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

Anonymous Author(s)
Affiliation
Address
email

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) [51, 38, 42, 34, 64] 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 [70, 47, 67] is generally harder: as a single party cannot observe the full features, it requires communication with other parties to compute the loss of *a single data*. This will result in two potential problems: (i) the procedure of VFL transfers the information of local data and may cause privacy leakage; and (ii) it may require a huge amount of communication to jointly train the machine learning model when the dataset is large. Most of the VFL literature focus on the privacy issue, and design secure training procedure for different machine learning models in the VFL setting [27, 70, 68, 9]. However, the communication efficiency of the training procedure in VFL is underexplored. To the best of our knowledge, even for the simple linear regression problem in the VFL setting, the communication complexity of the existing methods is at least linearly dependent on the dataset size [47, 70]. For unsupervised clustering problems, Ding et al. [18] propose constant approximation schemes for k-means clustering problem in the VFL setting. However, their communication complexity is also linear in terms of the dataset size. Thus previous algorithms do not scale well to the big data scenarios. [1] This leads us to consider the following question:

How to train machine learning models using sublinear communication complexity in terms of the dataset size in the vertical federated learning (VFL) setting?

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

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper, we try to answer this question, and our method is based on the notion of *coreset* [26, 21, 22]. 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

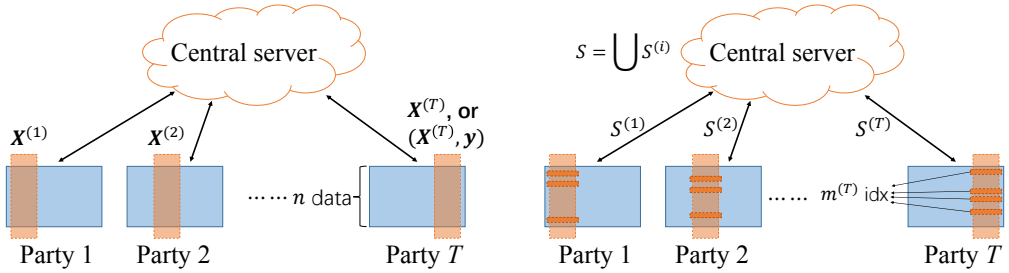
Federated learning Federated learning was introduced by McMahan et al. [51], and received increasing attention in recent years. There exist many works studied in the horizontal federated learning (HFL) setting, such as algorithms with multiple local update steps [51, 16, 37, 24, 53, 73] . There are also many algorithms with communication compression [36, 52, 45, 43, 23, 44, 56, 20, 72, 57] and algorithms with privacy preserving [66, 29, 74, 60].

Vertical federated learning Due to the difficulties of VFL, people designed secure VFL algorithms for some particular machine learning models, including linear regression [47, 70], logistic regression [71, 69, 28], gradient boosting trees [59, 10, 9], and k-means [18]. For linear regression, existing methods apply secure first-order methods (e.g., SGD) or second-order methods to optimize the loss function, all of which require $O(nT)$ communication complexity where n is the number of data and T is the number of parties. For k-means, Ding et al. [18] first computes local centers and then computes global centers based on the product of local centers, which also requires $O(nT)$ communication complexity. People also studied other settings in VFL, e.g., how to align the data among different parties [58], how to adopt asynchronous training [8, 25], and how to defend again attack in VFL [46, 50]. However, there is limited research to optimize communication efficiency in VFL. In this work, we aim to propose algorithms with sublinear communication complexity and thus can handle large-scale VFL problems.

Coreset Coresets have been applied to a large family of problems in machine learning and statistics, including clustering [21, 6, 30, 14, 15], regression [19, 41, 5, 12, 33, 11], low rank approximation [13], and mixture model [49, 32]. Specifically, Chhaya et al. [11] investigated coreset construction for regularized regression with different norms. Feldman and Langberg [21], Braverman et al. [6] 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 [30, 14, 15] to $\tilde{O}(k\varepsilon^{-4})$, and Cohen-Addad et al. [15] proved a lower bound of size $\Omega(\varepsilon^{-2}k)$. Due to the mergable property of coresets, there have been studies on coreset construction in the distributed/horizontal setting [2, 54, 1, 48]. 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).



(a) Communication model in VFL

(b) General coreset framework in VFL

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. [18, 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. 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. [18] 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* [26, 21, 22].

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 [21, 6, 11, 30, 14, 15].

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 [26, 21, 22], 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 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 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 $(1+3\varepsilon)\alpha$-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 [21, 6]. 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 $|S| \leq m$

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

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 [63], e.g., Vempala et al. [63] 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.

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

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 orthornormal basis of the data $\boldsymbol{X}^{(j)}$ stored on party j, 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 speaking, this assumption requires that the subspace generated by the data on any single party should not be fully included in the subspace generated by other parties.

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, it will return another kind of coreset called *robust coreset*. The following remark summarizes this result, and the details are deferred to the appendix.

Remark 4.3 (Informal version of Theorem G.3). *If Assumption 4.1 is not satisfied, then Algorithm 2 will return a robust coreset for VRLR.*

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.

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 $|S| \leq m$

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

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. In reality, this assumption should be approximately satisfied since different features should be correlated.

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 [39, 62]. The widely used k-means++ algorithm [62] also 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 [61], 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 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 appendix.

Remark 5.3 (Informal version of Theorem G.4)**.** *If Assumption 5.1 is not satisfied, then Algorithm 3 will return a robust coreset for VKMC.*

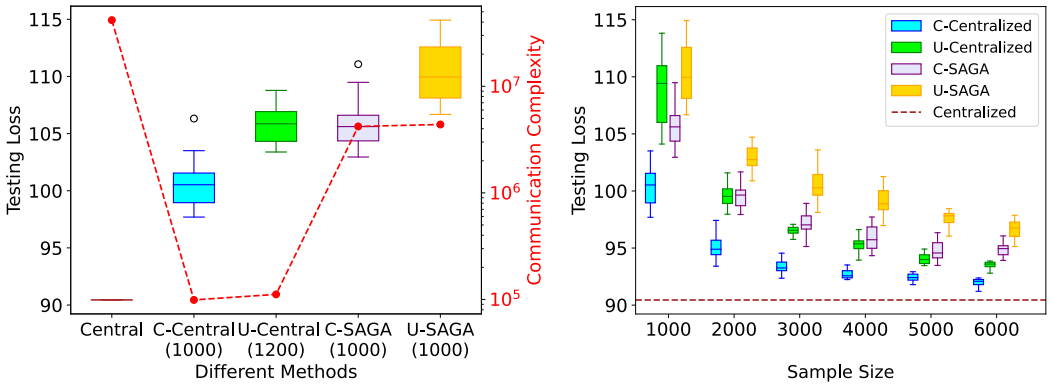


Figure 2: Left: Testing loss and communication complexity of VRLR for different methods. C and U means using coreset or uniform sampling. The number in the parentheses denote the sample size. Right: Testing loss of VRLR for different methods under multiple sample sizes.

6 Numerical Experiments

In this section, we present the numerical experiments, which corroborate our theoretical results.

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 three parties (i.e. $T = 3$) 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 $\lambda = 0.1$. 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 for the solution; 2) SAGA as using [17]'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++ [62]; 2) DISTDIM by [18].

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.

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 U-CENTRAL can use less than 4% of training data (2000/51630) 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 Figure 2 and 3 (left),

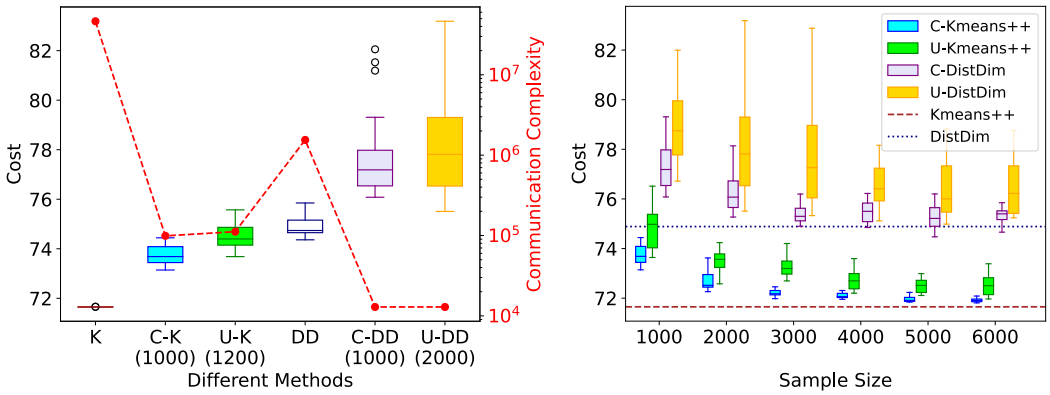


Figure 3: Left: Cost and communication complexity of VKMC for different methods. C and U means using coreset sampling or uniform sampling. The number in the parentheses denote the sample size. Right: Cost of VKMC for different methods under multiple sample sizes.

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	U-CENTRAL	108.79/2.79	9.3e4
2000		95.01/0.95	2.0e5		99.65/1.01	1.9e5
3000		93.39/0.63	3.0e5		96.68/0.73	2.8e5
4000		92.73/0.41	4.0e5		95.32/0.65	3.7e5
5000		92.42/0.33	5.0e5		94.08/0.48	4.7e5
6000		91.97/0.32	5.9e5		93.54/0.44	5.6e5
SAGA		N/A	N/A			
1000	C-SAGA	105.93/2.17	4.2e6	U-SAGA	110.43/2.43	4.4e6
2000		99.55/0.96	4.2e6		102.96/1.09	4.9e6
3000		97.13/0.98	4.0e6		100.64/1.44	4.9e6
4000		95.90/1.04	3.9e6		99.10/1.20	5.0e6
5000		94.75/0.85	3.7e6		97.64/0.59	4.8e6
6000		94.83/0.65	3.7e6		96.88/1.12	4.7e6

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	U-KMEANS++	74.91/0.81	9.3e4
2000		72.68/0.35	2.0e5		73.52/0.44	1.9e5
3000		72.23/0.17	3.0e5		73.25/0.39	2.8e5
4000		72.14/0.19	4.0e5		72.74/0.39	3.7e5
5000		71.97/0.13	5.0e5		72.54/0.37	4.7e5
6000		71.92/0.09	5.9e5		72.57/0.44	5.6e5
DISTDIM		74.89/0.00	1.5e6			
1000	C-DISTDIM	77.75/1.78	1.3e4	U-DISTDIM	78.87/1.44	7.0e3
2000		76.82/0.85	2.5e4		78.13/2.10	1.3e4
3000		75.52/0.64	3.7e4		77.85/2.23	1.9e4
4000		75.49/0.45	4.9e4		76.70/1.27	2.5e4
5000		75.27/0.50	6.1e4		76.38/1.17	3.1e4
6000		75.32/0.33	7.3e4		76.44/1.03	3.7e4

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.

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 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] Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k -means clustering via lightweight coresets. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1119–1127. ACM, 2018.

[2] Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 1995–2003, 2013.

[3] Ziv Bar-Yossef, Thathachar S Jayram, Ravi 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] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.

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

[6] Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *CoRR*, abs/1612.00889, 2016.

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

[8] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. VAFL: a method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.

[9] Weijing Chen, Guoqiang Ma, Tao Fan, Yan Kang, Qian Xu, and Qiang Yang. Secureboost+: A high performance gradient boosting tree framework for large scale vertical federated learning. *arXiv preprint arXiv:2110.10927*, 2021.

[10] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6): 87–98, 2021.

[11] Rachit Chhaya, Anirban Dasgupta, and Supratim Shit. On coresets for regularized regression. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1866–1876. PMLR.

[12] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015.

[13] Michael B Cohen, Cameron Musco, and Christopher 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.

[14] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In Samir Khuller and Virginia Vassilevska 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.

[15] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris 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.

[16] Rudrajit Das, Abolfazl Hashemi, Sujay Sanghavi, and Inderjit S Dhillon. Improved convergence rates for non-convex federated learning with compression. *arXiv e-prints*, pages arXiv–2012, 2020.

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

[18] Hu Ding, Yu Liu, Lingxiao Huang, and Jian Li. K-means clustering with distributed dimensions. In *International Conference on Machine Learning*, 2016.

[19] Petros Drineas, Michael W Mahoney, and Shan 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.

[20] Ilyas Fatkhullin, Igor Sokolov, Eduard Gorbunov, Zhize Li, and Peter Richtárik. EF21 with bells & whistles: Practical algorithmic extensions of modern error feedback. *arXiv preprint arXiv:2110.03294*, 2021.

[21] Dan Feldman and Michael 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.

[22] Dan Feldman, Melanie Schmidt, and Christian 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.

[23] Eduard Gorbunov, Konstantin P Burlachenko, Zhize Li, and Peter Richtárik. MARINA: Faster non-convex distributed learning with compression. In *International Conference on Machine Learning*, pages 3788–3798. PMLR, 2021.

[24] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. Local sgd: Unified theory and new efficient methods. In *International Conference on Artificial Intelligence and Statistics*, pages 3556–3564. PMLR, 2021.

[25] Bin Gu, An Xu, Zhouyuan Huo, Cheng Deng, and Heng Huang. Privacy-preserving asynchronous federated learning algorithms for multi-party vertically collaborative learning. *arXiv preprint arXiv:2008.06233*, 2020.

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

[27] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[28] Daojing He, Runmeng Du, Shanshan Zhu, Min Zhang, Kaitai Liang, and Sammy Chan. Secure logistic regression for vertical federated learning. *IEEE Internet Computing*, 2021.

[29] Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, and Yanmin Gong. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal*, 7(10):9530–9539, 2020.

[30] Lingxiao Huang and Nisheeth 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.

[31] Lingxiao Huang, Shaofeng H-C Jiang, Jian Li, and Xuan 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.

[32] Lingxiao Huang, K. Sudhir, and Nisheeth K. Vishnoi. Coresets for regressions with panel data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[33] Ibrahim Jubran, Alaa Maalouf, and Dan Feldman. Fast and accurate least-mean-squares solvers. In *Advances in Neural Information Processing Systems*, pages 8305–8316, 2019.

[34] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

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

[36] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[37] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[38] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[39] Amit Kumar, Yogish Sabharwal, and Sandeep 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.

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

[41] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 127–136. IEEE, 2013.

[42] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

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

[44] Zhize Li and Peter Richtárik. CANITA: Faster rates for distributed convex optimization with communication compression. *Advances in Neural Information Processing Systems*, 34, 2021.

[45] Zhize Li, Dmitry Kovalev, Xun Qian, and Peter Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization. In *International Conference on Machine Learning*, pages 5895–5904. PMLR, 2020.

[46] Jing Liu, Chulin Xie, Krishnaram Kenthapadi, Oluwasanmi O Koyejo, and Bo Li. Rvfr: Robust vertical federated learning via feature subspace recovery. *1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021)*, 2021.

[47] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A communication efficient vertical federated learning framework. *arXiv preprint arXiv:1912.11187*, 2019.

[48] Hanlin Lu, Ming-Ju Li, Ting He, Shiqiang Wang, Vijaykrishnan Narayanan, and Kevin S. Chan. Robust coreset construction for distributed machine learning. *IEEE J. Sel. Areas Commun.*, 38 (10):2400–2417, 2020.

[49] Mario Lucic, Matthew Faulkner, Andreas Krause, and Dan Feldman. Training Gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research*, 18(1):5885–5909, 2017.

[50] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin 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.

[51] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[52] Konstantin Mishchenko, Eduard Gorbunov, Martin Takáč, and Peter Richtárik. Distributed learning with compressed gradient differences. *arXiv preprint arXiv:1901.09269*, 2019.

[53] Aritra Mitra, Rayana Jaafar, George J Pappas, and Hamed Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. *Advances in Neural Information Processing Systems*, 34:14606–14619, 2021.

[54] Jeff M. Phillips. Coresets and sketches. *CoRR*, abs/1601.00617, 2016.

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

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

[57] Peter Richtárik, Igor Sokolov, Ilyas Fatkhullin, Elnur Gasanov, Zhize Li, and Eduard Gorbunov. 3PC: Three point compressors for communication-efficient distributed training and a better theory for lazy aggregation. *arXiv preprint arXiv:2202.00998*, 2022.

[58] Jiankai Sun, Xin Yang, Yuanshun Yao, Aonan Zhang, Weihao Gao, Junyuan Xie, and Chong Wang. Vertical federated learning without revealing intersection membership. *arXiv preprint arXiv:2106.05508*, 2021.

[59] Zhihua Tian, Rui Zhang, Xiaoyang Hou, Jian Liu, and Kui Ren. Federboost: Private federated learning for gbdt. *arXiv preprint arXiv:2011.02796*, 2020.

[60] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi 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.

[61] Kasturi Varadarajan and Xin 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.

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

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

[64] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[65] Zixiu Wang, Yiwen Guo, and Hu Ding. Robust and fully-dynamic coreset for continuous-and-bounded learning (with outliers) problems. *Advances in Neural Information Processing Systems*, 34, 2021.

[66] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[67] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Sha Wei, Fan Wu, Guihai Chen, and Thilina Ranbaduge. Vertical federated learning: Challenges, methodologies and experiments. *arXiv preprint arXiv:2202.04309*, 2022.

[68] Haiqin Weng, Juntao Zhang, Feng Xue, Tao Wei, Shouling Ji, and Zhiyuan Zong. Privacy leakage of real-world vertical federated learning. *arXiv preprint arXiv:2011.09290*, 2020.

[69] Kai Yang, Tao Fan, Tianjian Chen, Yuanming Shi, and Qiang Yang. A quasi-newton method based vertical federated learning framework for logistic regression. *arXiv preprint arXiv:1912.00513*, 2019.

[70] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, 10(2):1–19, 2019.

[71] Shengwen Yang, Bing Ren, Xuhui Zhou, and Liping Liu. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019.

[72] Haoyu Zhao, Konstantin Burlachenko, Zhize Li, and Peter Richtárik. Faster rates for compressed federated learning with client-variance reduction. *arXiv preprint arXiv:2112.13097*, 2021.

[73] Haoyu Zhao, Zhize Li, and Peter Richtárik. FedPAGE: A fast local stochastic gradient method for communication-efficient federated learning. *arXiv preprint arXiv:2108.04755*, 2021.

[74] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan 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)? [No] The codes are public upon publication.
   (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)? [N/A]
   (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]
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]