X) &\leq (1-\epsilon)\text{cost}(\tilde{P},X)+(1-\epsilon)\Delta &(4.63)\\ &\leq \text{cost}(\mathbf{S},X)+\Delta, &(4.64)\end{aligned}$$

where we can obtain (4.64) from (4.63) because $\mathbf{S}$ is an ϵ-coreset of $\tilde{P}$. Similarly, multiplying (4.62) by $1+\epsilon$, we have

$$\begin{aligned}(1+\epsilon)^2\text{cost}(P,X) &\geq (1+\epsilon)\text{cost}(\tilde{P},X)+(1+\epsilon)\Delta\\ &\geq \text{cost}(\mathbf{S},X)+\Delta. &(4.65)\end{aligned}$$

Combining (4.64, 4.65) yields the desired bound. □

Remark: Comparing Lemma 4.4.1 with Definition 4.2.2, we see that π_2 does not construct an $O(\epsilon)$-coreset of its input dataset. Nevertheless, its output can approximate the k-means cost of the input dataset up to a constant shift, which is sufficient for computing an approximate k-means solution.

Theorem 4.4.4. *For any $\epsilon \in (0,1/3)$ and $\delta \in (0,1)$, suppose that in Algorithm 11, π_1 satisfies Lemma 4.3.2, π_2 satisfies Lemma 4.4.1, and kmeans$(\bigcup_{i=1}^m S_i', w, k)$ returns the optimal k-means centers of the dataset $\bigcup_{i=1}^m S_i'$ with weights w. Then*

1. *the output X is a $(1+\epsilon)^6/(1-\epsilon)^2$-approximation for k-means clustering of $\bigcup_{i=1}^m P_i$ with probability $\geq (1-\delta)^2$,*
2. *the total communication cost over all the data sources is $O(mk\epsilon^{-4}\log n)$, and*
3. *the complexity at each data source is $\tilde{O}(nd\epsilon^{-4})$,*

assuming $\min(n, d) \gg m, k, 1/\epsilon,$ *and* $1/\delta$.

Proof. For 1), let $\mathbf{S}' := (\bigcup_{i=1}^m S_i', \Delta, w)$, where $(\bigcup_{i=1}^m S_i', 0, w)$ is the overall coreset constructed by line 3 of Algorithm 11, and Δ is a constant satisfying Lemma 4.4.1 for the input dataset $\{P_i'\}_{i=1}^m$ as in line 3 of Algorithm 11. Let $P := \bigcup_{i=1}^m P_i$, and X^* be the optimal k-means centers for P. Then with probability $\geq (1-\delta)^2$, we have

$$\begin{aligned}
\text{cost}(P, X) &\leq (1+\epsilon)^2 \text{cost}(\pi_1(P), X') && (4.66)\\
&\leq \frac{(1+\epsilon)^2}{(1-\epsilon)^2} \text{cost}(\mathbf{S}', X') && (4.67)\\
&\leq \frac{(1+\epsilon)^2}{(1-\epsilon)^2} \text{cost}(\mathbf{S}', \pi_1(X^*)) && (4.68)\\
&\leq \frac{(1+\epsilon)^4}{(1-\epsilon)^2} \text{cost}(\pi_1(P), \pi_1(X^*)) && (4.69)\\
&\leq \frac{(1+\epsilon)^6}{(1-\epsilon)^2} \text{cost}(P, X^*), && (4.70)
\end{aligned}$$

where (4.66) is by Lemma 4.3.2 (note that $\pi_1(X) = X'$), (4.67) is by Lemma 4.4.1 (note that $\text{cost}(\mathbf{S}', X') = \text{cost}((\bigcup_{i=1}^m S_i', 0, w), X') + \Delta$), (4.68) is because X' is optimal in minimizing $\text{cost}(\mathbf{S}', \cdot)$, (4.69) is again by Lemma 4.4.1, and (4.70) is again by Lemma 4.3.2.

For 2), only line 3 incurs communication cost. By Theorem 4.4.3, we know that applying BKLW to a distributed dataset $\{P_i'\}_{i=1}^m$ with dimension d' incurs a cost of $O(mkd'/\epsilon^2)$, and by Lemma 4.3.2, we know that $d' = O(\log n/\epsilon^2)$, which yields the desired result.

For 3), the JL projection at each data source incurs a complexity of $O(ndd') = O(nd \log n/\epsilon^2)$. By Theorem 4.4.3, applying BKLW incurs a complexity of $O(nd' \cdot \min(n, d')) = O(n \log^2 n/\epsilon^4)$ at each data source. Together, the complexity is $O(\frac{nd}{\epsilon^2} \log n + \frac{n}{\epsilon^4} \log^2 n) = \tilde{O}(nd/\epsilon^4)$. □

Discussion: Comparing Theorems 4.4.4 and 4.4.3, we see that for $d \gg \log n$ (e.g., $d = \Omega(n)$), Algorithm 11 can significantly reduce the communication cost and the complexity at data sources, while achieving a similar $(1 + O(\epsilon))$-approximation as BKLW. Note that although the possibility of applying another DR method before BKLW was mentioned in [108], no result was given there.

Meanwhile, although one could develop a distributed counterpart of Algorithm 10 that applies JL projection after BKLW, its performance will not be competitive. Specifically, using similar analysis, this approach incurs the same order of communication cost and complexity as BKLW. Meanwhile, the JL projection introduces additional error, causing

its overall approximation error to be larger. It is thus unnecessary to consider this algorithm.

One might wonder why in the centralized setting, applying JL projection after FSS (i.e., Algorithm 10) improves the performance, but the idea does not work in the distributed setting. Fundamentally, it is because in the centralized setting, applying JL projection after FSS avoids transmitting the principal components. However, since in the distributed setting, disPCA already requires the local principal components to be transmitted, applying JL projection later only reduces a lower-order term in the communication cost that is dominated by the cost of transmitting the local principal components.

4.4.2 Repeated DR/CR

We now consider the possibility of applying certain DR/CR methods repeatedly.

4.4.2.1 Distributed Setting

We claim that with suitable DR/CR methods, repeated application of DR/CR is unnecessary in the distributed setting. This is because after one round of BKLW (with or without applying JL projection beforehand), we already reduce the cardinality to $O(\epsilon^{-4}(k^2/\epsilon^2 + \log(1/\delta)) + mk\log(mk/\delta))$ and the dimension to $O(k/\epsilon^2)$, both constant in the size (n, d) of the original dataset. Meanwhile, this round incurs a communication cost that scales with (n, d) as $O(\log n)$ (with JL projection) or $O(d)$ (without JL projection), and a complexity that scales as $\tilde{O}(nd)$ (with JL projection) or $O(nd \cdot \min(n, d))$ (without JL projection). Therefore, any possible reduction in the cost or the complexity achieved by further reducing the cardinality or dimension will be dominated by the cost or the complexity in the first round. Hence, repeated application of DR/CR will not improve the order of the communication cost or the complexity.

4.4.2.2 Centralized Setting

In contrast, we will show that repeated application can be beneficial in the centralized setting. Recall from Section 4.3.3 that applying JL projection before FSS (Algorithm 8) results in a lower complexity, while applying JL projection after FSS (Algorithm 10) results in a lower communication cost. One may wonder whether it is possible to combine the strengths of both of the algorithms. Below we give an affirmative answer by applying some of these methods repeatedly.

Algorithm 12: k-Means under Communication-efficient DR+CR+DR

input : Original dataset P, number of centers k, JL projection $\pi_1^{(1)}$ for P, FSS-based CR method π_2, JL projection $\pi_1^{(2)}$ for the output of π_2

output : Centers for k-means clustering of P

data source:

$P' \leftarrow \pi_1^{(1)}(P)$;

$(S, \Delta, w) \leftarrow \pi_2(P')$;

$S' \leftarrow \pi_1^{(2)}(S)$;

report (S', Δ, w) to the server;

server:

$X' \leftarrow \text{kmeans}(S', w, k)$;

$X \leftarrow (\pi_1^{(2)} \circ \pi_1^{(1)})^{-1}(X')$;

return X;

Specifically, we know from Theorem 4.2.3 that applying FSS once already reduces the cardinality to a constant (in n and d), and hence there is no need to repeat FSS. The same theorem also implies that if we apply FSS first, we will incur a super-linear complexity, and hence we need to apply JL projection before FSS. Meanwhile, we see from Lemmas 4.3.2 and 4.3.3 that applying JL projection on a dataset of cardinality n' can reduce its dimension to $O(\epsilon^{-2}\log(n'k/\delta))$ while achieving a $(1+O(\epsilon))$-approximation with high probability. Thus, we can further reduce the dimension by applying JL projection again after reducing the cardinality by FSS. The above reasoning suggests a three-step procedure: JL→FSS→JL, presented in Algorithm 12, where $\pi_1^{(1)}$ projects from $\mathbb{R}^d$ to $\mathbb{R}^{O(\log n/\epsilon^2)}$, and $\pi_1^{(2)}$ projects from $\mathbb{R}^{O(\log n/\epsilon^2)}$ to $\mathbb{R}^{O(\log |S|/\epsilon^2)}$. Note that by convention, $\pi_1^{(2)} \circ \pi_1^{(1)}(X)$ means $\pi_1^{(2)}(\pi_1^{(1)}(X))$.

Below, we will show that this seemingly small change is able to combine the low communication cost of Algorithm 10 and the low complexity of Algorithm 8, at a small increase in the approximation error.

Theorem 4.4.5. *For any $\epsilon, \delta \in (0,1)$, if in Algorithm 12, $\pi_1^{(1)}$ satisfies Lemma 4.3.2, $\pi_1^{(2)}$ satisfies Lemma 4.3.3, π_2 generates an ϵ-coreset of its input dataset with probability at least $1-\delta$, and kmeans(S', w, k) returns the optimal k-means centers of the dataset S' with weights w, then*

1. *the output X is a $(1+\epsilon)^9/(1-\epsilon)$-approximation for k-means clustering of P with probability $\geq (1-\delta)^3$,*
2. *the communication cost is $\tilde{O}(k^3/\epsilon^6)$, and*
3. *the complexity at the data source is $\tilde{O}(nd/\epsilon^2)$,*

assuming $\min(n, d) \gg k, 1/\epsilon$, *and* $1/\delta$.

Proof. Let $n' := |S|$, d' be the dimension after $\pi_1^{(1)}$, and d'' be the dimension after $\pi_1^{(2)}$. Let X^* be the optimal k-means centers for P.

For 1), note that with probability $\geq (1-\delta)^3$, $\pi_1^{(1)}$ and $\pi_1^{(2)}$ will preserve the k-means cost up to a multiplicative factor of $(1+\epsilon)^2$, and π_2 will generate an ϵ-coreset of P'. Thus, with this probability, we have

$$\begin{aligned}
\text{cost}(P, X) &\leq (1+\epsilon)^2 \text{cost}(P', \pi_1^{(1)}(X)) && (4.71)\\
&\leq \frac{(1+\epsilon)^2}{1-\epsilon} \text{cost}((S, \Delta, w), \pi_1^{(1)}(X)) && (4.72)\\
&\leq \frac{(1+\epsilon)^4}{1-\epsilon} \text{cost}((S', \Delta, w), \pi_1^{(2)} \circ \pi_1^{(1)}(X)) && (4.73)\\
&\leq \frac{(1+\epsilon)^4}{1-\epsilon} \text{cost}((S', \Delta, w), \pi_1^{(2)} \circ \pi_1^{(1)}(X^*)) && (4.74)\\
&\leq \frac{(1+\epsilon)^6}{1-\epsilon} \text{cost}((S, \Delta, w), \pi_1^{(1)}(X^*)) && (4.75)\\
&\leq \frac{(1+\epsilon)^7}{1-\epsilon} \text{cost}(P', \pi_1^{(1)}(X^*)) && (4.76)\\
&\leq \frac{(1+\epsilon)^9}{1-\epsilon} \text{cost}(P, X^*), && (4.77)
\end{aligned}$$

where (4.71) is by Lemma 4.3.2, (4.72) is because (S, Δ, w) is an ϵ-coreset of P', (4.73) is by Lemma 4.3.3, (4.74) is because $\pi_1^{(2)} \circ \pi_1^{(1)}(X) = X'$, which is optimal in minimizing $\text{cost}((S', \Delta, w), \cdot)$, (4.75) is by Lemma 4.3.3, (4.76) is because (S, Δ, w) is an ϵ-coreset of P', and (4.77) is by Lemma 4.3.2.

For 2), note that by Theorem 4.2.3, the cardinality of the coreset constructed by FSS is $n' = O(k^3 \log^2 k\epsilon^{-4} \log(1/\delta))$, which is independent of the dimension of the input dataset. Thus, the communication cost remains the same as that of Algorithm 10, which is $\tilde{O}(k^3/\epsilon^6)$.

For 3), note that the first JL projection $\pi_1^{(1)}$ takes $O(ndd')$ time, where $d' = O(\log n/\epsilon^2)$ by Lemma 4.3.2, and the second JL projection $\pi_1^{(2)}$ takes $O(n'd'd'')$ time, where n' is specified by Theorem 4.2.3 as above and $d'' = O(\epsilon^{-2} \log(n'k/\delta))$ by Lemma 4.3.3. Moreover, from the proof of Theorem 4.3.2, we know that applying FSS after a JL projection takes $O(\frac{n}{\epsilon^2}(\log^2 n/\epsilon^2 + k \log n/\epsilon^2 + k^2 \log \frac{1}{\delta}))$ time. Thus, the total complexity at the data source is

$$O\left(ndd' + \frac{n}{\epsilon^2}\left(\frac{\log^2 n}{\epsilon^2} + \frac{k \log n}{\epsilon^2} + k^2 \log \frac{1}{\delta}\right) + n'd'd''\right)$$

Table 4.1. Summary of Comparison

Algorithm	Communication cost	Computation complexity
FSS [101]	$O(kd/\epsilon_2^2)$	$O(nd \cdot \min(n,d))$
JL + FSS (Alg. 8)	$O(k \log n/\epsilon_1^4)$	$\tilde{O}(nd/\epsilon_1^2)$
FSS + JL (Alg. 10)	$\tilde{O}(k^3/\epsilon_1^6)$	$O(nd \cdot \min(n,d))$
JL + FSS + JL (Alg. 12)	$\tilde{O}(k^3/\epsilon_3^6)$	$\tilde{O}(nd/\epsilon_3^2)$
BKLW [108]	$O(mkd/\epsilon_4^2)$	$O(nd \cdot \min(n,d))$
JL + BKLW (Algorithm 11)	$O(mk \log n/\epsilon_5^4)$	$\tilde{O}(nd/\epsilon_5^4)$

$$= O\left(\frac{nd \log n}{\epsilon^2} + \frac{n \log^2 n}{\epsilon^4}\right) = \tilde{O}\left(\frac{nd}{\epsilon^2}\right). \tag{4.78}$$

□

Remark: Theorem 4.4.5 implies that Algorithm 12 is essentially "optimal" in the sense that it achieves a $(1 + O(\epsilon))$-approximation with an arbitrarily high probability, at a near-linear complexity and a constant communication cost at the data source. Thus, no qualitative improvement will be achieved by applying further DR/CR methods.

4.4.3 Summary of Comparison

We are now ready to compare the performances of all the proposed algorithms and their best existing counterparts in both centralized (i.e., single data source) and distributed (i.e., multiple data sources) settings.

To ensure the same approximation error for all the algorithms, we set the error parameter 'ϵ' to ϵ_1 for Algorithms 8 and 10, ϵ_2 for FSS, ϵ_3 for Algorithm 12, ϵ_4 for BKLW, and ϵ_5 for Algorithm 11, where for any $\epsilon \in (0,1)$, ϵ_1 satisfies $(1+\epsilon_1)^5/(1-\epsilon_1) = 1+\epsilon$, ϵ_2 satisfies $(1+\epsilon_2)/(1-\epsilon_2) = 1+\epsilon$, ϵ_3 satisfies $(1+\epsilon_3)^9/(1-\epsilon_3) = 1+\epsilon$, ϵ_4 satisfies $(1+\epsilon_4)^2/(1-\epsilon_4)^2 = 1+\epsilon$, and ϵ_5 satisfies $(1+\epsilon_5)^6/(1-\epsilon_5)^2 = 1+\epsilon$.

The comparison, summarized in Table 4.1, is in terms of the communication cost and the complexity at the data source(s) for achieving a $(1+\epsilon)$-approximation for k-means clustering of an input dataset of cardinality n and dimension d, where the first four rows are for the centralized setting and the last two rows are for the distributed setting. Clearly, for high-dimensional datasets satisfying $d \gg \log n$, the best proposed algorithm significantly outperforms the best existing algorithm in both the centralized and the distributed settings.

4.5 Performance Evaluation

In this section, we use experiments on real datasets to validate our analysis about the proposed joint DR and CR algorithms in comparison with the existing algorithms in both the single-source and the multiple-source cases.

4.5.1 Datasets and Metrics

We use two datasets in our experiments: (1) MNIST training dataset [53], a handwritten digits dataset which has $60,000$ images in 784-dimensional space; (2) NeurIPS Conference Papers 1987-2015 dataset [124], a word counts dataset of the NeurIPS conference papers published from 1987 to 2015, which has $11,463$ instances (words) with $5,812$ attributes (papers). Both of these two datasets are normalized to $[-1, 1]$ with zero mean. In the case of multiple data sources, we randomly partition each dataset among 10 data sources.

We measure the performance by (i) the approximation error, measured by the normalized k-means cost $\text{cost}(P, X)/\text{cost}(P, X^*)$, where X is the set of centers returned by the evaluated algorithm and X^* is the set of centers computed from P, (ii) the communication cost, measured by the number of scalars transmitted by the data source(s), and (iii) the complexity at the data source(s), measured by the running time of the evaluated DR/CR algorithm. We set $k = 2$ in all the experiments. Because of the randomness of the algorithms, we repeat each test for 10 Monte Carlo runs.

4.5.2 Algorithms

In the case of single data source, we evaluate 4 algorithms:

1. "FSS": the benchmark algorithm introduced in [101],
2. "JL+FSS": Algorithm 8, where we use JL projection before applying FSS,
3. "FSS+JL": Algorithm 10, where we use JL projection after applying FSS, and
4. "JL+FSS+JL": Algorithm 12, where we apply JL projection both before and after applying FSS.

In the case of multiple data sources, we evaluate 2 algorithms:

1. "BKLW": the benchmark algorithm from [108], and
2. "JL+BKLW": Algorithm 11, where we apply JL projection before BKLW.

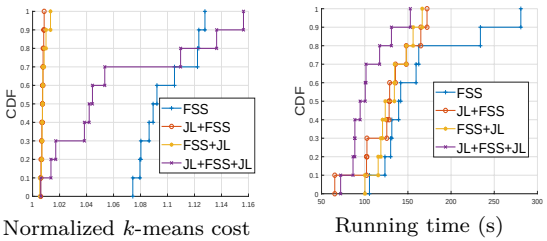


Figure 4.1. Single-source case: MNIST

In both cases, we have tuned the parameters of both the benchmark and proposed algorithms to make all the algorithms achieve a similar empirical approximation error. As a baseline, we also include the naive method of "no reduction (NR)", i.e., transmitting the raw data.

4.5.3 Results

4.5.3.1 Single data source

The results for the case of single data source (that computes and transmits a data summary to a remote server) are given in Figures 4.1–4.2 and Table 4.2. Note that by definition, the baseline (NR) has a normalized cost of 1 and no computation at the data source. We observe the following: (i) Compared to the naive method of transmitting the raw data (NR), the proposed algorithms can dramatically reduce the communication cost (by 99%) with a moderate increase in the k-means cost ($< 16\%$). (ii) Compared to the benchmark (FSS), the proposed algorithms can achieve a similar or smaller k-means cost while significantly reducing the communication cost (in the case of MNIST) or complexity (in the case of NeurIPS). (iii) Between the approaches of DR+CR (JL+FSS) and CR+DR (FSS+JL), we see that the DR+CR approach yields a better performance in these experiments, especially for the NeurIPS dataset, where JL+FSS has a substantially smaller running time than FSS+JL but similar approximation error and communication cost. This is because $\log n \ll \min(n, d)$ here, allowing JL+FSS to significantly reduce the complexity compared with FSS+JL without blowing up the communication cost. (iv) Repeated applications of DR/CR (JL+FSS+JL) can further reduce the communication cost (for NeurIPS) while obtaining comparable performance in the other metrics.

Table 4.2. Communication Cost: Single-source Case

Dataset	NR	FSS	JL+FSS	FSS+JL	JL+FSS+JL
MNIST	47,040,000	402,021	179,416	179,416	182,419
NeurIPS	66,622,956	756,741	724,072	724,836	543,373

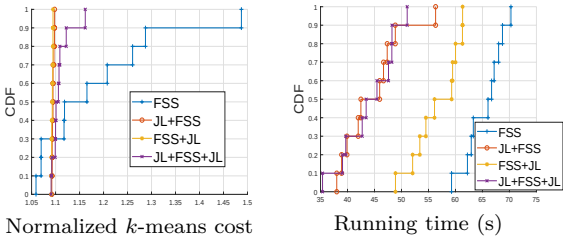


Figure 4.2. Single-source case: NeurIPS

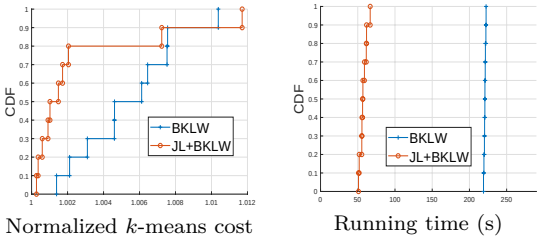


Figure 4.3. Multiple-source case: MNIST

4.5.3.2 Multiple data sources

Figures 4.3–4.4 and Table 4.3 show the results for the case of multiple data sources. We see that on both datasets, the proposed algorithm (JL+BKLW) achieves a k-means cost comparable to the benchmark (BKLW), while incurring substantially lower complexity and communication cost. This demonstrates the benefit of suitably combining the JL projection with other data reduction methods; recall that applying JL projection after BKLW will not reduce the communication cost or the complexity as explained after Theorem 4.4.4. Note that although JL+BKLW slightly underperforms BKLW in terms of k-means cost on the NeurIPS dataset, the difference is insignificant ($< 2\%$), which may be because of the randomness of JL projection.

4.5.4 Summary of Observations

Our experimental results verified the following:

- Solving k-means based on data summaries generated by DR/CR methods can effectively reduce the communication cost without incurring a high complexity at the data sources, while providing a solution of reasonable quality.
- Suitable combination of DR and CR methods will further improve the communication cost and the complexity while maintaining a similar solution quality.

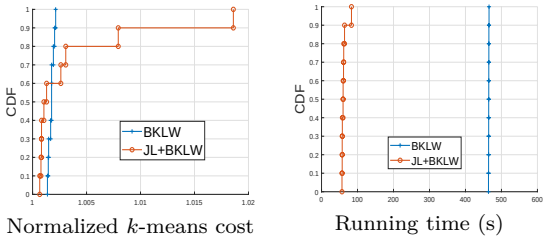


Figure 4.4. Multiple-source case: NeurIPS

Table 4.3. Communication Cost: Multiple-source Case

Dataset	NR	BKLW	JL+BKLW
MNIST	47,040,000	27,152,484	20,365,463
NeurIPS	66,622,956	20,993,951	14,036,651

4.6 Extension to Joint DR, CR, and Quantization

Besides cardinality and dimensionality, the volume of a dataset also depends on its *precision*, defined as the number of bits used to represent each attribute in the dataset. While DR and CR methods can be used to reduce the dimensionality and the cardinality, quantization techniques [64] can be used to reduce the precision and hence further reduce the communication cost. While the optimal (efficient) quantization in support of k-means is worth a separate study, our focus here is on properly combining a given quantizer with the proposed DR/CR methods. To this end, we will use a simple rounding-based quantizer as a concrete example.

4.6.1 Rounding-based Quantization

Given a scalar $x \in \mathbb{R}$, we denote the b_0-bit binary floating number representation of x by

$$x = (-1)^{\text{sign}(x)} \times 2^{e_x} \times (a(0) + a(1) \times 2^{-1} + \ldots + a(b_0 - 1 - m_e) \times 2^{-(b_0-1-m_e)}), \tag{4.79}$$

where $\text{sign}(x) = 0$ if $x \geq 0$ and $\text{sign}(x) = 1$ if $x < 0$, m_e is the number of exponent bits, e_x is the m_e-bit exponent of x, and $a(\cdot) \in \{0, 1\}$ are the significant bits ($a(0) \equiv 1$). The rounding-based quantizer Γ with s significant bits is defined as

$$\Gamma(x) := (-1)^{\text{sign}(x)} \times 2^{e_x} \times (a(0) + a(1) \times 2^{-1} + \ldots + a(s) \times 2^{-s} + a'(s) \times 2^{-s}), \tag{4.80}$$

where $a'(s)$ is the result of rounding the remaining bits (0: rounding down; 1: rounding up).

For simplicity of notation, we also use $p' := \Gamma(p)$ to denote the element-wise rounding-based quantization of a data point $p = (p_i)_{i=1}^d \in \mathbb{R}^d$. Defining the maximum quantization error as $\Delta_{QT} := \max_{p \in P} \|p - p'\|$, we know that by the definition of rounding-based quantizer, the quantization error in each element satisfies $|p_i - p'_i| \leq 2^{e_{p_i} - s} \leq |p_i| 2^{-s}$ since $|p_i| \geq 2^{e_{p_i}}$. Therefore, the maximum quantization error is bounded as

$$\Delta_{QT} = \max_{p \in P} \sqrt{\sum_{i=1}^{d} (p_i - p'_i)^2} \leq \max_{p \in P} \sqrt{\sum_{i=1}^{d} 2^{-2s} p_i^2} = 2^{-s} \max_{p \in P} \|p\|. \tag{4.81}$$

4.6.2 Approximation Error Analysis

We now analyze the performance after adding quantization to the proposed communication-efficient k-means algorithms. As DR and CR can generate data points of arbitrary values that may not be representable with a given number of significant bits, we add quantization after all the DR/CR steps. That is, we assume that right before a data source reports its dimensionality-reduced coreset (S, Δ, w) to the server, it will apply the rounding-based quantizer Γ and report (S_{QT}, Δ, w) instead[8], where $S_{QT} := \{\Gamma(p) : p \in S\}$. As the quantization reduces the communication cost and incurs a sub-linear complexity in the size of the original dataset (subsumed by the complexity of DR/CR), the only performance metric it can negatively impact is the approximation error, which is analyzed in the following theorem (see proof in [125]).

Theorem 4.6.1. *Let X denote the optimal k-means centers computed by the server based on the received coreset, X^* denote the optimal k-means centers based on the original dataset, and Δ_D denote the diameter of the input space.*

1. *In Algorithm 8, suppose that π_1 satisfies Lemma 4.3.2 with ϵ_1, π_2 generates an ϵ_2-coreset with probability $\geq 1 - \delta$, and π_{QT} is a quantizer with maximum error Δ_{QT}. If we update Line 4 to: $S'_{QT} \leftarrow \pi_{QT}(S')$ and report (S'_{QT}, Δ, w) to the server, then the approximation error will be:*

$$cost(P, X) \leq \frac{(1+\epsilon_1)^4(1+\epsilon_2)}{(1-\epsilon_2)} cost(P, X^*) + \frac{(1+\epsilon_1)^2}{(1-\epsilon_2)} 4n\Delta_D\Delta_{QT} \tag{4.82}$$

[8]Here we only apply quantization to the coreset points in S as their transfer dominates the communication cost, but our approach can be extended to other cases.

with probability at least $(1-\delta)^2$.

2. *In Algorithm 10, suppose that* π_1 *satisfies Lemma 4.3.3 with* ϵ_1, π_2 *generates an* ϵ_2*-coreset with probability* $\geq 1-\delta$, *and* π_{QT} *is a quantizer with maximum error* Δ_{QT}. *If we update Line 4 to:* $S'_{QT} \leftarrow \pi_{QT}(S')$ *and report* (S'_{QT}, Δ, w) *to the server, then the approximation error will be:*

$$cost(P,X) \leq \frac{(1+\epsilon_1)^4(1+\epsilon_2)}{(1-\epsilon_2)} cost(P,X^*) + \frac{(1+\epsilon_1)^2}{(1-\epsilon_2)} 4n\Delta_D\Delta_{QT} \tag{4.83}$$

with probability at least $(1-\delta)^2$.

3. *In Algorithm 12, suppose that* $\pi_1^{(1)}$ *satisfies Lemma 4.3.2 with* $\epsilon_1^{(1)}$, $\pi_1^{(2)}$ *satisfies Lemma 4.3.3 with* $\epsilon_1^{(2)}$, π_2 *generates an* ϵ_2*-coreset with probability* $\geq 1-\delta$, *and* π_{QT} *is a quantizer with maximum error* Δ_{QT}. *If we update Line 5 to:* $S'_{QT} \leftarrow \pi_{QT}(S')$ *and report* (S'_{QT}, Δ, w) *to the server, then the approximation error will be:*

$$cost(P,X) \leq$$
$$\frac{(1+\epsilon_1^{(1)})^4(1+\epsilon_2)(1+\epsilon_1^{(2)})^4}{(1-\epsilon_2)} cost(P,X^*) + \frac{(1+\epsilon_1^{(1)})^2(1+\epsilon_1^{(2)})^2}{(1-\epsilon_2)} 4n\Delta_D\Delta_{QT} \tag{4.84}$$

with probability at least $(1-\delta)^3$.

4. *In Algorithm 11, suppose that* π_1 *satisfies Lemma 4.3.2,* π_2 *satisfies Lemma 4.4.1, and* π_{QT} *is a quantizer with maximum error* Δ_{QT}. *If we update Line 3 to: run* π_2 *on the distributed dataset* $\{P'_i\}_{i=1}^m$ *to compute a local coreset* $(S'_i, 0, w)$ *at each data source* i *and report* $(\pi_{QT}(S'_i), 0, w)$ *to the server, then the approximation error will be:*

$$cost(P,X) \leq \frac{(1+\epsilon_1)^4(1+\epsilon_2)^2}{(1-\epsilon_2)^2} cost(P,X^*) + \frac{(1+\epsilon_1)^2}{(1-\epsilon_2)^2} 4n\Delta_D\Delta_{QT} \tag{4.85}$$

with probability at least $(1-\delta)^2$.

4.6.3 Configuration of Joint DR, CR, and Quantization

Based on the analysis of the approximation error under given DR, CR, and QT (quantization) methods, we aim to answer the following question: how can we configure the DR, CR, and QT methods such that we can minimize the communication cost while keeping the approximation error within a given bound? We will present a detailed solution for

the four-step procedure JL+FSS+JL+QT, as the solutions for the other procedures are similar.

4.6.3.1 Problem Formulation

Let $\mathcal{Y}_0$ denote a desired bound on the approximation error and $1-\delta_0$ the desired confidence level, i.e., $\text{cost}(P,X) \leq \mathcal{Y}_0 \text{cost}(P,X^*)$ with probability $\geq 1-\delta_0$, where X is the computed k-means solution and X^* the optimal solution. By Theorem 4.6.1, the QT step introduces an additive error. We now convert it into a multiplicative error to enforce the bound $\mathcal{Y}_0$. To this end, suppose that we are given a lower bound $\mathcal{E}$ on the optimal k-means cost $\text{cost}(P,X^*)$. For example, by [126], we can estimate $\mathcal{E}$ by selecting $O(k)$ points from P according to a certain probability distribution, repeating this process for $\log(1/\delta)$ times, and outputting the set X of selected points with the minimum $\text{cost}(P,X)$. This result is proven to be at most 20-time worse than the optimal solution, i.e., $\mathcal{E} := \text{cost}(P,X)/20 \leq \text{cost}(P,X^*)$, with probability at least $1-\delta$. Define $\epsilon_{QT} := \frac{4n\Delta_D\Delta_{QT}}{\mathcal{E}}$. Then based on (4.84), we have:

$$\begin{aligned}
\text{cost}(P,X) &\leq \frac{(1+\epsilon_1^{(1)})^4(1+\epsilon_2)(1+\epsilon_1^{(2)})^4}{(1-\epsilon_2)}\text{cost}(P,X^*) + \frac{(1+\epsilon_1^{(1)})^2(1+\epsilon_1^{(2)})^2}{(1-\epsilon_2)}4n\Delta_D\Delta_{QT} \\
&\leq \frac{(1+\epsilon_1^{(1)})^4(1+\epsilon_2)(1+\epsilon_1^{(2)})^4}{(1-\epsilon_2)}\text{cost}(P,X^*) + \frac{(1+\epsilon_1^{(1)})^2(1+\epsilon_1^{(2)})^2}{(1-\epsilon_2)}\frac{4n\Delta_D\Delta_{QT}}{\mathcal{E}}\text{cost}(P,X^*) \\
&= \frac{(1+\epsilon_1^{(1)})^2(1+\epsilon_1^{(2)})^2}{(1-\epsilon_2)}((1+\epsilon_1^{(1)})^2(1+\epsilon_2)(1+\epsilon_1^{(2)})^2 + \epsilon_{QT})\text{cost}(P,X^*). \qquad (4.86)
\end{aligned}$$

Let $f(\epsilon_1^{(1)}, \epsilon_2, \epsilon_1^{(2)}, \epsilon_{QT})$ denote the communication cost as a function of the configuration parameters $\epsilon_1^{(1)}$, ϵ_2, $\epsilon_1^{(2)}$, and ϵ_{QT}. Our goal is to find the optimal configuration that minimizes the communication cost while satisfying the given bound on the approximation error:

$$\min_{\epsilon_1^{(1)},\epsilon_2,\epsilon_1^{(2)},\epsilon_{QT}} \quad \mathcal{X} := f(\epsilon_1^{(1)}, \epsilon_2, \epsilon_1^{(2)}, \epsilon_{QT}) \qquad (4.87a)$$

$$\text{s.t.} \quad \mathcal{Y} := \frac{(1+\epsilon_1^{(1)})^2(1+\epsilon_1^{(2)})^2}{(1-\epsilon_2)}((1+\epsilon_1^{(1)})^2(1+\epsilon_2)(1+\epsilon_1^{(2)})^2 + \epsilon_{QT}) \leq \mathcal{Y}_0, \qquad (4.87b)$$

where $\mathcal{X}$ denotes the communication cost and $\mathcal{Y}$ denotes (an upper bound on) the approximation error. The parameter δ is set to $1-(1-\delta_0)^{1/3}$ to satisfy the desired confidence level.

4.6.3.2 Analysis

The communication cost $\mathcal{X}$ is dominated by the transfer of the dimensionality-reduced, quantized coreset S'_{QT}. Let its cardinality, dimensionality, and precision be n', d', and b'. By [101], the cardinality of an ϵ_2-coreset generated by FSS is $n' = O(\frac{k^3 \log^2(k) \log(1/\delta)}{\epsilon_2^4})$. To satisfy Lemma 4.3.2 with $\epsilon_1^{(2)}$, the dimensionality needs to satisfy $d' = O(\frac{\log(n'k/\delta)}{(\epsilon_1^{(2)})^2})$. By the analysis of quantization error in Section 4.6.1, $b' = O(\log(\frac{n\sqrt{d}}{\epsilon_{QT}}))$. Denoting the constant factors in these big-O terms by C_1, C_2, and C_3, we have

$$\mathcal{X} \approx n' \cdot d' \cdot b' = C_1 \frac{k^3 \log^2(k) \log(1/\delta)}{\epsilon_2^4} \cdot C_2 \frac{\log(n'k/\delta)}{(\epsilon_1^{(2)})^2} \cdot C_3 \log\left(\frac{n\sqrt{d}}{\epsilon_{QT}}\right) \tag{4.88}$$

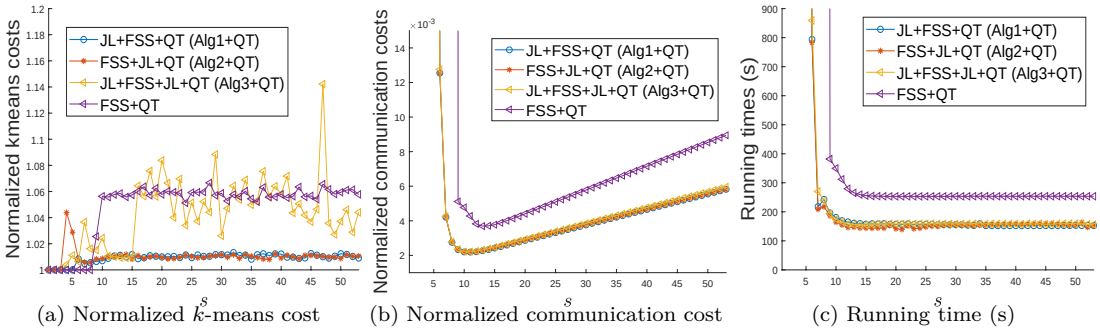
$$= \tilde{C}_1 \cdot \frac{\log(\tilde{C}_2/\epsilon_2^4)}{\epsilon_2^4 (\epsilon_1^{(2)})^2} \cdot \log\left(\frac{\tilde{C}_3}{\frac{\mathcal{Y}_0(1-\epsilon_2)}{(1+\epsilon_1^{(1)})^2(1+\epsilon_1^{(2)})^2} - (1+\epsilon_1^{(1)})^2(1+\epsilon_2)(1+\epsilon_1^{(2)})^2}\right), \tag{4.89}$$

where $\tilde{C}_1 = k^3 \log^2(k) \log(1/\delta) C_1 C_2 C_3$, $\tilde{C}_2 = k^4 \log^2(k) \log(1/\delta)/\delta$, and $\tilde{C}_3 = n\sqrt{d}$. Assuming $k \geq 2$, by plugging the constant factors from [37, 127, 128] into Theorem 36 from [101], we can use $C_1 = 54912(1 + \log_2(3))(1 + \log_2(26/3))/225$. By JL projection, $d' \leq \lceil 8 * \log(\frac{4n'k}{\delta})/\epsilon^2 \rceil$, and therefore C_2 could be 24. Assuming $n > 8$ and $\mathcal{E} \geq 1$, C_3 could be 2. Equation (4.89) is obtained by replacing ϵ_{QT} by its maximum value derived from (4.87b).

While solving (4.87) in the general case is nontrivial, we note that for the rounding-based quantizer defined in Section 4.6.1, ϵ_{QT} only has a finite number of possible values, corresponding to the number of significant bits $s = 1, \ldots, b_0 - 1 - m_e$. Thus, under the simplifying constraint of $\epsilon_1^{(1)} = \epsilon_2 = \epsilon_1^{(2)} =: \epsilon$, we can enumerate each possible value of ϵ_{QT}, compute the maximum ϵ under this ϵ_{QT} from (4.87b), and plug it into (4.89) to evaluate $\mathcal{X}$. We can then select the configuration that yields the minimum $\mathcal{X}$.

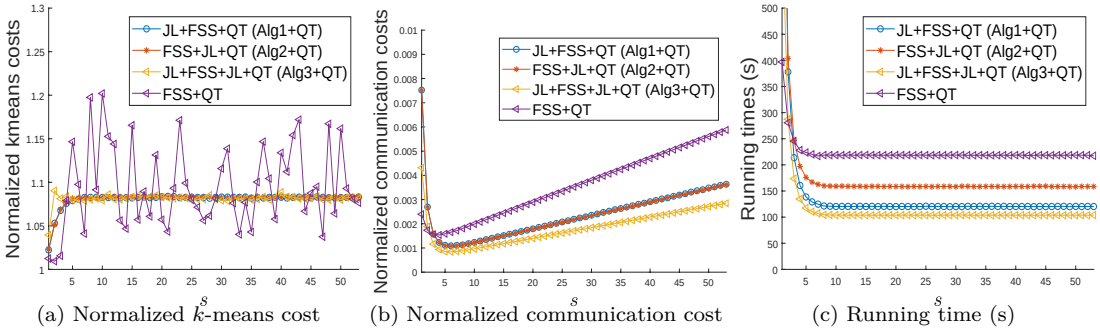
4.6.4 Experiments for Joint DR, CR, and Quantization

We now use the datasets from Section 4.5 to evaluate the impact of adding quantization to the previously proposed algorithms. We also use the same performance metrics, except that the normalized communication cost becomes the ratio between the number of transmitted bits and the total number of bits in the original dataset (in double precision). For tractability, in the experiments we set all the ϵ values except ϵ_{QT} to be equal when



(a) Normalized k-means cost (b) Normalized communication cost (c) Running time (s)

Figure 4.5. Single-source case with quantization: MNIST



(a) Normalized k-means cost (b) Normalized communication cost (c) Running time (s)

Figure 4.6. Single-source case with quantization: NeurIPS

solving (4.87). All the results are averaged over 10 Monte Carlo runs[9].

4.6.4.1 Evaluated Algorithms

In the case of a single data source, we evaluate the following:

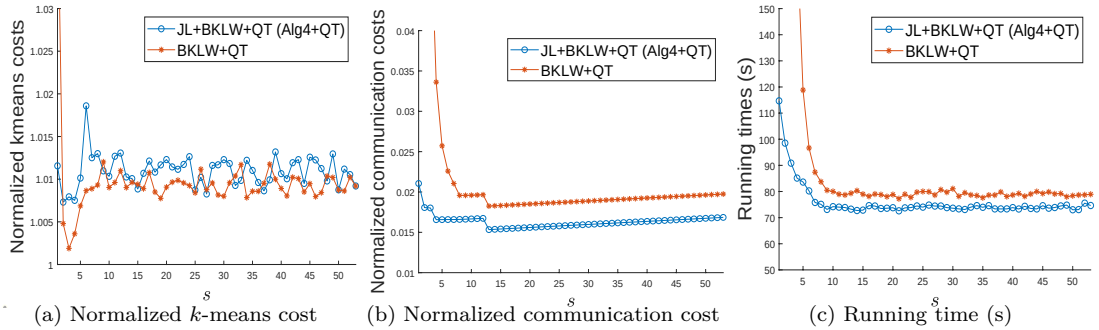
- "FSS+QT": the quantization-added version of FSS [101],
- "JL+FSS+QT": the quantization-added version of Algorithm 8,
- "FSS+JL+QT": the quantization-added version of Algorithm 10, and
- "JL+FSS+JL+QT": the quantization-added version of Algorithm 12.

In the case of multiple data sources, we evaluate the following algorithms:

- "BKLW+QT": the quantization-added version of BKLW [108], and
- "JL+BKLW+QT": the quantization-added version of Algorithm 11.

For each algorithm, we construct a data summary under each configuration that corresponds to a possible number of significant bits s, and then solve k-means based on the data summary. Specifically, since the IEEE Standard 754 floating number representation [77] consists of 53 significant bits, we enumerate $s = 1, \ldots, 53$.

[9]The number of Monte Carlo runs is limited by the running time of the experiment, which takes around 30 hours to complete one Monte Carlo run for all the algorithms and all the settings.



(a) Normalized k-means cost (b) Normalized communication cost (c) Running time (s)

Figure 4.7. Multiple-source case with quantization: MNIST

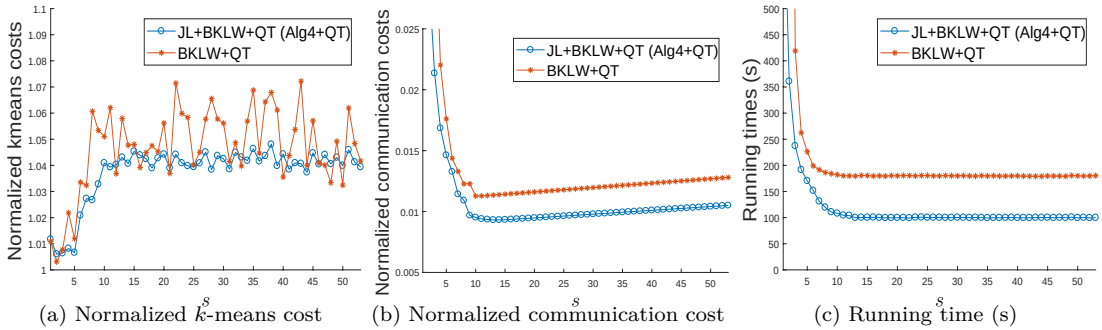
4.6.4.2 Results

The results for the single data source scenario are given in Figures 4.5–4.6. We have the following key observations: (i) Compared with the methods without quantization (the right-most points under $s = 53$), adding quantization can further reduce the communication cost by up to 2/3 without increasing the k-means cost or the running time. (ii) However, it is not trivial to find the optimal configuration to achieve a comparable k-means cost with the least communication cost and running time, e.g., very small and very large values of s both lead to high communication costs, justifying the need of solving (4.87). (iii) Overall, the four-step procedure JL+FSS+JL+QT achieves the best communication cost and running time with a comparable k-means cost.

The results for the multiple data source scenario are given in Figures 4.7–4.8. We have similar observations as in the case of a single data source: (i) Adding quantization to DR/CR methods can further reduce the communication cost by up to 10% without increasing the k-means cost or the running time. (ii) Choosing a proper configuration (by selecting the optimal number of significant bits to retain in quantization) is nontrivial. (iii) Compared with BKLW+QT, JL+BKLW+QT can reduce both the communication cost and the computational complexity at data sources, which is consistent with the comparison between BKLW and JL+BKLW (Section 4.4.1.2). We also note that in both scenarios, the proposed algorithms show better performance relative to the benchmark algorithms (i.e., FSS+QT, BKLW+QT) for NeurIPS than MNIST. This is because NeurIPS has considerably more attributes than MNIST, leaving more room of improvement for the design of data reduction algorithms.

4.7 Conclusion

In this chapter, we considered the problem of using data reduction methods to efficiently compute the k-means centers for a large high-dimensional dataset located at remote



(a) Normalized k-means cost (b) Normalized communication cost (c) Running time (s)

Figure 4.8. Multiple-source case with quantization: NeurIPS

data source(s), with focus on DR and CR. Through a comprehensive analysis of the approximation error, the communication cost, and the complexity of various combinations of state-of-the-art DR/CR methods, we proved that it is possible to achieve a near-optimal approximation of k-means at a near-linear complexity at the data source(s) and a very low (constant or logarithmic) communication cost. In the process, we developed algorithms based on novel combinations of existing DR/CR methods that outperformed two state-of-the-art algorithms in the scenario of a single data source or multiple data sources, respectively. We also demonstrated how to combine DR/CR methods with quantizers to further reduce the communication cost without compromising the other performance metrics. Our findings were validated through experiments on real datasets.

Chapter 5
Conclusion and Future Works

5.1 Summary of Contributions

In this dissertation, we are focused on the edge-based learning scenario, where large datasets are distributed at different clients and will be utilized to help build machine learning models efficiently. Towards this goal, we investigated and incorporated multiple techniques to keep reducing the size of the transmitted datasets and improving the performance of our approaches.

1. As our first step, in centralized setting, we proved that the optimal k-clustering (including k-means/median) gives a robust coreset that provides a guaranteed approximation for any machine learning problem with a sufficiently continuous cost function; in distributed setting, we adapted an existing distributed k-clustering algorithm to construct a robust coreset over distributed datasets with a very low communication cost. The robust ML performances are validated through experiments on real datasets.

2. Later on we incorporated quantization techniques into the coreset construction process and formulated an optimization problem of optimizing the configuration between cardinality and precision. Three algorithms were proposed to solve this optimization problem in centralized and distributed setting. Through extensive experiments, we showed the effectiveness of the proposed algorithms in supporting diverse ML tasks, and observed that jointly optimizing coreset and quantization achieves notably better ML performances than applying coreset only or quantization only.

3. At last, we jointly considered cardinality reduction, dimensionality reduction and

quantization to support k-means. In centralized setting, we gave the k-means performance with DR and CR methods, showed the tradeoffs with different orders of applying DR and CR methods and analyzed the performance with repeated DR and CR methods. In distributed setting, we improved the state-of-the-art algorithm by applying another DR method before it. Finally, the results were extended to incorporate the quantization and showed how to configure the quantizer to minimize the communication cost while guaranteeing a given approximation error. With the configurations of state-of-the-art DR/CR methods, we achieved a near-optimal approximation of k-means at a near-linear complexity at the data source(s) and a very low (constant or logarithmic) communication cost. At last, our experiments showed the advantage of adding quantization as the last step over approaches without quantization.

5.2 Future Directions

While we have made some progress towards reducing dataset and supporting machine learning models, there are many interesting problems to investigate:

1. Although we have formulated an optimization problem with joint DR, CR and QT techniques, how to efficiently compute the optimal ϵ configurations for these techniques remains open. It'll be promising if we could answer this question optimally.

2. Currently we only considered the rounding-based quantizer in our analysis. While the maximum quantization error will link any quantizer to our analysis easily, the formal analysis could be our next step.

3. As Deep Learning (DL) models become more and more popular, how coreset will fuel the Deep Learning model could be an interesting question. For example, viewed as a data reduction technique, coreset could be potentially applied to gradient space to accelerate the training of DL models.

4. As coresets transmit the local data summary to a global aggregator, sensitive information might be leaked to public. The adversary might be interested in attacking our coresets as well. Thus privacy-preserving coreset becomes a new challenge.

5. Recall in the edge-based learning scenario, we are interested in constructing machine learning models over distributed datasets. Towards this goal, we could either share local data (distributed coreset), or share model parameters (federated learning). It's worth investigation to compare sharing data and sharing model by different performance metrics in edge-based learning scenario.

5.3 Conclusion

In this dissertation, we studied a variety of techniques, including coreset construction, dimensionality reduction and quantization. We developed our robust coreset construction algorithm and showed that k-clustering centers form a robust coreset that supports a broad set of machine learning applications. Later we proposed the first framework to optimally and efficiently configure coreset and quantization in support of diverse machine learning tasks. At last, we dived into communication-efficient k-means calculation with the configuration of coreset, projection and quantization, showing a near-optimal approximation of k-means at a near-linear complexity at the data source(s) and a very low (constant or logarithmic) communication cost. The impact of this dissertation is the three steps towards data reduction for communication-efficient machine learning: under the same edge-based learning scenario, we started with the robust coreset construction for reducing caidinality, later continued to optimize the configuration of coreset and quantization, and finally considered optimizing the combinations of coreset, projection and quantization. With the rapid data generation and development of machine learning applications in the edge-based learning scenario, our techniques could significantly reduce the communication cost in support of machine learning tasks on distributed large dataset, therefore enabling new distributed applications that requires more data transmission.

Appendix A
Proofs

Proof of Lemma 2.3.1:

Proof. By definition, $\text{opt}(P,k) = \sum_{i=1}^{k} \text{opt}(P_i, 1)$. By letting $X_i \coloneqq \{X_{i,1}, X_{i,2}\}$ be the centers of the optimal 2-clustering of P_i, we have

$$\sum_{i=1}^{k} \text{opt}(P_i, 2) = \sum_{i=1}^{k} \sum_{p \in P_i} w_p (\min_{x \in X_i} \text{dist}(p,x))^z$$

$$\geq \sum_{i=1}^{k} \sum_{p \in P_i} w_p (\min_{x \in \bigcup_{i=1}^{k} X_i} \text{dist}(p,x))^z \geq \text{opt}(P, 2k). \tag{A.1}$$

Thus we have

$$\sum_{i=1}^{k} (\text{opt}(P_i, 1) - \text{opt}(P_i, 2)) \leq \text{opt}(P,k) - \text{opt}(P, 2k) \leq \epsilon'. \tag{A.2}$$

Since $\text{opt}(P_i, 1) - \text{opt}(P_i, 2) \geq 0$, $\text{opt}(P_i, 1) - \text{opt}(P_i, 2) \leq \epsilon'$, $\forall i \in [k]$. □

Proof of Lemma 2.3.2:

Proof. By definition of k-clustering, we have

$$\text{opt}(P_i, 1) = \sum_{p \in P_i} w_p \text{dist}(p, \mu(P_i))^z, \tag{A.3}$$

$$\text{opt}(P_i, 2) \leq \sum_{p \in P_i^1} w_p \text{dist}(p, \mu(P_i))^z + \sum_{p \in P_i^2} w_p \text{dist}(p, p_0)^z \tag{A.4}$$

for any $p_0 \in P_i$, where P_i^1 is the subset of points in P_i that are closer to $\mu(P_i)$ than p_0

(ties broken arbitrarily) and $P_i^2 \coloneqq P_i \setminus P_i^1$. Then subtracting (A.3) by (A.4), we have

$$\mathrm{opt}(P_i,1)-\mathrm{opt}(P_i,2)\geq\sum_{p\in P_i^2} w_p(\mathrm{dist}(p,\mu(P_i))^z-\mathrm{dist}(p,p_0)^z). \tag{A.5}$$

As $\mathrm{dist}(p,\mu(P_i))^z - \mathrm{dist}(p,p_0)^z \geq 0$ for all $p \in P_i^2$,

$$\begin{aligned} w_p(\mathrm{dist}(p,\mu(P_i))^z-\mathrm{dist}(p,p_0)^z) &\leq \mathrm{opt}(P_i,1)-\mathrm{opt}(P_i,2) \\ &\leq \epsilon',\ \forall p \in P_i^2. \end{aligned} \tag{A.6}$$

In particular, for $p = p_0$, $w_{p_0}\mathrm{dist}(p_0,\mu(P_i))^z \leq \epsilon'$ and thus $\mathrm{dist}(p_0,\mu(P_i)) \leq (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$. The proof completes by noting that this holds for any $p_0 \in P_i$. □

Proof of Lemma 2.3.3:

Proof. Since $\forall x \in \mathcal{X}$ and $p \in P_i$,

$$(1-\epsilon)\mathrm{cost}(p,x) \leq \mathrm{cost}(\mu(P_i),x) \leq (1+\epsilon)\mathrm{cost}(p,x), \tag{A.7}$$

multiplying this inequality by w_p and then summing over $p \in P_i$, we have

$$\begin{aligned} (1-\epsilon)\sum_{p\in P_i} w_p\mathrm{cost}(p,x) &\leq \mathrm{cost}(\mu(P_i),x)\sum_{p\in P_i} w_p \\ &\leq (1+\epsilon)\sum_{p\in P_i} w_p\mathrm{cost}(p,x),\ \forall x \in \mathcal{X}. \end{aligned} \tag{A.8}$$

Summing up (A.8) over all $i \in [k]$, we have

$$(1-\epsilon)\mathrm{cost}(P,x) \leq \mathrm{cost}(S,x) \leq (1+\epsilon)\mathrm{cost}(P,x),\ \forall x \in \mathcal{X}.$$

Therefore, S is an ϵ-coreset for P w.r.t. $\mathrm{cost}(P,x)$. □

Proof of Lemma 2.3.4

Proof. Taking maximum over $p \in P_i$ for (2.5), we have that $\forall i \in [k]$ and $x \in \mathcal{X}$,

$$\begin{aligned} (1-\epsilon)\max_{p\in P_i}\mathrm{cost}(p,x) &\leq \mathrm{cost}(\mu(P_i),x) \\ &\leq (1+\epsilon)\max_{p\in P_i}\mathrm{cost}(p,x). \end{aligned} \tag{A.9}$$

Taking maximum over $i \in [k]$ for (A.9): $\forall x \in \mathcal{X}$,

$$(1-\epsilon)\max_{p\in P}\text{cost}(p,x) \le \max_{i\in[k]}\text{cost}(\mu(P_i),x) \le (1+\epsilon)\max_{p\in P}\text{cost}(p,x).$$

That is, S is an ϵ-coreset for P w.r.t. $\text{cost}(P,x)$. □

Proof of Lemma 2.3.5:

Proof. By definition, $\text{approx}(P,k) = \sum_{i=1}^{k}\text{approx}(\widetilde{P}_i,1)$. By Assumption 2, we can have: $\sum_{i=1}^{k}\text{approx}(\widetilde{P}_i,2) \ge \text{approx}(P,2k)$. Thus, we have

$$\sum_{i=1}^{k}(\text{approx}(\widetilde{P}_i,1)-\text{approx}(\widetilde{P}_i,2)) \le \text{approx}(P,k)-\text{approx}(P,2k) \le \epsilon'. \tag{A.10}$$

Let $\{\widetilde{P}_i^1,\widetilde{P}_i^2\}$ be the partition of $\widetilde{P}_i$ generated by the algorithm for 2-clustering of $\widetilde{P}_i$. Under Assumption 1,

$$\begin{aligned}\text{approx}(\widetilde{P}_i,2) &= \sum_{p\in\widetilde{P}_i^1} w_p\text{dist}(p,\mu(\widetilde{P}_i^1))^z + \sum_{p\in\widetilde{P}_i^2} w_p\text{dist}(p,\mu(\widetilde{P}_i^2))^z \\ &\le \sum_{p\in\widetilde{P}_i^1} w_p\text{dist}(p,\mu(\widetilde{P}_i))^z + \sum_{p\in\widetilde{P}_i^2} w_p\text{dist}(p,\mu(\widetilde{P}_i))^z \\ &= \text{approx}(\widetilde{P}_i,1).\end{aligned} \tag{A.11}$$

Combining (A.10, A.11) yields $\text{approx}(\widetilde{P}_i,1)-\text{approx}(\widetilde{P}_i,2) \le \epsilon'$ for any $i \in [k]$. □

Proof of Lemma 2.3.6:

Proof. First, by Assumption 1, $\text{approx}(\widetilde{P}_i,1) = \sum_{p\in\widetilde{P}_i} w_p \cdot\text{dist}(p,\mu(\widetilde{P}_i))^z$. Moreover, by Assumption 3,

$$\begin{aligned}\text{approx}(\widetilde{P}_i,2) &\le \sum_{p\in\widetilde{P}_i} w_p(\min_{x\in\{\mu(\widetilde{P}_i),p^*\}}\text{dist}(p,x))^z \\ &\le \sum_{p\in\widetilde{P}_i\setminus\{p^*\}} w_p\text{dist}(p,\mu(\widetilde{P}_i))^z,\end{aligned} \tag{A.12}$$

where $p^* \coloneqq \arg\max_{p\in\widetilde{P}_i} w_p \text{dist}(p, \mu(\widetilde{P}_i))^z$. Thus, we have

$$\begin{aligned}\epsilon' \geq \text{approx}(\widetilde{P}_i, 1) - \text{approx}(\widetilde{P}_i, 2) &\geq w_{p^*}\text{dist}(p^*, \mu(\widetilde{P}_i))^z \\ &\geq w_p \text{dist}(p, \mu(\widetilde{P}_i))^z,\ \forall p \in \widetilde{P}_i. \end{aligned} \tag{A.13}$$

Therefore, $\text{dist}(p, \mu(\widetilde{P}_i)) \leq (\frac{\epsilon'}{w_{\min}})^{\frac{1}{z}}$, $\forall p \in \widetilde{P}_i$. □

Proof of Theorem 2.4.1:

Proof. The proof is based on a sampling lemma from [6]:

Lemma A.0.1 ([6]). *Let F be a set of nonnegative functions $f : P \to \mathbb{R}_{\geq 0}$. Let S be a set of i.i.d. samples from P, where each sample equals $p \in P$ with probability $\frac{m_p}{\sum_{z\in P} m_z}$. Let $u'_q = \frac{\sum_{p\in P} m_p}{m_q|S|}$ for $q \in P$. If for a sufficiently large constant α, $|S| \geq \frac{\alpha}{\epsilon^2}(\dim(F, P) + \log\frac{1}{\delta})$, then with probability at least $1-\delta$, $\forall f \in F$: $\left|\sum_{p\in P} f(p) - \sum_{q\in S} u'_q f(q)\right| \leq \epsilon(\sum_{p\in P} m_p)(\max_{p\in P}\frac{f(p)}{m_p})$.*

In our case, $S = \bigcup_{j=1}^n S_j$, and $B = \bigcup_{j=1}^n B_j^{k_j}$. Define $f_x(p) \coloneqq w_p(\text{cost}(p, x) - \text{cost}(b_p, x) + \rho\text{dist}(p, b_p))$, where b_p is the center in $B_j^{k_j}$ closest to $p \in P_j$. By the ρ-Lipschitz-continuity of $\text{cost}(p, x)$, $f_x(p) \geq 0$ and $f_x(p) \leq 2\rho w_p \text{dist}(p, b_p)$. By lines 6 and 9 in Algorithm 2, k-median-based DRCC generates S via i.i.d. sampling from P with probabilities proportional to $m_p = w_p\text{dist}(p, b_p)$. Therefore, by Lemma A.0.1, there exists $|S| = O(\frac{1}{\epsilon^2}(\dim(F, P) + \log\frac{1}{\delta}))$ such that with probability at least $1-\delta$, $\forall x \in \mathcal{X}$:

$$\Delta \coloneqq \Big|\sum_{p\in P} f_x(p) - \sum_{q\in S} u'_q f_x(q)\Big| \leq \epsilon(\sum_{p\in P} m_p)(\max_{p\in P}\frac{f_x(p)}{m_p}). \tag{A.14}$$

The righthand sides of (A.14) and (2.11) are equal as $\sum_{p\in P} m_p = \sum_{j=1}^n c(P_j, B_j^{k_j})$ and $\frac{f_x(p)}{m_p} \leq 2\rho$.

We will show that the lefthand side of (A.14) also equals the lefthand side of (2.11). Specifically,

$$\sum_{p\in P} f_x(p) = \sum_{p\in P} w_p \text{cost}(p, x) - \sum_{b\in B}\text{cost}(b, x)\sum_{p\in P_b} w_p + \rho\sum_{p\in P} m_p, \tag{A.15}$$

and since $u'_q w_q = u_q$ for each $q \in S$ (line 10 in Algorithm 2),

$$\sum_{q\in S} u'_q f_x(q) = \sum_{q\in S} u_q \text{cost}(q, x) - \sum_{b\in B}\text{cost}(b, x)\sum_{q\in P_b\cap S} u_q$$

$$+ \rho \sum_{q \in S} u_q \text{dist}(q, b_q). \tag{A.16}$$

Since $\sum_{q \in S} u_q \text{dist}(q, b_q) = \sum_{p \in P} m_p$,

$$\begin{aligned} \Delta &= \Big| \sum_{p \in P} w_p \text{cost}(p, x) - \sum_{q \in S} u_q \text{cost}(q, x) \\ &\quad - \sum_{b \in B} \text{cost}(b, x) (\sum_{p \in P_b} w_p - \sum_{q \in P_b \cap S} u_q) \Big| \\ &= \Big| \sum_{p \in P} w_p \text{cost}(p, x) - \sum_{q \in S \cup B} u_q \text{cost}(q, x) \Big|, \end{aligned} \tag{A.17}$$

as $u_b = \sum_{p \in P_b} w_p - \sum_{q \in P_b \cap S} u_q$ (line 11 in Algorithm 2). □

Appendix B
Analysis of Lipschitz Constant

• 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 triangle inequality, we have:

$$|\text{dist}(p, x) - \text{dist}(p', x)| \leq \text{dist}(p, p'). \tag{B.1}$$

Hence, its cost function is 1-Lipschitz-continuous ($\rho = 1$).

• For k-median, $\text{cost}(p, x) = \min_{q \in x} \text{dist}(p, q)$, where $x \subset \mathbb{R}^d$ denotes the set of k centers. For any data points $p, p' \in \mathbb{R}^d$, *without loss of generality (WLOG)*, suppose $\min_{q \in x} \text{dist}(p, q) \geq \min_{q \in x} \text{dist}(p', q) = \text{dist}(p', q')$ for some $q' \in x$. Then

$$\begin{aligned} |\min_{q \in x} \text{dist}(p, q) - \min_{q \in x} \text{dist}(p', q)| &= \min_{q \in x} \text{dist}(p, q) - \text{dist}(p', q') \\ &\leq \text{dist}(p, q') - \text{dist}(p', q') \leq \text{dist}(p, p'). \end{aligned} \tag{B.2}$$

Hence, $\rho = 1$ for k-median.

• For k-means, $\text{cost}(p, x) = \min_{q \in x} \text{dist}(p, q)^2$, where x denotes the set of k centers. Similar to the above, for any data points $p, p' \in \mathbb{R}^d$, suppose WLOG that $\min_{q \in x} \text{dist}(p, q) \geq \min_{q \in x} \text{dist}(p', q) = \text{dist}(p', q')$ for some $q' \in x$. Then

$$\begin{aligned} |\min_{q \in x} \text{dist}(p, q)^2 - \min_{q \in x} \text{dist}(p', q)^2| &\leq \text{dist}(p, q')^2 - \text{dist}(p', q')^2 \\ &= (\text{dist}(p, q') + \text{dist}(p', q'))(\text{dist}(p, q') - \text{dist}(p', q')) \\ &\leq 2\Delta \cdot \text{dist}(p, p'), \end{aligned} \tag{B.3}$$

where Δ is the diameter of the sample space (assuming that the centers also reside in the sample space). Hence, $\rho = 2\Delta$ for k-means.

• For PCA, $\text{cost}(p, x) = \text{dist}(p, xp)^2$, where $x = WW^T$ is the projection matrix, and

W is the matrix consisting of the first l ($l < d$) principle components as columns. For any data points $p, p' \in \mathbb{R}^d$, assuming WLOG that $\text{dist}(p, xp) \geq \text{dist}(p', xp')$, we have

$$\begin{aligned} |\text{dist}(p, xp)^2 - \text{dist}(p', xp')^2| = (\text{dist}(p, xp) + \text{dist}(p', xp')) \\ \cdot (\text{dist}(p, xp) - \text{dist}(p', xp')). \end{aligned} \tag{B.4}$$

The first factor in (B.4) is upper-bounded by 2Δ (Δ: diameter of sample space), as long as the projections xp, xp' reside in the sample space. The second factor is upper-bounded by

$$\begin{aligned} \text{dist}(p, xp') - \text{dist}(p', xp') + \text{dist}(xp', xp) \\ \leq \text{dist}(p, p') + ||x||_2 \cdot \text{dist}(p, p'), \end{aligned} \tag{B.5}$$

which is due to triangle inequality and the sub-multiplicative property of ℓ-2 norm, i.e., $\text{dist}(xp', xp) = ||x(p' - p)||_2 \leq ||x||_2 \cdot ||p' - p||_2$. As the principle components are mutually orthogonal unit vectors, we have $||x||_2 \leq l$. Plugging these into (B.4) gives

$$|\text{dist}(p, xp)^2 - \text{dist}(p', xp')^2| \leq 2\Delta(l + 1)\text{dist}(p, p'), \tag{B.6}$$

i.e., $\rho = 2\Delta(l + 1)$ for PCA.

• For SVM, $\text{cost}(p, x) = \max(0, 1 - p_d(p_{1:d-1}^T x_{1:d-1} + x_d))$, where $p_{1:d-1} \in \mathbb{R}^{d-1}$ denotes the numerical portion of a data point p, and $p_d \in \{1, -1\}$ denotes its label. Consider two points p, p' that are identical with each other except the label, i.e., $p_{1:d-1} = p'_{1:d-1}$ and $p_d = -p'_d$. Suppose that $p_{1:d-1}^T x_{1:d-1} + x_d < -1$, $p_d = 1$, and $p'_d = -1$. Then we have $\text{dist}(p, p') = 2$, $\text{cost}(p', x) = 0$, and $\text{cost}(p, x) = 1 - (p_{1:d-1}^T x_{1:d-1} + x_d)$. As $\text{cost}(p, x)$ for SVM is unbounded in general, the ratio

$$\frac{|\text{cost}(p, x) - \text{cost}(p', x)|}{\text{dist}(p, p')} = \frac{1}{2}\Big[1 - (p_{1:d-1}^T x_{1:d-1} + x_d)\Big] \tag{B.7}$$

is unbounded. Therefore, $\rho = \infty$ for SVM.

Appendix C
Analysis of Dimension of Function Space

Definition C.0.1 ([6]). *Let $F \coloneqq \{f_x : x \in \mathcal{X}\}$, where each element $f_x : P \to \mathbb{R}_{\geq 0}$ is a function from a set P to nonnegative real numbers. Define $B(x, r) \coloneqq \{p \in P : f_x(p) \leq r\}$. The* dimension of the function space $\dim(F, P)$ *is the smallest integer m such that*

$$|\{S \cap B(x, r) : x \in \mathcal{X},\ r \geq 0\}| \leq |S|^m, \quad \forall S \subseteq P. \tag{C.1}$$

The dimension of function space is related to the *Vapnik-Chervonenkis (VC) dimension*, defined as follows. We refer to $(P, \mathcal{R})$ as a *range space* if P is a set of points and $\mathcal{R}$ is a family of subsets of P.

Definition C.0.2 (VC dimension [129]). *The VC dimension of a range space $(P, \mathcal{R})$, denoted by $d_{VC}(P, \mathcal{R})$, is the maximum cardinality of a set $S \subseteq P$ that is* shattered *by $\mathcal{R}$, i.e., $\{S \cap R : R \in \mathcal{R}\}$ contains all the subsets of S.*

Lemma C.0.1 (Corollary 5.2.3 [129]). *If $(P, \mathcal{R})$ is a range space of VC dimension m, then for every $S \subseteq P$, we have $|\{S \cap R : R \in \mathcal{R}\}| \leq |S|^m$.*

By Definition C.0.1 and Lemma C.0.1, we have $\dim(F, P) \leq d_{\mathrm{VC}}(P, \mathcal{R})$ for $\mathcal{R} \coloneqq \{B(x, r) : x \in \mathcal{X}, r \geq 0\}$. This allows us to bound the dimension of a function space by bounding the VC dimension of the corresponding range space. The VC dimension has an intuitive meaning that it is the number of free parameters to uniquely specify a set in $\mathcal{R}$, e.g., the VC dimension of intervals is 2, the VC dimension of planar disks is 3, and the VC dimension of half spaces in $\mathbb{R}^d$ is $d + 1$ [129]. In our case, we conjecture that the VC dimension is $O(d_{\mathcal{X}})$, where $d_{\mathcal{X}}$ is the number of parameters to uniquely specify an $x \in \mathcal{X}$.

Appendix D
Additional Evaluations

In addition to the normalized costs, we have also evaluated the relative approximation error of the overall cost, defined as $|\text{cost}(P, x_S) - \text{cost}(S, x_S)|/\text{cost}(P, x_S)$, where x_S is the model parameter computed on a coreset S. By Definition 2.2.1, this error should be upper-bounded by ϵ if S is an ϵ-coreset for P. In Table D.1 we show the maximum relative approximation error over all the Monte Carlo runs together with the value of ϵ computed according to Corollary 2.3.2.1. Indeed, the error is always upper-bounded by ϵ. Meanwhile, we note that the bound is very loose, and the maximum error is usually orders of magnitude smaller than ϵ. Similar results have been obtained for the other datasets.

Table D.1. ϵ for Fisher's iris dataset with coreset size 5

problem	z	max relative error	ϵ
MEB	1	0.0041	0.4127
MEB	2	0.0097	0.6424
Kmeans	1	0.0002	3.5175
Kmeans	2	0.0307	5.4752
PCA	1	0.6416	14.0698
PCA	2	0.8844	21.9008

Bibliography

[1] Chiang, M. and T. Zhang (2016) "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things Journal*, **3**(6), pp. 854–864.

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

[3] McMahan, H. B., E. Moore, D. Ramage, S. Hampson, and B. A. Arcas (2016) "Communication-efficient Learning of Deep Networks from Decentralized Data," in *AISTATS*.

[4] Wang, S., T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan (2018) "When Edge Meets Learning: Adaptive Control for Resource-constrained Distributed Machine Learning," in *IEEE INFOCOM*.

[5] Konečnỳ, J., H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon (2016) "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*.

[6] Balcan, M. F., S. Ehrlich, and Y. Liang (2013) "Distributed K-means and K-median Clustering on General Topologies," in *NIPS*.

[7] Kannan, R., S. Vempala, and D. Woodruff (2014) "Principal Component Analysis and Higher Correlations for Distributed Data," in *COLT*.

[8] Barger, A. and D. Feldman (2016) "k-Means for Streaming and Distributed Big Sparse Data," in *SDM*.

[9] Lu, H., C. Liu, T. He, S. Wang, and K. S. Chan (2020) "Sharing Models or Coresets: A Study based on Membership Inference Attack," *arXiv preprint arXiv:2007.02977*.

[10] Perera, C., A. Zaslavsky, M. Compton, P. Christen, and D. Georgakopoulos (2013) "Semantic-driven configuration of internet of things middleware," in *2013 Ninth International Conference on Semantics, Knowledge and Grids*, IEEE, pp. 66–73.

[11] Lane, N. D., S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar (2015) "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the 2015 international workshop on internet of things towards applications*, pp. 7–12.

[12] Van Laerhoven, K. (2001) "Combining the self-organizing map and k-means clustering for on-line classification of sensor data," in *International Conference on Artificial Neural Networks*, Springer, pp. 464–469.

[13] Yau, K.-L. A., P. Komisarczuk, and P. D. Teal (2012) "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *Journal of Network and Computer Applications*, **35**(1), pp. 253–267.

[14] Bādoiu, M., S. Har-Peled, and P. Indyk (2002) "Approximate Clustering via Core-sets," in *ACM STOC*.

[15] Gray, R. M. and D. L. Neuhoff (1998) "Quantization," *IEEE transactions on information theory*, **44**(6), pp. 2325–2383.

[16] Feldman, D., M. Schmidt, and C. Sohler (2013) "Turning Big Data into Tiny Data: Constant-size Coresets for k-means, PCA, and Projective clustering," in *SODA*.

[17] Zou, H. and L. Xue (2018) "A selective overview of sparse principal component analysis," *Proceedings of the IEEE*, **106**(8), pp. 1311–1320.

[18] Chen, S., S. Ma, L. Xue, and H. Zou (2020) "An Alternating Manifold Proximal Gradient Method for Sparse Principal Component Analysis and Sparse Canonical Correlation Analysis," *Informs Journal on Optimization*, **2**(3), pp. 192–208.

[19] Johnson, W. and J. Lindenstrauss (1982) "Extensions of Lipschitz Mappings into a Hilbert Space," in *Conference in Modern Analysis and Probability*.

[20] Chan, P. K. and S. J. Stolfo (1997) "Toward Parallel and Distributed Learning by Meta-Learning," in *AAAI Workshop in Knowledge Discovery in Databases*.

[21] Kittler, J., M. Hatef, R. P. Duin, and J. Matas (1998) "On Combining Classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3), pp. 226–239.

[22] Wolpert, D. (1992) "Stacked Generalization," *Neural Networks*, **5**(2), pp. 241–259.

[23] Tsoumakas, G. and I. Vlahavas (2002) "Effective Stacking of Distributed Classifiers," in *ECAI*.

[24] Guo, Y. and J. Sutiwaraphun (1999) "Probing Knowledge in Distributed Data Mining," in *Methodologies for Knowledge Discovery and Data Mining*.

[25] Chawla, N., L. Halla, K. Bowyer, and W. Kegelmeyer (2004) "Learning Ensembles from Bites: A Scalable and Accurate Approach," *Journal of Machine Learning Research*, **5**, pp. 421–451.

[26] Lazarevic, A. and Z. Obradovic (2002) "Boosting Algorithms for Parallel and Distributed Learning," *Distributed and Parallel Databases*, **11**(2), pp. 203–229.

[27] Phillips, J. M. (2016) "Coresets and Sketches," *CoRR*, **abs/1601.00617**.

[28] Munteanu, A. and C. Schwiegelshohn (2018) "Coresets-Methods and History: A Theoreticians Design Pattern for Approximation and Streaming Algorithms," *KI - Künstliche Intelligenz*, **32**(1), pp. 37–53.

[29] Feldman, D. and M. Langberg (2011) "A Unified Framework for Approximating and Clustering Data," in *STOC*.

[30] Bādoiu, M. and K. L. Clarkson (2003) "Smaller Core-sets for Balls," in *SODA*.

[31] Har-Peled, S. and K. R. Varadarajan (2002) "Projective Clustering in High Dimensions Using Core-Sets," in *SOCG*.

[32] Tsang, I. W., J. T. Kwok, and P.-M. Cheung (2005) "Core Vector Machines: Fast SVM Training on Very Large Data Sets," *The Journal of Machine Learning Research*, **6**, pp. 363–392.

[33] Har-Peled, S., D. Roth, and D. Zimak (2007) "Maximum Margin Coresets for Active and Noise Tolerant Learning," in *IJCAI*.

[34] Feldman, D., M. Volkov, and D. Rus (2016) "Dimensionality Reduction of Massive Sparse Datasets using Coresets," in *NIPS*.

[35] Feldman, D., A. Munteanu, and C. Sohler (2014) "Smallest Enclosing Ball for Probabilistic Data," in *SOCG*.

[36] Clarkson, K. L. (2010) "Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm," *ACM Transactions on Algorithms*, **6**(4).

[37] Langberg, M. and L. J. Schulman (2010) "Universal ϵ Approximators for Integrals," in *SODA*.

[38] Feldman, D., M. Feigin, and N. Sochen (2013) "Learning Big (Image) Data via Coresets for Dictionaries," *Journal of Mathematical Imaging and Vision*, **46**(3), pp. 276–291.

[39] Molina, A., A. Munteanu, and K. Kersting (2018) "Core Dependency Networks," in *AAAI*.

[40] Braverman, V., D. Feldman, and H. Lang (2016) "New Frameworks for Offline and Streaming Coreset Constructions," *CoRR*, **abs/1612.00889**.

[41] Feldman, D., A. Fiat, and M. Sharir (2006) "Coresets for Weighted Facilities and Their Applications," in *FOCS*.

[42] Frahling, G. and C. Sohler (2005) "Coresets in Dynamic Geometric Data Streams," in *STOC*.

[43] Har-Peled, S. and S. Mazumdar (2004) "On Coresets for K-means and K-median Clustering," in *STOC*.

[44] Boutsidis, C., P. Drineas, and M. Magdon-Ismail (2013) "Near-optimal Coresets for Least-Squares Regression," *IEEE. Trans. IT*, **59**(10), pp. 6880–6892.

[45] Feldman, D., A. Krause, and M. Faulkner (2011) "Scalable Training of Mixture Models via Coresets," in *NIPS*.

[46] Aloise, D., A. Deshpande, P. Hansen, and P. Popat (2009) "NP-hardness of Euclidean sum-of-squares clustering," *Machine Learning*, **75**(2), pp. 245–248.

[47] Megiddo, N. and K. J. Supowit (1984) "On the Complexity of Some Common Geometric Location Problems," *SIAM Journal of Computing*, **13**(1), pp. 182–196.

[48] Arthur, D. and S. Vassilvitskii (2007) "k-means++: The Advantages of Careful Seeding," in *SODA*.

[49] Cohen, M., Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford (2016) "Geometric Median in Nearly Linear Time," in *STOC*.

[50] Fisher, R. (1936), "Iris Data Set," `https://archive.ics.uci.edu/ml/datasets/iris`.
URL `https://archive.ics.uci.edu/ml/datasets/iris`

[51] Moro, S., P. Rita, and B. Vala (2016), "Facebook Metrics Data Set," `https://archive.ics.uci.edu/ml/datasets/Facebook+metrics`.
URL `https://archive.ics.uci.edu/ml/datasets/Facebook+metrics`

[52] Alpaydin, E. and F. Alimoglu (1996), "Pen-based Recognition of Handwritten Digits Data Set," `https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits`.
URL `https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits`

[53] LeCun, Y., C. Cortes, and C. Burges (1998), "The MNIST Database of Handwritten Digits," `http://yann.lecun.com/exdb/mnist/`.
URL `http://yann.lecun.com/exdb/mnist/`

[54] Smith, V., C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar (2017) "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, pp. 4424–4434.

[55] Wang, S., T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan (2018) "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, pp. 63–71.

[56] Phillips, J. M. (2016) "Coresets and sketches," *arXiv preprint arXiv:1601.00617*.

[57] Feldman, D., M. Monemizadeh, C. Sohler, and D. P. Woodruff (2010) "Coresets and sketches for high dimensional subspace approximation problems," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, pp. 630–649.

[58] Sarlos, T. (2006) "Improved approximation algorithms for large matrices via random projections," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, IEEE, pp. 143–152.

[59] Agarwal, P. K., S. Har-Peled, and K. R. Varadarajan (2005) "Geometric approximation via coresets," *Combinatorial and computational geometry*, **52**, pp. 1–30.

[60] Clarkson, K. L. (2010) "Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm," *ACM Transactions on Algorithms (TALG)*, **6**(4), p. 63.

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

[62] Lu, H., M.-J. Li, T. He, S. Wang, V. Narayanan, and K. S. Chan (2019), "Robust Coreset Construction for Distributed Machine Learning," `arXiv:1234.56789`.
URL `http://arxiv.org/abs/1904.05961`

[63] Lloyd, S. (1982) "Least squares quantization in PCM," *IEEE transactions on information theory*, **28**(2), pp. 129–137.

[64] Sayood, K. (2012) *Introduction to Data Compression, Fourth Edition*, 4th ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[65] Fischer, T. (1986) "A pyramid vector quantizer," *IEEE transactions on information theory*, **32**(4), pp. 568–583.

[66] Goodman, D. and R. Wilkinson (1975) "A robust adaptive quantizer," *IEEE Transactions on Communications*, **23**(11), pp. 1362–1365.

[67] Farvardin, N. and V. Vaishampayan (1987) "Optimal quantizer design for noisy channels: An approach to combined source-channel coding," *IEEE Transactions on Information Theory*, **33**(6), pp. 827–838.

[68] Dasgupta, A., P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney (2009) "Sampling algorithms and coresets for \ell_p regression," *SIAM Journal on Computing*, **38**(5), pp. 2060–2078.

[69] Lucic, M., M. Faulkner, A. Krause, and D. Feldman (2017) "Training mixture models at scale via coresets," *stat*, **1050**, p. 23.

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

[71] Feldman, D., M. Feigin, and N. Sochen (2013) "Learning big (image) data via coresets for dictionaries," *Journal of mathematical imaging and vision*, **46**(3), pp. 276–291.

[72] Han, S., H. Mao, and W. J. Dally (2015) "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*.

[73] Zhou, A., A. Yao, Y. Guo, L. Xu, and Y. Chen (2017) "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*.

[74] Lin, D., S. Talathi, and S. Annapureddy (2016) "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning*, pp. 2849–2858.

[75] Gray, R. M. (1990) "Vector quantization," *Readings in speech recognition*, **1**(2), pp. 75–100.

[76] Gersho, A. and R. M. Gray (2012) *Vector quantization and signal compression*, vol. 159, Springer Science & Business Media.

[77] IEEE (2019), "754-2019 - IEEE Standard for Floating-Point Arithmetic," . URL `https://ieeexplore.ieee.org/servlet/opac?punumber=8766227`

[78] Ding, C. and X. He (2004) "K-means clustering via principal component analysis," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, p. 29.

[79] Stewart, G. W. (2002) "A Krylov–Schur algorithm for large eigenproblems," *SIAM Journal on Matrix Analysis and Applications*, **23**(3), pp. 601–614.

[80] Lim, A., B. Rodrigues, F. Wang, and Z. Xu (2005) "k-Center problems with minimum coverage," *Theoretical Computer Science*, **332**(1-3), pp. 1–17.

[81] Khuller, S. and Y. J. Sussmann (2000) "The capacitated k-center problem," *SIAM Journal on Discrete Mathematics*, **13**(3), pp. 403–418.

[82] Khuller, S., R. Pless, and Y. J. Sussmann (2000) "Fault tolerant k-center problems," *Theoretical Computer Science*, **242**(1-2), pp. 237–245.

[83] Hochbaum, D. S. and D. B. Shmoys (1985) "A best possible heuristic for the k-center problem," *Mathematics of operations research*, **10**(2), pp. 180–184.

[84] Kleinberg, J. and E. Tardos (2006) *Algorithm design*, Pearson Education India.

[85] Demmel, J., I. Dumitriu, and O. Holtz (2007) "Fast linear algebra is stable," *Numerische Mathematik*, **108**(1), pp. 59–91.

[86] Luss, H. (1991) "A nonlinear minimax allocation problem with multiple knapsack constraints," *Operations Research Letters*, **10**(4), pp. 183–187.

[87] ——— (1987) "An algorithm for separable non-linear minimax problems," *Operations Research Letters*, **6**(4), pp. 159–162.

[88] Farvardin, N. (1990) "A Study of Vector Quantization for Noisy Channels," *IEEE Transactions on Information Theory*, **36**(4), pp. 799–809.

[89] Jegou, H., M. Douze, and C. Schmid (2011) "Product Quantization for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**(1), pp. 117–128.

[90] Jain, A. K. (2010) "Data Clustering: 50 Years Beyond k-Means," *Pattern Recognition Letters*, **31**(8), pp. 651–666.

[91] Lu, H., M.-J. Li, T. He, S. Wang, V. Narayanan, and K. S. Chan (2019) "Robust Coreset Construction for Distributed Machine Learning," in *IEEE Globecom*.

[92] Aloise, D., A. Deshpande, P. Hansen, and P. Popat (2009) "NP-hardness of Euclidean Sum-of-Squares Clustering," *Machine Learning*, **75**(2), pp. 245–248.

[93] Mahajan, M., P. Nimbhorkar, and K. R. Varadarajan (2012) "The Planar k-Means Problem is NP-hard," *Theoretical Computer Science*, **442**(13), pp. 13–21.

[94] Feldman, D., M. Monemizadeh, and C. Sohler (2007) "A PTAS for k-Means Clustering Based on Weak Coresets," in *ACM Symposium on Computational Geometry (SoCG)*.

[95] Kumar, A., Y. Sabharwal, and S. Sen (2010) "Linear-time Approximation Schemes for Clustering Problems in Any Dimensions," *Journal of the ACM*, **57**(2), pp. 5:1–5:32.

[96] Friggstad, A., M. Rezapour, and M. R. Salavatipour (2016) "Local Search Yields a PTAS for k-Means in Doubling Metrics," in *FOCS*.

[97] Cohen-Addad, V., P. N. Klein, and C. Mathieu (2016) "Local Search Yields Approximation Schemes for k-Means and k-Median in Euclidean and Minor-free Metrics," in *FOCS*.

[98] Awasthi, P., M. Charikar, R. Krishnaswamy, and A. K. Sinop (2015) "The Hardness of Approximation of Euclidean k-Means," in *ACM Symposium on Computational Geometry (SoCG)*.

[99] Lee, E., M. Schmidt, and J. Wright (2017) "Improved and Simplified Inapproximability for k-Means," *Information Processing Letters*, **120**, pp. 40–43.

[100] Makarychev, K., Y. Makarychev, and I. Razenshteyn (2019) "Performance of Johnson-Lindenstrauss Transform for k-Means and k-Medians Clustering," in *ACM SIGACT Symposium on Theory of Computing (STOC)*.

[101] Feldman, D., M. Schmidt, and C. Sohler (2018), "Turning Big Data into Tiny Data: Constant-size Coresets for k-Means, PCA, and Projective Clustering," `arXiv:1807.04518v1`.
URL `https://arxiv.org/abs/1807.04518`

[102] Boutsidis, C., A. Zouzias, and P. Drineas (2010) "Random Projections for k-Means Clustering," in *NIPS*.

[103] Cohen, M. B., S. Elder, C. Musco, C. Musco, and M. Persu (2015) "Dimensionality Reduction for k-Means Clustering and Low Rank Approaximation," in *STOC*.

[104] Drineas, P., A. Frieze, R. Kannan, S. Vempala, and V. Vinay (2004) "Clustering Large Graphs via the Singular Value Decomposition," *Machine Learning*, **56**(1-3), pp. 9–33.

[105] Boutsidis, C., A. Zouzias, M. W. Mahoney, and P. Drineas (2015) "Randomized Dimensioinality Reduction for k-Means Clustering," *IEEE Transactions on Information Theory*, **61**(2), pp. 1045–1062.

[106] Har-Peled, S. and A. Kushal (2007) "Smaller Coresets for k-Median and k-Means Clustering," *Discrete & Computational Geometry*, **37**(1), pp. 3–19.

[107] Chen, K. (2009) "On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and their Applications," *SIAM Journal on Computing*, **39**(3), pp. 923–947.

[108] Balcan, M. F., V. Kanchanapally, Y. Liang, and D. Woodruff (2014) "Improved Distributed Principal Component Analysis," in *NIPS*.

[109] Awasthi, P., A. Bakshi, M.-F. Balcan, C. White, and D. P. Woodruff (2019) "Robust Communication-Optimal Distributed Clustering Algorithms," in *International Colloquium on Automata, Languages, and Programming (ICALP)*.

[110] Zhao, W., H. Ma, and Q. He (2009) "Parallel K-Means Clustering based on MapReduce," in *IEEE International Conference on Cloud Computing (CloudCom)*.

[111] Anchalia, P. P., A. K. Koundinya, and N. K. Srinath (2013) "MapReduce Design of K-Means Clustering Algorithm," in *IEEE International Conference on Information Science and Applications (ICISA)*.

[112] Mao, Y., Z. Xu, P. Ping, and L. Wang (2015) "An Optimal Distributed K-Means Clustering Algorithm Based on CloudStack," in *International Conference on Frontier of Computer Science and Technology*.

[113] Oliva, G., R. Setola, and C. N. Hadjicostis (2014), "Distributed k-Means Algorithm," `arXiv:1312.4176v3`.
URL `https://arxiv.org/abs/1312.4176`

[114] Naldi, M. C. and R. J. G. B. Campello (2013) "Distributed k-means Clustering with Low Transmission Cost," in *Brazilian Conference on Intelligent Systems*.

[115] Giannella, C. R., H. Kargupta, and S. Datta (2009) "Approximate Distributed K-Means Clustering over a Peer-to-Peer Network," *IEEE Transactions on Knowledge and Data Engineering*, **21**, pp. 1372–1388.

[116] Lin, P., Y. Wang, H. Qi, and Y. Hong (2018) "Distributed Consensus-Based K-Means Algorithm in Switching Multi-Agent Networks," *Journal of Systems Science and Complexity*, **31**(5), p. 1128–1145.

[117] Ben-Israel, A. and T. N. E. Greville (2003) *Generalized Inverses: Theory and Applications*, Springer.

[118] Dasgupta, S. and A. Gupta (2003) "An Elementary Proof of a Theorem of Johnson and Lindenstrauss," *Random Structures & Algorithms*, **22**(1), pp. 60–65.

[119] Indyk, P. and R. Motwani (1998) "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," in *ACM STOC*.

[120] Achlioptas, D. (2003) "Database-friendly Random Projections: Johnson-Lindenstrauss with Binary Coins," *Journal of Computer and System Sciences*, **66**(4), pp. 671–687.

[121] Klartag, B. and S. Mendelson (2005) "Empirical Processes and Random Projections," *Journal of Functional Analysis*, **225**(1), pp. 229–245.

[122] Balcan, M. F., V. Kanchanapally, Y. Liang, and D. Woodruff (2014), "Improved Distributed Principal Component Analysis," arXiv:1408.5823.
URL https://arxiv.org/abs/1408.5823

[123] Aggarwal, A., A. Deshpande, and R. Kannan (2009) "Adaptive Sampling for k-Means Clustering," in *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*.

[124] Perrone, V., P. A. Jenkins, D. Spano, and Y. W. Teh (2016) "Poisson random fields for dynamic feature models," *arXiv preprint arXiv:1611.07460*.

[125] Lu, H., T. He, W. S, C. Liu, M. Mahdavi, V. arayanan, K. S. Chan, and S. Pasteris (2021), "Communication-efficient k-means for edge-based machine learning," 2102.04282.
URL http://arxiv.org/abs/2102.04282

[126] Aggarwal, A., A. Deshpande, and R. Kannan (2009) "Adaptive sampling for k-means clustering," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Springer, pp. 15–28.

[127] Blumer, A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth (1989) "Learnability and the Vapnik-Chervonenkis dimension," *Journal of the ACM (JACM)*, **36**(4), pp. 929–965.

[128] Li, Y., P. M. Long, and A. Srinivasan (2001) "Improved bounds on the sample complexity of learning," *Journal of Computer and System Sciences*, **62**(3), pp. 516–527.

[129] Har-Peled, S. (2011) *Geometric Approximation Algorithms*, American Mathematical Society.

Hanlin Lu VITA

PhD candidate at Penn State hll.cs1995@gmail.com, hzl263@psu.edu

State College, PA 16801

EDUCATION

Pennsylvania State University Aug 2017 - Present

The Department of Computer Science and Engineering

PhD candidate

Beihang University Sep 2013 - Jun 2017

School of Mathematics and Systems Science

Undergraduate in Hua Luogeng Class

WORK EXPERIENCES

Penn State CSE Research Assistant Aug 2017 - Present

Advisor: Prof. Ting He

IBM T. J. Watson Research Center Research intern May 2019 - Aug 2019

IBM T. J. Watson Research Center Research intern Jun 2020 - Aug 2020

Mentors: Dr. Changchang Liu and Dr. Shiqiang Wang

Penn State CSE Teaching Assistant Jan 2018 - May 2018

Course: CMPEN362 (EE362): Communication Networks

Publications

1. Communication-efficient k-Means for Edge-based Machine Learning. Lu, Hanlin, et al. *IEEE ICDCS*, November 2020. Acceptance rate: 18%.

2. Sharing Models or Coresets: A Study based on Membership Inference Attack. Lu, Hanlin, et al. *FL-ICML'20*, long paper presentation, July 2020. Acceptance rate: 23.9%.

3. Joint Coreset Construction and Quantization for Distributed Machine Learning. Lu, Hanlin, et al. *IFIP Networking*, June 2020.

4. Robust Coreset Construction for Distributed Machine Learning. Lu, Hanlin, et al. *IEEE JSAC*, March 2020. Acceptance rate: 21.4%.