A Novel Sequential Coreset Method for Gradient Descent Algorithms

Jiawei Huang [*1] **Ruomin Huang** [*2] **Wenjie Liu** [*1] **Nikolaos M. Freris** [1] **Hu Ding** [1]

Abstract

A wide range of optimization problems arising in machine learning can be solved by gradient descent algorithms, and a central question in this area is how to efficiently compress a large-scale dataset so as to reduce the computational complexity. *Coreset* is a popular data compression technique that has been extensively studied before. However, most of existing coreset methods are problem-dependent and cannot be used as a general tool for a broader range of applications. A key obstacle is that they often rely on the pseudo-dimension and total sensitivity bound that can be very high or hard to obtain. In this paper, based on the "locality" property of gradient descent algorithms, we propose a new framework, termed "sequential coreset", which effectively avoids these obstacles. Moreover, our method is particularly suitable for sparse optimization whence the coreset size can be further reduced to be only poly-logarithmically dependent on the dimension. In practice, the experimental results suggest that our method can save a large amount of running time compared with the baseline algorithms.

1. Introduction

Coreset (Feldman, 2020) is a popular technique for compressing large-scale datasets so as to speed up existing algorithms. Especially for the optimization problems arising in machine learning, coresets have been extensively studied in recent years. Roughly speaking, given a large dataset P and a specified optimization objective (*e.g.,* k-means clustering), the coreset approach is to construct a new dataset $\tilde{P}$ with the size $|\tilde{P}| \ll |P|$, such that any solution obtained over $\tilde{P}$ will approximately preserve the same quality over the original set P; that is, we can replace P by $\tilde{P}$ when running an available algorithm for solving this optimization problem. Because $|\tilde{P}| \ll |P|$, the runtime can be significantly reduced.

In this paper, we consider *Empirical Risk Minimization (ERM)* problems which capture a broad range of applications in machine learning (Vapnik, 1991). Let $\mathbb{X}$ and $\mathbb{Y}$ be the data space and response space, respectively. Given an input training set $P = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, where each $x_i \in \mathbb{X}$ and each $y_i \in \mathbb{Y}$, the objective is to learn the hypothesis β (from the hypothesis space $\mathbb{R}^d$) so as to minimize the *empirical risk*

$$F(\beta) = \frac{1}{n}\sum_{i=1}^{n} f(\beta, x_i, y_i), \tag{1}$$

where $f(\cdot,\cdot,\cdot)$ is the non-negative real-valued *loss function*. In practice, the data size n can be very large, thus it is instrumental to consider data compression methods (like coresets) to reduce the computational complexity.

Let $\epsilon \in [0, 1]$. A standard **ϵ-coreset** is represented as a vector $W = [w_1, w_2, \cdots, w_n] \in \mathbb{R}^n$ with the property that the function $\tilde{F}(\beta) = \frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i f(\beta, x_i, y_i)$ must satisfy

$$\tilde{F}(\beta) \in (1 \pm \epsilon)F(\beta),\ \forall \beta \in \mathbb{R}^d. \tag{2}$$

The number of non-zero entries $\|W\|_0$ is the coreset size, and thus the goal of compression is to have W to be as sparse as possible. Suppose we have an algorithm that can achieve c-approximation for the ERM problem ($c \geq 1$). Then, we can run the same algorithm on the ϵ-coreset, and let $\hat{\beta}$ be the returned c-approximation, *i.e.,* $\tilde{F}(\hat{\beta}) \leq c \cdot \min_{\beta \in \mathbb{R}^d} \tilde{F}(\beta)$. It holds that $F(\hat{\beta}) \leq c \cdot \frac{1+\epsilon}{1-\epsilon} \min_{\beta \in \mathbb{R}^d} F(\beta)$, thus the approximation ratio of $\hat{\beta}$ for the original objective function $F(\cdot)$ is only slightly worse than c when ϵ is sufficiently small.

A large part of coreset methods are based on the "sensitivity" idea (Langberg & Schulman, 2010). First, it computes a constant factor approximation with respect to the objective function (1); then it estimates the sensitivity σ_i for each data item (x_i, y_i) based on the obtained constant factor approximation; finally, it takes a random sample (as the coreset) over the input set P, where each data item (x_i, y_i) is selected with probability proportional to its sensitivity σ_i,

[*]Equal contribution [1]School of Computer Science and Technology, University of Science and Technology of China, Anhui, China. [2]School of Data Science, University of Science and Technology of China, Anhui, China. Correspondence to: Hu Ding <huding@ustc.edu.cn, http://staff.ustc.edu.cn/~huding/>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

and the total sample size depends on the total sensitivity bound $\sum_{i=1}^{n} \sigma_i$ along with the "pseudo-dimension" of the objective function (Feldman & Langberg, 2011; Li et al., 2001). This sensitivity-based coreset framework has been successfully applied to solve problems such as k-means clustering and projective clustering (Feldman & Langberg, 2011). However, there are several obstacles when trying to apply this approach to general ERM problems. For instance, it is not easy to obtain a constant factor approximation; moreover, different from clustering problems, it is usually challenging to achieve a reasonably low total sensitivity bound and compute the pseudo-dimension for many practical ERM problems. For example, the coreset size can be as large as $\tilde{\Omega}(d^2\sqrt{n}/\epsilon^2)$ for logistic regression (Tukan et al., 2020) with $O(nd^2)$ construction time.

Another common class of coreset construction methods is based on "greedy selection" (Coleman et al., 2020; Mirzasoleiman et al., 2020a). The greedy selection procedure is quite similar to the *k-center clustering* algorithm (Gonzalez, 1985) and the greedy *submodular set cover* algorithm (Wolsey, 1982). Intuitively, the method greedily selects a subset of the input training set, *i.e.,* the coreset, which are expected to be as diverse as possible; consequently, the whole training set can be covered by small balls centered at the selected subset. Nonetheless, this approach also suffers from several drawbacks. First, it is difficult to bound the size of the obtained coreset, when specifying the error bound induced by the coreset (*e.g.,* one may need too many balls to cover the training set if their radii are required to be no larger than an upper bound). Second, the time complexity can be too high, *e.g.,* the greedy k-center clustering procedure usually needs to read the input training set for a large number of passes, and the greedy submodular set cover algorithm usually needs a large number of function evaluations.

1.1. Our Contributions

The aforementioned issues seriously limit the applications of coresets in practice. In this paper, we propose a novel and easy-to-implement coreset framework, termed *sequential coreset*, for the general ERM problem (1). Our idea comes from a simple observation. For many ERM problems, either convex or non-convex, gradient descent algorithms are commonly invoked. In particular, these gradient descent algorithms usually share the following **locality property**:

Since the learning rate of a gradient descent algorithm is usually restricted by an upper bound, the trajectory of the hypothesis β in (1) is likely to be "smooth" (except for the first few rounds). That is, the change of β should be "small" between successive rounds.

This allows to focus, in each round, on a local region rather than the whole hypothesis space $\mathbb{R}^d$. We can thus visualize

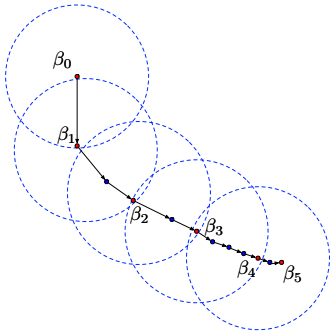

Figure 1. Suppose the trajectory starts from β_0, and we construct a "local" coreset within the ball centered at β_0; when the trajectory approaches the ball's boundary, *i.e.,* β_1, we update the coreset. Similarly, we update the coreset at β_2, β_3, and β_4, until sufficiently approximating the stationary point β_5. We can view β_1, β_2, β_3, and β_4 as a sequence of "anchors".

the trajectory to be decomposed into a sequence of "segments", where each segment is bounded by an individual ball. See Figure 1 for an illustration. When the trajectory enters a new ball (*i.e.,* a new local region), we construct a coreset $W = [w_1, w_2, \cdots, w_n]$ with

$$\tilde{F}(\beta) \in (1 \pm \epsilon)F(\beta), \forall \beta \in \text{the current local region}. \quad (3)$$

The formal definition of such a "local" coreset is shown in Section 2. When the trajectory approaches the boundary of the ball, we update the coreset for the next ball. Therefore, we call the method "sequential coreset".

Although building the coreset for a local region is easier than that for the global hypothesis space, there remain several technical challenges to resolve. Partly inspired by the layered sampling idea of (Chen, 2009; Ding & Wang, 2020), we can achieve a coreset of (3) where the coreset size depends on the range of the local region. In particular, our method enjoys several significant advantages compared with previous coreset methods:

- Our method is not problem-dependent and can be applied to any (convex or non-convex) ERM problem that uses gradient descent, under some mild assumptions. In fact, our method can be extended to apply to other iterative algorithms beyond gradient descent, such as subgradient descent and expectation maximization, as long as they satisfy the locality property.
- Our method can avoid to compute the total sensitivity bound and pseudo-dimension, thus it does not incur any complicated computations (*e.g.,* SVD) and has only linear construction time.
- For special cases of practical interest such as sparse optimization, the coreset size can be further reduced to be only poly-logarithmically dependent on the dimension.

1.2. Related Works

Gradient descent. Given a differentiable objective function, gradient descent is arguably the most common first-order iterative optimization algorithm for finding the optimal solution (Curry, 1944). A number of ERM models can be solved via gradient descent methods, such as Ridge regression (Tikhonov, 1998) and Logistic regression (Cramer, 2004). Note that though the objective function of the Lasso regression (Tibshirani, 1996) is not differentiable, several natural generalizations of the traditional gradient descent method, such as subgradient methods (Bertsekas, 2015) and proximal gradient methods (Mosci et al., 2010; Beck & Teboulle, 2009), have been developed and shown to perform well in practice. Several extensions of gradient descent have been also widely studied in recent years. For example, Nesterov introduced the acceleration technique for achieving faster gradient method (Nesterov, 1983). In view of the rapid development of deep learning and many other machine learning applications, stochastic gradient descent method and variants have played a central role in the field of large-scale machine learning, due to their scalability to very large, possibly distributed datasets (Bottou et al., 2018; Kingma & Ba, 2015; Duchi et al., 2011).

Coresets. Compared with other data compression approaches, an obvious advantage of coreset is that it is to be selected from the original input; that is, the obtained coreset can well preserve some favorable properties, such as sparsity and interpretability, in the input domain. In the past years, coreset techniques have been widely applied to many optimization problems, such as: clustering (Chen, 2009; Feldman & Langberg, 2011; Huang et al., 2018), logistic regression (Huggins et al., 2016; Munteanu et al., 2018; Samadian et al., 2020; Tukan et al., 2020; Samadian et al., 2020), Bayesian methods (Campbell & Broderick, 2018; Campbell & Beronov, 2019), linear regression (Dasgupta et al., 2009; Drineas et al., 2006; Chhaya et al., 2020; Kacham & Woodruff, 2020; Tukan et al., 2020), robust optimization (Ding & Wang, 2020), Gaussian mixture model (Lucic et al., 2017), and active learning (Coleman et al., 2020; Sener & Savarese, 2018). Recently, (Maalouf et al., 2019) also proposed the notion of "accurate" coresets, which do not introduce any approximation error when compressing the input dataset. Coresets are also applied to speed up large-scale or distributed machine learning algorithms (Reddi et al., 2015; Mirzasoleiman et al., 2020a;b; Borsos et al., 2020).

Very recently, (Raj et al., 2020) also considered the "local" heuristic for coresets. However, their results are quite different from ours. Their method still relies on the problem-dependent pseudo-dimension and the sensitivities; moreover, their method requires the objective function to be strongly convex.

Sketch. Another widely used data summarization method is "sketch" (Phillips, 2016). Different from coresets, sketch does not require to generate the summary from the original input. The sketch technique is particularly popular for solving linear regression problems (with and without regularization) (Avron et al., 2017; Chowdhury et al., 2018).

2. Preliminaries

Given an instance of the ERM problem (1), we assume that the loss function is Lipschitz smooth. This is a quite common assumption for analyzing many gradient descent methods (Wolfe, 1969).

Assumption 1 (Lipschitz Smoothness) *There exists a real constant $L > 0$, such that for any $1 \leq i \leq n$ and any β_1, β_2 in the hypothesis space, we have*

$$\|\nabla f(\beta_1, x_i, y_i) - \nabla f(\beta_2, x_i, y_i)\| \leq L\|\beta_1 - \beta_2\|, \quad (4)$$

where $\|\cdot\|$ is the Euclidean norm in the space.

For simplicity, we just use ball to define the local region for constructing our coreset as (3). Suppose we have the "anchor" $\beta_{\texttt{anc}} \in \mathbb{R}^d$ and region range $R \geq 0$. Let $\mathbb{B}(\beta_{\texttt{anc}}, R)$ denote the ball centered at $\beta_{\texttt{anc}}$ with radius R. Below, we provide the formal definition for the "local" coreset in (3).

Definition 1 (Local ϵ-Coreset) *Let $P = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ be an input dataset of the ERM problem (1). Suppose $\epsilon \in (0, 1)$. Given $\beta_{\texttt{anc}} \in \mathbb{R}^d$ and $R \geq 0$, the local ϵ-coreset, denoted $\mathcal{CS}_\epsilon(\beta_{\texttt{anc}}, R)$, is a vector $W = [w_1, w_2, \cdots, w_n]$ satisfying that*

$$\tilde{F}(\beta) \in (1 \pm \epsilon)F(\beta), \qquad \forall \beta \in \mathbb{B}(\beta_{\texttt{anc}}, R), \quad (5)$$

where $\tilde{F}(\beta) = \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^n w_i f(\beta, x_i, y_i)$. The number of non-zero entries of W is the size of $\mathcal{CS}_\epsilon(\beta_{\texttt{anc}}, R)$.

3. Local ϵ-Coreset Construction

We first present the construction algorithm for local ϵ-coreset, and expose the detailed analysis on its quality in Section 3.1. Besides the quality guarantee of (5), in Section 3.2 we show that our coreset can approximately preserve the gradient $\nabla F(\beta)$, which is an important property for gradient descent algorithms. In Section 3.3, we discuss some extensions beyond gradient descent. Relying on the local ϵ-coreset, we propose the sequential coreset framework and consider several important applications in Section 4.

Coreset construction. Let $N = \lceil \log n \rceil$ (the basis of the logarithm is 2 in this paper). Given the central point $\beta_{\texttt{anc}} \in \mathbb{R}^d$ and the local region range (*i.e.,* the radius) $R \geq 0$, we set $H = F(\beta_{\texttt{anc}})$ and then partition the input dataset

Algorithm 1 Local ϵ-Coreset Construction

Input: A training dataset $P = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, the Lipschitz constant L as described in Assumption 1, $\beta_{\texttt{anc}} \in \mathbb{R}^d$, and the parameters $R \geq 0$ and $\epsilon \in (0, 1)$.

1. Let $N = \lceil \log n \rceil$ and $H = F(\beta_{\texttt{anc}})$; initialize $W = [0, 0, \cdots, 0] \in \mathbb{R}^n$.
2. The set P is partitioned into $N + 1$ layers $\{P_0, \ldots, P_N\}$ as in (6) and (7).
3. For each $P_j \neq \emptyset$, $0 \leq j \leq N$:
   (a) take a random sample Q_j from P_j uniformly at random, where the size $|Q_j|$ depends on the parameters ϵ, R, and L (the exact value will be discussed in our following analysis in Section 3.1);
   (b) for each sampled data item $(x_i, y_i) \in Q_j$, assign the weight $w_i = \frac{|P_j|}{|Q_j|}$;

Output: the weight vector $W = [w_1, w_2, \cdots, w_n]$ as the coreset.

$P = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ into $N + 1$ layers:

$$P_0 = \big\{(x_i, y_i) \in P \mid f(\beta_{\texttt{anc}}, x_i, y_i) \leq H\big\}, \quad (6)$$

$$P_j = \big\{(x_i, y_i) \in P \mid 2^{j-1}H < f(\beta_{\texttt{anc}}, x_i, y_i) \leq 2^j H\big\}, 1 \leq j \leq N. \quad (7)$$

It is easy to see that $P = \cup_{j=0}^{N} P_j$, since $f(\beta_{\texttt{anc}}, x_i, y_i)$ is always no larger than $2^N H$ for any $1 \leq i \leq n$. For each $0 \leq j \leq N$, if $P_j \neq \emptyset$, we take a random sample Q_j from P_j uniformly at random, where the size $|Q_j|$ will be determined in our following analysis (in Section 3.1); for each sampled data item $(x_i, y_i) \in Q_j$, we assign the weight w_i to be $\frac{|P_j|}{|Q_j|}$; for all the data items of $P_j \setminus Q_j$, we let their weights to be 0. At the end, we obtain the weight vector $W = [w_1, w_2, \cdots, w_n]$ as our coreset, and consequently $\tilde{F}(\beta) = \frac{1}{n}\sum_{i=1}^{n} w_i f(\beta, x_i, y_i)$ (it is easy to verify $\sum_{i=1}^{n} w_i = n$ from our construction). The construction procedure is shown in Algorithm 1.

Remark 1 *Our layered sampling procedure in Algorithm 1 is similar to the coreset construction idea of (Chen, 2009; Ding & Wang, 2020), which was originally designed for the k-median/means clustering problems. Compared with the sensitivity based coreset construction idea (Langberg & Schulman, 2010), a significant advantage of our method is that there is no need to compute the total sensitivity bound and pseudo-dimension. These values are problem-dependent and, for some objectives, they can be very high or hard to obtain (Munteanu et al., 2018; Tukan et al., 2020).*

3.1. Theoretical Analysis

In this section, we prove the quality guarantee and complexity of the coreset returned from Algorithm 1. We define two values before presenting our theorem, $M := \max_{1 \leq i \leq n} \|\nabla f(\beta_{\texttt{anc}}, x_i, y_i)\|$ and $m := \min_{\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)} F(\beta)$.

Theorem 1 *With probability $1 - \frac{1}{n}$, Algorithm 1 returns a qualified coreset $\mathcal{CS}_\epsilon(\beta_{\texttt{anc}}, R)$ with size $\tilde{O}\left(\left(\frac{H+MR+LR^2}{m}\right)^2 \cdot \frac{d}{\epsilon^2}\right)$[1]. Furthermore, when the vector β is restricted to have at most $k \in \mathbb{Z}^+$ non-zero entries in the hypothesis space $\mathbb{R}^d$, the coreset size can be reduced to be $\tilde{O}\left(\left(\frac{H+MR+LR^2}{m}\right)^2 \cdot \frac{k \log d}{\epsilon^2}\right)$. The runtime of Algorithm 1 is $O(n \cdot t_f)$, where t_f is the time complexity for computing the loss $f(\beta, x, y)$.*

Remark 2 *From Theorem 1 we can see that the coreset size depends on the initial vector $\beta_{\texttt{anc}}$ and the local region range R. Also note that the value m is non-increasing with R.*

First, the linear time complexity of Algorithm 1 is easy to see: to obtain the partition and the samples, it just needs to compute $f(\beta_{\texttt{anc}}, x_i, y_i)$ for $1 \leq i \leq n$. Below, we focus on proving the quality guarantee and coreset size. For the sake of simplicity, we use $f_i(\beta)$ to denote $f(\beta, x_i, y_i)$ in our analysis. By using Taylor expansion and Assumption 1, we directly have

$$f_i(\beta) \in f_i(\beta_{\texttt{anc}}) \pm \left(\frac{\|\nabla f_i(\beta_{\texttt{anc}})\|}{R} + \frac{L}{2}\right) R^2. \quad (8)$$

for any $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$ and $1 \leq i \leq n$. Then we have the following lemma.

Lemma 1 *We fix a vector $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$ and an index j from $\{0, 1, \ldots, N\}$. Given any two numbers $\lambda \in (0, 1)$ and $\delta > 0$, if we set the sample size in Step 3(a) of Algorithm 1 to be*

$$|Q_j| = O\left((2^{j-1}H + MR + LR^2)^2 \delta^{-2} \log \frac{1}{\lambda}\right), \quad (9)$$

we have

$$\texttt{Prob}\left[\left|\frac{1}{|Q_j|}\sum_{(x_i,y_i)\in Q_j} f_i(\beta) - \frac{1}{|P_j|}\sum_{(x_i,y_i)\in P_j} f_i(\beta)\right| \geq \delta\right] \leq \lambda.$$

Proof. For a fixed $1 \leq j \leq N$, we view $f_i(\beta)$ as an independent random variable for each $(x_i, y_i) \in P_j$. Through

[1] $\tilde{O}(g) := O\left(g \cdot \texttt{polylog}\left(\frac{nHMR}{\epsilon m}\right)\right)$.

the partition construction (6) and (7), and the bounds (8), we have

$$\left.\begin{aligned} f_i(\beta) \geq \quad & 2^{j-1}H - MR - \frac{1}{2}LR^2; \\ f_i(\beta) \leq \quad & 2^{j}H + MR + \frac{1}{2}LR^2. \end{aligned}\right\} \tag{10}$$

Let the sample size $|Q_j| = \lceil \frac{1}{2}(2^{j-1}H + LR^2 + 2MR)^2\delta^{-2}\ln\frac{2}{\lambda}\rceil$. Through the Hoeffding's inequality (Hoeffding, 1994), we know that

$$\texttt{Prob}\left[\left|\frac{1}{|Q_j|}\sum_{(x_i,y_i)\in Q_j} f_i(\beta) - \frac{1}{|P_j|}\sum_{(x_i,y_i)\in P_j} f_i(\beta)\right| \geq \delta\right]$$

is no larger than $2e^{-\frac{2|Q_j|\delta^2}{(2^{j-1}H+LR^2+2MR)^2}} \leq \lambda$.

Now we consider the case $j = 0$. For any data item $(x_i, y_i) \in P_0$, we have $0 \leq f_i(\beta) \leq H + MR + \frac{1}{2}LR^2$. If letting the sample size $|Q_0| = \lceil \frac{1}{2}(H + \frac{1}{2}LR^2 + MR)^2\delta^{-2}\ln\frac{2}{\lambda}\rceil$, it is easy to verify that the same probability bound also holds. □

After proving Lemma 1, we further show $\tilde{F}(\beta) \approx F(\beta)$ for any fixed $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$.

Lemma 2 *Suppose $\epsilon_1 \geq 0$. In Lemma 1, if we set $\delta = \epsilon_1 2^{j-1}H$ for $j = 0, 1, \cdots, N$, then for any fixed $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$,*

$$\left|\tilde{F}(\beta) - F(\beta)\right| \leq \frac{3}{2}\epsilon_1 F(\beta_{\texttt{anc}}) \tag{11}$$

holds with probability at least $1 - (N+1)\lambda$.

Proof. From Lemma 1, it holds that the probability that

$$\left|\frac{|P_j|}{|Q_j|}\sum_{(x_i,y_i)\in Q_j} f_i(\beta) - \sum_{(x_i,y_i)\in P_j} f_i(\beta)\right| \geq |P_j|\cdot\epsilon_1 2^{j-1}H \tag{12}$$

is at most λ. Recall $\tilde{F}(\beta) = \frac{1}{n}\sum_{i=1}^{n} w_i f(\beta, x_i, y_i)$, where for each $(x_i, y_i) \in P_j$, $w_i = \frac{|P_j|}{|Q_j|}$ if $(x_i, y_i) \in Q_j$, and $w_i = 0$ if $(x_i, y_i) \in P_j \setminus Q_j$. Thus, by taking the union bound of (12) over $0 \leq j \leq N$, we have

$$\begin{aligned} & n\left|\tilde{F}(\beta) - F(\beta)\right| \\ = & \left|\sum_{j=0}^{N}\frac{|P_j|}{|Q_j|}\sum_{(x_i,y_i)\in Q_j} f_i(\beta) - \sum_{j=0}^{N}\sum_{(x_i,y_i)\in P_j} f_i(\beta)\right| \\ \leq & \sum_{j=0}^{N}\left|\frac{|P_j|}{|Q_j|}\sum_{(x_i,y_i)\in Q_j} f_i(\beta) - \sum_{(x_i,y_i)\in P_j} f_i(\beta)\right| \\ \leq & \sum_{j=0}^{N}|P_j|\epsilon_1 2^{j-1}H \end{aligned} \tag{13}$$

with probability at least $(1-\lambda)^{N+1} > 1 - (N+1)\lambda$. To complete the proof, we also need the following claim.

Claim 1 $\sum_{j=0}^{N}|P_j|2^j \leq 3n$.

Proof. By the definition of P_j, we have

$$\begin{aligned} & 2^j H = H, && \text{if } j = 0; \\ & 2^j H \leq 2f_i(\beta_{\texttt{anc}}), \forall (x_i, y_i) \in P_j, && \text{if } j \geq 1. \end{aligned} \tag{14}$$

Therefore, $2^j H$ is always no larger than $2f_i(\beta_{\texttt{anc}}) + H$ for any $0 \leq j \leq N$ and any $(x_i, y_i) \in P_j$. Overall,

$$\begin{aligned} \sum_{j=0}^{N}|P_j|2^j H &= \sum_{j=0}^{N}\sum_{(x_i,y_i)\in P_j} 2^j H \\ &\leq \sum_{j=0}^{N}\sum_{(x_i,y_i)\in P_j}(2f_i(\beta_{\texttt{anc}}) + H) \\ &= 2nF(\beta_{\texttt{anc}}) + nH = 3nH. \end{aligned} \tag{15}$$

Thus the claim $\sum_{j=0}^{N}|P_j|2^j \leq 3n$ is true. □

By using Claim 1, (13) can be rewritten as

$$n\left|\tilde{F}(\beta) - F(\beta)\right| \leq \frac{3}{2}\epsilon_1 nF(\beta_{\texttt{anc}}). \tag{16}$$

So we complete the proof. □

To prove $\tilde{F}(\beta)$ is a qualified coreset, we need to extend Lemma 2 to any $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$. For this purpose, we discretize the region $\mathbb{B}(\beta_{\texttt{anc}}, R)$ first (the discretization is only used for our analysis, and we do not need to build the grid in reality). Imagine that we build a uniform grid inside $\mathbb{B}(\beta_{\texttt{anc}}, R)$ with the side length being equal to $\frac{\epsilon_2 R}{\sqrt{d}}$, where the exact value of ϵ_2 is to be determined later. Inside each grid cell of $\mathbb{B}(\beta_{\texttt{anc}}, R)$, we pick an arbitrary point as its representative point and let G be the set consisting of all the representative points. Based on the formula of the volume of a ball in $\mathbb{R}^d$, we have

$$|G| = O\left(\left(\frac{2\sqrt{\pi e}}{\epsilon_2}\right)^d\right). \tag{17}$$

So we can simply increase the sample size of Lemma 2, and take the union bound over all $\beta \in G$ so as to extend the result as follows.

Lemma 3 *Suppose $\epsilon_1 \geq 0$. In the sample size (9) of Lemma 1, we set $\delta = \epsilon_1 2^{j-1}H$ for $j = 0, 1, \cdots, N$, respectively, and replace λ by $\frac{\lambda}{(N+1)|G|}$. The following*

$$\left|\tilde{F}(\beta) - F(\beta)\right| \leq \frac{3}{2}\epsilon_1 F(\beta_{\texttt{anc}}) \tag{18}$$

holds for any $\beta \in G$, with probability at least $1 - \lambda$.

Following Lemma 3, we further derive a uniform bound over all $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$ (not just in G). For any $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$, we let $\beta' \in G$ be the representative point of the cell containing β. Then we have $\|\beta - \beta'\| \leq \epsilon_2 R$. We define $M' := \max_{1\leq i\leq n} \max_{\beta\in\mathcal{B}(\beta_{\texttt{anc}},R)} \|\nabla f(\beta, x_i, y_i)\|$. By Assumption 1 we immediately know $M' \leq M + LR$. By using the similar manner of (8), for any $1 \leq i \leq n$ we have

$$\left|f_i(\beta) - f_i(\beta')\right| \leq \epsilon_2 M' R + \frac{1}{2} L \epsilon_2^2 R^2. \tag{19}$$

This implies both

$$\begin{aligned} &|F(\beta) - F(\beta')| \text{ and } |\tilde{F}(\beta) - \tilde{F}(\beta')| \\ &\leq \epsilon_2 M' R + \frac{1}{2} L \epsilon_2^2 R^2. \end{aligned} \tag{20}$$

Using triangle inequality, we obtain

$$\begin{aligned} & |\tilde{F}(\beta) - F(\beta)| \\ \leq\ & |\tilde{F}(\beta) - \tilde{F}(\beta')| + |\tilde{F}(\beta') - F(\beta')| \\ & + |F(\beta') - F(\beta)| \\ \leq\ & \frac{3}{2}\epsilon_1 F(\beta_{\texttt{anc}}) + 2 \times (\epsilon_2 M' R + \frac{1}{2} L \epsilon_2^2 R^2), \end{aligned} \tag{21}$$

where the last inequality follows from Lemma 3 (note $\beta' \in G$) and (20). By letting $\epsilon_1 = \frac{2m\epsilon}{7F(\beta_{\texttt{anc}})}$ and $\epsilon_2 = \frac{2\epsilon_1 F(\beta_{\texttt{anc}})}{R\left(\sqrt{M'^2 + 2L\epsilon_1 F(\beta_{\texttt{anc}})} + M'\right)}$, we have $|\tilde{F}(\beta) - F(\beta)| \leq \epsilon F(\beta)$ via simple calculations. That is, the returned vector $W = [w_1, w_2, \cdots, w_n]$ is a qualified coreset $\mathcal{CS}_\epsilon(\beta_{\texttt{anc}}, R)$.

Last, it remains to specify the obtained coreset size. To guarantee the success probability to be at least $1 - 1/n$, we set $\lambda = 1/n$. Then we can compute the coreset size, *i.e.,* the number of non-zero entries of W, which equals

$$\sum_{j=0}^{N} |Q_j| = \tilde{O}\left(\left(\frac{H + MR + LR^2}{m}\right)^2 \cdot \frac{d}{\epsilon^2}\right) \tag{22}$$

(by combining (9), with the selection of δ in Lemma 2, the choice of λ in Lemma 3 along with (17), and the definition of ϵ_1).

For the case that β is restricted to have at most k non-zero entries (*i.e.,* sparse optimization) , we revisit the size $|G|$ in (17). For a d-dimensional vector, there are $\binom{d}{k}$ different combinations for the positions of the k non-zero entries. Thus β can be only located in the union of $\binom{d}{k}$ k-dimensional subspaces (similar idea was also used for analyzing compressed sensing (Baraniuk et al., 2006)). In other words, we just need to build the grid (only for the sake of analysis) in the union of $\binom{d}{k}$ k-dimensional balls instead of the whole $\mathbb{B}(\beta_{\texttt{anc}}, R)$. Consequently, the new size $|G|$ is $O\left(\binom{d}{k} \cdot \left(\frac{2\sqrt{\pi e}}{\epsilon_2}\right)^k\right)$, and the coreset size is reduced to $\tilde{O}\left(\left(\frac{H+MR+LR^2}{m}\right)^2 \cdot \frac{k \log d}{\epsilon^2}\right)$.

3.2. Gradient Preservation

Besides the quality guarantee (5), our local coreset also enjoys another favorable property. In this section, we show that the gradient $\nabla\tilde{F}(\beta)$ can be approximately preserved as well, *i.e.,* $\nabla\tilde{F}(\beta) \approx \nabla F(\beta)$ for any $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$. Because the trajectory of β is guided by the gradients, this property gives a hint that our eventually obtained β is likely to be close to the optimal hypothesis β^* (we also validate this property in our experiments). In some scenarios like statistical inference and parameter estimation, we expect to achieve not only an almost minimal loss $F(\beta)$, but also a small difference between β and β^*.

Given a vector $v \in \mathbb{R}^d$, we use $v_{[l]}$ to denote its l-th coordinate value, for $l = 1, 2, \ldots, d$. Under Assumption 1, we obtain (similar with (8)), for any $1 \leq i \leq n$,

$$\nabla f_i(\beta)_{[l]} \in \nabla f_i(\beta_{\texttt{anc}})_{[l]} \pm LR. \tag{23}$$

We can apply a similar line of analysis as in Section 3.1 to obtain Theorem 2. We need the following modifications. First, we need to change the sample size Q_j (and similarly the total coreset size in (22)) of Algorithm 1 because we now consider a different objective. Also, we achieve an additive error for the gradient, instead of the $(1 \pm \epsilon)$-multiplicative error as (5). The reason is that the gradient can be almost equal to 0, if the solution approaches to a local or global optimum (but the objective value (1) is usually not equal to 0, *e.g.,* we often add a non-zero penalty item to the objective function).

Theorem 2 *Let $\sigma > 0$ be any given small number. With probability $1 - \frac{1}{n}$, Algorithm 1 can return a vector W with $\tilde{O}\left(\frac{L^2R^2}{\sigma^2} \cdot d\right)$ non-zero entries, such that for any $\beta \in \mathbb{B}(\beta_{\texttt{anc}}, R)$ and $1 \leq l \leq d$,*

$$\nabla\tilde{F}(\beta)_{[l]} \in \nabla F(\beta)_{[l]} \pm \sigma. \tag{24}$$

Furthermore, if the vector β is restricted to have at most $k \in \mathbb{Z}^+$ non-zero entries in the hypothesis space $\mathbb{R}^d$, the number of non-zero entries of W can be reduced to be $\tilde{O}\left(\frac{L^2R^2}{\sigma^2} \cdot k \log d\right)$.

Remark 3 *If we want to guarantee both Theorem 1 and 2, we can just set the coreset size as the maximum over both cases.*

3.3. Beyond Gradient Descent

In Section 3.1, our analysis relied on the fact that the function $f(\beta, x_i, y_i)$ is differentiable. However, for some ERM problems, the loss function can be non-differentiable. A representative example is the l_1-norm regularized regression, such as (Tibshirani, 1996; Lee et al., 2006). We consider the l_p regularized regression with $0 < p \leq 2$. Given a

regularization parameter $\lambda > 0$, the objective function can be written as

$$F(\beta) = \frac{1}{n}\sum_{i=1}^{n} g(\beta, x_i, y_i) + \lambda\|\beta\|_p, \tag{25}$$

where the function $g(\beta, x_i, y_i)$ is assumed to be differentiable and satisfy Assumption 1. We can easily cast (25) to have the form of (1) by setting $f(\beta, x_i, y_i) = g(\beta, x_i, y_i) + \lambda\|\beta\|_p$.

First, we note that problem (25) is usually solved by generalizations of gradient descent method, such as subgradient methods (Bertsekas, 2015) and proximal gradient methods (Mosci et al., 2010). The key point is that these algorithms also enjoy the locality property described in Section 1.1. Thus, a natural question arises whether we can build a local coreset for (25) as well.

We answer this question in the affirmative. In (8), we provide the upper and lower bounds of $f_i(\beta)$ (*i.e.,* $f(\beta, x_i, y_i)$) for the non-differentiable case (25). For $0 < p \leq 2$, by using the Hölder's inequality we obtain the similar bounds for non-differentiable case : $f_i(\beta) \in f_i(\beta_{\texttt{anc}}) \pm \left(\left(\frac{\|\nabla g_i(\beta_{\texttt{anc}})\|}{R} + \frac{L}{2}\right)R^2 + \frac{\lambda d^{1/p-1/2}}{n}R\right)$. After replacing (8) by these bounds, we can proceed the same analysis in Section 3.1 and attain a similar result with Theorem 1.

4. Sequential Coreset Framework and Applications

The local ϵ-coreset constructed in Section 3 can be directly used for compressing input data. However, the trajectory of the hypothesis β (although enjoying the locality property) may span a relatively large range globally in the space. As discussed in Remark 2, the coreset size depends on the prespecified local region range. Therefore, the coreset size can be high, if we want to build in one shot a local coreset that covers the whole trajectory. This motivates us to propose the **sequential coreset framework** (see Algorithm 2).

In each round of Algorithm 2, we build the local coreset $\mathcal{CS}_\epsilon(\beta_t, R)$ and run the "host" algorithm $\mathcal{A}$ on it until either (i) the result becomes stable inside $\mathbb{B}(\beta_t, R)$ or (ii) the hypothesis β reaches the boundary of $\mathbb{B}(\beta_t, R)$[2]. For (i), we just terminate the algorithm and output the result; for (ii), we update β_t and proceed the next iteration.

Following the sequential coreset framework, we consider its applications for several ERM problems in machine learning.

Ridge regression. In the original linear regression problem, the data space $\mathbb{X} = \mathbb{R}^d$ and the response space $\mathbb{Y} = \mathbb{R}$, and the goal is to find a vector $\beta \in \mathbb{R}^d$ such that the objective function $F(\beta) = \frac{1}{n}\sum_{i=1}^{n} |\langle x_i, \beta\rangle - y_i|^2$ is minimized. For Ridge regression (Tikhonov, 1998), we add a squared l_2-norm penalty and the objective function becomes

$$F(\beta) = \frac{1}{n}\sum_{i=1}^{n} |\langle x_i, \beta\rangle - y_i|^2 + \lambda\|\beta\|_2^2, \tag{26}$$

where $\lambda > 0$ is a regularization parameter. Consequently, the loss function $f(\beta, x_i, y_i)$ of (26) is taken as $|\langle x_i, \beta\rangle - y_i|^2 + \lambda\|\beta\|_2^2$.

Lasso regression. Another popular regularized regression model is Lasso (Tibshirani, 1996). Compared to (26), the only difference is that we use an l_1-norm penalty *i.e.,*

$$F(\beta) = \frac{1}{n}\sum_{i=1}^{n} |\langle x_i, \beta\rangle - y_i|^2 + \lambda\|\beta\|_1, \tag{27}$$

where $\lambda > 0$ is a regularization parameter. The loss function $f(\beta, x_i, y_i)$ of (27) is $|\langle x_i, \beta\rangle - y_i|^2 + \lambda\|\beta\|_1$. A key advantage of Lasso is that the returned β is a sparse vector. The objective function (27) is not differentiable, but it can still be solved by our sequential coreset framework as discussed in Section 3.3.

Logistic regression. For Logistic regression, the response is binary, *i.e.,* $y_i = 0$ or 1 (Cramer, 2004). The objective function

$$F(\beta) = -\frac{1}{n}\sum_{i=1}^{n}\Big\{y_i \log g(\langle x_i, \beta\rangle) + (1 - y_i)\log\big(1 - g(\langle x_i, \beta\rangle)\big)\Big\}, \tag{28}$$

where $g(t) := \frac{1}{1+e^{-t}}$ (the logistic function). We may add an l_1 or l_2-norm penalty to (28), in the same way as (26) and (27). The loss function $f(\beta, x_i, y_i)$ for Logistic regression is $-y_i \log g(\langle x_i, \beta\rangle) - (1 - y_i)\log\big(1 - g(\langle x_i, \beta\rangle)\big)$.

Gaussian Mixture Model (GMM). As emphasized before, our local coreset method does not require the objective function to be convex. Here, we consider a typical

Algorithm 2 Sequential Coreset Framework

Input: An instance $P = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ of the ERM problem (1) with the initial solution β_0 and range $R > 0$, an available gradient descent algorithm $\mathcal{A}$ as the "host", and the parameter $\epsilon \in (0, 1)$.

1. For $t = 0, 1, \ldots$, build the local coreset $\mathcal{CS}_\epsilon(\beta_t, R)$ and run the host algorithm $\mathcal{A}$ on it until:
   (a) if the result becomes stable inside $\mathbb{B}(\beta_t, R)$, terminate the loop and return the current β;
   (b) else, the current β reaches the boundary of $\mathbb{B}(\beta_t, R)$, and then set $\beta_{t+1} = \beta$ and $t = t + 1$.

[2] In practice, we can set a small number $\sigma \in (0, 1)$ and deduce that the boundary is reached when $\|\beta_t - \beta\| > (1 - \sigma)R$.

non-convex example: GMM training (Bishop, 2006). A mixture of k Gaussian kernels is represented with $\beta := [(\omega_1, \mu_1, \Sigma_1), \ldots, (\omega_k, \mu_k, \Sigma_k)]$, where $\omega_1, \ldots, \omega_k \geq 0$, $\sum_{j=1}^{k} \omega_j = 1$, and each (μ_j, Σ_j) is the mean and covariance matrix of the j-th Gaussian in $\mathbb{R}^D$. GMM is an unsupervised learning problem, where the training dataset contains $\{x_1, \cdots, x_n\} \subset \mathbb{R}^D$, and the goal is to minimize the objective function

$$F(\beta) = -\frac{1}{n} \sum_{i=1}^{n} \log \Big(\sum_{j=1}^{k} \omega_j \mathcal{N}(x_i, \mu_j, \Sigma_j) \Big), \qquad (29)$$

where $\mathcal{N}(x_i, \mu_j, \Sigma_j)$ is $\frac{1}{\sqrt{(2\pi)^D |\Sigma_j|}} \exp(-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j))$; so $f(x_i, \beta) = -\log \Big(\sum_{j=1}^{k} \omega_j \mathcal{N}(x_i, \mu_j, \Sigma_j) \Big)$ for (29). It is worth noting that (29) is differentiable and Lipschitz smooth and thus can be solved via the gradient descent method. However, the expectation-maximization (EM) method is more popular due to its simplicity and efficiency for GMM training. Moreover, the EM method also has the locality property in practice. In our experiment, we still use Algorithm 2 to generate the sequential coreset, but run the EM algorithm as the "host" algorithm $\mathcal{A}$.

5. Experimental Evaluation

We evaluate the performance of our sequential coreset method for the applications mentioned in Section 4. All results were obtained on a server equipped with 2.4GHz Intel CPUs and 256GB main memory; the algorithms were implemented in Python. We consider Ridge and Lasso regression first. APPLIANCES ENERGY is a dataset for predicting energy consumption which contains 19735 points in $\mathbb{R}^{29}$ (Candanedo et al., 2017). FACEBOOK COMMENT is a dataset for predicting comment which contains 602813 points in $\mathbb{R}^{54}$ (Singh et al., 2015). Furthermore, we generate a synthetic dataset of 10^6 points in $\mathbb{R}^{50}$; each point is randomly sampled from the linear equation $y = \langle h, x \rangle$, where each coefficient of h is sampled from $[-5, 5]$ uniformly at random; for each data point we also add a Gaussian noise $\mathcal{N}(0, 4)$ to y.

Compared methods. As the host algorithm $\mathcal{A}$ in Algorithm 2, we apply the standard gradient descent algorithm. Fixing a coreset size, we consider several different data compression methods for comparison. (1) ORIGINAL: directly run $\mathcal{A}$ on the original input data; (2) UNISAMP: the simple uniform sampling; (3) IMPSAMP: the importance sampling method (Tukan et al., 2020); (4) SEQCORE-R: our sequential coreset method with a specified region range R; (5) ONESHOT: build the local coreset as Algorithm 1 in one-shot (without using the sequential idea)[3].

[3]For ONESHOT, we do not need to specify the range R, if we fix the coreset size. The range is only used for our sequential coreset method because we need to re-build the coreset when β reaches the boundary.

Results. We consider three metrics to measure the performance: (1) the total loss, (2) the normalized error to the optimal β^* (let $\texttt{Error}_\beta = \frac{\|\beta - \beta^*\|_2}{\|\beta^*\|_2}$ where β is the obtained solution and β^* is the optimal solution obtained from ORIGINAL), and (3) the normalized runtime (over the runtime of ORIGINAL). The results of Ridge regression are shown in Figures 2, 3 and 4 (averaged across 10 trials). We can see that in general our proposed sequential coreset method has better performance on the loss and $\texttt{Error}_\beta$, though sometimes it is slightly slower than IMPSAMP if we set R to be too small. UNISAMP is always the fastest one (because it is just simple uniform sampling), but at the cost of inferior performance in total loss and model estimate error. ONESHOT is faster than SEQCORE-R but often has worse loss and error. Similar results of Lasso regression are shown in Figure 5 and 6. Due to the space limit, more detailed experimental results (including the results on Logistic regression and GMM) are shown in our full paper.

6. Conclusions and Future Work

Based on the simple observation of the locality property, we propose a novel sequential coreset framework for reducing the complexity of gradient descent algorithms and some relevant variants. Our framework is easy to implement and has provable quality guarantees. Due to the space limit, we place some omitted proofs and more experimental results to our full paper. Following this work, it is interesting to consider building coresets for other optimization methods, such as the popular stochastic gradient descent method as well as second order methods.

7. Acknowledgements

The authors would like to thank Mingyue Wang and the anonymous reviewers for their helpful discussions and suggestions on improving this paper. This work was supported in part by the Ministry of Science and Technology of China through grant 2019YFB2102200, the Anhui Dept. of Science and Technology through grant 201903a05020049, and Tencent Holdings Ltd through grant FR202003.

References

Avron, H., Clarkson, K. L., and Woodruff, D. P. Sharper bounds for regularized data fitting. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017*, volume 81, pp. 27:1–27:22, 2017.

Baraniuk, R., Davenport, M., DeVore, R., and Wakin, M.

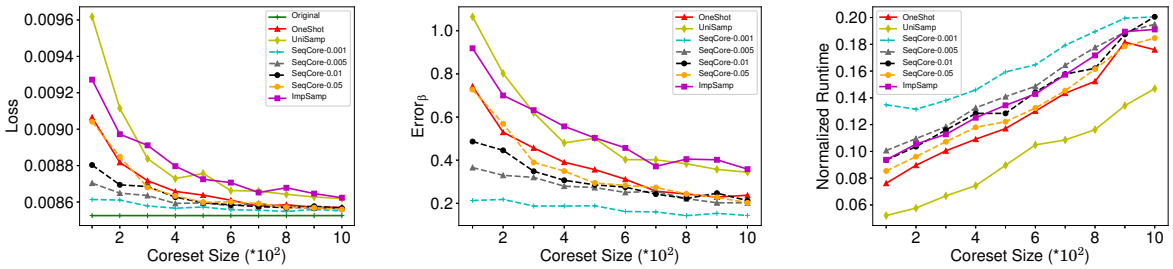

Figure 2. The experimental results on APPLIANCES ENERGY for Ridge regression ($\lambda = 0.01$).

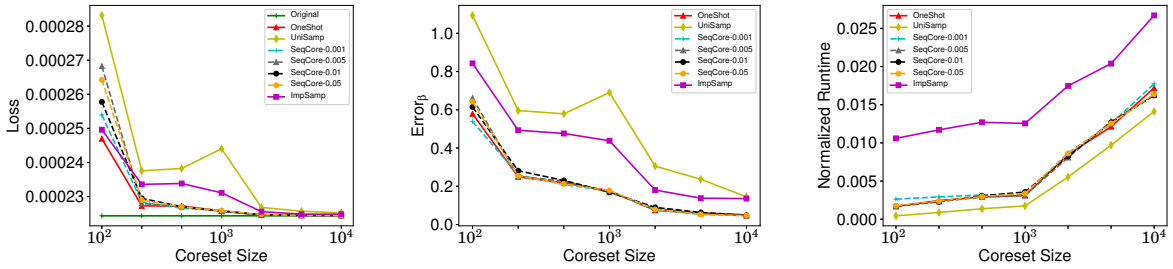

Figure 3. The experimental results on FACEBOOK COMMENT for Ridge regression ($\lambda = 0.01$).

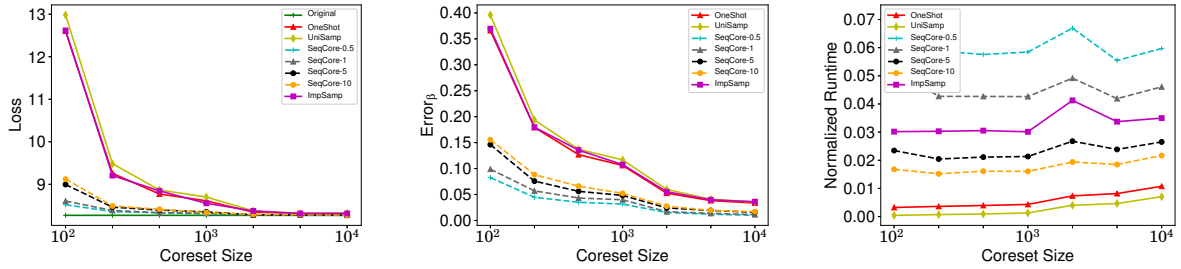

Figure 4. The experimental results on the synthetic dataset for Ridge regression ($\lambda = 0.01$)

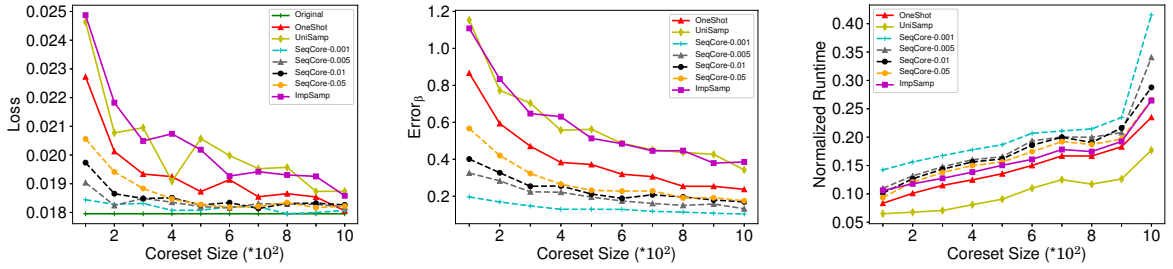

Figure 5. The experimental results on APPLIANCES ENERGY for Lasso regression ($\lambda = 0.01$).

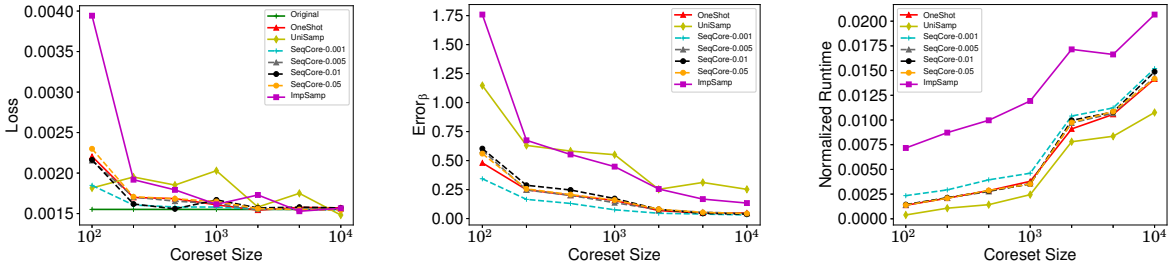

Figure 6. The experimental results on FACEBOOK COMMENT for Lasso regression ($\lambda = 0.01$).

The johnson-lindenstrauss lemma meets compressed sensing. *preprint*, 100(1):1–9, 2006.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Bertsekas, D. P. *Convex Optimization Algorithms*. Athena Scientific Belmont, MA, 2015.

Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

Borsos, Z., Mutny, M., and Krause, A. Coresets via bilevel optimization for continual learning and streaming. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60 (2):223–311, 2018.

Campbell, T. and Beronov, B. Sparse variational inference: Bayesian coresets from scratch. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11457–11468, 2019.

Campbell, T. and Broderick, T. Bayesian coreset construction via greedy iterative geodesic ascent. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 697–705. PMLR, 2018.

Candanedo, L. M., Feldheim, V., and Deramaix, D. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, 140:81–97, 2017.

Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

Chhaya, R., Dasgupta, A., and Shit, S. On coresets for regularized regression. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pp. 1866–1876, 2020.

Chowdhury, A., Yang, J., and Drineas, P. An iterative, sketching-based framework for ridge regression. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80, pp. 988–997, 2018.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. Selection via proxy: Efficient data selection for deep learning. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020.

Cramer, J. S. The early origins of the logit model. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences*, 35(4):613 – 626, 2004.

Curry, H. B. The method of steepest descent for non-linear minimization problems. *Quart. Appl. Math.*, 2:258–261, 1944.

Dasgupta, A., Drineas, P., Harb, B., Kumar, R., and Mahoney, M. W. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.

Ding, H. and Wang, Z. Layered sampling for robust optimization problems. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pp. 2556–2566, 2020.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. Sampling algorithms for l_2 regression and applications. In *Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithms*, pp. 1127–1136, 2006.

Duchi, J. C., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

Feldman, D. Core-sets: An updated survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 10(1), 2020.

Feldman, D. and Langberg, M. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pp. 569–578, 2011.

Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985.

Hoeffding, W. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pp. 409–426. Springer, 1994.

Huang, L., Jiang, S., Li, J., and Wu, X. Epsilon-coresets for clustering (with outliers) in doubling metrics. In *IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 814–825, 2018.

Huggins, J., Campbell, T., and Broderick, T. Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pp. 4080–4088, 2016.

Kacham, P. and Woodruff, D. P. Optimal deterministic coresets for ridge regression. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 108, pp. 4141–4150, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

Langberg, M. and Schulman, L. J. Universal ε-approximators for integrals. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pp. 598–607. SIAM, 2010.

Lee, S., Lee, H., Abbeel, P., and Ng, A. Y. Efficient L1 regularized logistic regression. In *Proceedings, The 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*, pp. 401–408. AAAI Press, 2006.

Li, Y., Long, P. M., and Srinivasan, A. Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527, 2001.

Lucic, M., Faulkner, M., Krause, A., and Feldman, D. Training Gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

Maalouf, A., Jubran, I., and Feldman, D. Fast and accurate least-mean-squares solvers. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, pp. 8305–8316, 2019.

Mirzasoleiman, B., Bilmes, J. A., and Leskovec, J. Coresets for data-efficient training of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pp. 6950–6960, 2020a.

Mirzasoleiman, B., Cao, K., and Leskovec, J. Coresets for robust training of deep neural networks against noisy labels. In *Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020b.

Mosci, S., Rosasco, L., Santoro, M., Verri, A., and Villa, S. Solving structured sparsity regularization with proximal methods. In *European Conference on Machine Learning and Knowledge Discovery in Databases ECML PKDD*, volume 6322, pp. 418–433, 2010.

Munteanu, A., Schwiegelshohn, C., Sohler, C., and Woodruff, D. On coresets for logistic regression. In *Advances in Neural Information Processing Systems*, pp. 6561–6570, 2018.

Nesterov, Y. A method of solving a convex programming problem with convergence rate O($1/k^2$). *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Phillips, J. M. Coresets and sketches. *Computing Research Repository*, 2016.

Raj, A., Musco, C., and Mackey, L. Importance sampling via local sensitivity. In Chiappa, S. and Calandra, R. (eds.), *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3099–3109. PMLR, 2020.

Reddi, S. J., Póczos, B., and Smola, A. J. Communication efficient coresets for empirical loss minimization. In Meila, M. and Heskes, T. (eds.), *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence, UAI 2015*, pp. 752–761. AUAI Press, 2015.

Samadian, A., Pruhs, K., Moseley, B., Im, S., and Curtin, R. R. Unconditional coresets for regularized loss minimization. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 108, pp. 482–492, 2020.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net, 2018.

Singh, K., Sandhu, R. K., and Kumar, D. Comment volume prediction using neural networks and decision trees. In *IEEE UKSim-AMSS 17th International Conference on Computer Modelling and Simulation, UKSim2015 (UKSim2015)*, Cambridge, United Kingdom, mar 2015.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

Tikhonov, A. Nonlinear ill-posed problems. *Applied Mathematical Sciences*, 1998.

Tukan, M., Maalouf, A., and Feldman, D. Coresets for near-convex functions. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

Vapnik, V. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems 4, [NIPS*, pp. 831–838, 1991.

Wolfe, P. Convergence conditions for ascent methods. *SIAM Rev.*, 11(2):226–235, 1969.

Wolsey, L. A. An analysis of the greedy algorithm for the submodular set covering problem. *Comb.*, 2(4):385–393, 1982.