

Joint Coreset Construction and Quantization for Distributed Machine Learning

Hanlin Lu*, Changchang Liu[†], Shiqiang Wang[†], Ting He*, Vijay Narayanan*, Kevin S. Chan[‡], Stephen Pasteris[§]

*Pennsylvania State University, University Park, PA, USA

[†]IBM T. J. Watson Research Center, Yorktown, NY, USA

[‡]Army Research Laboratory, Adelphi, MD, USA

[§]University College London, UK

Abstract—Coresets are small, weighted summaries of larger datasets, aiming at providing provable error bounds for machine learning (ML) tasks while significantly reducing the communication and computation costs. To achieve a better trade-off between ML error bounds and costs, we propose the first framework to incorporate quantization techniques into the process of coreset construction. Specifically, we theoretically analyze the ML error bounds caused by a combination of coreset construction and quantization. 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 budget to the nodes while minimizing the overall ML error. Through extensive experiments on multiple real-world datasets, we demonstrate the effectiveness and efficiency of our proposed algorithms for a variety of ML tasks. In particular, our algorithms have achieved more than 90% data reduction with less than 10% degradation in ML performance in most cases.

Index Terms—Coreset, quantization, distributed machine learning, optimization

I. INTRODUCTION

Distributed analytics applications in tactical coalition environments employ various *machine learning* (ML) models to unleash the valuable information hidden in the data. One key challenge for such applications is the high communication cost in training/adapting ML models over large distributed datasets. One approach to address this challenge is federated learning [1]–[3], 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 construct data summaries, several techniques have been proposed, which can be generally classified into: 1) sketching techniques for reducing the feature dimension [4]–[6]; and 2) coreset construction techniques for reducing the sample dimension [7]–[15]. 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 will preserve the feature space, making them directly applicable to the original ML tasks. Therefore, we focus on coreset-based data summarization.

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 [16], [17], where various techniques have been proposed to transform the data points from arbitrary values in the sample space to a set of discrete values [18]–[21].

In this work, 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.

In summary, our contributions include:

- 1) We are the first to incorporate quantization techniques into the coreset construction process. Based on rigorous analysis for the performance of a combination of coreset construction and quantization, we formulate an optimization problem to jointly configure the coreset construction algorithm and the quantizer to minimize the ML error under a given communication budget.
- 2) We propose two algorithms to solve the optimization by identifying proxies of the objective function that can be evaluated efficiently for large datasets. Through theoretical analysis as well as experimental evaluations, we demonstrate the effectiveness of the proposed algorithms in supporting diverse ML tasks.
- 3) We further propose a novel algorithm to allocate the communication budget across multiple nodes to adapt our solutions to the distributed setting. Experimental results demonstrate the effectiveness of the proposed algorithm as well as its advantages over existing solutions.

II. RELATED WORK

Coresets have been widely applied in shape fitting and clustering problems [13]. Previous coreset construction algorithms can be classified into four categories: sampling-based algorithms [14], [22]–[24], SVD-based algorithms [15], [25], [26], space-decomposition algorithms [27], [28], and iterative algorithms [7], [29]. Sampling-based algorithms [14], [22]–[24] leverage importance/sensitivity sampling and other advanced sampling techniques to build a subset. SVD-based algorithms [15], [25], [26] use the singular value decomposition (SVD) to build coresets for ML tasks including dictionary design and sparse representation. Space-decomposition algorithms [27], [28] partition the original sample space and then select representative points from these partitions. Iterative algorithms [7], [29] select points to form a coreset in an iterative manner. However, most existing coreset construction algorithms are model-specific [11]. To address this issue, *robust coreset construction (RCC)* has been recently proposed in [28] to support a variety of ML models with provable error bounds. Therefore, in this work, we focus on RCC as our choice of coreset construction algorithm.

Quantization techniques [16]–[21], [30]–[32] 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 [18]–[21], which can be classified into *scalar quantizers* and *vector quantizers*. 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.

Roadmap. Section III reviews the background on coreset and quantization. Section IV formulates the main problem. Section V presents two algorithms based on strategic reformulation of the original problem. Section VI extends the solutions to distributed setting. Section VII presents our experimental results. At last, Section VIII concludes this paper.

III. PRELIMINARIES

In this section, we briefly review several definitions and algorithms that will be used in subsequent sections.

A. 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 $\mathbf{Y} := [\mathbf{y}_1, \dots, \mathbf{y}_i, \dots, \mathbf{y}_n]$ 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} .

B. Coreset Construction (CS)

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) [9] 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 III.1 (ϵ_{CS} -coreset [11]). *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.) $\text{cost}(\mathcal{Y}, \mathbf{x})$ ($\mathbf{x} \in \mathcal{X}$) if $\forall \mathbf{x} \in \mathcal{X}$,*

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

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

Definition III.1 also provides a performance measure for coresets: $\epsilon_{CS, \mathcal{S}} := \sup_{\mathbf{x} \in \mathcal{X}} |\text{cost}(\mathcal{Y}, \mathbf{x}) - \text{cost}(\mathcal{S}, \mathbf{x})| / \text{cost}(\mathcal{Y}, \mathbf{x})$ measures the maximum relative error in approximating the ML cost function by coreset \mathcal{S} , called the *ML error* of \mathcal{S} . The smaller $\epsilon_{CS, \mathcal{S}}$, the better \mathcal{S} is in supporting the ML task. A recent work [28] showed that using k -means clustering centers as the coreset achieves guaranteed performances for a broad class of ML tasks. In the sequel, we denote the optimal k -means clustering cost for \mathcal{Y} by $\text{opt}(k)$. It is known that the optimal 1-means center of \mathcal{Y} is the sample mean $\mu(\mathcal{Y})$.

C. Quantization (QT)

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 $(-1)^{\text{sign}(y_{ij})} \times 2^{e_{ij}} \times (a_{ij}(0) + a_{ij}(1) \times 2^{-1} + \dots + a_{ij}(b_0 - 1 - m_e) \times 2^{-(b_0 - 1 - m_e)})$ [33]. Here, $\text{sign}(y_{ij})$ 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 $y'_{ij} = (-1)^{\text{sign}(y_{ij})} \times 2^{e_{ij}} \times (a_{ij}(0) + a'_{ij}(1) \times 2^{-1} + \dots + a'_{ij}(s) \times 2^{-s})$, 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

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

IV. 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.

A. 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.

B. Error Bound Analysis

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

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

Theorem IV.1 is directly implied by the following Lemma IV.1, which gives the ML error after one single quantization.

Lemma IV.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 IV.1.*

C. Configuration Optimization

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

$$\min_{b,k} \epsilon_{CS}(k) + \rho\Delta(b) + \epsilon_{CS}(k) \cdot \rho\Delta(b) \quad (3a)$$

$$\text{s.t. } b \cdot k \cdot d \leq B, \quad (3b)$$

$$b, k \in \mathbb{Z}^+, \quad (3c)$$

where $\epsilon_{CS}(k)$ 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 (3a) according to Theorem IV.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.

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 [28], 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 III-C, 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 (2). Then, by Theorem IV.1, the MECB problem in this case becomes:

$$\min_{b,k} \rho\sqrt{f(k)} + \rho\Delta(b) + \rho^2\Delta(b)\sqrt{f(k)} \quad (4a)$$

$$\text{s.t. } b \cdot k \cdot d \leq B, \quad (4b)$$

$$b, k \in \mathbb{Z}^+. \quad (4c)$$

3) *Straightforward Solution:* In (4), only b (or k) is the free decision variable. Thus, a straightforward way to solve (4) is to evaluate the objective function (4a) 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 (4) in the following section by identifying proxies of the objective function that can be evaluated efficiently.

V. EFFICIENT ALGORITHMS FOR MECB

In this section, we propose two algorithms to effectively and efficiently solve the concrete MECB problem given in (4).

A. Eigenvalue Decomposition Based Algorithm for MECB (EVD-MECB)

1) *Re-formulating the Optimization Problem:* This algorithm is motivated by the following bound derived in [34].

Theorem V.1 (Bound for k -means costs [34]). *The optimal k -means cost for \mathcal{Y} is bounded by*

$$\text{opt}(k) \geq n\bar{\mathbf{y}}^2 - \sum_{i=1}^{k-1} \lambda_i, \quad (5)$$

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

Algorithm 1: 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 .

- 1 Calculate eigenvalues $\{\lambda_i\}_{i=1}^d$ for $\mathbf{Y}\mathbf{Y}^T$;
- 2 $\Lambda_j \leftarrow \sum_{i=1}^j \lambda_i, \forall 1 \leq j \leq d$;
- 3 **foreach** $b = [1 + m_e, 2 + m_e, \dots, b_0]$ **do**
- 4 $k \leftarrow \lfloor B/d/b \rfloor$;
- 5 $f(k) \leftarrow \Lambda_{2k-1} - \Lambda_{k-1}$;
- 6 $\Delta(b) \leftarrow 2^{-(b-1-m_e)} \max_{\mathbf{y}_i \in \mathcal{Y}} \|\mathbf{y}_i\|$;
- 7 $\epsilon(k, b) \leftarrow \rho \cdot \sqrt{f(k)} + \rho \cdot \Delta(b) + \rho^2 \cdot \Delta(b) \cdot \sqrt{f(k)}$;
- 8 $(k^*, b^*) \leftarrow \arg \min \epsilon(k, b)$;
- 9 **return** (k^*, b^*) ;

Algorithm 2: 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 $k = 1, \dots, K$.

- 1 $\mathcal{G} \leftarrow \emptyset$;
- 2 **foreach** $\mathbf{y} \in \mathcal{Y}$ **do**
- 3 $d(\mathbf{y}) \leftarrow \infty$;
- 4 **while** $|\mathcal{G}| < K$ **do**
- 5 $\mathbf{y} \leftarrow \arg \max_{\mathbf{q} \in \mathcal{Y}} d(\mathbf{q})$;
- 6 $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathbf{y}\}$;
- 7 **foreach** $j = 1, \dots, |\mathcal{Y}|$ **do**
- 8 $d(\mathbf{y}_j) \leftarrow \min(d(\mathbf{y}_j), \text{dist}(\mathbf{y}_j, \mathbf{y}))$;
- 9 $g(|\mathcal{G}|) \leftarrow \max_{\mathbf{y}_j \in \mathcal{Y}} d(\mathbf{y}_j)$;
- 10 **return** $(g(k))_{k=1}^K$;

We use the bound in (5) as an approximation of k -means cost that is much faster to evaluate than the exact k -means cost. Replacing $\text{opt}(k)$ by this bound, we obtain an approximation of (4), where $f(k)$ is approximated by

$$f(k) \approx n\bar{\mathbf{y}}^2 - \sum_{i=1}^{k-1} \lambda_i - (n\bar{\mathbf{y}}^2 - \sum_{i=1}^{2k-1} \lambda_i) = \sum_{i=k}^{2k-1} \lambda_i. \quad (6)$$

2) *EVD-MECB Algorithm*: The righthand side of (6) is easier to calculate than the exact value of $f(k)$, as we can compute the eigenvalue decomposition once [35], 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 (4). We summarize the algorithm in Algorithm 1.

B. Max-distance Based Algorithm for MECB (MD-MECB)

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. Let $f_2(k) := \max_{i=1, \dots, k} \max_{\mathbf{y} \in \mathcal{Y}_i} \text{dist}(\mathbf{y}, \mu(\mathcal{Y}_i))$ be the maximum distance between any data point and its nearest k -means center, where $\{\mu(\mathcal{Y}_i)\}_{i=1}^k$ are the k -means clusters. Then, a similar proof as that of Lemma IV.1 implies the following.

Algorithm 3: 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 \mathcal{S}' for \mathcal{Y} within budget B .

- 1 Run Algorithm 2 with input \mathcal{Y} and $K = \min\{\lfloor \frac{B}{d \cdot (1+m_e)} \rfloor, n\}$;
- 2 **foreach** $b = [1 + m_e, 2 + m_e, \dots, b_0]$ **do**
- 3 $k \leftarrow \lfloor B/d/b \rfloor$;
- 4 $\text{opt}_\infty(k) \leftarrow g(k)$ by the output of Algorithm 2;
- 5 $\Delta(b) \leftarrow 2^{-(b-1-m_e)} \max_{\mathbf{y}_i \in \mathcal{Y}} \|\mathbf{y}_i\|$;
- 6 $\epsilon(k, b) \leftarrow \rho \cdot \text{opt}_\infty(k) + \rho \Delta(b) + \rho^2 \cdot \text{opt}_\infty(k) \Delta(b)$;
- 7 $(k^*, b^*) \leftarrow \arg \min \epsilon(k, b)$;
- 8 **return** (k^*, b^*) ;

Lemma V.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 satisfying the conditions in Theorem IV.1.*

This lemma provides an alternative error bound for the RCC algorithm in [28], which constructs the coreset as in Lemma V.1. Using $\epsilon_{\text{CS}} = \rho f_2(k)$, if we apply the rounding-based quantization after RCC, we can apply Theorem IV.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 [36]–[38]. Hence, we would like to use the optimal k -center cost, denoted by $\text{opt}_\infty(k)$, as a heuristic to calculate $f_2(k)$. By using this alternative error bound and approximating $f_2(k) \approx \text{opt}_\infty(k)$, we can reformulate the MECB problem as follows:

$$\min_{b, k} \quad \rho \cdot \text{opt}_\infty(k) + \rho \Delta(b) + \rho^2 \cdot \text{opt}_\infty(k) \Delta(b) \quad (7a)$$

$$\text{s.t.} \quad b \cdot k \cdot d \leq B, \quad (7b)$$

$$b, k \in \mathbb{Z}^+, \quad (7c)$$

where $\Delta(b)$ is defined as in (4).

2) *MD-MECB Algorithm*: The re-formulation (7) allows us to leverage algorithms for k -center clustering to efficiently evaluate $\text{opt}_\infty(k)$. Although k -center clustering is a NP-hard problem [39], a number of efficient heuristics have been proposed. In particular, it has been proved [39] that the best possible approximation for k -center clustering is 2-approximation, achieved by a simple greedy algorithm [40] that keeps adding the point farthest from the existing 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 2. 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 [39], the returned costs satisfy $\text{opt}_\infty(k) \leq g(k) \leq 2\text{opt}_\infty(k)$, where $g(\cdot)$ is defined in line 9. Based on this algorithm, the MD-MECB algorithm, shown in Algorithm 3, solves an approximation of (7) with $\text{opt}_\infty(k)$ approximated by $g(k)$.

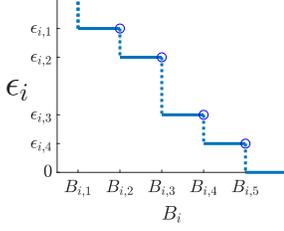


Fig. 1: Illustration of step objective functions.

C. Discussions

1) *Performance Comparison:* The straightforward solution EM (Section IV-C3) directly minimizes the error bound (4a) 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 VII).

2) *Complexity Comparison:* In terms of complexity, EM is computationally complicated because of executions of k -means. In comparison, EVD-MECB and MD-MECB can be done once for all (k, b) pairs. As EVD can be computed with complexity $O(n^3)$ [41], EVD-MECB has a complexity of $O(n^3 + d + b_0)$. Since the computational complexity of Algorithm 2 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 VII.

VI. 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 $\{\mathcal{Y}_1, \dots, \mathcal{Y}_N\}$. Our goal is to configure the construction of local coresets $\{\mathcal{S}_1, \dots, \mathcal{S}_N\}$ 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 V 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.

A. 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 VI.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 $\mathcal{C}_1 \cup \mathcal{C}_2$ is a $\max\{\epsilon_1, \epsilon_2\}$ -coreset for $\mathcal{Y}_1 \cup \mathcal{Y}_2$ w.r.t. the same cost function.*

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 $i \in \{1, \dots, N\}$.

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:

$$\min \max_{i \in \{1, \dots, N\}} \epsilon_i(B_i) \quad (8a)$$

$$\text{s.t.} \quad \sum_{i=1}^N B_i \leq B. \quad (8b)$$

Note that to compute $\epsilon_i(B_i)$ for a given B_i , we need to solve an instance of the MECB problem in (4) for dataset \mathcal{Y}_i and budget B_i .

B. Optimal Budget Allocation Algorithm for MECBD (OBA-MECBD)

The MECBD problem in (8) is a *minimax knapsack problem* [42], [43] with a nonlinear non-increasing objective function. Special cases of this problem with strictly decreasing objective functions have been solved in [43]. 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 1). This is because the configuration parameters k and b in the CS + QT procedure are integers. Therefore, there exist intervals $[B_{i,j}, B_{i,j+1})$ ($j = 0, 1, 2, \dots$) 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 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 4, 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

Algorithm 4: 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 $i = 1, \dots, N$:

- 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 $I \leftarrow \lfloor \frac{I_{max} + I_{min}}{2} \rfloor$;
- 14 $\epsilon_I \leftarrow$ the I -th element in \mathbf{E} ;
- 15 **for** $i = 1, \dots, N$ **do**
- 16 $B_i(\epsilon_I) \leftarrow \min\{B_{i,j} : (B_{i,j}, \epsilon_{i,j}) \in \mathcal{E}_i, \epsilon_{i,j} \leq \epsilon_I\}$;
- 17 **if** $I_{min} = I_{max} + 1$ **then**
- 18 send $B_i(\epsilon_I)$ to node i for each $i = 1, \dots, N$;
- 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 $i = 1, \dots, N$:

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

$O((n^2 + b_0)db_0n)$ 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 $O(db_0n \log(db_0n))$. 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 VI.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 (8) with $\epsilon_i(B_i)$ replaced by $\epsilon_i^\pi(B_i)$.*

VII. 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.

A. Datasets

In our experiments, we use two real-world datasets to evaluate our algorithms: (1) Pendigits dataset [44], which has 7,494 data points and 17 attributes; (2) MNIST handwritten digits dataset in a 784-dimensional space [45], where we use 60,000 data points for training and 10,000 data points for testing. We leverage the approach in [28] 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 [33].

B. ML Tasks

We consider four ML tasks: (1) minimum enclosing ball (MEB) [9]; (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.

C. Algorithms

For the centralized setting, we consider five different algorithms for comparison. The first two are EVD-MECB in Algorithm 1 and MD-MECB in Algorithm 3. The third algorithm is *EM* (seem Section IV-C3). The fourth algorithm aims to Maximize the Precision (*MP*), i.e., using $k = \lfloor \frac{B}{d \cdot b_0} \rfloor$ and $b = b_0$. The fifth algorithm aims to Maximize the Cardinality (*MC*), i.e., using $k = \min(n, \lfloor \frac{B}{d \cdot (1 + m_e)} \rfloor)$ and $b = \max(1 + m_e, \lfloor \frac{B}{d \cdot n} \rfloor)$.

For the distributed setting, we consider six algorithms for comparison. The first five algorithms are instances of OBA-MECBD in Algorithm 4 that use EVD, MD, 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 [28].

D. Performance Metrics

We use the *normalized ML cost* to measure the performance of unsupervised ML tasks. The normalized ML cost is defined as $\text{cost}(\mathcal{Y}, \mathbf{x}_S) / \text{cost}(\mathcal{Y}, \mathbf{x}^*)$, where \mathbf{x}_S is the model learned from coreset 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. All metrics are computed over 40 Monte Carlo runs unless stated otherwise.

E. Results in Centralized Setting

1) *Unsupervised Learning:* We first evaluate the unsupervised learning tasks: MEB, k -means, and PCA. We perform this evaluation on Pendigits. Figure 2 shows 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., $B = 163069$. We have the following observations: 1) Compared to the proposed algorithms, EM achieves a slightly better ML performance, but has a much higher running time. 2)

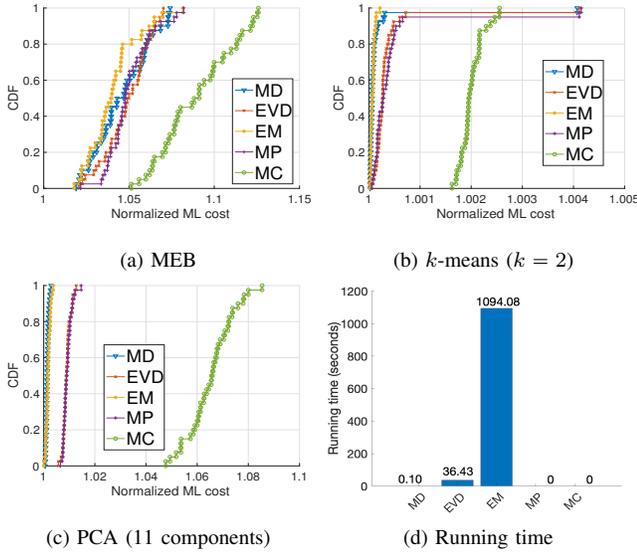


Fig. 2: Evaluation on Pendigits dataset (centralized setting).

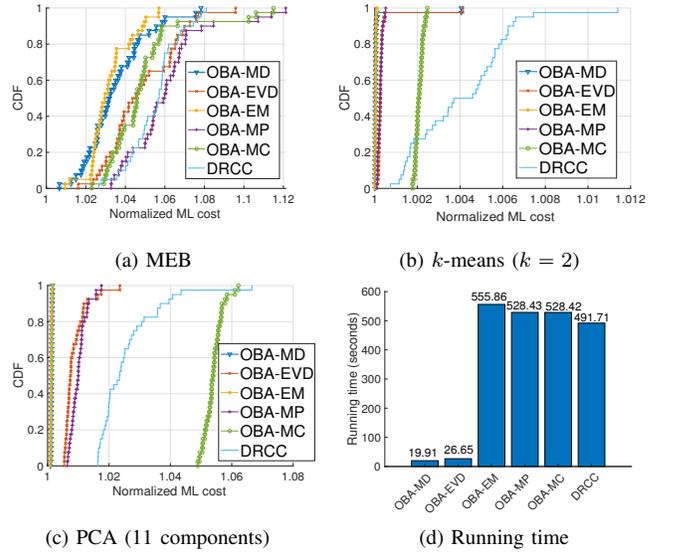


Fig. 4: Evaluation on Pendigits dataset (distributed setting).

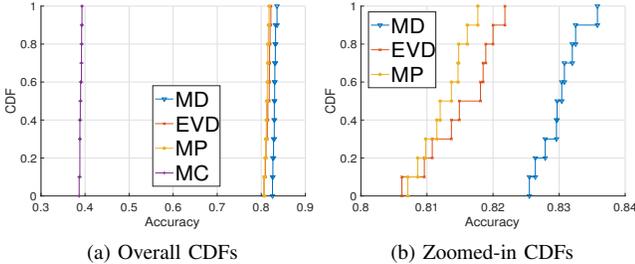


Fig. 3: Evaluation on MNIST (Neural Net Accuracies)

Compared to EVD-MECB, MD-MECB is not only faster, but also more closely approximates EM. 3) Compared with single operation, joint coreset construction and quantization achieves much better ML performance.

2) *Supervised Learning*: For supervised learning, we evaluate neural network based classification on the MNIST dataset. We do not evaluate EM here because of its high running time. Same as unsupervised learning, we only use 2% of the original data, i.e., $B = 60,211,200$. Figure 3 shows the CDFs of classification accuracy over 10 Monte Carlo runs. 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.

F. Results in Distributed Setting

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

We present the results for distributed setting in Figure 4, from which we have the following observations: 1) Most of the algorithms equipped with OBA outperform DRCC with a small degradation in the ML performance. 2) Our proposed OBA-EVD and OBA-MD which jointly optimize the operations of coreset construction and quantization perform significantly better than single operation. 3) OBA-MD is the most efficient over all these algorithms.

G. 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.

VIII. CONCLUSION

In this paper, we have proposed the first framework, MECB, to jointly configure coreset construction algorithms and quantizers in order to minimize the ML error under a given communication budget. We have proposed two algorithms to efficiently compute approximate solutions to the MECB problem. We have further proposed an algorithm to extend our solutions to the distributed setting. The effectiveness and efficiency of our algorithms have been demonstrated through experiments based on multiple real-world datasets.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. 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.

REFERENCES

- [1] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint arXiv:1610.05492*, 2016.
- [2] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, “Federated multi-task learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.
- [3] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, “When edge meets learning: Adaptive control for resource-constrained distributed machine learning,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 63–71.
- [4] J. M. Phillips, “Coresets and sketches,” *arXiv preprint arXiv:1601.00617*, 2016.
- [5] D. Feldman, M. Monemizadeh, C. Sohler, and D. P. Woodruff, “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, 2010, pp. 630–649.
- [6] T. Sarlos, “Improved approximation algorithms for large matrices via random projections,” in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06)*. IEEE, 2006, pp. 143–152.
- [7] M. Bădoiu, S. Har-Peled, and P. Indyk, “Approximate clustering via core-sets,” in *ACM STOC*, 2002.
- [8] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, “Geometric approximation via coresets,” *Combinatorial and computational geometry*, vol. 52, pp. 1–30, 2005.
- [9] K. L. Clarkson, “Coresets, sparse greedy approximation, and the frank-wolfe algorithm,” *ACM Transactions on Algorithms (TALG)*, vol. 6, no. 4, p. 63, 2010.
- [10] S. Har-Peled and S. Mazumdar, “On coresets for k-means and k-median clustering,” in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 2004, pp. 291–300.
- [11] D. Feldman and M. Langberg, “A unified framework for approximating and clustering data,” in *STOC*, June 2011.
- [12] M. F. Balcan, S. Ehrlich, and Y. Liang, “Distributed k-means and k-median clustering on general topologies,” in *NIPS*, 2013.
- [13] D. Feldman, M. Volkov, and D. Rus, “Dimensionality reduction of massive sparse datasets using coresets,” in *NIPS*, 2016.
- [14] M. Langberg and L. J. Schulman, “Universal ϵ approximators for integrals,” in *SODA*, 2010.
- [15] D. Feldman, M. Schmidt, and C. Sohler, “Turning big data into tiny data: Constant-size coresets for k-means, PCA, and projective clustering,” in *SODA*, 2013.
- [16] R. M. Gray and D. L. Neuhoff, “Quantization,” *IEEE transactions on information theory*, vol. 44, no. 6, pp. 2325–2383, 1998.
- [17] S. Lloyd, “Least squares quantization in pcm,” *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [18] K. Sayood, *Introduction to Data Compression, Fourth Edition*, 4th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.
- [19] T. Fischer, “A pyramid vector quantizer,” *IEEE transactions on information theory*, vol. 32, no. 4, pp. 568–583, 1986.
- [20] D. Goodman and R. Wilkinson, “A robust adaptive quantizer,” *IEEE Transactions on Communications*, vol. 23, no. 11, pp. 1362–1365, 1975.
- [21] N. Farvardin and V. Vaishampayan, “Optimal quantizer design for noisy channels: An approach to combined source-channel coding,” *IEEE Transactions on Information Theory*, vol. 33, no. 6, pp. 827–838, 1987.
- [22] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney, “Sampling algorithms and coresets for ℓ_p regression,” *SIAM Journal on Computing*, vol. 38, no. 5, pp. 2060–2078, 2009.
- [23] M. Lucic, M. Faulkner, A. Krause, and D. Feldman, “Training mixture models at scale via coresets,” *stat*, vol. 1050, p. 23, 2017.
- [24] K. Chen, “On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications,” *SIAM Journal on Computing*, vol. 39, no. 3, pp. 923–947, 2009.
- [25] D. Feldman, M. Feigin, and N. Sochen, “Learning big (image) data via coresets for dictionaries,” *Journal of mathematical imaging and vision*, vol. 46, no. 3, pp. 276–291, 2013.
- [26] C. Boutsidis, P. Drineas, and M. Magdon-Ismael, “Near-optimal coresets for least-squares regression,” *IEEE Trans. IT*, vol. 59, no. 10, pp. 6880–6892, October 2013.
- [27] S. Har-Peled and S. Mazumdar, “On coresets for k-means and k-median clustering,” in *STOC*, 2004.
- [28] H. Lu, M.-J. Li, T. He, S. Wang, V. Narayanan, and K. S. Chan, “Robust coreset construction for distributed machine learning,” 2019. [Online]. Available: <http://arxiv.org/abs/1904.05961>
- [29] K. L. Clarkson, “Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm,” *ACM Transactions on Algorithms*, vol. 6, no. 4, August 2010.
- [30] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” *arXiv preprint arXiv:1510.00149*, 2015.
- [31] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, “Incremental network quantization: Towards lossless cnns with low-precision weights,” *arXiv preprint arXiv:1702.03044*, 2017.
- [32] D. Lin, S. Talathi, and S. Annapureddy, “Fixed point quantization of deep convolutional networks,” in *International Conference on Machine Learning*, 2016, pp. 2849–2858.
- [33] IEEE, “754-2019 - ieee standard for floating-point arithmetic,” 2019. [Online]. Available: <https://ieeexplore.ieee.org/servlet/opac?punumber=8766227>
- [34] C. Ding and X. He, “K-means clustering via principal component analysis,” in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 29.
- [35] G. W. Stewart, “A krylov–schur algorithm for large eigenproblems,” *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 601–614, 2002.
- [36] A. Lim, B. Rodrigues, F. Wang, and Z. Xu, “k-center problems with minimum coverage,” *Theoretical Computer Science*, vol. 332, no. 1–3, pp. 1–17, 2005.
- [37] S. Khuller and Y. J. Sussmann, “The capacitated k-center problem,” *SIAM Journal on Discrete Mathematics*, vol. 13, no. 3, pp. 403–418, 2000.
- [38] S. Khuller, R. Pless, and Y. J. Sussmann, “Fault tolerant k-center problems,” *Theoretical Computer Science*, vol. 242, no. 1–2, pp. 237–245, 2000.
- [39] D. S. Hochbaum and D. B. Shmoys, “A best possible heuristic for the k-center problem,” *Mathematics of operations research*, vol. 10, no. 2, pp. 180–184, 1985.
- [40] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2006.
- [41] J. Demmel, I. Dumitriu, and O. Holtz, “Fast linear algebra is stable,” *Numerische Mathematik*, vol. 108, no. 1, pp. 59–91, 2007.
- [42] H. Luss, “A nonlinear minimax allocation problem with multiple knapsack constraints,” *Operations Research Letters*, vol. 10, no. 4, pp. 183–187, 1991.
- [43] —, “An algorithm for separable non-linear minimax problems,” *Operations Research Letters*, vol. 6, no. 4, pp. 159–162, 1987.
- [44] E. Alpaydin and F. Alimoglu, “Pen-based recognition of handwritten digits data set,” <https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits>, 1996. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits>
- [45] Y. LeCun, C. Cortes, and C. Burges, “The MNIST database of handwritten digits,” <http://yann.lecun.com/exdb/mnist/>, 1998. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>