Coresets for Vertical Federated Learning: Regularized Linear Regression and K-Means Clustering

Lingxiao Huang*
Nanjing University
huanglingxiao1990@126.com

Zhize Li*
Carnegie Mellon University
zhizeli@cmu.edu

Jialin Sun*
Fudan University
sunjl20@fudan.edu.cn

Haoyu Zhao*
Princeton Univeristy
haoyu@princeton.edu

Abstract

Vertical federated learning (VFL), where data features are stored in multiple parties distributively, is an important area in machine learning. However, the communication complexity for VFL is typically very high. In this paper, we propose a unified framework by constructing *coresets* in a distributed fashion for communication-efficient VFL. We study two important learning tasks in the VFL setting: regularized linear regression and k-means clustering, and apply our coreset framework to both problems. We theoretically show that using coresets can drastically alleviate the communication complexity, while nearly maintain the solution quality. Numerical experiments are conducted to corroborate our theoretical findings.

1 Introduction

Federated learning (FL) [54, 40, 44, 36, 68] is a learning framework where multiple clients/parties collaboratively train a machine learning model under the coordination of a central server without exposing their raw data (i.e., each party's raw data is stored locally and not transferred). There are two large categories of FL, horizontal federated learning (HFL) and vertical federated learning (VFL), based on the distribution characteristics of the data. In HFL, different parties usually hold different datasets but all datasets share the same features; while in VFL, all parties use the same dataset but different parties hold different subsets of the features (see Figure 1a).

Compared to HFL, VFL [74, 50, 71] is generally harder and requires more communication: as a single party cannot observe the full features, it requires communication with other parties to compute the loss and the gradient of *a single data*. This will result in two potential problems: (i) it may require a huge amount of communication to jointly train the machine learning model when the dataset is large; and (ii) the procedure of VFL transfers the information of local data and may cause privacy leakage. Most of the VFL literature focus on the privacy issue, and designing secure training procedure for different machine learning models in the VFL setting [28, 74, 72, 10]. However, the communication efficiency of the training procedure in VFL is somewhat underexplored. For unsupervised clustering problems, Ding et al. [19] propose constant approximation schemes for k-means clustering, and their communication complexity is *linear* in terms of the dataset size. For linear regression, although the communciaiton complexity can be improved to *sublinear* via sampling, such as SGD-type uniform sampling for the dataset [50, 74], the final performance is not comparable to that using the whole

*Alphabetical order.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

dataset. Thus previous algorithms usually do not scale or perform well to the big data scenarios. [2] This leads us to consider the following question:

> *How to train machine learning models using sublinear communication complexity in terms of the dataset size without sacrificing the performance in the vertical federated learning (VFL) setting?*

In this paper, we try to answer this question, and our method is based on the notion of *coreset* [27, 22, 23]. Roughly speaking, coreset can be viewed as a small data summary of the original dataset, and the machine learning model trained on the coreset performs similarly to the model trained using the full dataset. Therefore, as long as we can obtain a coreset in the VFL setting in a communication-efficient way, we can then run existing algorithms on the coreset instead of the full dataset.

Our contribution We study the communication-efficient methods for vertical federated learning with an emphasis on scalability, and design a general paradigm through the lens of coreset. Concretely, we have the following key contributions:

1. We design a unified framework for coreset construction in the vertical federated learning setting (Section 3), which can help reduce the communication complexity (Theorem 2.5).
2. We study the regularized linear regression (Definition 2.1) and k-means (Definition 2.2) problems in the VFL setting, and apply our unified coreset construction framework to them. We show that we can get ε-approximation for these two problems using only $o(n)$ sublinear communications under mild conditions, where n is the size of the dataset (Section 4 and 5).
3. We conduct numerical experiments to validate our theoretical results. Our numerical experiments corroborate our findings that using coresets can drastically reduce the communication complexity, while maintaining the quality of the solution (Section 6). Moreover, compared to uniform sampling, applying our coresets can achieve a better solution with the same or smaller communication complexity.

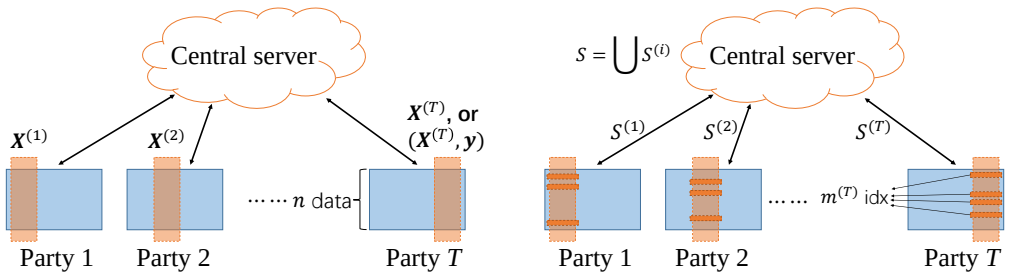
1.1 More related works

Federated learning Federated learning was introduced by McMahan et al. [54], and received increasing attention in recent years. There exist many works studied in the horizontal federated learning (HFL) setting, such as algorithms with multiple local update steps [54, 17, 39, 25, 56, 77] . There are also many algorithms with communication compression [38, 55, 47, 45, 24, 46, 60, 21, 76, 61, 78] and algorithms with privacy preserving [70, 30, 79, 64, 48].

Vertical federated learning Due to the difficulties of VFL, people designed VFL algorithms for some particular machine learning models, including linear regression [50, 74], logistic regression [75, 73, 29], gradient boosting trees [63, 11, 10], and k-means [19]. For k-means, Ding et al. [19] proposed an algorithm that computes global centers based on the product of local centers, which requires $O(nT)$ communication complexity. For linear regression, Liu et al. [50] and Yang et al. [74] used uniform sampling to get unbiased gradient estimation and improved the communication efficiency, but the performance may not be good compared to that without sampling. Yang et al. [73] also applied uniform sampling to quasi-Newton algorithm and improved communication complexity for logistic regression. People also studied other settings in VFL, e.g., how to align the data among different parties [62], how to adopt asynchronous training [9, 26], and how to defend against attacks in VFL [49, 53]. In this work, we aim to develop communication-efficient algorithms to handle large-scale VFL problems.

Coreset Coresets have been applied to a large family of problems in machine learning and statistics, including clustering [22, 7, 31, 15, 16], regression [20, 43, 6, 13, 34, 12], low rank approximation [14], and mixture model [52, 33]. Specifically, Chhaya et al. [12] investigated coreset construction for regularized regression with different norms. Feldman and Langberg [22], Braverman et al. [7] proposed an importance sampling framework for coreset construction for clustering (including k-means). The coreset size for k-means clustering has been improved by several following works [31, 15, 16] to $\tilde{O}(k\varepsilon^{-4})$, and Cohen-Addad et al. [16] proved a lower bound of size $\Omega(\varepsilon^{-2}k)$. Due to the mergable

[2] In our numerical experiments (Section 6), we provide some results to justify this claim.



(a) Communication model in VFL

(b) General coreset framework in VFL

Figure 1: Illustration of coreset construction in VFL

property of coresets, there have been studies on coreset construction in the distributed/horizontal setting [2, 58, 1, 51]. To our knowledge, we are the first to consider coreset construction in VFL.

2 Problem Formulation/Model

In this section, we formally define our problems: coresets for vertical regularized linear regression and coresets for vertical k-means clustering (Problem 1).

Vertical federated learning model. We first introduce the model of vertical federated learning (VFL). Let $\boldsymbol{X} \subset \mathbb{R}^d$ be a dataset of size n that is vertically separated stored in T data parties ($T \geq 2$). Concretely, we represent each point $\boldsymbol{x}_i \in \boldsymbol{X}$ by $\boldsymbol{x}_i = (\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(T)})$ where $\boldsymbol{x}_i^{(j)} \in \mathbb{R}^{d_j}$ ($j \in [T]$), and each party $j \in [T]$ holds a local dataset $\boldsymbol{X}^{(j)} = \left\{\boldsymbol{x}_i^{(j)}\right\}_{i \in [n]}$. Note that $\sum_{j \in [T]} d_j = d$. Additionally, if there is a label $y_i \in \mathbb{R}$ for each point $\boldsymbol{x}_i \in \boldsymbol{X}$, we assume the label vector $\boldsymbol{y} \in \mathbb{R}^n$ is stored in Party T. The objective of vertical federated learning is to collaboratively solve certain training problems in the central server with a total communication complexity as small as possible.

Similar to Ding et al. [19, Figure 1], we only allow the communication between the central server and each of the T parties, and require the central server to hold the final solution. Note that the central server can also be replaced with any party in practice. For the communication complexity, we assume that transporting an integer/floating-point costs 1 unit, and consequently, transporting a d-dimensional vector costs d communication units. See Figure 1a for an illustration.

Vertical regularized linear regression and vertical k-means clustering. In this paper, we consider the following two important machine learning problems in the VFL model.

Definition 2.1 (**Vertical regularized linear regression (VRLR)**). Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ together with labels $\boldsymbol{y} \in \mathbb{R}^n$ in the VFL model, a regularization function $R : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, the goal of the vertical regularized linear regression problem (VRLR) is to compute a vector $\boldsymbol{\theta} \in \mathbb{R}^d$ in the server that (approximately) minimizes $\mathsf{cost}^R(\boldsymbol{X}, \boldsymbol{\theta}) := \sum_{i \in [n]} \mathsf{cost}_i^R(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i \in [n]} (\boldsymbol{x}_i^\top \boldsymbol{\theta} - \boldsymbol{y}_i)^2 + R(\boldsymbol{\theta})$, and the total communication complexity is as small as possible.

Definition 2.2 (**Vertical k-means clustering (VKMC)**). Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ in the VFL model, an integer $k \geq 1$, let $\mathcal{C}$ denote the collection of all k-center sets $\boldsymbol{C} \in \mathcal{C}$ with $|\boldsymbol{C}| = k$ and $d(\cdot, \cdot)$ denote the Euclidean distance. The goal of the vertical k-means clustering problem (VKMC) is to compute a k-center set $\boldsymbol{C} \in \mathcal{C}$ in the server that (approximately) minimizes $\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}) := \sum_{i \in [n]} \mathsf{cost}_i^C(\boldsymbol{X}, \boldsymbol{C}) = \sum_{i \in [n]} d(\boldsymbol{x}_i, \boldsymbol{C})^2 = \sum_{i \in [n]} \min_{\boldsymbol{c} \in \boldsymbol{C}} d(\boldsymbol{x}_i, \boldsymbol{c})^2$, and the total communication complexity is as small as possible.

Ding et al. [19] proposed a similar vertical k-means clustering problem and provided constant approximation schemes. They additionally compute an assignment from all points x_i to solution C, which requires a communication complexity of at least $\Omega(nT)$. Due to huge n, directly solving VRLR or VKMC is a non-trivial task and may need a large communication complexity. To this end, we introduce a powerful data-reduction technique, called *coresets* [27, 22, 23].

Coresets for VRLR and VKMC. Roughly speaking, a coreset is a small summary of the original dataset, that approximates the learning objective for every possible choice of learning parameters. We first define coresets for offline regularized linear regression and k-means clustering as follows. As mentioned in Section 1.1, both problems have been well studied in the literature [22, 7, 12, 31, 15, 16].

Definition 2.3 (**Coresets for offline regularized linear regression).** Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ together with labels $\boldsymbol{y} \in \mathbb{R}^n$ and $\varepsilon \in (0,1)$, a subset $S \subseteq [n]$ together with a weight function $w : S \to \mathbb{R}_{\geq 0}$ is called an ε-coreset for offline regularized linear regression if for any $\boldsymbol{\theta} \in \mathbb{R}^d$,

$$\mathsf{cost}^R(S, \boldsymbol{\theta}) := \sum_{i \in S} w(i) \cdot (\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2 + R(\boldsymbol{\theta}) \in (1 \pm \varepsilon) \cdot \mathsf{cost}^R(\boldsymbol{X}, \boldsymbol{\theta}).$$

Definition 2.4 (**Coresets for offline k-means clustering).** Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$, an integer $k \geq 1$ and $\varepsilon \in (0,1)$, a subset $S \subseteq [n]$ together with a weight function $w : S \to \mathbb{R}_{\geq 0}$ is called an ε-coreset for offline k-means clustering if for any k-center set $\boldsymbol{C} \subset \mathbb{R}^d$,

$$\mathsf{cost}^C(S, \boldsymbol{C}) := \sum_{i \in S} w(i) \cdot d(\boldsymbol{x}_i, \boldsymbol{C})^2 \in (1 \pm \varepsilon) \cdot \mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}).$$

Now we are ready to give the following main problem.

Problem 1 (**Coreset construction for VRLR and VKMC).** Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ (together with labels $\boldsymbol{y} \in \mathbb{R}^n$) in the VFL model and $\varepsilon \in (0,1)$, our goal is to construct an ε-coreset for regularized linear regression (or k-means clustering) in the server, with as small communication complexity as possible. See Figure 1b for an illustration.

Note that our coreset is a subset of indices which is slightly different from that in previous work [27, 22, 23], whose coreset consists of weighted points. This is because we would like to reduce data transportation from parties to the server due to privacy considerations. Specifically, if the communication schemes for VRLR and VKMC do not need to make data transportation, then we can avoid data transportation by first applying our coreset construction scheme and then doing the communication schemes based on the coreset. Moreover, we have the following theorem that shows how coresets reduce the communication complexity in the VFL models, and the proof is in Section C.

Theorem 2.5 (**Coresets reduce the communication complexity for VRLR and VKMC).** *Given $\varepsilon \in (0,1)$, suppose there exist*

1. *a communication scheme A that given a (weighted) dataset $\boldsymbol{X} \subset \mathbb{R}^d$ together with labels $\boldsymbol{y} \in \mathbb{R}^n$ in the VFL model, computes an α-approximate solution ($\alpha \geq 1$) for VRLR (or VKMC) in the server with a communication complexity $\Lambda(n)$;*

2. *a communication scheme A' that given a (weighted) dataset $\boldsymbol{X} \subset \mathbb{R}^d$ together with labels $\boldsymbol{y} \in \mathbb{R}^n$ in the VFL model, constructs an ε-coreset for VRLR (or VKMC respecitively) of size m in the server with a communication complexity Λ_0.*

Then there exists a communication scheme that given a (weighted) dataset $\boldsymbol{X} \subset \mathbb{R}^d$ together with labels $\boldsymbol{y} \in \mathbb{R}^n$ in the VFL model, computes an $(1+3\varepsilon)\alpha$-approximate solution ($\alpha \geq 1$) for VRLR (or VKMC respectively) in the server with a communication complexity $\Lambda_0 + 2mT + \Lambda(m)$.

Usually, $\Lambda(m) = \Omega(mT)$ and Λ_0 is small or comparable to Tm (see Theorems 4.2 and 5.2 for examples). Consequently, the total communication complexity by introducing coresets is dominated by $\Lambda(m)$, which is much smaller compared to $\Lambda(n)$. Hence, coreset can efficiently reduce the communication complexity with a slight sacrifice on the approximate ratio.

3 A Unified Scheme for VFL Coresets via Importance Sampling

In this section, we propose a unified communication scheme (Algorithm 1) that will be used as a meta-algorithm for solving Problem 1. We assume each party $j \in [T]$ holds a real number $g_i^{(j)} \geq 0$ for data $\boldsymbol{x}_i^{(j)}$ in Algorithm 1, that will be computed locally for both VRLR (Algorithm 2) and VKMC (Algorithm 3). There are three communication rounds in Algorithm 1. In the first round (Lines 2-4), the server knows all "local total sensitivities" $\mathcal{G}^{(j)}$, takes samples of $[T]$ with probability proportional

to $\mathcal{G}^{(j)}$, and sends a_j to each party j, where a_j is the number of local samples of party j for the second round. In the second round (Lines 5-6), each party samples a collection $S^{(j)} \subseteq [n]$ of size a_j with probability proportional to $g_i^{(j)}$. The server achieves the union $S = \bigcup_{j\in[T]} S^{(j)}$. In the third round (Lines 7-8), the goal is to compute weights $w(i)$ for all samples. In the end, we achieve a weighted subset (S, w). We propose the following theorem to analyze the performance of Algorithm 1 and show that (S, w) is a coreset when size m is large enough.

Theorem 3.1 (**The performance of Algorithm 1).** *The communication complexity of Algorithm 1 is $O(mT)$. Let $\varepsilon, \delta \in (0, 1/2)$ and $k \geq 1$ be an integer. We have*

- *Let $\zeta = \max_{i\in[n]} \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X},\boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X},\boldsymbol{\theta})} / \sum_{j\in[T]} g_i^{(j)}$ and $m = O\left(\varepsilon^{-2}\zeta\mathcal{G}(d^2\log(\zeta\mathcal{G}) + \log(1/\delta))\right)$. With probability at least $1-\delta$, (S, w) is an ε-coreset for offline regularized linear regression.*
- *Let $\zeta = \max_{i\in[n]} \sup_{\boldsymbol{C}\in\mathcal{C}} \frac{\mathsf{cost}_i^C(\boldsymbol{X},\boldsymbol{C})}{\mathsf{cost}^C(\boldsymbol{X},\boldsymbol{C})} / \sum_{j\in[T]} g_i^{(j)}$ and $m = O\left(\varepsilon^{-2}\zeta\mathcal{G}(dk\log(\zeta\mathcal{G}) + \log(1/\delta))\right)$. With probability at least $1-\delta$, (S, w) is an ε-coreset for offline k-means clustering.*

The proof can be found in Appendix D. The main idea is to show that Algorithm 1 simulates a well-known importance sampling framework for offline coreset construction by [22, 7]. The term $\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X},\boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X},\boldsymbol{\theta})}$ (or $\sup_{\boldsymbol{C}\in\mathcal{C}} \frac{\mathsf{cost}_i^C(\boldsymbol{X},\boldsymbol{C})}{\mathsf{cost}^C(\boldsymbol{X},\boldsymbol{C})}$) is called the *sensitivity* of point $\boldsymbol{x}_i$ for VRLR (or VKMC) that represents the maximum contribution of $\boldsymbol{x}_i$ over all possible parameters. Algorithm 1 aims to use $\sum_{j\in[T]} g_i^{(j)}$ to estimate the sensitivity of $\boldsymbol{x}_i$, and hence, ζ represents the maximum sensitivity gap over all points. The performance of Algorithm 1 mainly depends on the quality of these estimations $\sum_{j\in[T]} g_i^{(j)}$. As both ζ and the total sum $\mathcal{G} = \sum_{i\in[n],j\in[T]} g_i^{(j)}$ become smaller, the required size m becomes smaller. Specifically, if both ζ and $\mathcal{G}$ only depends on parameters k, d, T, the coreset size m is independent of n as expected. Combining with Theorem 2.5, we can heavily reduce the communication complexity for VRLR or VKMC.

Algorithm 1 A unified importance sampling for coreset construction in the VFL model

Input: Each party $j \in [T]$ holds data $\boldsymbol{x}_i^{(j)}$ together with a real number $g_i^{(j)} \geq 0$, an integer $m \geq 1$
Output: a weighted collection $S \subseteq [n]$ of size $|S| \leq m$

1: **procedure** DIS($m, \{g_i^{(j)} : i \in [n], j \in [T]\}$)
2: Each party $j \in [T]$ sends $\mathcal{G}^{(j)} \leftarrow \sum_{i\in[n]} g_i^{(j)}$ to the server. ▷ 1st round begins
3: The server computes $\mathcal{G} = \sum_{j\in[T]} \mathcal{G}^{(j)}$ and samples a multiset $A \subseteq [T]$ of m samples, where each sample $j \in [T]$ is selected with probability $\mathcal{G}^{(j)}/\mathcal{G}$.
4: The server sends $a_j \leftarrow \#j \in A$ to each party $j \in [T]$. ▷ 1st round ends
5: Each party $j \in [T]$ samples a multiset $S^{(j)} \subseteq [n]$ of size a_j, where each sample $i \in [n]$ is selected with probability $g_i^{(j)}/\mathcal{G}^{(j)}$, and sends $S^{(j)}$ to the server. ▷ 2nd round begins
6: The server broadcasts a multiset $S \leftarrow \bigcup_{j\in[T]} S^{(j)}$ to all parties. ▷ 2nd round ends
7: Each party $j \in [T]$ sends $G^{(j)} = \left\{g_i^{(j)} : i \in S\right\}$ to the server. ▷ 3rd round begins
8: The server computes weights $w(i) \leftarrow \mathcal{G}/|S|\cdot\sum_{j\in[T]} g_i^{(j)}$ for each $i \in S$. ▷ 3rd round ends
9: **return** (S, w)
10: **end procedure**

Privacy issue. We consider the privacy of the proposed scheme from two aspects: coreset construction and model training. As for the coreset construction part (Algorithm 1), the privacy leakage comes from the "sensitivity score" $g_i^{(j)}$ of the data points in different parties. To tackle this problem, we can use secure aggregation [5] to transport the sum $g_i = \sum_{j=1}^T g_i^{(j)}$ to the server without revealing the exact values of $g_i^{(j)}$s (Line 7 of Algorithm 1). The server only knows (S, w) and $\mathcal{G}^{(j)}$s. For the model training part, we can apply the secure VFL algorithms if existed, e.g., using homomorphic encryption on SAGA for regression (it is an extension from SGD to SAGA [28]).

Algorithm 2 Vertical federated coreset construction for Regularized Linear Regression (VRLR)

Input: Each party $j \in [T]$ holds the data $\boldsymbol{x}_i^{(j)}$ for all $i \in [n]$, coreset size m.

1: **for** each party $j \in [T]$ **do**

2: Compute orthornormal basis $\boldsymbol{U}^{(j)} = [\boldsymbol{u}_1^{(j)}, \ldots, \boldsymbol{u}_n^{(j)}]^\top$ of $\boldsymbol{X}^{(j)}$

3: $g_i^{(j)} \leftarrow \|\boldsymbol{u}_i^{(j)}\|^2 + \frac{1}{n}$ for all $i \in [n]$

4: **end for**

5: **return** $(S, w) \leftarrow \texttt{DIS}(m, \{g_i^{(j)}\})$

Note that the VFL communication model in Section 2 is assumed to be semi-honest. Suppose some party j is malicious, then it can report a large enough $\mathcal{G}^{(j)}$ (Line 2 of Algorithm 1) such that the server sets the number of samples $a_j \approx m$ in party j (Line 4 of Algorithm 1). Consequently, party j can sample a large multi-set $S^{(j)}$ which heavily affects the resulting set S. For instance, by reporting $S^{(j)}$ of uniform samples, party j can make S close to uniform sampling and loss the theoretical guarantees in Theorem 3.1.

4 Coreset Construction for VRLR

In this section, we discuss the coreset construction for VRLR. We first show that it is generally hard to construct a strong coreset for VRLR. Then, we show how to communication-efficiently construct coresets for VRLR under mild assumption. All missing proofs can be found in Section E.

With slightly abuse of notation, we denote $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, $\boldsymbol{X}^{(j)} \in \mathbb{R}^{n \times d_j}$ to be the data matrix of whole data and the data matrix stored on party j respectively. Since there are labels $\boldsymbol{y}$ stored on party T, $\boldsymbol{X}^{(T)}$ has dimension $n \times (d_T + 1)$.

Communication complexity lower bound for VRLR We first show that it is *hard* to compute the coreset for VRLR by proving an $\Omega(n)$ deterministic communication complexity lower bound.

Theorem 4.1 (**Communication complexity of coreset construction for VRLR**). *Let $T \geq 2$. Given constant $\varepsilon \in (0, 1)$, any deterministic communication scheme that constructs an ε-coreset for VRLR requires a communication complexity $\Omega(n)$.*

The communication complexity lower bound for linear regression has also been considered in the HFL setting [67], e.g., Vempala et al. [67] also gets a deterministic communication complexity lower bound. Theorem 4.1 shows that linear regression in the VFL setting is "hard" and thus we may need to add data assumptions to get theoretical guarantees for coreset construction.

Communication-efficient coreset construction for VRLR Now we show that under mild condition, we can construct a strong coreset for VRLR using $o(n)$ number of communication. Specifically, we assume the data $\boldsymbol{X}$ satisfies the following assumption, which will be justified in the appendix.

Assumption 4.1. *Let $\boldsymbol{U}^{(j)} \in \mathbb{R}^{n \times d_j'}$ denote the orthonormal basis of the column space of $\boldsymbol{X}^{(j)}$ stored on party j ($\boldsymbol{U}^{(T)}$ denotes the orthonormal basis of $[\boldsymbol{X}^{(T)}, y]$), and then the matrix $\boldsymbol{U} = [\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(T)}]$ has smallest singular value $\sigma_{\min}(\boldsymbol{U}) \geq \gamma > 0$.*

Intuitively, $\gamma \in (0, 1]$ represents the degree of orthonormal among data in different parties. As the larger γ is, the more orthonormal among the column spaces of $X^{(j)}$, and thus U is more close to the orthonormal basis computed on X directly. Now we introduce our coreset construction algorithm for VRLR (Algorithm 2). At a very high level perspective, we let each party j to compute a coreset $S^{(j)}$ based on its own data $\boldsymbol{X}^{(j)}$, and combine all the $S^{(j)}$ together to obtain a final coreset S. More specifically, for each party j, we let it to compute $\boldsymbol{U}^{(j)} = [\boldsymbol{u}_1^{(j)}, \ldots, \boldsymbol{u}_n^{(j)}]^\top$ based on the data $\boldsymbol{X}^{(j)}$, and set $g_i^{(j)} = \|\boldsymbol{u}_i^{(j)}\|^2 + \frac{1}{n}$ to be the weight of data i on party j. Then, we set $g_i = \sum_{j \in [T]} g_i^{(j)}$ to be the final weight of data $\boldsymbol{x}_i$ and want to sample m samples using weight g_i. To do this, we apply the DIS procedure (Algorithm 1).

Theorem 4.2 (**Coresets for VRLR**). *For a given dataset $\boldsymbol{X} \subset \mathbb{R}^d$ satisfying Assumption 4.1, number of parties $T \geq 1$ and constants $\varepsilon, \delta \in (0, 1)$, with probability at least $1 - \delta$, Algorithm 2 constructs*

an ε-coreset for VRLR of size $m = O(\varepsilon^{-2}\gamma^{-2}d(d^2 \log \gamma^{-2}d + \log 1/\delta))$, and uses communication complexity $O(mT)$.

Note that the coreset size and the total communication are all independent on n, and thus when combined with Theorem 2.5, using coreset construction can reduce the communication complexity for VRLR. When Assumption 4.1 is not satisfied, Algorithm 4.2 is not guaranteed to return a strong coreset. However, as shown in the following remark, it will return another kind of coreset called *robust coreset* [22, 32, 69], which allows a small portion of data to be treated as outliers and excluded both in S and $\boldsymbol{X}$ when evaluating the quality of S. The outliers represent a small percentage of data with unbounded sensitivity gap. More details can be found in the Theorem G.3.

Remark 4.3 (Robust coreset for VRLR)**.** *Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ together with labels $\boldsymbol{y} \in \mathbb{R}^n$, $\varepsilon \in (0,1)$ and $\beta \in [0,1)$, a subset $S \subseteq [n]$ together with a weight function $w : S \to \mathbb{R}_{\geq 0}$ is called a (β, ε)-robust coreset for offline regularized linear regression if for any $\boldsymbol{\theta} \in \mathbb{R}^d$, there exists a subset $O_{\boldsymbol{\theta}} \subseteq [n]$ such that $|O_{\boldsymbol{\theta}}|/n \leq \beta$, $|S \cap O_{\boldsymbol{\theta}}|/|S| \leq \beta$ and*

$$\mathsf{cost}^R(S \backslash O_{\boldsymbol{\theta}}, \boldsymbol{\theta}) \in \mathsf{cost}^R(\boldsymbol{X} \backslash O_{\boldsymbol{\theta}}, \boldsymbol{\theta}) \pm \varepsilon \cdot \mathsf{cost}^R(\boldsymbol{X}, \boldsymbol{\theta}).$$

If Assumption 4.1 is not satisfied, for $m = O((\varepsilon\beta T)^{-2}d^6)$, Algorithm 2 will return a (β, ε)-robust coreset for VRLR with communication complexity $O(mT)$.

5 Coreset Construction for VKMC

In this section, we discuss the coreset construction for VKMC. Similar to VRLR, we first show it generally requires $\Omega(n)$ communication complexity to construct a coreset for VKMC, and then we show that it is possible to vastly reduce the communication complexity (Algorithm 3) under mild data assumption. All missing proofs can be found in Section F.

Communication complexity lower bound for VKMC. We first present an $\Omega(n)$ communication complexity lower bound for constructing an ε-coreset for VKMC in the following theorem.

Theorem 5.1 (**Communication complexity of coreset construction for VKMC**)**.** *Let $d \geq T \geq 2$. Given a constant $\varepsilon \in (0,1)$ and an integer $k \geq 3$, any randomized communication scheme that constructs an ε-coreset for VKMC with probability 0.99 requires a communication complexity $\Omega(n)$.*

Different from VRLR, we have a randomized communication complexity lower bound for VKMC. Similarly, we also need to introduce certain data assumptions to get theoretical guarantees for coreset construction due to this hardness result.

Communication-efficient coreset construction for VKMC Now we show how to communication-efficiently construct coresets for VKMC under mild condition. Specifically, we assume that the data satisfies the following assumption, which will be justified in the appendix.

Assumption 5.1. *There exists $\tau \geq 1$ and some party $t \in [T]$ such that $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq \tau \left\|\boldsymbol{x}_i^{(t)} - \boldsymbol{x}_j^{(t)}\right\|^2$ for any $i, j \in [n]$.*

This assumption says that, there is a party that is "important", and any two data points which can be differentiated can also be differentiated on that party to some extent. Specifically, as τ is more close to 1, Assumption 5.1 implies that there exists a party $t \in [T]$ whose local pairwise distances $\|x_i^{(t)} - x_j^{(t)}\|$s are close to the corresponding global pairwise distances $\|x_i - x_j\|$s. Then we introduce our coreset construction algorithm for VKMC (Algorithm 3). For the input, note that there exist several constant approximation algorithms for k-means [41, 66]. The widely used k-means++ algorithm [66] provides an $O(\ln k)$-approximation and performs well in practice. Similar to Algorithm 2 for VRLR, Algorithm 3 also applies Algorithm 1 after computing $g_i^{(j)}$ locally. The key is to construct local sensitivities $g_i^{(j)}$ to upper bound both ζ and $\mathcal{G}$ in Theorem 3.1. The derivation of the local sensitivities $g_i^{(j)}$ defined in Line 10 is partly inspired by [65], which upper bounds the total sensitivity of a point set in clustering problems by projecting points onto an optimal solution. Intuitively, if some party t satisfies Assumption 5.1, a constant factor approximate solution computed locally in party t can also induce a global one. Then by projecting points onto this global constant

Algorithm 3 Vertical federated coreset construction for k-means Clustering (VKMC)

Input: Each party $j \in [T]$ holds the data $\boldsymbol{x}_i^{(j)}$ for all $i \in [n]$, coreset size m, number of centers k, an α-approximation algorithm $\mathcal{A}$ (e.g. k-means++).

Output: a weighted collection $S \subseteq [n]$ of size $|S| \leq m$

for all party $j \in [T]$ **do**
  $\boldsymbol{C}^{(j)} \leftarrow \mathcal{A}(\{\boldsymbol{x}_i^{(j)}\}_{i\in[n]})$. Note that $\boldsymbol{C}^{(j)} = \{\boldsymbol{c}_1^{(j)}, \boldsymbol{c}_2^{(j)}, \ldots, \boldsymbol{c}_k^{(j)}\}$.
  Initialize $\boldsymbol{B}_l^{(j)} = \varnothing$ for $l \in [k]$.
  for all $i \in [n]$ **do**
    $\pi(i) \leftarrow \arg\min_{l\in[k]} d(\boldsymbol{x}_i^{(j)}, \boldsymbol{c}_l^{(j)})$ ▷ a mapping to find the closest center locally.
    $\boldsymbol{B}_{\pi(i)}^{(j)} \leftarrow \boldsymbol{B}_{\pi(i)}^{(j)} \cup i$.
  end for
  $\text{cost}^{(j)} \leftarrow \sum_{i\in[n]} d(\boldsymbol{x}_i^{(j)}, \boldsymbol{C}^{(j)})^2$ ▷ $d(\boldsymbol{x}_i^{(j)}, \boldsymbol{C}^{(j)}) = d(\boldsymbol{x}_i^{(j)}, \boldsymbol{c}_{\pi(i)}^{(j)})$
  for all $i \in [n]$ **do**
    $l \leftarrow \pi(i)$, $g_i^{(j)} \leftarrow \frac{\alpha d(\boldsymbol{x}_i^{(j)}, \boldsymbol{c}_l^{(j)})^2}{\text{cost}^{(j)}} + \frac{\alpha \sum_{i'\in \boldsymbol{B}_l^{(j)}} d(\boldsymbol{x}_{i'}^{(j)}, \boldsymbol{c}_l^{(j)})^2}{|\boldsymbol{B}_l^{(j)}|\text{cost}^{(j)}} + \frac{2\alpha}{|\boldsymbol{B}_l^{(j)}|}$.
  end for
end for
return $(S, w) \leftarrow \texttt{DIS}(m, \{g_i^{(j)}\})$

approximation, we can prove that $g_i^{(t)}$ (scaled by some constant factor) is an upper bound of the global sensitivity of $\boldsymbol{x}_i$ for any $i \in [n]$. Though unaware of which party satisfies Assumption 5.1, it suffices to sum up $g_i^{(j)}$ over $j \in [T]$, only costing an additional T in $\mathcal{G}$. Finally, we can upper bound ζ by $O(\tau)$ and $\mathcal{G}$ by $O(kT)$ respectively. The main theorem is as follows.

Theorem 5.2 (**Coresets for VKMC**). *For a given dataset $\boldsymbol{X} \subset \mathbb{R}^d$ satisfying Assumption 5.1, an α-approximation algorithm for k-means with $\alpha = O(1)$, integers $k \geq 1$, $T \geq 1$ and constants $\varepsilon, \delta \in (0, 1)$, with probability at least $1 - \delta$, Algorithm 3 constructs an ε-coreset for VKMC of size $m = O(\varepsilon^{-2}\alpha\tau kT(dk \log(\alpha\tau kT) + \log 1/\delta))$, and uses communication complexity $O(mT)$.*

Again, note that both the coreset size and communication complexity are independent of n. Thus, using Algorithm 3 together with other baseline algorithms can drastically reduce the communication complexity. Similar to VRLR, we have the following remark when the data assumption (Assumption 5.1) is not satisfied. More details can be found in the Theorem G.4.

Remark 5.3 (Robust coreset for VKMC). *Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$, an integer $k \geq 1$, $\varepsilon \in (0, 1)$ and $\beta \in [0, 1)$, a subset $S \subseteq [n]$ together with a weight function $w : S \to \mathbb{R}_{\geq 0}$ is called a (β, ε)-robust coreset for offline k-means clustering if for any $\boldsymbol{C} \subset \mathbb{R}^d$, there exists a subset $O_{\boldsymbol{C}} \subseteq [n]$ such that $|O_{\boldsymbol{C}}|/n \leq \beta$, $|S \cap O_{\boldsymbol{C}}|/|S| \leq \beta$ and*

$$\mathsf{cost}^C(S \backslash O_{\boldsymbol{C}}, \boldsymbol{C}) \in \mathsf{cost}^C(\boldsymbol{X} \backslash O_{\boldsymbol{C}}, \boldsymbol{C}) \pm \varepsilon \cdot \mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}).$$

If Assumption 5.1 is not satisfied, for $m = O((\varepsilon\beta)^{-2}k^5 d)$ Algorithm 3 will return a (β, ε)-robust coreset for VKMC with communication complexity $O(mT)$.

6 Numerical Experiments

In this section, we present the numerical experiments, which corroborate our theoretical results. We conduct experiments on a single system that simulates the distributed settings.[3]

Empirical setup. We conduct experiments on the `YearPredictionMSD` dataset [4] for both VRLR and VKMC. `YearPredictionMSD` dataset has 515345 data, and each data contains 90 features and a corresponding label. We assume there are $T = 3$ parties and each party stories 30 distinct features. For VRLR, we split the data into a training set with size 463715 and a testing set with size 51630. We consider ridge regression in VRLR by letting $R(\boldsymbol{\theta}) = \lambda \|\boldsymbol{\theta}\|^2$ for $\lambda = 0.1n$ where n is the dataset

[3]The codes are available at `https://github.com/haoyuzhao123/coreset-vfl-codes`.

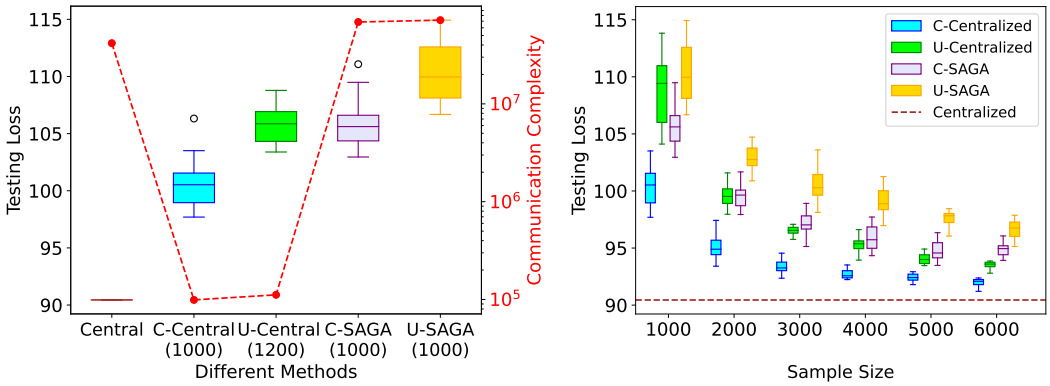


Figure 2: Left: Testing loss and communication complexity of VRLR for different methods. C and U means using coreset or uniform sampling. The number in the parentheses denotes the sample size. Right: Testing loss of VRLR for different methods under multiple sample sizes.

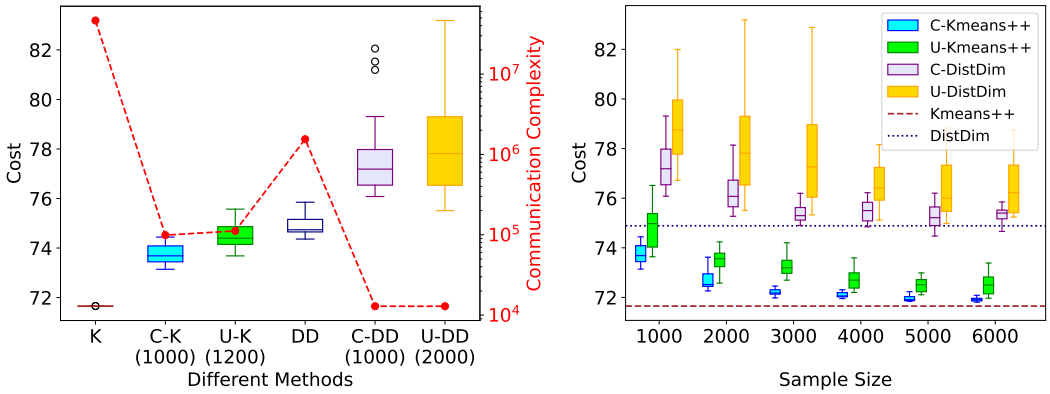


Figure 3: Left: Cost and communication complexity of VKMC for different methods. C and U means using coreset sampling or uniform sampling. The number in the parentheses denote the sample size. Right: Cost of VKMC for different methods under multiple sample sizes.

size. For VKMC, there is only one training set with size 515345 and without labels. We choose $k = 10$ (10 centers) and we normalize each feature with mean 0 and standard deviation 1 for VKMC.

For VRLR, we consider two baselines: 1) CENTRAL as the procedure that transfers all data to the central server and solves the problem using scikit-learn package [57]; 2) SAGA as using [18]'s algorithm to optimize in a VFL fashion. For VKMC, we also consider two baselines: 1) KMEANS++ as the procedure that transfers all data to the central server and clusters using KMEANS++ [66]; 2) DISTDIM by [19].

For each baseline, we compare our coreset algorithm with uniform sampling. We use C-X to denote coreset sampling followed by algorithm X and U-X for uniform sampling followed by algorithm X, e.g. C-DISTDIM means that we apply coreset construction and then use DISTDIM algorithm. We compare C-X and U-X with different sizes, and each experiment is repeated 20 times.

Empirical results. Figure 2 shows our results for VRLR and Figure 3 shows our results for VKMC. Table 1 summarize the results. For VRLR, since it is a supervised learning problem, we report the testing loss; for VKMC, it is an unsupervised learning task and the cost refers to the training loss on the full training data.

Coreset sampling performs close to the baseline with less communication. From the results, we find that using our coreset can achieve a similar loss compared to the baseline, while the communication complexity is reduced drastically. Specifically by Table 1, our coreset algorithm C-CENTRAL can

Table 1: Results of VRLR and VKMC on `YearPredictionMSD` dataset. Left: results for VRLR. Right: results for VKMC. The average and std. are computed using the 20 repeated experiments. The communication complexity denotes the average communication complexity, and the number in the parenthesis denotes the fraction of coreset construction (or uniform sampling respectively).

Alg (size)		Cost avg/std	Com. compl.	Alg	Cost avg/std	Com. compl.
CENTRAL		90.45/0.00	4.2e7			
1000	C-CENTRAL	100.50/2.11	9.9e4(0.09)	U-CENTRAL	108.79/2.79	9.3e4(0.03)
2000		95.01/0.95	2.0e5(0.09)		99.65/1.01	1.9e5(0.03)
3000		93.39/0.63	3.0e5(0.09)		96.68/0.73	2.8e5(0.03)
4000		92.73/0.41	4.0e5(0.09)		95.32/0.65	3.7e5(0.03)
5000		92.42/0.33	5.0e5(0.09)		94.08/0.48	4.7e5(0.03)
6000		91.97/0.32	5.9e5(0.09)		93.54/0.44	5.6e5(0.03)
SAGA		N/A	N/A			
1000	C-SAGA	105.93/2.17	6.9e7(<0.01)	U-SAGA	110.43/2.43	7.2e7(<0.01)
2000		99.55/0.96	1.4e8(<0.01)		102.96/1.09	1.6e8(<0.01)
3000		97.13/0.98	1.9e8(<0.01)		100.64/1.44	2.4e8(<0.01)
4000		95.90/1.04	2.5e8(<0.01)		99.10/1.20	3.3e8(<0.01)
5000		94.75/0.85	3.0e8(<0.01)		97.64/0.59	3.9e8(<0.01)
6000		94.83/0.65	3.6e8(<0.01)		96.88/1.12	4.6e8(<0.01)

Alg (size)		Cost avg/std	Com. compl.	Alg	Cost avg/std	Com. compl.
KMEANS++		71.65/0.00	4.6e7			
1000	C-KMEANS++	73.76/0.38	9.9e4(0.09)	U-KMEANS++	74.91/0.81	9.3e4(0.03)
2000		72.68/0.35	2.0e5(0.09)		73.52/0.44	1.9e5(0.03)
3000		72.23/0.17	3.0e5(0.09)		73.25/0.39	2.8e5(0.03)
4000		72.14/0.19	4.0e5(0.09)		72.74/0.39	3.7e5(0.03)
5000		71.97/0.13	5.0e5(0.09)		72.54/0.37	4.7e5(0.03)
6000		71.92/0.09	5.9e5(0.09)		72.57/0.44	5.6e5(0.03)
DISTDIM		74.89/0.00	1.5e6			
1000	C-DISTDIM	77.75/1.78	1.3e4(0.70)	U-DISTDIM	78.87/1.44	7.0e3(0.43)
2000		76.82/0.85	2.5e4(0.72)		78.13/2.10	1.3e4(0.47)
3000		75.52/0.64	3.7e4(0.73)		77.85/2.23	1.9e4(0.48)
4000		75.49/0.45	4.9e4(0.74)		76.70/1.27	2.5e4(0.48)
5000		75.27/0.50	6.1e4(0.74)		76.38/1.17	3.1e4(0.49)
6000		75.32/0.33	7.3e4(0.74)		76.44/1.03	3.7e4(0.49)

use less than 0.4% of training data (2000/463715) and achieve a $95.01/90.45 \approx 1.05$-approximate solution for VRLR compared to the baseline CENTRAL. Observe that a larger coreset size leads to a smaller cost and a larger communication complexity. From Figures 2 and 3 (left), using coresets can reduce 50-100x communication complexity compared with the original baselines.

Coreset performs better than uniform sampling under the same communication. From Figures 2 and 3 (right), we observe that our coresets always achieve a better solution than uniform sampling under the same sample size. Table 1 also reflects this trend. Under the same sample size, the communication complexity by uniform sampling is slightly lower than that of coreset, since there is no need to transfer weights in uniform sampling. Thus, we also compare the performance of our coresets and uniform sampling under the same communication complexity. From Figures 2 and 3 (left), we find that for different baselines, our coreset algorithms still achieve better testing loss/training cost while using fewer or the same communication, compared to uniform sampling.

Coreset and uniform sampling may also make the problem feasible. It is also interesting to observe that SAGA will not converge (or very slowly) on the original VRLR problem (Table 1), possibly because of the large dataset and the ill-conditioned optimization problem. However, by applying the coreset/uniform sampling, SAGA works for VRLR. This also indicates the effectiveness of our framework and the importance to reduce the dependency on n (the dataset size).

7 Conclusion and Future Directions

In this paper, we first consider coreset construction in the vertical federated learning setting. We propose a unified coreset framework for communication-efficient VFL, and apply the framework to two important learning tasks: regularized linear regression and k-means clustering. We verify the efficiency of our coreset algorithms both theoretically and empirically, which can drastically alleviate the communication complexity while still maintaining the solution quality.

Our work initializes the topic of introducing coresets to VFL, which leaves several future directions. Firstly, our VFL coreset size is still larger than that of offline coresets for both VRLR and VKMC, even under certain data assumptions. One direction is to further improve the coreset size. Another interesting direction is to extend coreset construction to other learning tasks in the VFL setting, e.g., logistic regression or gradient boosting trees.

References

[1] O. Bachem, M. Lucic, and A. Krause. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1119–1127, 2018.

[2] M.-F. F. Balcan, S. Ehrlich, and Y. Liang. Distributed k-means and k-median clustering on general topologies. *Advances in neural information processing systems*, 26, 2013.

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68 (4):702–732, 2004.

[4] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. 2011.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[6] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal coresets for least-squares regression. *IEEE transactions on information theory*, 59(10):6880–6892, 2013.

[7] V. Braverman, D. Feldman, and H. Lang. New frameworks for offline and streaming coreset constructions. *CoRR*, abs/1612.00889, 2016.

[8] P. Bürgisser and F. Cucker. Smoothed analysis of moore–penrose inversion. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2769–2783, 2010.

[9] T. Chen, X. Jin, Y. Sun, and W. Yin. VAFL: a method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.

[10] W. Chen, G. Ma, T. Fan, Y. Kang, Q. Xu, and Q. Yang. Secureboost+: A high performance gradient boosting tree framework for large scale vertical federated learning. *arXiv preprint arXiv:2110.10927*, 2021.

[11] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[12] R. Chhaya, A. Dasgupta, and S. Shit. On coresets for regularized regression. In *International conference on machine learning*, pages 1866–1876. PMLR, 2020.

[13] M. B. Cohen, Y. T. Lee, C. Musco, C. Musco, R. Peng, and A. Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015.

[14] M. B. Cohen, C. Musco, and C. Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[15] V. Cohen-Addad, D. Saulpic, and C. Schwiegelshohn. A new coreset framework for clustering. In S. Khuller and V. V. Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 169–182. ACM, 2021.

[16] V. Cohen-Addad, K. G. Larsen, D. Saulpic, and C. Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. In *Proceedings of the firty-fourth annual ACM symposium on Theory of computing*, 2022.

[17] R. Das, A. Hashemi, S. Sanghavi, and I. S. Dhillon. Improved convergence rates for non-convex federated learning with compression. *arXiv e-prints*, pages arXiv–2012, 2020.

[18] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

[19] H. Ding, Y. Liu, L. Huang, and J. Li. K-means clustering with distributed dimensions. In *International Conference on Machine Learning*, 2016.

[20] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. Sampling algorithms for l_2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136. Society for Industrial and Applied Mathematics, 2006.

[21] I. Fatkhullin, I. Sokolov, E. Gorbunov, Z. Li, and P. Richtárik. EF21 with bells & whistles: Practical algorithmic extensions of modern error feedback. *arXiv preprint arXiv:2110.03294*, 2021.

[22] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2011.

[23] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453. SIAM, 2013.

[24] E. Gorbunov, K. P. Burlachenko, Z. Li, and P. Richtárik. MARINA: Faster non-convex distributed learning with compression. In *International Conference on Machine Learning*, pages 3788–3798. PMLR, 2021.

[25] E. Gorbunov, F. Hanzely, and P. Richtárik. Local sgd: Unified theory and new efficient methods. In *International Conference on Artificial Intelligence and Statistics*, pages 3556–3564. PMLR, 2021.

[26] B. Gu, A. Xu, Z. Huo, C. Deng, and H. Huang. Privacy-preserving asynchronous federated learning algorithms for multi-party vertically collaborative learning. *arXiv preprint arXiv:2008.06233*, 2020.

[27] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *36th Annual ACM Symposium on Theory of Computing,*, pages 291–300, 2004.

[28] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[29] D. He, R. Du, S. Zhu, M. Zhang, K. Liang, and S. Chan. Secure logistic regression for vertical federated learning. *IEEE Internet Computing*, 2021.

[30] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal*, 7(10):9530–9539, 2020.

[31] L. Huang and N. K. Vishnoi. Coresets for clustering in euclidean spaces: Importance sampling is nearly optimal. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1416–1429, 2020.

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

[33] L. Huang, K. Sudhir, and N. Vishnoi. Coresets for regressions with panel data. *Advances in Neural Information Processing Systems*, 33:325–337, 2020.

[34] I. Jubran, A. Maalouf, and D. Feldman. Fast and accurate least-mean-squares solvers. In *Advances in Neural Information Processing Systems*, pages 8305–8316, 2019.

[35] Kaggle. Kc house data. `https://www.kaggle.com/datasets/shivachandel/kc-house-data`.

[36] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[37] B. Kalyanasundaram and G. Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[38] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[39] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[40] J. Konečnỳ, 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.

[41] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time (1+/spl epsiv/)-approximation algorithm for k-means clustering in any dimensions. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 454–462. IEEE, 2004.

[42] E. Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

[43] M. Li, G. L. Miller, and R. Peng. Iterative row sampling. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 127–136. IEEE, 2013.

[44] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[45] Z. Li and P. Richtárik. A unified analysis of stochastic gradient methods for nonconvex federated optimization. *arXiv preprint arXiv:2006.07013*, 2020.

[46] Z. Li and P. Richtárik. CANITA: Faster rates for distributed convex optimization with communication compression. In *Advances in Neural Information Processing Systems*, pages 13770–13781, 2021.

[47] Z. Li, D. Kovalev, X. Qian, and P. Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization. In *International Conference on Machine Learning*, pages 5895–5904. PMLR, 2020.

[48] Z. Li, H. Zhao, B. Li, and Y. Chi. SoteriaFL: A unified framework for private federated learning with communication compression. *arXiv preprint arXiv:2206.09888*, 2022.

[49] J. Liu, C. Xie, K. Kenthapadi, O. O. Koyejo, and B. Li. Rvfr: Robust vertical federated learning via feature subspace recovery. *1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021)*, 2021.

[50] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang. A communication efficient collaborative learning framework for distributed features. *arXiv preprint arXiv:1912.11187*, 2019.

[51] H. Lu, M. Li, T. He, S. Wang, V. Narayanan, and K. S. Chan. Robust coreset construction for distributed machine learning. *IEEE J. Sel. Areas Commun.*, 38(10):2400–2417, 2020.

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

[53] X. Luo, Y. Wu, X. Xiao, and B. C. Ooi. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 181–192. IEEE, 2021.

[54] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[55] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[56] A. Mitra, R. Jaafar, G. J. Pappas, and H. Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. *Advances in Neural Information Processing Systems*, 34:14606–14619, 2021.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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

[59] A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

[60] P. Richtárik, I. Sokolov, and I. Fatkhullin. EF21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34, 2021.

[61] P. Richtárik, I. Sokolov, E. Gasanov, I. Fatkhullin, Z. Li, and E. Gorbunov. 3PC: Three point compressors for communication-efficient distributed training and a better theory for lazy aggregation. In *International Conference on Machine Learning*, pages 18596–18648. PMLR, 2022.

[62] J. Sun, X. Yang, Y. Yao, A. Zhang, W. Gao, J. Xie, and C. Wang. Vertical federated learning without revealing intersection membership. *arXiv preprint arXiv:2106.05508*, 2021.

[63] Z. Tian, R. Zhang, X. Hou, J. Liu, and K. Ren. Federboost: Private federated learning for gbdt. *arXiv preprint arXiv:2011.02796*, 2020.

[64] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66, 2020.

[65] K. Varadarajan and X. Xiao. On the sensitivity of shape fitting problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[66] S. Vassilvitskii and D. Arthur. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2006.

[67] S. S. Vempala, R. Wang, and D. P. Woodruff. The communication complexity of optimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1733–1752. SIAM, 2020.

[68] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[69] Z. Wang, Y. Guo, and H. Ding. Robust and fully-dynamic coreset for continuous-and-bounded learning (with outliers) problems. *Advances in Neural Information Processing Systems*, 34, 2021.

[70] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[71] K. Wei, J. Li, C. Ma, M. Ding, S. Wei, F. Wu, G. Chen, and T. Ranbaduge. Vertical federated learning: Challenges, methodologies and experiments. *arXiv preprint arXiv:2202.04309*, 2022.

[72] H. Weng, J. Zhang, F. Xue, T. Wei, S. Ji, and Z. Zong. Privacy leakage of real-world vertical federated learning. *arXiv preprint arXiv:2011.09290*, 2020.

[73] K. Yang, T. Fan, T. Chen, Y. Shi, and Q. Yang. A quasi-newton method based vertical federated learning framework for logistic regression. *arXiv preprint arXiv:1912.00513*, 2019.

[74] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019.

[75] S. Yang, B. Ren, X. Zhou, and L. Liu. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019.

[76] H. Zhao, K. Burlachenko, Z. Li, and P. Richtárik. Faster rates for compressed federated learning with client-variance reduction. *arXiv preprint arXiv:2112.13097*, 2021.

[77] H. Zhao, Z. Li, and P. Richtárik. FedPAGE: A fast local stochastic gradient method for communication-efficient federated learning. *arXiv preprint arXiv:2108.04755*, 2021.

[78] H. Zhao, B. Li, Z. Li, P. Richtárik, and Y. Chi. BEER: Fast $O(1/T)$ rate for decentralized nonconvex optimization with communication compression. *arXiv preprint arXiv:2201.13320*, 2022.

[79] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal*, 8(11):8836–8853, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Assumption 4.1 and Assumption 5.1, and we also provide justification of these data assumtions in Section B.
   (b) Did you include complete proofs of all theoretical results? [Yes] All missing proofs can be found in the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See section 6 footnote.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] All experiments are done using a single computer
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See the baseline algorithms mentioned in Section 6.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Additional Experiments

In this section, we present some additional experiments. This section is organized as follow: in Section A.1, we conduct experiments using different number of parties (as opposed to three parties in Section 6); in Section A.2, we test our methods using other regularizer for VRLR, e.g., Lasso; in Section A.3, we test our methods in VKMC with different number of centers; and finally in Section A.4, we conduct experiments on another dataset (`KC House` Dataset [35]).

A.1 Different number of parties

In this section, we test our algorithms using different number of parties. We choose to use five parties ($T = 5$) in this section instead of three parties in Section 6.

Empirical setup Most of the experimental setups are the same as those in Section 6, except that now we use 5 parties instead of 3 parties. There are 90 dimensions for a single data in `YearPredictionMSD` dataset, and we let each party hold 18 dimensions. Besides, changing the number of parties does not affect the performance of U-Central and U-SAGA (but the number of communication will change due to different number of parties), and we reuse the results from Section 6 and recalculate the number of communications.

Empirical results Figure 4 and 5 summarize our results for VRLR and VKMC respectively. Note that all the observations in Section 6 hold for 5 parties.

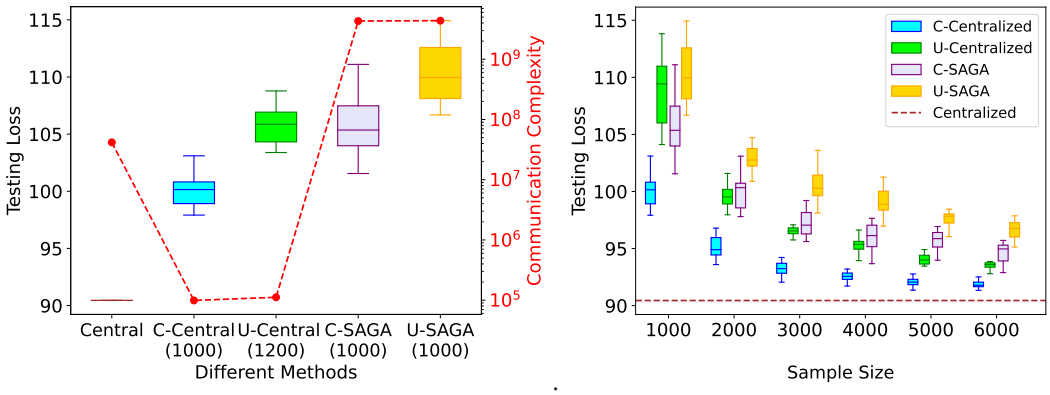


Figure 4: *Results for 5 parties (Section A.1)* Left: Testing loss and communication complexity of VRLR for different methods. C and U means using coreset or uniform sampling. The number in the parentheses denote the sample size. Right: Testing loss of VRLR for different methods under multiple sample sizes.

A.2 Different regularizer for VRLR

In this part, we consider using different regularizers in VRLR.

Empirical setup We consider three different regression problems: plain linear regression, Lasso regression, and elastic nets. In Section 6, we consider the Ridge regression ($R(\boldsymbol{\theta}) = 0.1n \|\boldsymbol{\theta}\|_2^2$ where n is the dataset size), and in this part, linear regression denotes the optimization problem where $R(\boldsymbol{\theta}) = 0$, Lasso regression denotes the problem where $R(\boldsymbol{\theta}) = 2n \|\boldsymbol{\theta}\|_1$, and elastic net denotes the problem where $R(\boldsymbol{\theta}) = 2n \|\boldsymbol{\theta}\|_1 + n \|\boldsymbol{\theta}\|_2^2$. All the experiments setup remains the same as Section 6, except the for Lasso regression and elastic nets, there is no SAGA solver and we only compare C-Central and U-Central with Central.

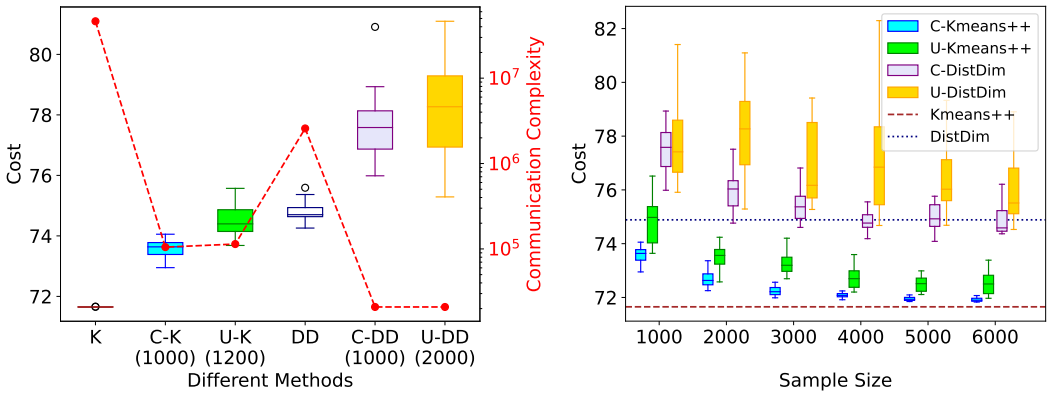


Figure 5: *Results for 5 parties (Section A.1)* Left: Cost and communication complexity of VKMC for different methods. C and U means using coreset sampling or uniform sampling. The number in the parentheses denote the sample size. Right: Cost of VKMC for different methods under multiple sample sizes.

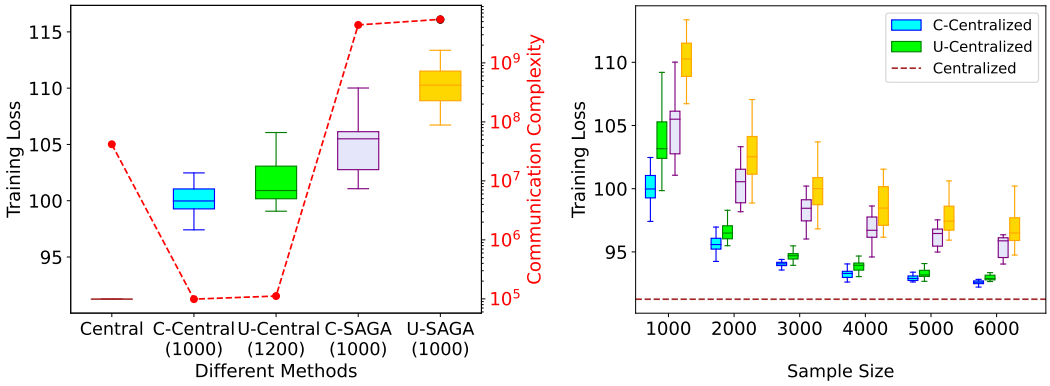


Figure 6: *Results for linear regression (Section A.2)* Left: Training loss and communication complexity of VRLR for different methods. C and U means using coreset or uniform sampling. The number in the parentheses denote the sample size. Right: Testing loss of VRLR for different methods under multiple sample sizes.

Empirical results We plot the training loss instead of the testing loss since we are comparing different objective functions. Figure 6, 7, and 8 show the empirical results in this part. Note that all the observations in Section 6 also hold: (1) coreset sampling and uniform sampling can drastically reduce the communication complexity where nearly maintain the solution performance, and (2) coreset performs better than uniform sampling under the same number of communication.

A.3 Different number of centers for VKMC

In this section we test our methods on VKMC using different number of centers.

Empirical setup The experimental setup in this part is the same as the setup in Section 6 for VKMC, except that we are using 5 centers instead of 10 centers.

Empirical results Figure 9 summarizes the result. All the observations in Section 6 also hold.

A.4 Experiments on other datasets

In this section, we present the experiment results on another dataset. We choose the `KC House` Dataset [35] for both VRLR and VKMC.

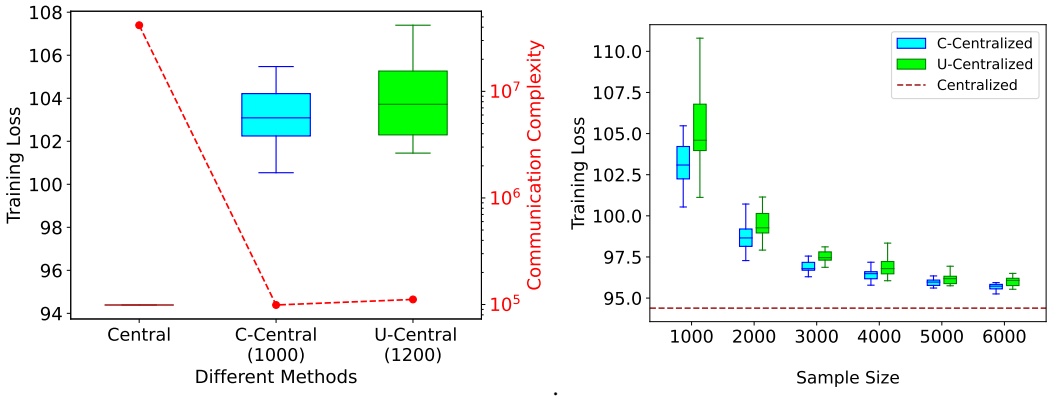


Figure 7: *Results for Lasso regression (Section A.2)* Left: Training loss and communication complexity of VRLR for different methods. C and U means using coreset or uniform sampling. The number in the parentheses denote the sample size. Right: Testing loss of VRLR for different methods under multiple sample sizes.

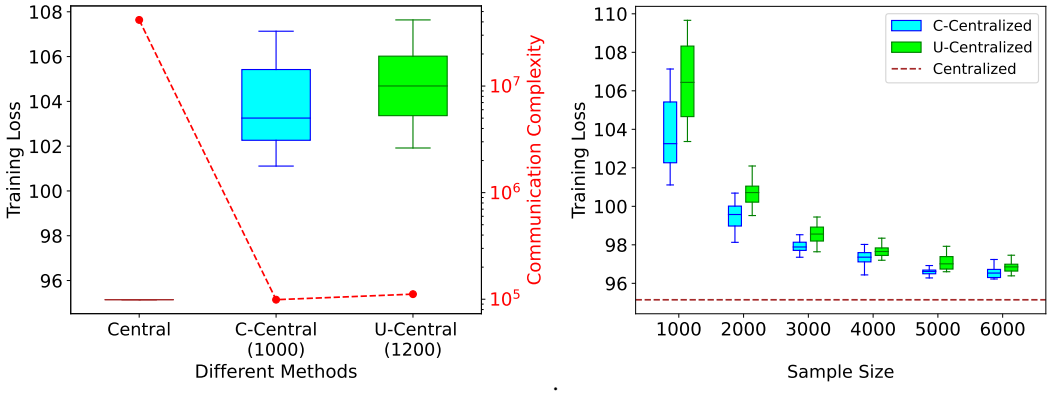


Figure 8: *Results for elastic net (Section A.2)* Left: Training loss and communication complexity of VRLR for different methods. C and U means using coreset or uniform sampling. The number in the parentheses denote the sample size. Right: Testing loss of VRLR for different methods under multiple sample sizes.

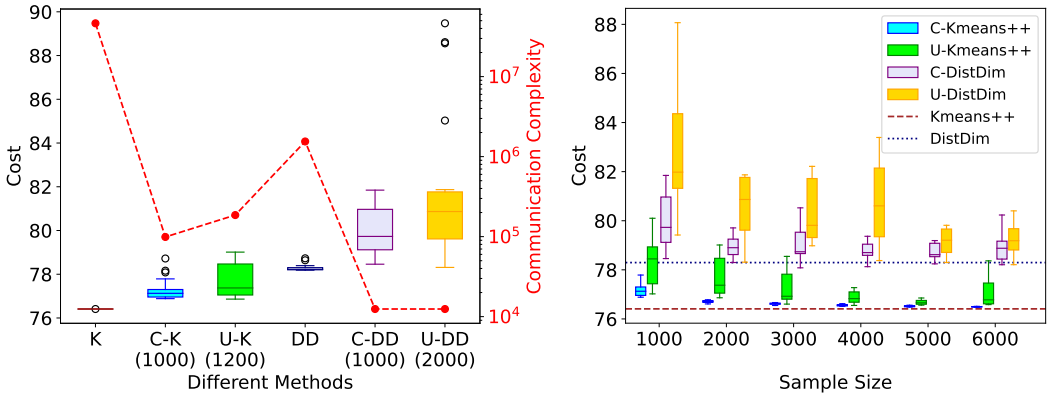


Figure 9: *Results for VKMC with 5 centers (Section A.3)* Left: Cost and communication complexity of VKMC for different methods. C and U means using coreset sampling or uniform sampling. The number in the parentheses denote the sample size. Right: Cost of VKMC for different methods under multiple sample sizes.

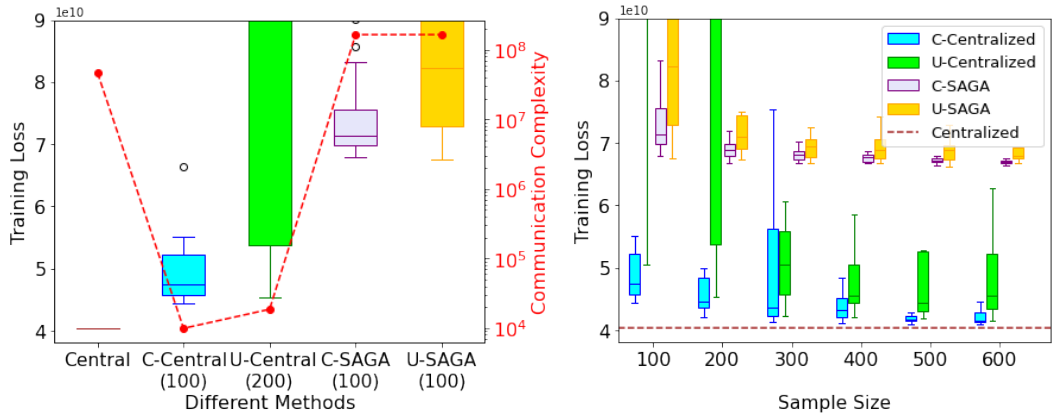


Figure 10: *Results for* `KC House` *dataset (Section A.4)* Left: Training loss and communication complexity of VRLR for different methods. C and U means using coreset or uniform sampling. The number in the parentheses denote the sample size. Right: Testing loss of VRLR for different methods under multiple sample sizes.

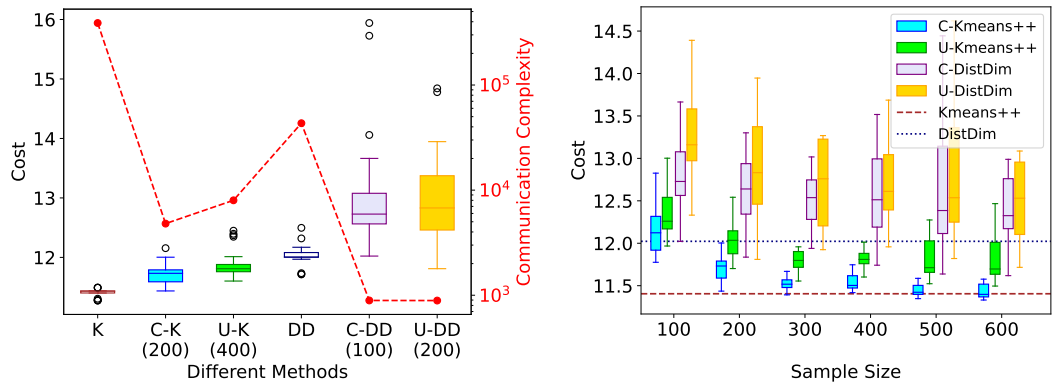


Figure 11: *Results for* `KC House` *dataset (Section A.4)* Left: Cost and communication complexity of VKMC for different methods. C and U means using coreset sampling or uniform sampling. The number in the parentheses denote the sample size. Right: Cost of VKMC for different methods under multiple sample sizes.

Empirical setup Our experiment setup is nearly the same as the setup in Section 6. However, there are a few differences: (1) the dataset we use is `KC House` Dataset [35], which contains 21613 data points and each datapoint constains 18 features and a label; (2) we conduct the experiment using only two parties because the limited number of features, we put the first nine features on the first party and the remaining on the second; and (3), we do not consider regularizer for VRLR (plain linear regression). Also note that similar to Section 6, we normalize each feature to have standard deviation 1 during the clustering task.

Empirical results For VRLR, we plot the training loss instead of the testing loss, since the dataset is not so large and coreset does not have theoretical guarantee for generalization error. Figure 10 and 11 summarize our results for VRLR and VKMC respectively. From the results, we still find that our coreset construction method can outperform uniform sampling, and both of them can drastically reduce the communication complexity compared with the original baselines.

Note that in Figure 10, C-SAGA and U-SAGA performs much worse than the baseline Central. However, C-Central can perform much better and has similar performance as Central, and this phenomenon may attribute to the fact that this problem is hard to solve by SAGA algorithm, and using other second-order methods [73] may help. Also note that when the size is small (100 and 200), U-Central may produce "ridiculous" solutions and the cost blows up.

B Justification of Data Assumptions

In this section, we justify our data assumptions in Section 4 (Assumption 4.1) and Section 5 (Assumption 5.1). We show that in the *smoothed analysis* regime, Assumption 4.1 and 5.1 are easy to satisfy with some standard assumptions. In Section B.1, we show the results related to Assumption 4.1, and in Section B.2, we justify Assumption 5.1.

B.1 Justification of Assumption 4.1

In this section, we interpret and justify Assumption 4.1. First, we recall Assumption 4.1.

Assumption 4.1. *Let $\boldsymbol{U}^{(j)} \in \mathbb{R}^{n\times d'_j}$ denote the orthonormal basis of the column space of $\boldsymbol{X}^{(j)}$ stored on party j ($\boldsymbol{U}^{(T)}$ denotes the orthonormal basis of $[\boldsymbol{X}^{(T)}, y]$), and then the matrix $\boldsymbol{U} = [\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(T)}]$ has smallest singular value $\sigma_{\min}(\boldsymbol{U}) \geq \gamma > 0$.*

Assumption 4.1 requires that the subspace generated by any party cannot be included in the subspace generated by all other parties. However, it it not sure what standard assumptions can lead to Assumption 4.1. The following lemma shows that, $\sigma_{\min}(\boldsymbol{U})$ can be lower bounded by th smallest and largest singular value of matrix $\boldsymbol{X}' = [\boldsymbol{X}, \boldsymbol{y}]$.

Lemma B.1. *If matrix $\boldsymbol{X}' = [\boldsymbol{X}, \boldsymbol{y}]$ has smallest singular value $\sigma_{\min}(\boldsymbol{X}') > 0$ and largest singular value $\sigma_{\max}(\boldsymbol{X}')$, we have*

$$\sigma_{\min}(\boldsymbol{U}) \geq \frac{\sigma_{\min}(\boldsymbol{X}')}{\sigma_{\max}(\boldsymbol{X}')}.$$

Proof. Because we assume $\boldsymbol{X}'$ has smallest singular value, we can represent $\boldsymbol{X}' = \boldsymbol{U}\boldsymbol{A}$, where $\boldsymbol{A}$ is a $d+1$ by $d+1$ matrix with rank $d+1$.

Now for any $\boldsymbol{w}$, we have

$$\|\boldsymbol{U}\boldsymbol{w}\| = \left\|\boldsymbol{X}'\boldsymbol{A}^{-1}\boldsymbol{w}\right\| \geq \sigma_{\min}(\boldsymbol{X}') \left\|\boldsymbol{A}^{-1}\boldsymbol{w}\right\|.$$

Note that $\boldsymbol{A}$ has rank $d+1$, and thus $\sigma_{\min}(\boldsymbol{A}^{-1}) = 1/\sigma_{\max}(\boldsymbol{A})$. Besides, $\boldsymbol{A} = \mathrm{diag}(\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(T)})$ is a block diagonal matrix, where $\boldsymbol{X}^{(j)} = \boldsymbol{A}^{(j)}\boldsymbol{U}^{(j)}$ for $j \in [T-1]$ and $[\boldsymbol{X}^{(T)}, \boldsymbol{y}] = \boldsymbol{A}^{(T)}\boldsymbol{U}^{(T)}$, and thus $\sigma_{\max}(\boldsymbol{A}) = \max_{j\in[T]}\{\sigma_{\max}(\boldsymbol{A}^{(j)})\}$. Because $\boldsymbol{U}^{(j)}$ is the orthonormal basis of $\boldsymbol{X}^{(j)}$ or $[\boldsymbol{X}^{(T)}, \boldsymbol{y}]$, we have

$$\sigma_{\max}(\boldsymbol{A}^{(j)}) = \sigma_{\max}(\boldsymbol{X}^{(j)}), \quad \sigma_{\max}(\boldsymbol{A}^{(T)}) = \sigma_{\max}([\boldsymbol{X}^{(T)}, \boldsymbol{y}]).$$

We also have $\sigma_{\max}(\boldsymbol{X}') \geq \sigma_{\max}(\boldsymbol{X}^{(j)})$ and $\sigma_{\max}(\boldsymbol{X}') \geq \sigma_{\max}([\boldsymbol{X}^{(T)}, \boldsymbol{y}])$. Combining all the properties together, we get $\sigma_{\max}(\boldsymbol{A}) \leq \sigma_{\max}(\boldsymbol{X}')$, and thus conclude the proof. □

Using the preivous lemma, it is easy to analyze the smallest singular value $\sigma(\boldsymbol{U})$ in the smoothed analysis regime. Specifically, we prove that for any dataset $[\boldsymbol{X}, \boldsymbol{y}]$ satisfying certain conditions, we add a random perturbation on the dataset, resulting $[\boldsymbol{X}_p, \boldsymbol{y}_p]$, and we show that with high probability, $\boldsymbol{U}_p$ (which is constructed from dataset $[\boldsymbol{X}_p, \boldsymbol{y}_p]$ has smallest singular value. The result is formalized in the following theorem.

Theorem B.1. *There exists constant n_0 such that for any dataset $[\boldsymbol{X}, \boldsymbol{y}] \in \mathbb{R}^{n\times(d+1)}$ where each data point $\|[\boldsymbol{x}_i; y_i]\|_2^2 \leq B$ and $n \geq 2d, n \geq n_0$. If we perturb the dataset by a small random Gaussian noise $[\boldsymbol{X}_p.\boldsymbol{y}_p]$ where $\boldsymbol{X}_p = \boldsymbol{X} + \boldsymbol{Z}$, $\boldsymbol{y}_p = \boldsymbol{y} + \boldsymbol{w}$, and each coordinate of $\boldsymbol{Z}$ and $\boldsymbol{w}$ comes from $\mathcal{N}(0, r^2B^2)$, then with high probability, the basis $\boldsymbol{U}_p$ computed from $[\boldsymbol{X}_p.\boldsymbol{y}_p]$ has smallest singular value at least $\Omega(r)$.*

In order to prove Theorem B.1, we use the following theorem (Theorem 1.1 in [8]).

Proposition B.1 (Smoothed analysis of condition number, Theorem 1.1 in [8])**.** *Suppose that $\bar{\boldsymbol{A}} \in \mathbb{R}^{n\times d}$ satisfies $\left\|\bar{\boldsymbol{A}}\right\| \leq 1$, and let $0 < r_p \leq 1$. Then,*

$$\Pr_{\boldsymbol{A}\sim\mathcal{N}(\bar{\boldsymbol{A}}, r^2\boldsymbol{I})}\{\kappa(\boldsymbol{A}) \geq C_1 t\} \leq \left(C_2/t + C_2/r_p\sqrt{n}t\right)^{n-d+1},$$

for some constants C_1, C_2, C_3 and all $t \geq C_3$.

Roughly speaking, Proposition B.1 claims that with high probability, the condition number under the smoothed analysis regime should be bounded above. Then with the help of Lemma B.1 and Proposition B.1, we can now prove Theorem B.1.

Proof of Theorem B.1. For simplicity, we treat denote $\boldsymbol{D} = [\boldsymbol{X}, \boldsymbol{y}]$ and $\boldsymbol{D}_p = [\boldsymbol{X}_p, \boldsymbol{y}_p]$, and $\boldsymbol{D}_p = \boldsymbol{D} + \boldsymbol{A}$, where each coordinate of $\boldsymbol{A}$ comes form $\mathcal{N}(0, r^2B^2)$.

Note that the condition number of a matrix is 'scale invariant', which means that

$$\kappa(\boldsymbol{A}) = \kappa(c\boldsymbol{A}),$$

for constants $c \neq 0$.

Now, since the row of $\boldsymbol{D}$ has bounded norm B, thus $\|\boldsymbol{D}\| \leq B\sqrt{n}$. By the scale invariance of condition number, we have

$$\kappa(\boldsymbol{D}_p) = \kappa(\boldsymbol{D}_p/(B\sqrt{n})).$$

Now, the perturbation factor r_p in Proposition B.1 is ${}^{rB}/_{B\sqrt{n}} = {}^{r}/_{\sqrt{n}}$, and we know that

$$\Pr\left\{\kappa(\boldsymbol{D}_p) \geq {}^{C_1}/_{r}\right\} \leq (C_2 r + C_2)^{n-d+1},$$

for some constants $C_1, C_3 > 0$, constant C_2 s.t. $0 < C_2 < 1$ and all $r \leq C_3$. Directly applying Lemma B.1 concludes the proof. □

B.2 Justification of Assumption 5.1

In this section, we justify Assumption 5.1. We first recall the assumption.

Assumption 5.1. *There exists $\tau \geq 1$ and some party $t \in [T]$ such that $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq \tau \left\|\boldsymbol{x}_i^{(t)} - \boldsymbol{x}_j^{(t)}\right\|^2$ for any $i, j \in [n]$.*

Roughly speaking, this assumption requires there is a party that is "important", and any two data points which can be differentiated can also be differentiated on that party to some extent. In reality, this assumption should be approximately satisfied since different features should be "correlated".

Next, similar to the justification of Assumption 4.1, we use smoothed analysis framework to show that for dataset $\boldsymbol{X}$ under certain conditions, by perturbing the dataset for a little bit, Assumption 5.1 will be satisfied with high probability. Formally, we have the following theorem.

Theorem B.2. *For any dataset where each data point $\|\boldsymbol{x}_i\|_2^2 \leq B$ for all $\boldsymbol{x}_i \in \boldsymbol{X}$ and $\max_{j\in[T]} d_j \geq \Omega(\log^2 n)$. If we perturb the dataset by a small random Gaussian noise $\boldsymbol{X}_p$ where $\boldsymbol{X}_p = \boldsymbol{X} + \boldsymbol{Z}$, and each coordinate of $\boldsymbol{Z}$ and $\boldsymbol{w}$ comes from $\mathcal{N}(0, r^2B^2)$. Then with high probability, $\boldsymbol{X}_p$ satisfies Assumption 5.1 with*

$$\tau = O\left(\frac{1}{r^2} + \frac{d}{\log^2 n}\right)$$

The intuition of the proof is that, the norm of a high-dimensional (sub-)gaussian random vector should concentrate around $\Theta(\sqrt{d})$, where d is the dimension of the (sub-)gaussian random vector. Thus, as long as we add some perturbation to the original dataset, the norm of the difference between any two perturbed data points on party j should be at least $\sqrt{d_j}$. Formally, we have the following proposition for the concentration of norm.

Proposition B.2 (Concentration of the norm)**.** *Let $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_d) \in \mathbb{R}^d$ be a random Gaussian vector, where each coordinate is sampled from $\mathcal{N}(0, r^2)$ independently. Then there exists constants c such that for any $t \geq 0$,*

$$\Pr\left\{\left|\|\boldsymbol{\xi}\|_2 - r\sqrt{d}\right| \geq rt\right\} \leq 2\exp\left(-ct^2\right)$$

Now with the help of this proposition, we can now prove Theorem B.2.

Proof of Theorem B.2. First, we upper bound $\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|^2$ where $\tilde{\boldsymbol{x}}_i$ denote the i-th perturbed data and we use $\boldsymbol{\xi}_i = \tilde{\boldsymbol{x}}_i - \boldsymbol{x}_i$ to denote the random perturbation. We have

$$\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|^2 \leq 2\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 + 2\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|^2.$$

From the assumption, we know that $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \leq 2B$, and thus we only need to bound the second term. From Proposition B.2, we know that for fixed $i \neq j$, we have

$$\Pr\left\{\left|\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\| - \sqrt{2}rB\sqrt{d}\right| \geq crB\log n\right\} \leq 2\exp\left(4\log n\right),$$

for some constants c since $\boldsymbol{\xi}_i - \boldsymbol{\xi}_j$ is a Gaussian random vector whose entries are drawn from $\mathcal{N}(0, 2r^2B^2)$. Thus, with probability at least $1 - 2/n^4$, we have

$$\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|^2 \leq 8B^2 + 4r^2B^2d + cr^2B^2\log^2 n,$$

for some constant c. Then applying the union bound, we know that with probability at least $1 - \frac{1}{n^2}$,

$$\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|^2 \leq 8B^2 + cr^2B^2\log^2 n, \forall i \neq j,$$

for some constant c. Without loss of generality, suppose that $d_1 = \max_{j\in[T]} d_j$, and then we lower bound $\left\|\tilde{\boldsymbol{x}}_i^{(1)} - \tilde{\boldsymbol{x}}_j^{(1)}\right\|^2$. First since $\left\|\tilde{\boldsymbol{x}}_i^{(1)} - \tilde{\boldsymbol{x}}_j^{(1)}\right\|^2$ is the noncentralized χ^2 distribution, we have

$$\Pr\left\{\left\|\tilde{\boldsymbol{x}}_i^{(1)} - \tilde{\boldsymbol{x}}_j^{(1)}\right\|^2 \geq t\right\} \geq \Pr\left\{\left\|\boldsymbol{\xi}_i^{(1)} - \boldsymbol{\xi}_j^{(1)}\right\|^2 \geq t\right\}.$$

Then from Proposition B.2, we have

$$\Pr\left\{\left|\left\|\boldsymbol{\xi}_i^{(1)} - \boldsymbol{\xi}_j^{(1)}\right\| - \sqrt{2}rB\sqrt{d_1}\right| \geq crB\log n\right\} \leq 2\exp\left(4\log n\right),$$

for some constant c. Thus, if $d_1 \geq C\log^2 n$ for some large enough constant c, we know that with probability at least $1 - 2/n^4$,

$$\left\|\tilde{\boldsymbol{x}}_i^{(1)} - \tilde{\boldsymbol{x}}_j^{(1)}\right\|^2 \geq cr^2B^2\log^2 n,$$

for some constant c. Then with a union bound, we know that with probability at least $1 - 1/n^2$,

$$\left\|\tilde{\boldsymbol{x}}_i^{(1)} - \tilde{\boldsymbol{x}}_j^{(1)}\right\|^2 \geq cr^2B^2\log^2 n, \forall i \neq j,$$

for some constant c. Combining with the previous part, we know that if $\max_{j\in[T]} d_j \geq C\log^2 n$ for some large constant C, then with probability at least $1 - \frac{1}{n}$, we have

$$\frac{\|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|^2}{\left\|\tilde{\boldsymbol{x}}_i^{(1)} - \tilde{\boldsymbol{x}}_j^{(1)}\right\|^2} \leq O\left(\frac{B^2 + r^2B^2d + r^2B^2\log^2 n}{r^2B^2\log^2 n}\right) = O\left(\frac{1}{r^2} + \frac{d}{\log^2 n}\right).$$

□

C Proof of Theorem 2.5

Proof of Theorem 2.5. We only take VRLR as an example. We consider the following communication scheme: First apply the communication scheme A' to construct an ε-coreset (S, w) for VRLR in the server; then the server broadcasts (S, w) to all parties; and finally apply the communication scheme A to (S, w) and obtain a solution $\boldsymbol{\theta} \in \mathbb{R}^d$ in the server.

Let $\boldsymbol{\theta}^\star$ be the optimal solution for the offline regularized linear regression problem.

By the coreset definition, we have that

$$\begin{aligned}
\mathsf{cost}^R(X, \boldsymbol{\theta}) \leq\ & (1+\varepsilon)\mathsf{cost}^R(S, \boldsymbol{\theta}) && \text{(by coreset definition)}\\
\leq\ & (1+\varepsilon)\alpha \cdot \mathsf{cost}^R(S, \boldsymbol{\theta}^\star) && \text{(by } A\text{)}\\
\leq\ & (1+\varepsilon)^2\alpha \cdot \mathsf{cost}^R(X, \boldsymbol{\theta}^\star) && \text{(by coreset definition)}\\
\leq\ & (1+3\varepsilon)\alpha \cdot \mathsf{cost}^R(X, \boldsymbol{\theta}^\star), && (\varepsilon \in (0,1))
\end{aligned}$$

which proves the approximation ratio.

For the total communication complexity, note that the broadcasting step costs $2Tm$. This completes the proof. □

D Proof of Theorem 3.1

For preparation, we first introduce a well-known importance sampling framework for offline coreset construction by [22, 7].

Theorem D.1 (**Feldman-Langberg framework [22, 7]**). *Let $\varepsilon, \delta \in (0, 1/2)$ and let $k \geq 1$ be an integer. Let $\boldsymbol{X} \subset \mathbb{R}^d$ be a dataset of n points together with a label vector $\boldsymbol{y} \in \mathbb{R}^n$, and $\boldsymbol{g} \in \mathbb{R}^n_{\geq 0}$ be a vector. Let $\mathcal{G} := \sum_{i\in[n]} g_i$. Let $S \subseteq [n]$ be constructed by taking $m \geq 1$ samples, where each sample $i \in [n]$ is selected with probability $\frac{g_i}{\mathcal{G}}$ and has weight $w(i) := \frac{\mathcal{G}}{|S| g_i}$. Then we have*

- *If $g_i \geq \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X},\boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X},\boldsymbol{\theta})}$ holds for any $i \in [n]$ and $m = O\left(\varepsilon^{-2}\mathcal{G}(d^2 \log \mathcal{G} + \log(1/\delta))\right)$, with probability at least $1 - \delta$, (S, w) is an ε-coreset for offline regularized linear regression.*
- *If $g_i \geq \sup_{\boldsymbol{C}\in\mathcal{C}} \frac{\mathsf{cost}_i^C(\boldsymbol{X},\boldsymbol{C})}{\mathsf{cost}^C(\boldsymbol{X},\boldsymbol{C})}$ holds for any $i \in [n]$ and $m = O\left(\varepsilon^{-2}\mathcal{G}(dk \log \mathcal{G} + \log(1/\delta))\right)$, with probability at least $1 - \delta$, (S, w) is an ε-coreset for offline k-means clustering.*

We call g_i the sensitivity of point $\boldsymbol{x}_i$ that represents the maximum contribution of $\boldsymbol{x}_i$ over all possible parameters, and call $\mathcal{G}$ the total sensitivity. By [65], we note that the total sensitivity can be upper bounded by $O(d)$ for offline regularized linear regression and by $O(k)$ for offline k-means clustering. By the Feldman-Langberg framework, it suffices to compute a sensitivity vector $\boldsymbol{g} \in \mathbb{R}^n$ for offline coreset construction.

Proof of Theorem 3.1. We first discuss the communication complexity of Algorithm 1. At the first round, the communication complexity in Line 2 is T and in Line 4 is T. At the second round, the communication complexity in Line 5 is at most $\sum_{j\in[T]} a_j = m$ and in Line 6 is at most mT. At the third round, the communication complexity in Line 7 is at most mT. Overall, the total communication complexity is $O(mT)$.

Next, we prove the correctness. We only take VRLR as an example and the proof for VKMC is similar. Note that each sample in S is equivalent to be drawn by the following procedure: Sample $i \in [n]$ with probability $\sum_{j\in[T]} g_i^{(j)}/\mathcal{G}$. This is because by Lines 3 and 5, the sampling probability of $i \in [n]$ is exactly

$$\sum_{j\in[T]} \frac{\mathcal{G}^{(j)}}{\mathcal{G}} \cdot \frac{g_i^{(j)}}{\mathcal{G}^{(j)}} = \frac{\sum_{j\in[T]} g_i^{(j)}}{\mathcal{G}}.$$

Then letting $g'_i = \zeta \cdot \sum_{j\in[T]} g_i^{(j)}$ for each $i \in [n]$, we have

$$g'_i \geq \sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X},\boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X},\boldsymbol{\theta})}$$

by assumption. This completes the proof by plugging g'_i to Theorem D.1. □

E Omitted Proof in Section 4

E.1 Communication lower bound for VRLR coreset construction

The proof is via a reduction from an EQUALITY problem to the problem of coreset construction for VRLR. For preparation, we first introduce some concepts in the field of communication complexity.

Communication complexity. Here it suffices to consider the two-party case ($T = 2$). Assume we have two players Alice and Bob, whose inputs are $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively. They exchange messages with a coordinator according to a protocol Π (deterministic/randomized) to compute some function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. For the input (x, y), the coordinator outputs $\Pi(x, y)$ when Alice and Bob run Π on it. We also use $\Pi(x, y)$ to denote the transcript (concatenation of messages). Let $|\Pi_{x,y}|$ be the length of the transcript. The communication complexity of Π is defined as $\max_{x,y} |\Pi_{x,y}|$. If Π is a randomized protocol, we define the *error* of Π by $\max_{x,y} \mathbb{P}(\Pi(x, y) \neq f(x, y))$, where the max is over all inputs (x, y) and the probability is over the randomness used in Π. The *δ-error randomized communication complexity* of f, denoted by $R_\delta(f)$, is the minimum communication complexity of any protocol with error at most δ.

EQUALITY problem. In the EQUALITY problem, Alice holds $a = \{a_1, \ldots, a_n\} \in \{0,1\}^n$ and Bob holds $b = \{b_1, \ldots, b_n\} \in \{0,1\}^n$. The goal is to compute EQUALITY(a, b) which equals 1 if $a_i = b_i$ for all $i \in [n]$ otherwise 0. The following lemma gives a well-known lower bound for deterministic communication protocols that correctly compute EQUALITY function.

Lemma E.1 (**Communication complexity of EQUALITY [42]**). *The deterministic communication complexity of EQUALITY is* $\Omega(n)$.

Reduction from EQUALITY. Now we are ready to prove Theorem 4.1.

Proof of Theorem 4.1. We prove this by a reduction from EQUALITY. For simplicity, it suffices to assume $d = 1$ and $T = 2$ in the VRLR problem. Given an EQUALITY instance of size n, let $a \in \{0,1\}^n$ be Alice's input and $b \in \{0,1\}^n$ be Bob's input. They construct inputs $\boldsymbol{X} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$ for VRLR, where $\boldsymbol{X} = a$ and $\boldsymbol{y} = b$. We denote $S \subseteq [n]$ with a weight function $w : S \to \mathbb{R}_{\geq 0}$ to be an ε-coreset such that for any $\boldsymbol{\theta} \in \mathbb{R}$, we have

$$\mathsf{cost}^R(S, \boldsymbol{\theta}) := \sum_{i \in S} w(i) \cdot (\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2 + R(\boldsymbol{\theta}) \in (1 \pm \varepsilon) \cdot \mathsf{cost}^R(\boldsymbol{X}, \boldsymbol{\theta}).$$

Based on the above guarantee, w.l.o.g, if we set $\theta = 1$ and $R = 0$, then there exist two cases with positive cost: $(a_i, b_i) = (0, 1)$ or $(1, 0)$. In other words, EQUALITY$(a, b) = 0$ if and only if the set $\{(\boldsymbol{x}_i, y_i) : i \in S\}$ includes $(0, 1)$ or $(1, 0)$. Thus, any deterministic protocol for VRLR coreset construction can be used as a deterministic protocol for EQUALITY. The lower bound follows from Lemma E.1. □

E.2 Proof of Theorem 4.2

In this section, we show the detailed proof of Theoem 4.2. The proof idea is to bound the sensitivity of each data point and then apply Theorem 3.1. Recall that in Theorem 3.1, we define

$$\zeta = \max_{i \in [n]} \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X}, \boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X}, \boldsymbol{\theta})} \Big/ \sum_{j \in [T]} g_i^{(j)}.$$

We first show the following main lemma.

Lemma E.2. *Under Assumption 4.1, the sensitivity of a data point can be bounded by*

$$\sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X}, \boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X}, \boldsymbol{\theta})} \leq \frac{g_i}{\gamma^2},$$

which means that $\zeta \leq 1/\gamma^2$.

Proof. The sensitivity function for each data point $(\boldsymbol{x}_i, y_i)$ is defined as

$$\sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X}, \boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X}, \boldsymbol{\theta})} = \sup_{\boldsymbol{\theta}} \frac{(\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2 + \frac{\lambda R(\boldsymbol{\theta})}{n}}{\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \lambda R(\boldsymbol{\theta})}.$$

First, we have

$$\begin{aligned}
\sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{(\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2 + \frac{\lambda R(\boldsymbol{\theta})}{n}}{\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \lambda R(\boldsymbol{\theta})} &= \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \left(\frac{(\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2}{\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \lambda R(\boldsymbol{\theta})} + \frac{\frac{\lambda R(\boldsymbol{\theta})}{n}}{\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \lambda R(\boldsymbol{\theta})} \right) \\
&\leq \sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \left(\frac{(\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2}{\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2} + \frac{1}{n} \right),
\end{aligned}$$

where we separate the regression loss and the regularized loss.

Then for the regression loss, define $\boldsymbol{X}' = [\boldsymbol{X}, \boldsymbol{y}]$ and $d' = \sum_{j \in [T]} d'_j$, we have

$$\sup_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{(\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2}{\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2} \leq \sup_{\boldsymbol{\theta} \in \mathbb{R}^{d+1}} \frac{((\boldsymbol{x}'_i)^\top \boldsymbol{\theta})^2}{\|\boldsymbol{X}'\boldsymbol{\theta}\|^2} = \sup_{\boldsymbol{\theta} \in \mathbb{R}^{d'}} \frac{((\boldsymbol{u}_i)^\top \boldsymbol{\theta})^2}{\|\boldsymbol{U}\boldsymbol{\theta}\|^2}$$

Note that under Assumption 4.1, matrix $\boldsymbol{U}$ has smallest singular value $\sigma_{\min} \geq \gamma > 0$, and we can get

$$\begin{aligned}\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{(\boldsymbol{x}_i^\top \boldsymbol{\theta} - y_i)^2}{\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2} &\leq \sup_{\boldsymbol{\theta}\in\mathbb{R}^{d'}} \frac{((\boldsymbol{u}_i)^\top \boldsymbol{\theta})^2}{\|\boldsymbol{U}\boldsymbol{\theta}\|^2} \\ &\leq \sup_{\boldsymbol{\theta}\in\mathbb{R}^{d'}} \frac{((\boldsymbol{u}_i)^\top \boldsymbol{\theta})^2}{\sigma_{\min}^2 \|\boldsymbol{\theta}\|^2} \\ &\leq \frac{\|\boldsymbol{u}_i\|^2}{\gamma^2} \\ &= \frac{\sum_{j\in[T]} \left\|\boldsymbol{u}_i^{(j)}\right\|^2}{\gamma^2}.\end{aligned}$$

Recall that $g_i = \sum_{j\in[T]} g_i^{(j)} = \sum_{j\in[T]} \left\|\boldsymbol{u}_i^{(j)}\right\|^2 + \frac{T}{n}$. Hence,

$$\sup_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{\mathsf{cost}_i^R(\boldsymbol{X},\boldsymbol{\theta})}{\mathsf{cost}^R(\boldsymbol{X},\boldsymbol{\theta})} \leq \frac{\sum_{j\in[T]} \left\|\boldsymbol{u}_i^{(j)}\right\|^2}{\gamma^2} + \frac{1}{n} \leq \frac{g_i}{\gamma^2}.$$

□

Now with the help of Lemma E.2, we can prove Theorem 4.2.

Proof of Theorem 4.2. Note that from Lemma E.2, we know that $\zeta \leq 1/\gamma^2$. Also note that from Algorithm 2, we have

$$\mathcal{G} = \sum_{j\in[T]} \sum_{i\in[n]} \left(\left\|\boldsymbol{u}_i^{(j)}\right\|^2 + \frac{1}{n}\right) = \sum_{j\in[T]} \left\|\boldsymbol{U}^{(j)}\right\|_{\mathrm{F}}^2 + T = \sum_{j\in[T]} d'_j + T \leq d + T + 1 \leq 2d + 1.$$

Then we apply Theorem 3.1, the ε-coreset size for VRLR can be bounded by

$$m = O(\varepsilon^{-2}\gamma^{-2} d(d^2 \log(\gamma^{-2} d) + \log 1/\delta)),$$

and the communication complexity is $O(mT)$. □

F Omitted Proof in Section 5

F.1 Communication lower bound for VKMC coreset construction

The proof is via a reduction from a set-disjointness (DISJ) problem to the problem of coreset construction for VKMC.

DISJ problem. In the DISJ problem, Alice holds $a = \{a_1, \ldots, a_n\} \in \{0,1\}^n$ and Bob holds $b = \{b_1, \ldots, b_n\} \in \{0,1\}^n$. The goal is to compute $\mathrm{DISJ}(a,b) = \bigvee_{i\in[n]}(a_i \bigwedge b_i)$. The following lemma gives a well-known communication lower bound for DISJ.

Lemma F.1 (Communication complexity of DISJ [37, 59, 3]). *The randomized communication complexity of DISJ is $\Omega(n)$, i.e., for $\delta \in [0, 1/2)$ and $n \geq 1$, $R_\delta(\mathit{DISJ}) = \Omega(n)$.*

Reduction from DISJ. Now we are ready to prove Theorem 5.1.

Proof of Theorem 5.1. We prove this by a reduction from DISJ. For simplicity, it suffices to assume $d = 2$ and $T = 2$ in the VKMC problem. Given a DISJ instance of size n, let $a \in \{0,1\}^n$ be Alice's input and $b \in \{0,1\}^n$ be Bob's input. They construct an input $\boldsymbol{X} \subset \mathbb{R}^2$ for VKMC, where $\boldsymbol{X} = \{\boldsymbol{x}_i : \boldsymbol{x}_i = (a_i, b_i), i \in [n]\}$. We denote $S \subseteq [n]$ with a weight function $w : S \to \mathbb{R}_{\geq 0}$ to be an ε-coreset such that for any $\boldsymbol{C} \in \mathcal{C}$ with $|\boldsymbol{C}| = k$, we have

$$\mathsf{cost}^C(S,\boldsymbol{C}) := \sum_{i\in S} w(i) \cdot d(\boldsymbol{x}_i, \boldsymbol{C})^2 \in (1 \pm \varepsilon) \cdot \mathsf{cost}^C(\boldsymbol{X},\boldsymbol{C}).$$

Based on the above guarantee, w.l.o.g., if we set $k = 3$ and $\boldsymbol{C} = \{(0,0),(0,1),(1,0)\}$, then only point $(1,1)$ can induce positive cost. In other words, $\mathrm{DISJ}(a,b) = 1$ if and only if the set $\{\boldsymbol{x}_i : i \in S\}$ includes point $(1,1)$. Thus, any δ-error protocol for VKMC coreset construction can be used as a δ-error protocol for DISJ. The lower bound follows from Lemma F.1. □

F.2 Proof of Theorem 5.2

Algorithm 3 applies the meta Algorithm 1 after computing $\{g_i^{(j)}\}$ locally. The key is to construct local sensitivities $g_i^{(j)}$ so that the sum $\sum_{j\in[T]} g_i^{(j)}$ can approximate global sensitivity g_i well, i.e, with both small ζ and $\mathcal{G}$ in Theorem 3.1.

Constructing local sensitivities. By the local sensitivities $g_i^{(j)}$ defined in Line 10 of Algorithm 3, we have the following lemma that upper bound both ζ and $\mathcal{G}$.

Lemma F.2 (**Upper bounding the global sensitivity of VKMC locally).** *Given a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ with Assumption 5.1, an α-approximation algorithm for k-means with $\alpha = O(1)$ and integers $k \geq 1$, $T \geq 1$, the local sensitivities $g_i^{(j)}$ in Algorithm 3 satisfies that for any $i \in [n]$, $\sup_{\boldsymbol{C}\in\mathcal{C}} \frac{\mathsf{cost}_i^C(\boldsymbol{X},\boldsymbol{C})}{\mathsf{cost}^C(\boldsymbol{X},\boldsymbol{C})} \leq 4\tau \sum_{j\in[T]} g_i^{(j)}$, i.e., $\zeta = O(\tau)$. Moreover, $\mathcal{G} := \sum_{i\in[n],j\in[T]} g_i^{(j)} = O(\alpha k T)$.*

The proof can be found in Section F.3, and it is partly modified from the dimension-reduction type argument [65], which upper bounds the total sensitivity of a point set in clustering problem by projecting points onto an optimal solution. Intuitively, if some party t satisfies Assumption 5.1, the partition over $[n]$ corresponding to an α-approximation computed using local data will induce a global $\alpha\tau$-approximate solution. Hence, combining this with the argument mentioned above, we derive that $g_i^{(t)}$ (scaled by 4τ) is an upper bound of the global sensitivity. Though unaware of which party satisfies Assumption 5.1, it suffices to sum up $g_i^{(j)}$ over $j \in [T]$, costing an addtional T in $\mathcal{G}$.

Proof of Theorem 5.2. By Lemma F.2, the sensitivity gap ζ is $O(\tau)$ and the total sensitivity $\mathcal{G}$ is $O(\alpha k T)$. Plugging them into Theorem 3.1 completes the proof. □

F.3 Proof of Lemma F.2

Our proof is partly inspired by [65]. For preparation, we first introduce the following useful notations.

Suppose the party t in the dataset $\boldsymbol{X}$ satisfies Assumption 5.1, and $\mathcal{A}$ is an α-approximation algorithm for k-means clustering. Let $\tilde{\boldsymbol{C}}^{(t)}$ be an α-approximate solution computed locally in party t using $\mathcal{A}$, i.e., $\tilde{\boldsymbol{C}}^{(t)} = \mathcal{A}(\boldsymbol{X}^{(t)}) = \{\tilde{\boldsymbol{c}}_l^{(t)} : l \in [k]\}$. We define a mapping $\pi : [n] \to [k]$ to find the closest center index for each point in the local solution, i.e., $\pi(i) = \arg\min_{l\in[k]} d(\boldsymbol{x}_i^{(t)}, \tilde{\boldsymbol{c}}_l^{(t)})$. We also denote $\boldsymbol{B}_l^{(t)} := \{i \in [n] : \pi(i) = l\}$ to be the local cluster corresponding to $\tilde{\boldsymbol{c}}_l^{(t)}$. Note that $\{\boldsymbol{B}_l^{(t)} : l \in [k]\}$ is a partition over data as $\boldsymbol{B}_l^{(t)} \cap \boldsymbol{B}_{l'}^{(t)} = \varnothing$ $(l, l' \in [k], l \neq l')$ and $\cup_{l\in[k]} \boldsymbol{B}_l^{(t)} = [n]$. Let $\tilde{\boldsymbol{C}} := \{\tilde{\boldsymbol{c}}_l : \tilde{\boldsymbol{c}}_l = \frac{1}{|\boldsymbol{B}_l^{(t)}|} \sum_{i\in\boldsymbol{B}_l^{(t)}} \boldsymbol{x}_i\}$ be a k-center set in $\mathbb{R}^d$ lifted from $\mathbb{R}^{d_t}$ based on $\{\boldsymbol{B}_l^{(t)}\}$. The following lemma shows that $\tilde{\boldsymbol{C}}$ is also a constant approximation to the global k-means clustering.

Lemma F.3 (**Local partition induces global constant apporximation for k-means).** *If party t of a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ satisfies Assumption 5.1, then given a local α-approximate solution $\tilde{\boldsymbol{C}}^{(t)}$, for any k-center set $\boldsymbol{C} \in \mathcal{C}$, we have*

$$\mathsf{cost}^C(\boldsymbol{X}, \tilde{\boldsymbol{C}}) \leq \tau \mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)}) \leq \alpha\tau \mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}).$$

Thus, $\tilde{\boldsymbol{C}}$ is an $\alpha\tau$-approximate solution to the global k-means clustering.

Proof.

$$\mathsf{cost}^C(\boldsymbol{X}, \tilde{\boldsymbol{C}}) = \sum_{i=1}^{n} d(\boldsymbol{x}_i, \tilde{\boldsymbol{C}})^2$$

$$
\begin{aligned}
&\leq \sum_{l=1}^{k} \sum_{i \in \boldsymbol{B}_l^{(t)}} d(\boldsymbol{x}_i, \tilde{\boldsymbol{c}}_l)^2 && \text{(assignment by } \boldsymbol{B}_l^{(t)} \text{ is not optimal)} \\
&= \sum_{l=1}^{k} \frac{1}{2|\boldsymbol{B}_l^{(t)}|} \sum_{i,j \in \boldsymbol{B}_l^{(t)}} d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 && \text{(a standard property of } k\text{-means)} \\
&\leq \sum_{l=1}^{k} \frac{\tau}{2|\boldsymbol{B}_l^{(t)}|} \sum_{i,j \in \boldsymbol{B}_l^{(t)}} d(\boldsymbol{x}_i^{(t)}, \boldsymbol{x}_j^{(t)})^2 && \text{(by Assumption 5.1)} \\
&= \tau \mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)}) \\
&\leq \alpha\tau \mathsf{cost}^C(\boldsymbol{X}^{(t)}, \boldsymbol{C}^{(t)}) && (\tilde{\boldsymbol{C}}^{(t)} \text{ is } \alpha\text{-approximation}) \\
&= \alpha\tau \sum_{i=1}^{n} d(\boldsymbol{x}_i^{(t)}, \boldsymbol{C}^{(t)})^2 \\
&\leq \alpha\tau \sum_{i=1}^{n} d(\boldsymbol{x}_i, \boldsymbol{C})^2 \\
&= \alpha\tau \mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}).
\end{aligned}
$$

Note that $|\tilde{\boldsymbol{C}}| = k$, and the above inequality holds for any $\boldsymbol{C} \in \mathcal{C}$ with $|\boldsymbol{C}| = k$. Minimizing the last item over $\boldsymbol{C} \in \mathcal{C}$ completes the proof. □

Next, since we get a global constant approximation $\tilde{\boldsymbol{C}}$, we can upper bound the global sensitivities via projecting $\boldsymbol{X}$ onto $\tilde{\boldsymbol{C}}$. Concretely, the following lemma shows that $g_i^{(t)}$ (scaled by 4τ) is an upper bound of the global sensitivity of $\boldsymbol{x}_i$ if Assumption 5.1 holds for party t.

Lemma F.4 (Upper bounding the global sensitivities for k-means). *If party t of a dataset $\boldsymbol{X} \subset \mathbb{R}^d$ satisfies Assumption 5.1, then given a local α-approximate solution $\tilde{\boldsymbol{C}}^{(t)}$, we have*

$$
\sup_{\boldsymbol{C} \in \mathcal{C}} \frac{d(\boldsymbol{x}_i, \boldsymbol{C})^2}{\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C})} \leq \frac{4\alpha\tau d(\boldsymbol{x}_i^{(t)}, \tilde{\boldsymbol{C}}^{(t)})^2}{\mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)})} + \frac{4\alpha\tau \sum_{j \in \boldsymbol{B}_{\pi(i)}^{(t)}} d(\boldsymbol{x}_j^{(t)}, \tilde{\boldsymbol{C}}^{(t)})^2}{|\boldsymbol{B}_{\pi(i)}^{(t)}| \mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)})} + \frac{8\alpha\tau}{|\boldsymbol{B}_{\pi(i)}^{(t)}|}. \tag{1}
$$

Proof. Let the multi-set $\pi(\boldsymbol{X}) := \{\tilde{\boldsymbol{c}}_{\pi(i)} : i \in [n]\}$ be the projection of $\boldsymbol{X}$ to $\tilde{\boldsymbol{C}}$. We denote $s_{\boldsymbol{X}}(\boldsymbol{x}_i)$ to be $\sup_{\boldsymbol{C} \in \mathcal{C}} \frac{d(\boldsymbol{x}_i, \boldsymbol{C})^2}{\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C})}$ for $i \in [n]$. Similarly, $s_{\pi(\boldsymbol{X})}(\tilde{\boldsymbol{c}}_l) := \sup_{\boldsymbol{C} \in \mathcal{C}} \frac{d(\tilde{\boldsymbol{c}}_l, \boldsymbol{C})^2}{\mathsf{cost}^C(\pi(\boldsymbol{X}), \boldsymbol{C})}$ for $l \in [\tilde{k}]$. First we show that for any $\boldsymbol{C} \in \mathcal{G}$, the k-means objective of the multi-set $\pi(\boldsymbol{X})$ w.r.t. $\boldsymbol{C}$ can be upper bounded by that of $\boldsymbol{X}$ with a constant factor.

$$
\begin{aligned}
\mathsf{cost}^C(\pi(\boldsymbol{X}), \boldsymbol{C}) &= \sum_{i=1}^{n} d(\tilde{\boldsymbol{c}}_{\pi(i)}, \boldsymbol{C})^2 \\
&= \sum_{i=1}^{n} \min_{l \in [k]} d(\tilde{\boldsymbol{c}}_{\pi(i)}, \boldsymbol{c}_l)^2 \\
&\leq \sum_{i=1}^{n} \min_{l \in [k]} \left(2d(\boldsymbol{x}_i, \boldsymbol{c}_l)^2 + 2d(\boldsymbol{x}_i, \tilde{\boldsymbol{c}}_{\pi(i)})^2\right) && \text{(triangle inequality for } d^2) \\
&= 2\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}) + 2\mathsf{cost}^C(\boldsymbol{X}, \tilde{\boldsymbol{C}}) \\
&\leq 2(1 + \alpha\tau)\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}) && \text{(Lemma F.3)} \\
&\leq 4\alpha\tau \mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}). && (\alpha\tau \geq 1) \qquad (2)
\end{aligned}
$$

Then for any $\boldsymbol{C} \in \mathcal{C}$ and $\boldsymbol{x}_i \in \boldsymbol{X}$, we have

$$
d(\boldsymbol{x}_i, \boldsymbol{C})^2
$$

$$
\begin{aligned}
&= \min_{l\in[k]} d(\boldsymbol{x}_i, \boldsymbol{c}_l)^2 \\
&\le \min_{l\in[k]} \left(2d(\boldsymbol{x}_i, \tilde{\boldsymbol{c}}_{\pi(i)})^2 + 2d(\tilde{\boldsymbol{c}}_{\pi(i)}, \boldsymbol{c}_l)^2\right) && \text{(triangle inequality of } d^2\text{)} \\
&= 2d(\boldsymbol{x}_i, \tilde{\boldsymbol{c}}_{\pi(i)})^2 + 2d(\tilde{\boldsymbol{c}}_{\pi(i)}, \boldsymbol{C})^2 \\
&\le 2d(\boldsymbol{x}_i, \tilde{\boldsymbol{c}}_{\pi(i)})^2 + 2s_{\pi(\boldsymbol{X})}(\tilde{\boldsymbol{c}}_{\pi(i)})\mathsf{cost}^C(\pi(\boldsymbol{X}), \boldsymbol{C}) && \text{(definition of } s_{\pi(\boldsymbol{X})}\text{)} \\
&\le 2d(\boldsymbol{x}_i, \tilde{\boldsymbol{c}}_{\pi(i)})^2 + 8\alpha\tau s_{\pi(\boldsymbol{X})}(\tilde{\boldsymbol{c}}_{\pi(i)})\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}) && \text{(from (2))} \\
&= 2d(\boldsymbol{x}_i, \frac{1}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} \boldsymbol{x}_j)^2 + 8\alpha\tau s_{\pi(\boldsymbol{X})}(\tilde{\boldsymbol{c}}_{\pi(i)})\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}) && \text{(definition of } \tilde{\boldsymbol{C}}\text{)} \\
&\le \frac{2}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 + 8\alpha\tau s_{\pi(\boldsymbol{X})}(\tilde{\boldsymbol{c}}_{\pi(i)})\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}) && \text{(convexity of } d^2\text{)} \\
&\le \frac{2}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2 + \frac{8\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}) && (s_{\pi(\boldsymbol{X})}(\tilde{\boldsymbol{c}}_{\pi(i)}) \le \frac{1}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}) \\
&\le \left(\frac{2}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} \frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C})} + \frac{8\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\right)\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C}).
\end{aligned}
$$

Thus,

$$
\begin{aligned}
&\frac{d(\boldsymbol{x}_i, \boldsymbol{C})^2}{\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C})} \\
\le\ & \frac{2}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} \frac{d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2}{\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C})} + \frac{8\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|} \\
\le\ & \frac{2\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} \frac{d(\boldsymbol{x}^{(t)}_i, \boldsymbol{x}^{(t)}_j)^2}{\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C})} + \frac{8\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|} && \text{(Assumption 5.1)} \\
\le\ & \frac{2\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} \frac{d(\boldsymbol{x}^{(t)}_i, \boldsymbol{x}^{(t)}_j)^2}{\mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)})} + \frac{8\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|} && \text{(Lemma F.3)} \\
\le\ & \frac{4\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|}\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} \frac{d(\boldsymbol{x}^{(t)}_i, \tilde{\boldsymbol{c}}_{\pi(i)})^2 + d(\boldsymbol{x}^{(t)}_j, \tilde{\boldsymbol{c}}_{\pi(i)})^2}{\mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)})} + \frac{8\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|} && \text{(triangle inequality of } d^2\text{)} \\
=\ & \frac{4\alpha\tau d(\boldsymbol{x}^{(t)}_i, \tilde{\boldsymbol{C}}^{(t)})^2}{\mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)})} + \frac{4\alpha\tau\sum_{j\in\boldsymbol{B}^{(t)}_{\pi(i)}} d(\boldsymbol{x}^{(t)}_j, \tilde{\boldsymbol{C}}^{(t)})^2}{|\boldsymbol{B}^{(t)}_{\pi(i)}|\mathsf{cost}^C(\boldsymbol{X}^{(t)}, \tilde{\boldsymbol{C}}^{(t)})} + \frac{8\alpha\tau}{|\boldsymbol{B}^{(t)}_{\pi(i)}|},
\end{aligned}
$$

taking supremum over $\boldsymbol{C} \in \mathcal{C}$ completes the proof. □

Now we are ready to prove Lemma F.2.

Proof of Lemma F.2. By Lemma F.4, since some party $t \in [T]$ satisfies Assumption 5.1, then

$$
\sup_{\boldsymbol{C}\in\mathcal{C}} \frac{\mathsf{cost}^C_i(\boldsymbol{X}, \boldsymbol{C})}{\mathsf{cost}^C(\boldsymbol{X}, \boldsymbol{C})} \le 4\tau g^{(t)}_i \le 4\tau \sum_{j\in[T]} g^{(j)}_i,
$$

where $g_i^{(t)}$ is defined as the right side of (1) for any $t \in [T]$. Moreover,

$$\begin{aligned}
\mathcal{G} &= \sum_{i\in[n]} \sum_{j\in[T]} g_i^{(j)} \\
&= \sum_{j\in[T]} \sum_{i\in[n]} \left(\frac{\alpha d(\boldsymbol{x}_i^{(j)}, \tilde{\boldsymbol{C}}^{(j)})^2}{\mathsf{cost}^C(\boldsymbol{X}^{(j)}, \tilde{\boldsymbol{C}}^{(j)})} + \frac{\alpha \sum_{i'\in \boldsymbol{B}_{\pi(i)}^{(j)}} d(\boldsymbol{x}_{i'}^{(j)}, \tilde{\boldsymbol{C}}^{(j)})^2}{|\boldsymbol{B}_{\pi(i)}^{(j)}| \mathsf{cost}^C(\boldsymbol{X}^{(j)}, \tilde{\boldsymbol{C}}^{(j)})} + \frac{2\alpha}{|\boldsymbol{B}_{\pi(i)}^{(j)}|} \right) \\
&= \sum_{j\in[T]} (\alpha + \alpha + 2k\alpha) \\
&= 2(k+1)\alpha T.
\end{aligned}$$

Hence, $\zeta = O(\tau)$ and $\mathcal{G} = O(\alpha k T)$, which completes the proof. □

G Robust Coresets for VRLR and VKMC

In this section, we prove that even if the data assumptions 4.1 and 5.1 fail to hold, Algorithms 2 and 3 still provide robust coresets for VRLR (Theorem G.3) and VKMC (Theorem G.4) in the flavor of approximating with *outliers*.

G.1 Robust coreset

In this section, we introduce a general definition of robust coreset. For preparation, we first give some notations for a function space, which can be easily specialized to the cases for VRLR and VKMC. Given a dataset $\boldsymbol{X}$ of size n, let F be a set of cost functions from $\boldsymbol{X}$ to $\mathbb{R}_{\geq 0}$. For a subset $S \subseteq [n]$ with a weight function $w : S \to \mathbb{R}_{\geq 0}$, we denote $f(S)$ to be the weighted total cost over S for any $f \in F$, i.e., $f(S) = \sum_{i\in S} w(i) f(\boldsymbol{x}_i)$. With a slight abuse of notation, we can see $\boldsymbol{X}$ as $[n]$ with unit weight such that $f(\boldsymbol{X}) = \sum_{i\in[n]} f(\boldsymbol{x}_i)$. Now we define the *robust coreset* as follows.

Definition G.1 (Robust coreset). Let $\beta \in [0, 1)$, and $\varepsilon \in (0, 1)$. Given a set F of functions from $\boldsymbol{X}$ to $\mathbb{R}_{\geq 0}$, we say that a weighted subset of $S \subseteq [n]$ is a (β, ε)-robust coreset of $\boldsymbol{X}$ if for any $f \in F$, there exists a subset $O_f \subseteq [n]$ such that

$$\frac{|O_f|}{n} \leq \beta, \frac{|S \cap O_f|}{|S|} \leq \beta,$$
$$|f(\boldsymbol{X} \backslash O_f) - f(S \backslash O_f)| \leq \varepsilon f(\boldsymbol{X}).$$

Roughly speaking, we allow a small portion of data to be treated as outliers and neglected both in $\boldsymbol{X}$ and S when considering the quality of S. Note that a $(0, \varepsilon)$-robust coreset is equivalent to a standard ε-coreset, and S provides a slightly weaker approximation guarantee with additive error if $\beta > 0$. Also note that our definition of robust coreset is a bit different from that in previous work [22, 32, 69], which focus on generating robust coresets from uniform sampling, but basically they all capture similar ideas. This is because we will be interested in the robustness of importance sampling under the case where a small percentage of data have unbounded sensitivity gap in Algorithm 1, and the above definition gives simpler results.

We propose the following theorem to show that (S, w) returned by Algorithm 1 is a (β, ε)-robust coreset when size m is large enough.

Theorem G.2 (The robustness of Algorighm 1). *Let β,$\varepsilon \in (0, 1)$. Given a dataset $\boldsymbol{X}$ of size n and a set F of functions from $\boldsymbol{X}$ to $\mathbb{R}_{\geq 0}$, let $g_i = \sum_{j\in[T]} g_i^{(j)}$ and $\mathcal{G} = \sum_{i\in[n]} g_i$. Let $S \subseteq [n]$ be a sample of size m drawn i.i.d from $[n]$ with probability proportional to $\{g_i : i \in [n]\}$, where each sample $i \in [n]$ is selected with probability $\frac{g_i}{\mathcal{G}}$ and has weight $w(i) := \frac{\mathcal{G}}{m g_i}$. If $\forall i \in [n]$, $j \in [T]$ we have $g_i^{(j)} \geq 1/n$, let $s_i := \sup_{f\in F} \frac{f(\boldsymbol{x}_i)}{f(\boldsymbol{X})}$ and $c = \frac{2\sum_{i\in[n]} s_i}{\beta T}$. If*

$$m = O\left(\frac{c^2 \mathcal{G}^2}{\varepsilon^2} \left(\dim(F) + \log \frac{1}{\delta} \right) \right), \tag{3}$$

where $\dim(F)$ is the pseudo-demension of F. Then with probability $1 - \delta$, (S, w) is a (β, ε)-robust coreset of $\boldsymbol{X}$.

The proof is in Section G.4. Recall that the term $\frac{s_i}{g_i}$ represents the sensitivity gap of point $\boldsymbol{x}_i$, and Algorithm 1 guarantees sublinear communication complexity only if the maximum sensitivity gap ζ over all points is independent of n. The main idea in the above theorem is that we can reduce the portion of potential outliers (with large sensitivity gap) to a small constant both in $\boldsymbol{X}$ and S via scaling sample size m by a sufficiently large constant.

G.2 Robust coresets for VRLR

The following theorem shows that Algorithm 2 returns a robust coreset for VRLR when sample size m is large enough. Note that m is still independent of n.

Theorem G.3 (**Robust coresets for VRLR**). *For a given dataset $\boldsymbol{X} \subset \mathbb{R}^d$, integer $T \geq 1$ and constants $\beta, \varepsilon, \delta \in (0,1)$, with probability at least $1-\delta$, Algorithm 2 constructs a (β, ε)-robust coreset for VRLR of size*

$$m = O\left(\frac{d^4}{\varepsilon^2 \beta^2 T^2}\left(d^2 + \log\frac{1}{\delta}\right)\right),$$

and uses communication complexity $O(mT)$.

Proof. By Theorem G.2, in VRLR, $F = \{f_{\boldsymbol{\theta}} : f_{\boldsymbol{\theta}}(\boldsymbol{x}) = (\boldsymbol{x}^\top \boldsymbol{\theta} - \boldsymbol{y})^2 + R(\boldsymbol{\theta})/n, \boldsymbol{\theta} \in \mathbb{R}^d\}$. Note that in Theorem 4.2, $g_i^{(j)} = \|\boldsymbol{u}_i^{(j)}\|^2 + \frac{1}{n} \geq \frac{1}{n}$, $\mathcal{G} = O(d)$ and $\sum_{i\in[n]} s_i = O(d)$, we have $c\mathcal{G} = O(\frac{d^2}{\beta T})$. Plugging $c\mathcal{G} = O(\frac{d^2}{\beta T})$ and $\dim(F) = d^2$ into (3) completes the proof. □

G.3 Robust coresets for VKMC

The following theorem shows that Algorithm 3 returns a robust coreset for VKMC when sample size m is large enough. Note that m is still independent of n.

Theorem G.4 (**Robust coresets for VKMC**). *For a given dataset $\boldsymbol{X} \subset \mathbb{R}^d$, an α-approximation algorithm for k-means with $\alpha = O(1)$, integers $k \geq 1$, $T \geq 1$ and constants $\beta, \varepsilon, \delta \in (0,1)$, with probability at least $1-\delta$, Algorithm 3 constructs a (β, ε)-robust coreset for VKMC of size*

$$m = O\left(\frac{\alpha^2 k^4}{\varepsilon^2 \beta^2}\left(dk + \log\frac{1}{\delta}\right)\right),$$

and uses communication complexity $O(mT)$.

Proof. By Theorem G.2, in VKMC, $F = \{f_{\boldsymbol{C}} : f_{\boldsymbol{C}}(\boldsymbol{x}) = d(\boldsymbol{x}, \boldsymbol{C})^2 = \min_{\boldsymbol{c}\in\boldsymbol{C}} d(\boldsymbol{x}, c)^2, \boldsymbol{C} \in \mathcal{C}, |\boldsymbol{C}| = k\}$. Note that in Theorem 5.2, $g_i^{(j)} \geq \frac{1}{n}$, $\mathcal{G} = O(\alpha k T)$ and $\sum_{i\in[n]} s_i = O(k)$, we have $c\mathcal{G} = O(\frac{\alpha k^2}{\beta})$. Plugging $c\mathcal{G} = O(\frac{d^2}{\beta T})$ and $\dim(F) = dk$ into (3) completes the proof. □

G.4 Proof of Theorem G.2

We first introduce the following lemma which mainly shows that importance sampling generates an ε-approximation of $\boldsymbol{X}$ on the corresponding weighted function space.

Lemma G.1 (**Importance sampling on a function space [2, 22]**). *Given a set F of functions from $\boldsymbol{X}$ to $\mathbb{R}_{\geq 0}$ and a constant $\varepsilon \in (0,1)$, let S be a sample of size m drawn i.i.d from $[n]$ with probability proportional to $\{g_i : i \in [n]\}$. If $g_i = \Omega(\frac{1}{n})$ for any $i \in [n]$, and let $\mathcal{G} = \sum_{i\in[n]} g_i$. If*

$$m = O\left(\frac{1}{\varepsilon^2}\left(\dim(F) + \log\frac{1}{\delta}\right)\right),$$

where $\dim(F)$ is the pseudo-demension of F. Then with probability $1-\delta$, $\forall f \in F$ and $\forall r \geq 0$,

$$\left|\sum_{i\in[n], \frac{f(\boldsymbol{x}_i)}{g_i} \leq r} f(\boldsymbol{x}_i) - \sum_{i\in S, \frac{f(\boldsymbol{x}_i)}{g_i} \leq r} \frac{\mathcal{G}}{m g_i} f(\boldsymbol{x}_i)\right| \leq \mathcal{G}\varepsilon r.$$

Now we are ready to prove Theorem G.2.

Proof of Theorem G.2. Recall that $c = \frac{2\sum_{i\in[n]} s_i}{\beta T}$. Let $O \subseteq [n]$ be defined as

$$O := \{i \in [n] : s_i \geq c g_i\}.$$

Note that $g_i = \sum_{j\in[T]} g_i^{(j)} \geq \frac{T}{n}$, and $\sum_{i\in[n]} s_i \geq \sum_{i\in O} s_i \geq \sum_{i\in O} c g_i \geq |O| \cdot \frac{cT}{n}$. Hence,

$$\frac{|O|}{n} \leq \frac{\sum_{i\in[n]} s_i}{cT} = \frac{\beta}{2} < \beta. \tag{4}$$

Let p be the probability that a point in S belongs to O, then

$$p = \frac{\sum_{i\in O} g_i}{\sum_{i\in[n]} g_i} \leq \frac{\sum_{i\in O} s_i}{c\sum_{i\in[n]} g_i} \leq \frac{\sum_{i\in O} s_i}{cT} \leq \frac{\sum_{i\in[n]} s_i}{cT} = \frac{\beta}{2}.$$

Hence, by a standard multiplicative Chernoff bound, if $m = \Omega(\frac{1}{\beta}\log 1/\delta)$, then with probability $1 - \delta/2$, we have

$$\frac{|S \cap O|}{|S|} \leq \beta. \tag{5}$$

For any $f \in F$, we define a subset $O_f \subseteq O$ as follows,

$$O_f := \{i \in [n] : \frac{f(\boldsymbol{x}_i)}{f(\boldsymbol{X})} \geq c g_i\}.$$

By (4) and (5), we have that $\frac{|O_f|}{n} \leq \beta$ and $\frac{|S\cap O_f|}{|S|} \leq \beta$. Note that $f(\boldsymbol{x}_i)/g_i \geq c f(\boldsymbol{X})$ if and only if $i \in O_f$. Let $r = c f(\boldsymbol{X})$ and plug it into Lemma G.1, then

$$\left| \sum_{i\in[n],\, \frac{f(\boldsymbol{x}_i)}{g_i} \leq r} f(\boldsymbol{x}_i) - \sum_{i\in S,\, \frac{f(\boldsymbol{x}_i)}{g_i} \leq r} \frac{\mathcal{G}}{m g_i} f(\boldsymbol{x}_i) \right| = |f(\boldsymbol{X}\backslash O_f) - f(S\backslash O_f)| \leq \mathcal{G} c \varepsilon f(\boldsymbol{X}),$$

scaling ε by $\frac{1}{c\mathcal{G}}$ completes the proof. □