Privacy-Preserving Asynchronous Vertical Federated Learning Algorithms for Multiparty Collaborative Learning

Bin Gu, An Xu, Zhouyuan Huo, Cheng Deng, *Senior Member, IEEE*, and Heng Huang

***Abstract*— The privacy-preserving federated learning for vertically partitioned (VP) data has shown promising results as the solution of the emerging multiparty joint modeling application, in which the data holders (such as government branches, private finance, and e-business companies) collaborate throughout the learning process rather than relying on a trusted third party to hold data. However, most of the existing federated learning algorithms for VP data are limited to synchronous computation. To improve the efficiency when the unbalanced computation/communication resources are common among the parties in the federated learning system, it is essential to develop asynchronous training algorithms for VP data while keeping the data privacy. In this article, we propose an asynchronous federated stochastic gradient descent (AFSGD-VP) algorithm and its two variance reduction variants, including stochastic variance reduced gradient (SVRG) and SAGA on the VP data. Moreover, we provide the convergence analyses of AFSGD-VP and its SVRG and SAGA variants under the condition of strong convexity and without any restrictions of staleness. We also discuss their model privacy, data privacy, computational complexities, and communication costs. To the best of our knowledge, AFSGD-VP and its SVRG and SAGA variants are the first asynchronous federated learning algorithms for VP data with theoretical guarantees. Extensive experimental results on a variety of VP datasets not only verify the theoretical results of AFSGD-VP and its SVRG and SAGA variants but also show that our algorithms have much higher efficiency than the corresponding synchronous algorithms.**

***Index Terms*— Asynchronous distributed computation, privacy-preserving, stochastic gradient descent (SGD), vertical federated learning.**

Manuscript received May 2, 2020; revised November 4, 2020; accepted February 19, 2021. *(Corresponding author: Heng Huang.)*

Bin Gu is with the Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates, and also with JD Finance America Corporation, Mountain View, CA 94043 USA (e-mail: bin.gu@mbzuai.ac.ae).

An Xu and Zhouyuan Huo are with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15260 USA (e-mail: an.xu@pitt.edu; zhouyuan.huo@pitt.edu).

Cheng Deng is with the School of Electronic Engineering, Xidian University, Xi'an 710071, China (e-mail: chdeng@mail.xidian.edu.cn).

Heng Huang is with the Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15260 USA, and also with JD Finance America Corporation, Mountain View, CA 94043 USA (e-mail: heng.huang@pitt.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3072238.

Digital Object Identifier 10.1109/TNNLS.2021.3072238

Nomenclature

$\widehat{w}$	w that inconsistently read from different workers.
$\widetilde{w}$	Snapshot of w after a certain number of iterations.
q	Number of workers.
d	Dimensionality of data.
b^ℓ	Random number generated on the ℓth worker.
$\xi(t,\ell)$	Local time counter for the global time counter t on the ℓth worker.
$\xi^{-1}(u,\ell)$	Corresponding global time counter to a local time counter u on the ℓth worker.
$\psi(t)$	Corresponding worker to obtain $\widehat{w}_t^T x_i$.
$\psi^{-1}(\ell,K)$	All the elements in K such that $\psi(\psi^{-1}(\ell,K))=\ell$.
Leaf$(\cdot)$	All leaves of a tree.

I. Introduction

FEDERATED learning facilitates the collaborative model learning without the sharing of raw data, and increasingly attracts attention from both tech giants and industries where privacy protection is required. Especially, in the emerging multiparty joint modeling application, the data locate at multiple (two or more) data holders, and each maintains its own records of different feature sets with common entities, which are called vertically partitioned (VP) data [1]. While an integrated dataset improves the performance of a trained learning model, organizations cannot share data due to legal restrictions or competition between participants. For example, a digital finance company, an E-commerce company, and a bank collect different information about the same person. The digital finance company has access to online consumption, loan, and repayment information. The E-commerce company has access to the online shopping information. The bank has customer information, such as average monthly deposit and account balance. If the person submits a loan application to the digital finance company, it might want to evaluate the credit risk of approving this financial loan by comprehensively utilizing the information stored in all three parties. Such scenarios have been popularly appearing in recent industrial applications and raise the need for efficient federated learning algorithms on the VP data.

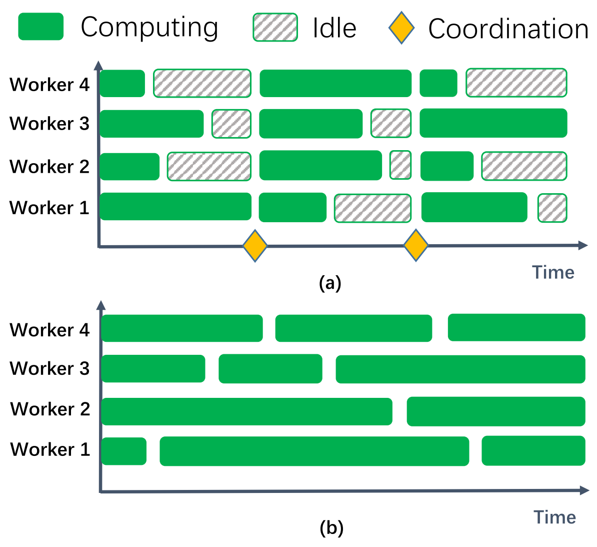

Fig. 1. (a) Synchronous computation versus (b) asynchronous computation.

For the VP data, direct access to the data in other providers or sharing of the data is often prohibited due to legal and commercial issues. For legal reason, most countries worldwide have made laws in the protection of data security and privacy. For example, the European Union made the General Data Protection Regulation (GDPR) [2] to protect users' personal privacy and data security. The recent data breach by Facebook has caused a wide range of protests [3]. For the commercial reason, customer data are usually a valuable business asset for corporations. For example, the real online shopping information of customers can be used to train a recommended model, which could provide valuable product recommendations to customers. Thus, both the causes require federated learning on the VP data without the disclosure of data.

In the literature, there are many privacy-preserving federated learning algorithms for VP data in various applications, for example, cooperative statistical analysis [4], linear regression [5]–[8], association rule-mining [9], k-means clustering [10], logistic regression [11], [12], XGBoost [13], random forest [14], and support vector machine [15]. From the optimization standpoint, Wan *et al.* [16] proposed privacy-preservation gradient descent algorithm for VP data. Zhang *et al.* [17] proposed a feature-distributed SVRG (FD-SVRG) algorithm for high-dimensional linear classification. However, these federated learning algorithms on the VP data are limited to synchronous computation, while the synchronous computation is much more inefficient than the asynchronous computation because it wastes a lot of computing resources to be idle (please see Fig. 1).

The stochastic gradient descent (SGD) algorithm [18] and its variance reduction variants [19]–[22] have been dominant in the training algorithms for solving large-scale machine learning problems. Specifically, each iteration of SGD independently selects a sample and uses the stochastic gradient with respect to the sampled sample to update the solution. The stochasticity makes each iteration of SGD cheap, while it also causes a large variance of stochastic gradients due to random sampling. To reduce the variance of stochastic gradients, the SGD variants with different variance reduction techniques (including stochastic variance reduced gradient (SVRG) [19], SAGA [20], stochastic average gradient (SAG) [21], stochastic recursive gradient (SARAH) [23], and stochastic path-integrated differential estimator (SPIDER) [22]) were proposed to speed up SGD. SVRG and SAGA are the most popular ones among them. In addition, SGD and its adaptive variants (e.g., adaptive gradient (Adagrad), root mean square propagation (RMSProp), and adaptive moment estimation (Adam) [24]) have shown their successes for the training of deep neural networks.

However, it is almost vacant for SGD and its different variance reduction variants to train VP data in parallel and asynchronously while keeping data and model privacy. Please refer to the detailed discussion of this point in Section II. To address this challenging problem, in this article, we propose an asynchronous decentralized federated SGD (AFSGD-VP) algorithm and its SVRG and SAGA variants for VP data. More importantly, we provide the convergence rates of AFSGD-VP and its SVRG and SAGA variants under the condition of strong convexity for the objective function. We also discuss their model privacy, data privacy, computational complexities, and communication costs. To the best of our knowledge, the proposed algorithms are the first asynchronous federated learning algorithms for VP data with theoretical guarantees. Extensive experimental results on a variety of VP datasets not only verify the theoretical results of AFSGD-VP and its SVRG and SAGA variants but also show that our algorithms have much higher efficiency than the corresponding synchronous algorithms. We summarize the main contributions of this article as follows.

1) We propose asynchronous decentralized federated stochastic gradient algorithm (i.e., AFSGD-VP) and its SVRG and SAGA variants for VP data. We provide their convergence rates under the condition of strong convexity.
2) Based on the *semihonest* assumption (i.e., Assumption 7), we prove that our AFSG-VP and its SVRG and SAGA variants can prevent the exact and approximate inference attacks.

II. Related Work

In this section, we give a brief review of asynchronous distributed stochastic optimization algorithms for VP data.

Although there have been a lot of asynchronous distributed stochastic optimization algorithms proposed to solve large-scale learning problems, most of them are limited to federated learning on horizontally partitioned data. Specifically, for the smooth convex optimization problems, Zhao and Li [25] proposed an asynchronous stochastic algorithm with SVRG and proved its linear convergence rate. Mania *et al.* [26] proposed a perturbed iterate framework to analyze the asynchronous stochastic SVRG algorithm with sparse gradients.

Huo and Huang [27] extended the asynchronous stochastic SVRG algorithm to the nonconvex optimization problems and proved its sublinear convergence rate. Leblond *et al.* [28] proposed an asynchronous SAGA algorithm and proved its linear convergence rate. For the convex optimization problems with nonsmooth regularization, Meng *et al.* [29] and Gu *et al.* [30] independently proposed asynchronous stochastic proximal gradient algorithms with SVRG and proved their linear convergence rates. Pedregosa *et al.* [31] proposed an asynchronous stochastic proximal gradient algorithm with SAGA and proved its linear convergence rate. For the nonconvex optimization problems with cardinality constraint, Pedregosa *et al.* [31] proposed an asynchronous stochastic proximal gradient algorithm with SAGA and proved its linear convergence rate. Li *et al.* [32] proposed an asynchronous stochastic gradient hard thresholding algorithm with the SVRG and SAGA techniques and proved the linear convergence rate to an approximately global optimum for the SVRG case. Kungurtsev *et al.* [33] proposed asynchronous stochastic subgradient methods for general nonsmooth nonconvex optimization problems and provided the corresponding convergence analyses.

FD-SVRG [17] is the first work of privacy-preservation SGD-like methods for VP data. However, the updating rules in FD-SVRG [17] are executed synchronously. To the best of our knowledge, FDML [34] is the only work of asynchronous distributed SGD-like algorithm for VP data. *However, FDML cannot guarantee convergence due to its strong dependence on the value of bounded staleness, while the theoretical value of bounded staleness is unknown for a real-world system.* Thus, FDML inevitably leads to the loss of expressiveness in the practical implementation, which is caused by the inconsistency between the theoretical and actual values of bounded staleness. Our experimental results in Fig. 6 also verify this issue by comparing the classification accuracy. On the contrary, our proposed algorithms are lossless because we do not have any restrictions on the bounded staleness in our theoretical analysis and practical implementation. We also summarize other main differences between our proposed algorithms and FDML as follows.

1) Although FDML and our proposed algorithms are all asynchronous distributed algorithms, the FDML algorithm is a master–slave algorithm, while our proposed algorithms are decentralized algorithms. Decentralized architecture is more robust than the master–slave architecture, which is the static and special case of decentralized architecture. More importantly, we derive the theoretical convergence rates for our decentralized algorithms.
2) The FDML algorithm supports a diminishing step size. However, our proposed algorithms support fixed step sizes, which will make our algorithms more suitable for asynchronous distributed algorithms because we do not need to adjust the step sizes for each iteration.
3) The FDML algorithm only supports SGD, but our algorithms can support SVRG and SAGA in addition to SGD.

III. Asynchronous Federated Learning for Vertically Partitioned Data

In this section, we first introduce the problem addressed in this article and then give a brief review of SGD, SVRG, and SAGA. Next, we give the system structure of our asynchronous federated learning algorithms. Finally, we propose our AFSGD-VP, AFSVRG-VP, and AFSAGA-VP algorithms.

A. Problem Statement

In this article, we consider the model in a linear form of $w^T x$. Given a training set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{l}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$ for binary classification or $y_i \in \mathbb{R}$ for regression,[1] the loss function with respect to the sample (x_i, y_i) and the model weights w can be formulated as $L(w^T x_i, y_i)$. Thus, we consider to optimize the following regularized empirical risk minimization problem:

$$\min_{w\in\mathbb{R}^d} f(w) = \frac{1}{l}\sum_{i=1}^{l} \underbrace{L(w^T x_i, y_i) + g(w)}_{f_i(w)} \tag{1}$$

where $g(w)$ is a regularization term, and each $f_i : \mathbb{R}^d \to \mathbb{R}$ is considered as a smooth, possibly nonconvex function in this article. It is obvious that the empirical risk minimization problem is a special case of the problem (1). In addition to the empirical risk minimization problem, problem (1) summarizes an extensive number of important regularized learning problems, such as ℓ_2-regularized logistic regression [35], ridge regression [36], and least-squares support vector machine [37].

As mentioned previously, in a lot of real-world machine learning applications, the input of training sample (x, y) is partitioned vertically into q parts, i.e., we have a partition $\{\mathcal{G}_1, \ldots, \mathcal{G}_q\}$ of d features. Thus, we have $x = [x_{\mathcal{G}_1}, x_{\mathcal{G}_2}, \ldots, x_{\mathcal{G}_q}]$, where $x_{\mathcal{G}_\ell} \in \mathbb{R}^{d_\ell}$ is stored on the ℓth worker, and $\sum_{\ell=1}^{q} d_\ell = d$. According to whether the label is included in a worker, we divide the workers into two types: one is the active worker and the other is passive worker, where the active worker is the data provider who holds the label of a sample beside the partial input of a sample, and the passive worker only has the partial input of a sample without label information. The active worker would be a dominating server in federated learning, while passive workers play the role of clients [13]. We let D^ℓ denote the data stored on the ℓth worker. Note that the labels y_i are distributed to active workers. Our goal in this article can be presented as follows.

Goal: *Make active workers cooperate with passive workers to solve the regularized empirical risk minimization problem (1) on the VP data $\{D^\ell\}_{\ell=1}^{q}$ in parallel and asynchronously with the SGD and its SVRG and SAGA variants while keeping the VP data private.*

[1]We do not force the label y from $\{+1, -1\}$ or $\mathbb{R}$ for binary classification or 1-dim regression problems. In fact, the label y can be many other choices even multidimensional problems, such as ℓ_2-norm regularized multinomial logistic regression problem (21) as long as the corresponding loss function is defined well.

B. Brief Review of SGD, SVRG, and SAGA

As mentioned before, SGD-like algorithms have been popular algorithms for solving large-scale machine learning problems. We first give a brief review of the update framework of SGD-like algorithms that include multiple variants of variance reduction methods. Specifically, given an unbiased stochastic gradient v (i.e., $\mathbb{E}v = \nabla f(w)$), the updating rule of SGD-like algorithms can be formulated as follows:

$$w \leftarrow w - \gamma v \tag{2}$$

where γ is the learning rate. In the following, we present the specific forms to the unbiased stochastic gradient v *with respect to* SGD, SVRG, and SAGA.

1) SGD: Each iteration of SGD [18] independently selects a sample (x_i, y_i) and uses the stochastic gradient $\nabla f_i(w)$ with respect to the sampled sample (x_i, y_i) to update the solution as follows:

$$v = \nabla f_i(w). \tag{3}$$

2) SVRG: For SVRG [19], [38], instead of directly using the stochastic gradient $\nabla f_i(w)$, they use an unbiased stochastic gradient v as follows to update the solution:

$$v = \nabla f_i(w) - \nabla f_i(\widetilde{w}) + \nabla f(\widetilde{w}) \tag{4}$$

where $\widetilde{w}$ denotes the snapshot of w after a certain number of iterations.

3) SAGA: For SAGA [20], the unbiased stochastic gradient v is formulated as follows:

$$v = \nabla f_i(w) - \alpha_i + \frac{1}{l}\sum_{i=1}^{l} \alpha_i \tag{5}$$

where α_i is the latest historical gradient of $\nabla f_i(w)$, which can be updated in an online fashion.

C. System Structure of Our Algorithms

As mentioned before, AFSG-VP, AFSVRG-VP, and AFSAGA-VP are privacy-preserving asynchronous federated learning algorithms on the VP data. Fig. 2 presents their system structure. Specifically, we give detailed descriptions of tree-structured communication, and data and model privacy, respectively, as follows.

1) Tree-Structured Communication: To obtain $w^T x_i$, we need to accumulate the local results from different workers. Zhang *et al.* [17] proposed an efficient tree-structured communication scheme to get the global sum, which is faster than the simple strategy of sending the results from all workers directly to the coordinator for sum.[2] Taking four workers as an example, we pair the workers so that, while worker 1 adds the result from worker 2, worker 3 can add the result from worker 4 simultaneously. Finally, the results from the two pairs of workers are sent to the coordinator, and we obtain the global sum [please see Fig. 3(a)]. In this article, we use the tree-structured communication scheme to obtain $w^T x_i$. Note that our tree-structured communication scheme works with the asynchronous pattern to obtain $w^T x_i$, which means that we do not align the iteration numbers of $w_{\mathcal{G}_\ell}$ from different workers to compute $w^T x_i$. It is significantly different from the synchronous pattern used in [17] where all $w_{\mathcal{G}_\ell}$ have one and the same iteration number.

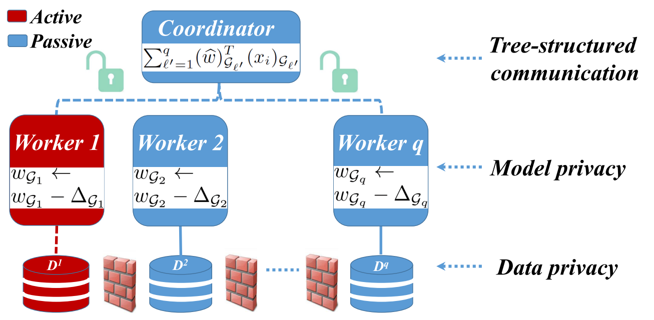
Active
Passive
Coordinator
$\sum_{\ell'=1}^{q}(\widehat{w})^T_{\mathcal{G}_{\ell'}}(x_i)_{\mathcal{G}_{\ell'}}$
Tree-structured communication
Worker 1
$w_{\mathcal{G}_1} \leftarrow w_{\mathcal{G}_1} - \Delta_{\mathcal{G}_1}$
Worker 2
$w_{\mathcal{G}_2} \leftarrow w_{\mathcal{G}_2} - \Delta_{\mathcal{G}_2}$
Worker q
$w_{\mathcal{G}_q} \leftarrow w_{\mathcal{G}_q} - \Delta_{\mathcal{G}_q}$
Model privacy
D^1
D^2
D^q
Data privacy

Fig. 2. System structure of our privacy-preserving asynchronous federated learning algorithms.

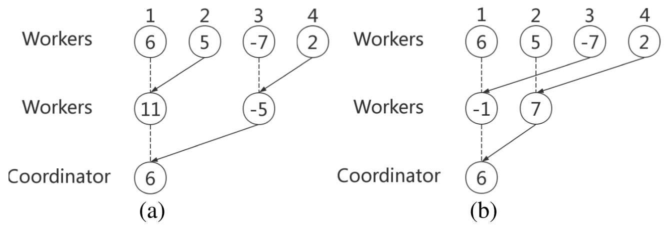

Fig. 3. Illustration of tree-structured communication with two significantly different tree structures. (a) Tree structures T_1. (b) Tree structures T_2.

Based on the tree-structured communication scheme, we summarize the basic algorithm of computing $\sum_{\ell'=1}^{q} w^T_{\mathcal{G}_{\ell'}}(x_i)_{\mathcal{G}_{\ell'}}$ on the ℓth active worker in Algorithm 1.

Algorithm 1 Basic Algorithm of Computing $\sum_{\ell'=1}^{q} w^T_{\mathcal{G}_{\ell'}}(x_i)_{\mathcal{G}_{\ell'}}$ on the ℓth Active Worker

```
Input: $w$, $x_i$
   {This loop asks multiple workers running in parallel.}
1: for $\ell' = 1, \ldots, q$ do
2:    Calculate $w^T_{\mathcal{G}_{\ell'}}(x_i)_{\mathcal{G}_{\ell'}}$.
3: end for
4: Use tree-structured communication scheme to compute $\xi = \sum_{\ell'=1}^{q} w^T_{\mathcal{G}_{\ell'}}(x_i)_{\mathcal{G}_{\ell'}}$.
Output: $\xi$.
```

2) Data and Model Privacy: To keep the VP data and model privacy, we save the data $(x_i)_{\mathcal{G}_\ell}$ and model weights $w_{\mathcal{G}_\ell}$ in the ℓth worker separately and privately. We do not directly transfer the local data $(x_i)_{\mathcal{G}_\ell}$ and local model weights $w_{\mathcal{G}_\ell}$ to other workers. To obtain $w^T x_i$, we locally compute $w^T_{\mathcal{G}_\ell}(x_i)_{\mathcal{G}_\ell}$ and only transfer $w^T_{\mathcal{G}_\ell}(x_i)_{\mathcal{G}_\ell}$ to other workers for computing $w^T x$, as shown in Algorithm 1. It is not trivial to infer the local model coefficients $w_{\mathcal{G}_\ell}$ and $(x_i)_{\mathcal{G}_\ell}$ based on the value of $w^T_{\mathcal{G}_\ell}(x_i)_{\mathcal{G}_\ell}$, which is discussed in detail in Section IV-B. Thus, we achieve the data and model privacy.

[2]The efficiency of different methods of summation is determined by communication efficiency and calculation efficiency. For the method of simple summation, both communication and calculation of summation are burdened by the master worker. However, for the tree-structured communication, the communication and calculation of summation are evenly distributed in different works. Thus, the tree-based structure is faster than simple summation if a lot of such operations are called simultaneously.

Algorithm 2 Safer Algorithm of Computing $\sum_{\ell'=1}^{q} w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$ on the ℓth Active Worker

```
Input: $w$, $x_i$
  {This loop asks multiple workers running in parallel.}
1: for $\ell' = 1, \ldots, q$ do
2:   Generate a random number $b^{\ell'}$.
3:   Calculate $w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}} + b^{\ell'}$.
4: end for
5: Use tree-structured communication scheme based on the tree structure $T_1$ on all workers $\{1, \ldots, q\}$ to compute $\xi = \sum_{\ell'=1}^{q} \left( w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}} + b^{\ell'} \right)$.
6: Use tree-structured communication scheme based on the totally different tree structure $T_2$ on all workers $\{1, \ldots, q\}$ to compute $\overline{b} = \sum_{\ell'=1}^{q} b^{\ell'}$.
Output: $\xi - \overline{b}$.
```

Although it is not trivial to exactly infer the local model coefficients $w_{\mathcal{G}_\ell}$ and $(x_i)_{\mathcal{G}_\ell}$ based on the value of $w_{\mathcal{G}_\ell}^T (x_i)_{\mathcal{G}_\ell}$, it has the risk of approximate inference attack (please refer to Definition 5). To address this issue, we propose a safer algorithm to compute $\sum_{\ell'=1}^{q} w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$ in Algorithm 2. Specifically, we add a random number $b^{\ell'}$ into $w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$ and then use the tree-structured communication scheme on a tree structure T_1 to compute $\sum_{\ell'=1}^{q} (w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}} + b^{\ell'})$, which can improve the data and model security for the operation of transferring the value of $w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}} + b^{\ell'}$. Finally, we need to recover the value of $\sum_{\ell'=1}^{q} w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$ from $\sum_{\ell'=1}^{q} (w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}} + b^{\ell'})$. In order to prevent leaking any sum of $b^{\ell'}$ of a subtree of T_1, we use a *significantly different tree structure* T_2 on all workers (please see Definition 1 and Fig. 3) to compute $\overline{b} = \sum_{\ell'=1}^{q} b^{\ell'}$.

Definition 1 (Two Significantly Different Tree Structures): For two tree structures T_1 and T_2 on all workers $\{1, \ldots, q\}$, they are significantly different if there does not exist a subtree $\widehat{T}_1$ of T_1 and a subtree $\widehat{T}_2$ of T_2 whose sizes are larger than 1 and smaller than T_1 and T_2, respectively, such that $\text{Leaf}(\widehat{T}_1) = \text{Leaf}(\widehat{T}_2)$.

D. Algorithms

In this section, we propose our three asynchronous federated stochastic gradient algorithms (i.e., AFSG-VP, AFSVRG-VP, and AFSAGA-VP) on the VP data.

1) AFSGD-VP: AFSGD-VP repeats the following four steps concurrently for each worker without any lock.

1) *Pick Up an Index:* AFSGD-VP picks up an index i randomly from $\{1, \ldots, l\}$ and obtains the local instance $(x_i)_{\mathcal{G}_\ell}$ from the local data D^ℓ.
2) *Compute $\widehat{w}^T x_i$:* AFSGD-VP uses the tree-structured communication scheme with asynchronous pattern (i.e., Algorithm 1 or 2) to obtain $\widehat{w}^T x_i = \sum_{\ell'=1}^{q} (\widehat{w})_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$, where $\widehat{w}$ denotes w inconsistently read from different workers and two $(\widehat{w})_{\mathcal{G}_{\ell'}}$ may be in different local iteration stages. Note that we always have that $(w)_{\mathcal{G}_\ell} = (\widehat{w})_{\mathcal{G}_\ell}$.
3) *Compute Stochastic Local Gradient:* Based on $\widehat{w}^T x_i$, we can compute the unbiased stochastic local gradient as $\widehat{v}^\ell = \nabla_{\mathcal{G}_\ell} f_i(\widehat{w})$, where $\nabla f_i(w) = ((\partial L(\widehat{w}^T x_i, y_i))/(\partial(\widehat{w}^T x_i)))x_i + \nabla g(\widehat{w})$.
4) *Update:* AFSGD-VP updates the local model weights $w_{\mathcal{G}_\ell}$ by $w_{\mathcal{G}_\ell} \leftarrow w_{\mathcal{G}_\ell} - \gamma \cdot \widehat{v}^\ell$, where γ is the learning rate.

We summarize our AFSGD-VP algorithm in Algorithm 3.

Algorithm 3 AFSGD-VP Algorithm for VP Data on the ℓth Active Worker

```
Input: Local data $D^\ell$, learning rate $\gamma$.
1: Initialize $w_{\mathcal{G}_\ell} \in \mathbb{R}^{d_\ell}$.
2: Keep doing in parallel
3:   Pick up an index $i$ randomly from $\{1, \ldots, l\}$ and obtain the local instance $(x_i)_{\mathcal{G}_\ell}$ from the local data $D^\ell$.
4:   Compute $(w)_{\mathcal{G}_\ell}^T (x_i)_{\mathcal{G}_\ell}$.
5:   Compute $\widehat{w}^T x_i = \sum_{\ell'=1}^{q} (\widehat{w})_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$ based on Algorithm 1 or 2.
6:   Compute $\widehat{v}^\ell = \nabla_{\mathcal{G}_\ell} f_i(\widehat{w})$ based on the value of $\widehat{w}^T x_i$.
7:   Update $w_{\mathcal{G}_\ell} \leftarrow w_{\mathcal{G}_\ell} - \gamma \cdot \widehat{v}^\ell$.
8: End parallel loop
Output: $w_{\mathcal{G}_\ell}$
```

2) AFSVRG-VP: Stochastic gradients in AFSGD-VP have a large variance due to the random sampling similar to the SGD algorithm [18]. To handle the large variance, AFSVRG-VP uses the SVRG technique [19] to reduce the variance of the stochastic gradient and propose a faster AFSGD-VP algorithm (i.e., AFSVRG-VP). We summarize our AFSVRG-VP algorithm in Algorithm 4. Compared to AFSGD-VP, AFSVRG-VP has the following three differences.

1) The first one is that AFSVRG-VP is to compute the full local gradient $\nabla_{\mathcal{G}_\ell} f(w^s) = (1/l) \sum_{i=1}^{l} \nabla_{\mathcal{G}_\ell} f_i(w^s)$ in the outer loop, which will be used as the snapshot of full gradient, where the superscript s denotes the sth out loop.
2) The second one is that we compute not only $\widehat{w}^T x_i$ but also $(w^s)^T x_i$ for each iteration.
3) The third one is that AFSVRG-VP computes the unbiased stochastic local gradient as $\widehat{v}^\ell = \nabla_{\mathcal{G}_\ell} f_i(\widehat{w}) - \nabla_{\mathcal{G}_\ell} f_i(w^s) + \nabla_{\mathcal{G}_\ell} f(w^s)$.

3) AFSAGA-VP: As mentioned above, the stochastic gradients in SGD have a large variance due to the random sampling. To handle the large variance, AFSAGA-VP uses the SAGA technique [20] to reduce the variance of the stochastic gradients. We summarize our AFSAGA-VP algorithm in Algorithm 5. Specifically, we maintain a table of latest historical local gradients α_i^ℓ, which is achieved by the updating rule of $\widehat{\alpha}_i^\ell \leftarrow \nabla_{\mathcal{G}_\ell} f_i(w)$ for each iteration. Based on the table of latest historical local gradients $\widehat{\alpha}_i^\ell$, the unbiased stochastic local gradient in AFSAGA-VP is computed as $\widehat{v}^\ell = \nabla_{\mathcal{G}_\ell} f_i(\widehat{w}) - \widehat{\alpha}_i^\ell + (1/l) \sum_{i=1}^{l} \widehat{\alpha}_i^\ell$.

IV. Theoretical Analyses

In this section, we provide the convergence, security, and complexity analyses to AFSG-VP, AFSVRG-VP, and AFSAGA-VP. All the proofs can be found in the Appendix.

Algorithm 4 Asynchronous Federated SVRG Algorithm (AFSVRG-VP) for VP Data on the ℓ-th Active Worker

```
Input: Local data $D^\ell$, learning rate $\gamma$.
1: Initialize $w^0_{\mathcal{G}_\ell} \in \mathbb{R}^{d_\ell}$.
2: for $s = 0, 1, 2, \ldots, S-1$ do
3:   Compute the full local gradient $\nabla_{\mathcal{G}_\ell} f(w^s) = \frac{1}{l}\sum_{i=1}^{l} \nabla_{\mathcal{G}_\ell} f_i(w^s)$ by using tree-structured communication scheme.
4:   $w_{\mathcal{G}_\ell} = w^s_{\mathcal{G}_\ell}$.
5:   Keep doing in parallel
6:     Pick up a local instance $(x_i)_{\mathcal{G}_\ell}$ randomly from the local data $D^\ell$.
7:     Compute $(w)^T_{\mathcal{G}_\ell}(x_i)_{\mathcal{G}_\ell}$ and $(w^s)^T_{\mathcal{G}_\ell}(x_i)_{\mathcal{G}_\ell}$.
8:     Compute $\widehat{w}^T x_i = \sum_{\ell'=1}^{q} (\widehat{w})^T_{\mathcal{G}_{\ell'}}(x_i)_{\mathcal{G}_{\ell'}}$ and $(w^s)^T x_i = \sum_{\ell'=1}^{q} (w^s)^T_{\mathcal{G}_{\ell'}}(x_i)_{\mathcal{G}_{\ell'}}$ based on Algorithm 1 or 2.
9:     Compute $\widehat{v}^\ell = \nabla_{\mathcal{G}_\ell} f_i(\widehat{w}) - \nabla_{\mathcal{G}_\ell} f_i(w^s) + \nabla_{\mathcal{G}_\ell} f(w^s)$.
10:    Update $w_{\mathcal{G}_\ell} \leftarrow w_{\mathcal{G}_\ell} - \gamma \cdot \widehat{v}^\ell$.
11:  End parallel loop
12:  $w^{s+1}_{\mathcal{G}_\ell} = w_{\mathcal{G}_\ell}$.
13: end for
Output: $w_{\mathcal{G}_\ell}$
```

Algorithm 5 Asynchronous Federated SAGA Algorithm (AFSAGA-VP) for VP Data on the ℓth Active Worker

```
Input: Local data $D^\ell$, learning rate $\gamma$.
1: Initialize $w_{\mathcal{G}_\ell} \in \mathbb{R}^{d_\ell}$.
2: Compute the local gradients $\alpha^\ell_i = \nabla_{\mathcal{G}_\ell} f_i(w)$, $\forall i \in \{1, \ldots, n\}$ by using tree-structured communication scheme, and locally save them.
3: Keep doing in parallel
4:   Pick up a local instance $(x_i)_{\mathcal{G}_\ell}$ randomly from the local data $D^\ell$.
5:   Compute $(w)^T_{\mathcal{G}_\ell}(x_i)_{\mathcal{G}_\ell}$.
6:   Compute $\widehat{w}^T x_i = \sum_{\ell=1}^{q} (\widehat{w})^T_{\mathcal{G}_\ell}(x_i)_{\mathcal{G}_\ell}$ based on Algorithm 1 or 2.
7:   Compute $\widehat{v}^\ell = \nabla_{\mathcal{G}_\ell} f_i(\widehat{w}) - \widehat{\alpha}^\ell_i + \frac{1}{l}\sum_{i=1}^{l} \widehat{\alpha}^\ell_i$.
8:   Update $w_{\mathcal{G}_\ell} \leftarrow w_{\mathcal{G}_\ell} - \gamma \cdot \widehat{v}^\ell$.
9:   Update $\widehat{\alpha}^\ell_i \leftarrow \nabla_{\mathcal{G}_\ell} f_i(\widehat{w})$.
10: End parallel loop
Output: $w_{\mathcal{G}_\ell}$.
```

A. Convergence Analyses

We first make several basic assumptions and then provide the results of the convergence of AFSG-VP, AFSVRG-VP, and AFSAGA-VP.

1) Preliminaries: In this part, we give the assumptions of strong convexity (see Assumption 1), different Lipschitz smoothness (see Assumption 2), and block-coordinate bounded gradients (see Assumption 3), which are standard for convex analysis [20], [25], [38]–[41].

Assumption 1 (Strong Convexity): The differentiable function f_i ($\forall i \in \{1, \ldots, l\}$ in the problem (1) is strongly convex with parameter $\mu > 0$, which means that, $\forall w$ and $\forall w'$, we have

$$f_i(w) \geq f_i(w') + \langle \nabla f_i(w'), w - w' \rangle + \frac{\mu}{2}\|w - w'\|^2. \quad (6)$$

Assumption 2 (Lipschitz Smoothness): The function f_i ($\forall i \in \{1, \ldots, l\}$ in the problem (1) is Lipschitz smooth with constant L, which means that, $\forall w$ and $\forall w'$, we have

$$\|\nabla f_i(w) - \nabla f_i(w')\| \leq L\|w - w'\|. \quad (7)$$

The function f_i ($\forall i \in \{1, \ldots, l\}$ in the problem (1) is block-coordinate Lipschitz smooth with respect to the ℓth block $\mathcal{G}_\ell$ with constant L_ℓ such that, $\forall w$ and $\forall \ell \in \{1, \ldots, q\}$, we have

$$\|\nabla_{\mathcal{G}_\ell} f_i(w + \mathbf{U}_\ell \Delta_\ell) - \nabla_{\mathcal{G}_\ell} f_i(w)\| \leq L_\ell \|\Delta_\ell\| \quad (8)$$

where $\Delta_\ell \in \mathbb{R}^{d_\ell}$, $\mathbf{U}_\ell \in \mathbb{R}^{d \times d_\ell}$, and $[\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_q] = \mathbf{I}_d$.

According to the definition of block-coordinate Lipschitz smooth constant L_ℓ in Assumption 2, we define $L_{\max} = \max_{\ell=1,\ldots,q} L_\ell$. Furthermore, we have $L \leq qL_{\max}$ that is proven in [42, Lemma 2].

Assumption 3 (Block-Coordinate Bounded Gradients): For smooth function $f_i(x)$ ($\forall i \in \{1, \ldots, l\}$) in (1), the block-coordinate gradient $\nabla_{\mathcal{G}_\ell} f_i(w)$ is called bounded if there exists a parameter G such that $\|\nabla_{\mathcal{G}_\ell} f_i(w)\|^2 \leq G$ $\forall i \in \{1, \ldots, l\}$ and $\forall \ell \in \{1, \ldots, q\}$.

2) Difficulties: In this part, we discuss the difficulties of globally labeling the iterates, global updating rules, and the relationship between w_t and $\widehat{w}_t$.

a) Globally labeling the iterates: As shown in Algorithms 3–5, we do not give the global iteration number for the loops of these algorithms run in different workers. Although it is fine for the implementation, how we define the global iteration counter t to label an iterate in one worker definitely matters in the analysis. More specifically, the global iteration counter plays a fundamental role in the convergence rate analyses of AFSG-VP, AFSVRG-VP, and AFSAGA-VP. To address this issue, we propose the strategy of "after communication" labeling [28], in which we update our iterate counter as one worker finishes computing $\widehat{w}^T x_i$. This means that $\widehat{w}_t$ (or $\widehat{w}^s_t$) is the $(t+1)$th fully completed the computation of $\widehat{w}^T x_i$. The strategy of "after communication" labeling guarantees both that the i_t are uniformly distributed and that i_t and $\widehat{w}_t$ are independent.

We define a minimum set of successive iterations of fully visiting all coordinates from the iteration counter t as $K(t)$ in Definition 2.

Definition 2 (Set K(t)): Let $\overline{K}(t) = \{\{t, t+1, \ldots, t+\sigma\} : \psi(\{t, t+1, \ldots, t+\sigma\}) = \{1, \ldots, q\}\}$. The minimum set of successive iterations of fully visiting all coordinates from the iteration counter t is defined as $K(t) = \text{argmin}_{K'(t) \in \overline{K}(t)} |K'(t)|$.

Let $\psi^{-1}(\ell, K)$ denote all the elements in K such that $\psi(\psi^{-1}(\ell, K)) = \ell$. We assume that there exists an upper bound η_1 to the size of $\psi^{-1}(\ell, K(t))$ (see Assumption 4).

Assumption 4 (Bounded Size of $\psi^{-1}(\ell, K(t))$): $\forall t$ and $\forall \ell \in \{1, \ldots, q\}$, the sizes of all $\psi^{-1}(\ell, K(t))$ are upper bounded by η_1, i.e., $|\psi^{-1}(\ell, K(t))| \leq \eta_1$.

Based on the definition of $K(t)$, we define the epoch number of fully visiting all coordinates for the global tth iteration as $v(t)$ and the start iteration counter in one epoch as $\varphi(t)$ in Definition 3. Our convergence rate analyses are built on the epoch number $v(t)$.

Definition 3 (Epoch Number $v(t)$ and Start Iteration Counter $\varphi(t)$): Let $P(t)$ be a partition of $\{0, 1, \ldots, t-\sigma'\}$, where $\sigma' \geq 0$. For any $\kappa \in P(t)$, we have that there exists $t' \leq t$ such that $K(t') = \kappa$, and there exists $\kappa_1 \in P(t)$ such that $K(0) = \kappa_1$. The epoch number $v(t)$ is defined as the maximum cardinality of $P(t)$. Given a global iteration counter $u \leq t$, if there exists $\kappa \in P(t)$ such that $u \in \kappa$, we define the start iteration counter $\varphi(t)$ as the minimum element of κ; otherwise, $\varphi(t) = t - \sigma' + 1$.

b) Global updating rule: The updating rules (such as $w_{\mathcal{G}_\ell} \leftarrow w_{\mathcal{G}_\ell} - \gamma \cdot \widehat{v}^\ell$) in Algorithms 3–5 are updating rules locally working on a certain worker. To provide the convergence rate analyses of AFSG-VP, AFSVRG-VP, and AFSAGA-VP, we need provide the global updating rules of AFSG-VP, AFSVRG-VP, and AFSAGA-VP. Due to the commutativity of the add operations used in $w_{\mathcal{G}_\ell} \leftarrow w_{\mathcal{G}_\ell} - \gamma \cdot \widehat{v}^\ell$, the order in which these updates are finished in the corresponding worker is irrelevant. Hence, we provide the global updating rules of AFSG-VP, AFSVRG-VP, and AFSAGA-VP as follows:

$$w_{t+1} = w_t - \gamma \mathbf{U}_{\psi(t)} \widehat{v}_t^{\psi(t)}. \quad (9)$$

Note that the global updating rule (9) that defines the relation of two adjacent iterates does not conflict with the rule of globally labeling the iterates due to the commutativity of the add operations.

c) Relationship between w_t and $\widehat{w}_t$: As mentioned before, AFSG-VP, AFSVRG-VP, and AFSAGA-VP use the tree-structured communication scheme with asynchronous pattern to obtain $\widehat{w}^T x_i = \sum_{\ell'=1}^{q} (\widehat{w})_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$, where $\widehat{w}$ denotes w inconsistently read from different workers. Thus, the vector $(\widehat{w}_t)_{\mathcal{G}_{\ell'}}$ for $\ell' \neq \ell$ may be inconsistent to the vector $(w_t)_{\mathcal{G}_{\ell'}}$, which means that some blocks of $\widehat{w}_t$ are same with the ones in w_t (e.g., $(w)_{\mathcal{G}_\ell} = (\widehat{w})_{\mathcal{G}_\ell}$), but others are different to the ones in w_t. To address the challenge, we assume an upper bound to the delay of updating. Specifically, we define a set $D(t)$ of iterations such that

$$\widehat{w}_t - w_t = \gamma \sum_{u \in D(t)} \mathbf{U}_{\psi(u)} \widehat{v}_u^{\psi(u)} \quad (10)$$

where, $\forall u \in D(t)$, we have $u < t$. It is reasonable to assume that there exists an upper bound τ such that $\tau \geq t - \min\{t' | t' \in D(t)\}$ (i.e., Assumption 5).

Assumption 5 (Bounded Overlap): There exists an upper bound τ such that $\tau \geq t - \min\{u | u \in D(t)\}$ for all iterations t in AFSG-VP, AFSVRG-VP, and AFSAGA-VP.

In addition, we assume that there exist an upper bound η_2 to the size of $\psi^{-1}(\ell, D(t))$ (see Assumption 6).

Assumption 6 (Bounded Size of $\psi^{-1}(\ell, D(t))$): $\forall t$ and $\forall \ell \in \{1, \ldots, q\}$, the sizes of all $\psi(\psi^{-1}(\ell, D(t)))$ are upper bounded by η_2, i.e., $|\psi(\psi^{-1}(\ell, K))| \leq \eta_2$.

3) AFSGD-VP: We provide the convergence result of AFSGD-VP in Theorem 1.

Theorem 1: Under Assumptions 1–6, to achieve the accuracy ϵ of (1) for AFSGD-VP, i.e., $\mathbb{E} f(w_t) - f(w^*) \leq \epsilon$, we set

$$\gamma = \frac{-L_{\max} + \sqrt{L_{\max}^2 + \frac{2\mu\epsilon\left(L^2 q \eta_1^2 + \eta_2 L^2 \tau\right)}{G \eta_1 q}}}{2L^2\left(q\eta_1^2 + \eta_2 \tau\right)} \quad (11)$$

and the epoch number $v(t)$ should satisfy the following condition:

$$v(t) \geq \frac{2}{\mu} \frac{2L^2\left(q\eta_1^2 + \eta_2\tau\right)}{-L_{\max} + \sqrt{L_{\max}^2 + \frac{2\mu\epsilon\left(L^2 q\eta_1^2 + \eta_2 L^2 \tau\right)}{G\eta_1 q}}} \cdot \log\left(\frac{2(f(w_0) - f(w^*))}{\epsilon}\right). \quad (12)$$

Remark 1: Theorem 2 shows that the convergence rate of AFSGD-VP is $\mathcal{O}((1/\sqrt{\epsilon})\log(1/\epsilon))$ to reach the accuracy ϵ. The theorem shows that, if we try to obtain a more accurate solution with a smaller step size, the convergence rate slows down.

4) AFSVRG-VP: We provide the convergence result of AFSVRG-VP in Theorem 2.

Theorem 2: Under Assumptions 1–6, to achieve the accuracy ϵ of (1) for AFSVRG-VP, i.e., $\mathbb{E} f(w_t) - f(w^*) \leq \epsilon$, let $C = (\eta_1 \gamma L^2 q \eta_1 + L_{\max})(\gamma^2/2)$ and $\rho = (\gamma\mu/2) - ((16 L^2 \eta_1 q C)/\mu)$, and we choose γ such that

$$\rho < 0 \quad (13)$$

$$\frac{8L^2 \eta_1 q C}{\rho\mu} \leq 0.5 \quad (14)$$

$$\gamma^3 \left(\left(\frac{1}{2} + \frac{2C}{\gamma}\right)\eta_2 \tau + 4\frac{C}{\gamma}\eta_1^2 q\right) \frac{\eta_1 q L^2 G}{\rho} \leq \frac{\epsilon}{8} \quad (15)$$

the inner epoch number should satisfy $v(t) \geq ((\log 0.25)/(\log(1-\rho)))$, and the outer loop number should satisfy $S \geq ((\log((2(f(w_0) - f(w^*)))/\epsilon))/(\log(4/3)))$.

Remark 2: Theorem 2 shows that the convergence rate of AFSVRG-VP is $\mathcal{O}(\log(1/\epsilon))$ to reach the accuracy ϵ.

5) AFSAGA-VP: We provide the convergence result of AFSAGA-VP in Theorem 3.

Theorem 3: Under Assumptions 1–6, to achieve the accuracy ϵ of (1) for AFSAGA-VP, i.e., $\mathbb{E} f(w_t) - f(w^*) \leq \epsilon$, let $c_0 = (((\eta_2/2) + 3(\gamma q \eta_1^2 + L_{\max})(\eta_1 + 2\eta_2))\tau + (\gamma L^2 q \eta_1^2 + 8 L_{\max})\eta_1 q \eta_1)\gamma^4 L^2 \eta_1 q G$, $c_1 = (\gamma L^2 q \eta_1^2 + L_{\max})\gamma^2 \eta_1 q 2 L^2$, $c_2 = 4(\gamma L^2 q \eta_1^2 + L_{\max})((L^2 \eta_1^2 q)/l)\gamma^2$, and $\rho \in (1 - (1/l), 1)$; we choose γ such that

$$\frac{4c_0}{\gamma\mu(1-\rho)\left(\frac{\gamma\mu^2}{4} - 2c_1 - c_2\right)} \leq \frac{\epsilon}{2} \quad (16)$$

$$0 < 1 - \frac{\gamma\mu}{4} < 1 \quad (17)$$

$$-\frac{\gamma\mu^2}{4} + 2c_1 + c_2\left(1 + \frac{1}{1 - \frac{1-\frac{1}{l}}{\rho}}\right) \leq 0 \quad (18)$$

$$-\frac{\gamma\mu^2}{4} + c_2 + c_1\left(2 + \frac{1}{1 - \frac{1-\frac{1}{l}}{\rho}}\right) \leq 0. \quad (19)$$

The epoch number should satisfy $v(t) \geq ((\log((2(2\rho - 1 + (\gamma\mu/4))(f(w_0) - f(w^*)))/(\epsilon(\rho - 1 + (\gamma\mu/4))((\gamma\mu^2/4) - 2c_1 - c_2))))/(\log(1/\rho)))$.

Remark 3: Theorem 2 shows that the convergence rate of AFSAGA-VP is $\mathcal{O}(\log(1/\epsilon))$ to reach the accuracy ϵ.

B. Security Analysis

We discuss the data and model security (in other words, prevent local data and model on one worker leaked to or inferred by other workers) of AFSG-VP, AFSVRG-VP, and AFSAGA-VP under the *semihonest* assumption. Note that the *semihonest* assumption (i.e., Assumption 7) is commonly used in previous works [11], [13], [16].

Assumption 7 (Semihonest Security): All workers will follow the algorithm to perform the correct computations. However, they may retain records of the intermediate computation results that they may use later to infer the other work's data and model.

Before discussing the data and model privacy in detail, we first introduce the concepts of exact and approximate inference attacks in Definitions 4 and 5.

Definition 4 (Exact Inference Attack): An exact inference attack on the ℓth worker is to exactly infer some feature group $\mathcal{G}$ of one sample x or model w, which belongs from other workers without directly accessing it.

Definition 5: (ϵ-Approximate Inference Attack): An ϵ-approximate inference attack on the ℓth worker is to infer some feature group $\mathcal{G}$ of one sample x (model w) as $\widehat{x}_{\mathcal{G}}$ ($\widehat{w}_{\mathcal{G}}$) with the accuracy of ϵ (i.e., $\|\widehat{x}_{\mathcal{G}} - x_{\mathcal{G}}\|_\infty \leq \epsilon$ or $\|\widehat{w}_{\mathcal{G}} - w_{\mathcal{G}}\|_\infty \leq \epsilon$), which belongs from other workers without directly accessing it.

1) Security Analysis Based on Algorithm 1: First, we show that AFSG-VP, AFSVRG-VP, and AFSAGA-VP based on Algorithm 1 can prevent the exact inference attack and, however, has the risk of approximate inference attack.

Specifically, in order to infer the information of $(w_t)_{\mathcal{G}_\ell}$ on the ℓ'th worker where $\ell' \neq \ell$, we only have a sequence of linear system of $o_t = (w_t)_{\mathcal{G}_\ell}^T (x_i)_{\mathcal{G}_\ell}$ with a sequence of trials of $(x_i)_{\mathcal{G}_\ell}$ and o_t, while only o_t are known. Thus, it is impossible to infer the exact information of $(w_t)_{\mathcal{G}_\ell}$ from the linear system of $o_t = (w_t)_{\mathcal{G}_\ell}^T (x_i)_{\mathcal{G}_\ell}$ even though the size of feature group $\mathcal{G}_\ell$ is one. Similarly, we cannot infer the exact information of $(x_i)_{\mathcal{G}_\ell}$.

However, it has the potential to approximately infer $(w_t)_{\mathcal{G}_\ell}$ from the linear system of $o_j = w_{\mathcal{G}_\ell}^T (x_i)_{\mathcal{G}_\ell}$ if the size of feature group $\mathcal{G}_\ell$ is one. Specifically, if we know the region of $(x_i)_{\mathcal{G}_\ell}$ as $\mathcal{I}$, we can have that $o_j / w_{\mathcal{G}_\ell} \in \mathcal{I}$, which can infer $w_{\mathcal{G}_\ell}$ approximately. Furthermore, we can infer $(x_i)_{\mathcal{G}_\ell}$ approximately. We say that Algorithm 1 has the risk of approximate inference attack.

2) Security Analysis Based on Algorithm 2: Next, we show that AFSG-VP, AFSVRG-VP, and AFSAGA-VP based on Algorithm 2 can prevent the approximate inference attack.

As discussed above, the key to preventing the approximate inference attack is to mask the value of o_j. As described in lines 2 and 3 of Algorithm 2, we add an extra random variable $b^{\ell'}$ into $w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}}$ and transfer the value of $w_{\mathcal{G}'_\ell}^T (x_i)_{\mathcal{G}'_\ell} + b^{\ell'}$ to another worker. This operation makes the received part cannot directly get the value of o_j. Finally, the ℓth active worker gets the global sum $\sum_{\ell'=1}^q (w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}} + b^{\ell'})$ by using a tree-structured communication scheme based on the tree structure T_1. Thus, lines 2–5 of Algorithm 2 keep data privacy.

Line 6 of Algorithm 2 is trying to get $w^T x$ by removing $\overline{b} = \sum_{\ell'=1}^q b^{\ell'}$ from the sum $\sum_{\ell'=1}^q (w_{\mathcal{G}_{\ell'}}^T (x_i)_{\mathcal{G}_{\ell'}} + b^{\ell'})$. To prove that Algorithm 2 can reduce the risk of approximate inference attack, we only need to prove that the calculation of $\overline{b} = \sum_{\ell'=1}^q b^{\ell'}$ in line 6 of Algorithm 2 does not disclose the value of $b^{\ell'}$ or the sum of $b^{\ell'}$ on a node of tree T_1 (please see Lemma 1; the proof is provided in the Appendix).

Lemma 1: Using a tree structure T_2 on all workers, which is significantly different to the tree T_1 to compute $\overline{b} = \sum_{\ell'=1}^q b^{\ell'}$, there is no risk to disclose the value of $b^{\ell'}$ or the sum of $b^{\ell'}$ on all nodes of a subtree of T_1 whose sizes are larger than 1 and smaller than T_1.

C. Complexity Analysis

We give the computational complexities and communication costs of AFSG-VP, AFSVRG-VP, and AFSAGA-VP as follows.

The computational complexity for one iteration of AFSGD-VP is $\mathcal{O}(d+q)$. Thus, the total computational complexity of AFSGD-VP is $\mathcal{O}((d+q)t)$, where t denotes the iteration number. Furthermore, the communication cost for one iteration of AFSGD-VP is $\mathcal{O}(q)$, and the total communication cost is $\mathcal{O}(qt)$.

For AFSVRG-VP, the computational complexity and communication cost of line 3 are $\mathcal{O}((d+q)l)$ and $\mathcal{O}(ql)$, respectively. Assume that the inner loop number of AFSVRG-VP is t. Thus, the total computational complexity of AFSVRG-VP is $\mathcal{O}((d+q)(l+t)S)$, and the communication cost is $\mathcal{O}(q(l+t)S)$.

For AFSAGA-VP, the computational complexity and communication cost of line 2 are $\mathcal{O}((d+q)l)$ and $\mathcal{O}(ql)$, respectively. Assume that the loop number of AFSAGA-VP is t. Thus, the total computational complexity of AFSAGA-VP is $\mathcal{O}((d+q)(l+t))$, and the communication cost is $\mathcal{O}(q(l+t))$.

V. Experimental Results

In this section, we first present the experimental setup and then provide the experimental results and discussions.

A. Experimental Setup

1) Design of Experiments: In the experiments, we not only verify the theoretical results of AFSG-VP, AFSVRG-VP, and AFSAGA-VP but also show that our algorithms have much better efficiency than the corresponding synchronous algorithms (i.e., FSG-VP, FSVRG-VP, and FSAGA-VP). We compare our asynchronous vertical SGD, SVRG, and SAGA algorithms (i.e., AFSG-VP, AFSVRG-VP, and AFSAGA-VP) with synchronous version of vertical SGD, SVRG, and SAGA (denoted as FSG-VP, FSVRG-VP, and FSAGA-VP, respectively) on classification and regression tasks, where FSVRG-VP is almost same to FD-SVRG [17]. For binary classification tasks, we consider the ℓ_2-norm regularized logistic regression model as follows:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \frac{1}{l} \sum_{i=1}^{l} \log(1 + e^{-y_i \mathbf{w}^T x_i}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \qquad (20)$$

TABLE I
DATASETS USED IN THE EXPERIMENTS

	Classification Tasks						Regression Tasks	
	Financial		Large-Scale		Multi-class			
	UCICreditCard	GiveMeSomeCredit	rcv1.binary	url	news20	rcv1.multiclass	E2006-tfidf	YearPredictionMSD
#class	2	2	2	2	20	53	-	-
#Train	24,000	96,257	677,399	1,916,904	15,935	518,571	16,087	463,715
#Test	6,000	24,012	20,242	479,226	3,993	15,564	3,308	51,630
#Feature	90	92	47,236	3,231,961	62,061	47,236	150,360	90

where the data $x_i \in \mathbb{R}^d$ and the corresponding label $y_i = \pm 1$. For multiclass classification task, we consider the ℓ_2-norm regularized multinomial logistic regression model as follows:

$$\min_{\mathbf{w}\in\mathcal{R}^{d\times c}} f(\mathbf{w}) = \frac{1}{l}\sum_{i=1}^{l}\left[\log\left(\sum_{j=1}^{c} e^{\mathbf{w}_{\cdot j}^T x_i}\right) - \sum_{j=1}^{c} Y_{ij}\mathbf{w}_{\cdot j}^T x_i\right] + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \quad (21)$$

where c is the number of classes, $\mathbf{w}_{\cdot j}$ is the jth column of $\mathbf{w}$, label $y_i \in \{1, 2, \ldots, c\}$, and $Y \in \mathcal{R}^{l\times c}$ with $Y_{i,j} = 1$ if $j = y_i$ and $Y_{i,j} = 0$ otherwise. For the regression tasks, we use the ridge linear regression method with ℓ_2-norm regularization as follows:

$$\min_{\mathbf{w},b} f(\mathbf{w}, b) = \frac{1}{l}\sum_{i=1}^{l}(\mathbf{w}^T x_i + b - y_i)^2 + \frac{\lambda}{2}\left(\|\mathbf{w}\|_2^2 + b^2\right). \quad (22)$$

2) Experiment Settings: We run all the experiments on Amazon Cloud EC2 nodes with eight instances in N. Virginia and eight instances in Ohio. Each node has eight Intel Xeon vCPUs, 32-GB memory, and is connected by a 25-GB network. We use OpenMPI [43] v3.1.1 with multithread support for communication between worker processes and Armadillo [44] v9.700.3 for efficient matrix computation. Each worker is placed on a different machine node. For the ℓ_2 regularization term, we set the coefficient $\lambda = 1e^{-4}$ for all experiments. We also choose the best learning rate $\in (5e^{-1}, 1e^{-1}, 5e^{-2}, 1e^{-2}, \ldots)$ for each algorithm on different learning tasks. There is a synthetic straggler node about 200% slower [i.e., the computation speed becomes (1/3)] than regular worker node to simulate the real application scenario. In practice, it is normal that different parties in a federated learning system will possess different computation and communication power and resources.

3) Implementation Details: Our asynchronous algorithms are implemented under the decentralized framework, where a worker owns his/her own part of data and model parameters. There is no master node for aggregating data/features/gradients, which may lead to undesired user information disclosure. Instead, we utilize a coordinator as in Fig. 2 to collect the product computed from local data and parameters from other workers. Each worker node can independently call the coordinator to enable the asynchronous model update. The aggregation of local product is performed in a demand-based manner, which means that, only when a worker node needs to update its local parameter, it will request the coordinator to pull the local product from other worker nodes. Different from horizontal federated learning [1], [45], [46], it will be much harder for an attacker to restore the information of the user data in a worker node using the local product than the gradient.[3]

TABLE II
ASYNCHRONOUS SPEEDUP

Dataset	Speedup		
	SGD	SVRG	SAGA
UCICreditCard	2.14	2.49	1.79
GiveMeSomeCredit	2.40	2.26	2.38
rcv1.binary	2.02	2.30	2.22
url	2.11	2.64	2.05
news20	2.85	1.83	1.95
rcv1.multiclass	2.32	2.81	2.61
E2006-tfidf	2.24	2.12	2.02
YearPredictionMSD	2.26	2.15	2.38

Specifically, in our asynchronous algorithms, each worker node performs computation rather independently. The main thread of a worker process performs the major workload of gradient computation and model update operation. Another listener thread keeps listening for the request and sends back the local product to the requesting source. The computation diagram can be summarized as follows for a worker.

1) Randomly select an index of the data.
2) Call the coordinator to broadcast the index to the listeners of other workers.
3) Reduce the sum of the local product back from the listeners.
4) Perform gradient computation and model parameters update.

Note that the local product is computed based on a worker's current parameters. Overall speaking, however, some workers may have updated their parameters more times than other workers. Different from common asynchronous horizontal algorithms [29], [30], although the worker processes run asynchronously, all the parameters that a worker uses to compute gradient are most up-to-date. The broadcast and reduce operation are also realized in a tree-structured scheme to reduce communication costs.

[3]If the dimensionality of data in the problem (1) is d, the size of the gradient is also d. On the other hand, the size of the local product is one regardless of how large the dimensionality of data, which means that the gradient carries more information of data than the local product. Thus, it is easier to infer the features of original data from gradient than the local product. The data reconstruction attack [45], [47] may reconstruct data with the gradient information by solving an optimization problem using the generative adversarial networks (GANs). As far as we know, there have been no algorithms that can reconstruct data based on the local product.

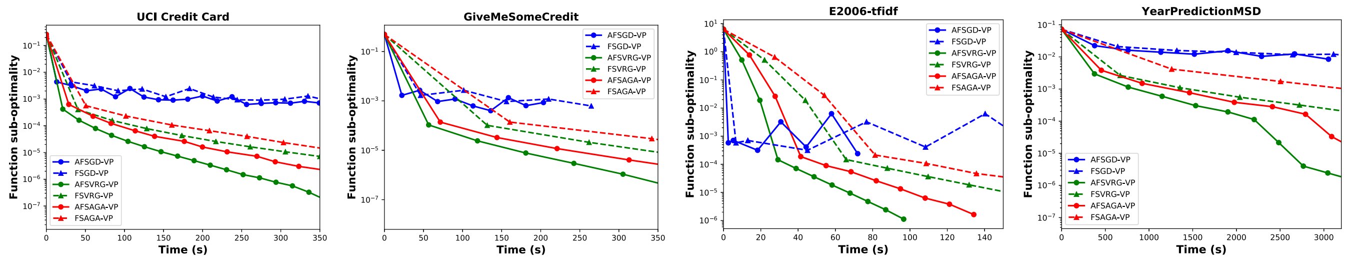

Fig. 4. Convergence of different algorithms for financial data classification task (left two) and regression task (right two).

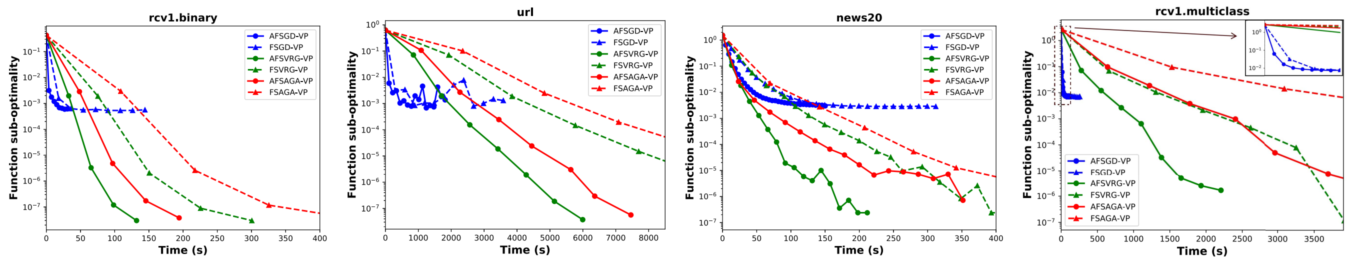

Fig. 5. Convergence of different algorithms for large-scale (left two: binary; right two: multiclass) classification task.

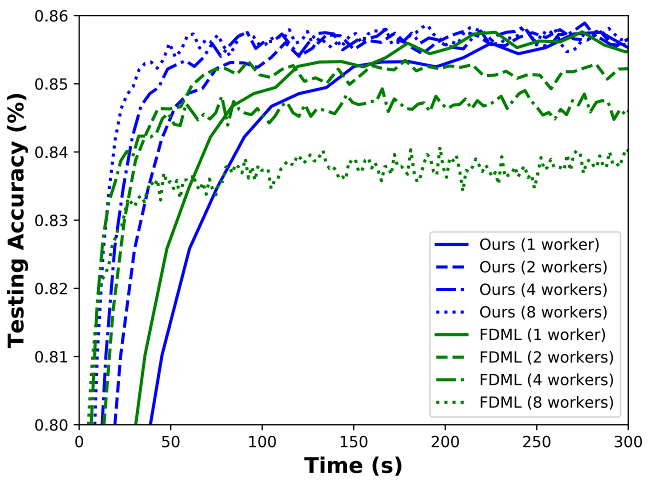

Fig. 6. Comparison of ours and FDML on the multiclass classification tasks.

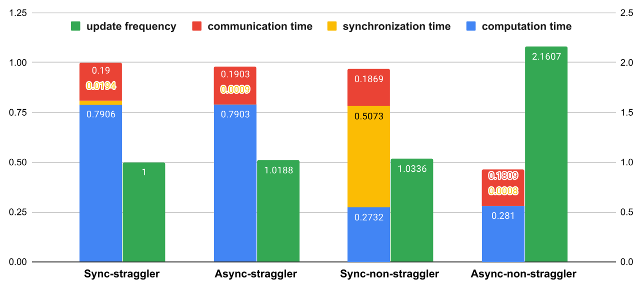

Fig. 7. Illustrate asynchronous efficiency with workload breakdown (eight workers on the url dataset of the binary classification task).

4) Datasets: To fully demonstrate the scalability of our asynchronous vertical federated learning algorithms, we conduct experiments on eight datasets, as summarized in Table I, for binary classification and regression tasks. Two real and relatively small financial datasets, UCICreditCard and GiveMeSomeCredit, are from the Kaggle[4] website. The other six datasets are from the LIBSVM[5] website [48]. We split news20 and url datasets into training data and testing data randomly with a ratio of 4:1. We also use rcv1's testing data for training and training data for testing as there are more instances in the testing data.

B. Result and Discussion

1) Classification Tasks: We first compared our asynchronous federated learning algorithm with the synchronous version on financial datasets to demonstrate the ability to address real applications. In asynchronous algorithms, each worker saves its local parameters every fixed interval for testing. In the synchronous setting, each worker saves the parameters every fixed number of iterations as all the workers run at the same pace. We follow this scheme for the other experiments.

The original total numbers of features of UCICreditCard and GiveMeSomeCredit datasets are 23 and 10, respectively. We apply one-hot encoding for categorical features and standardize other features columnwise. The numbers of features become 90 and 92, respectively, after the simple data preprocessing.

Four worker nodes are used in this part of the experiment. As shown by the left two figures in Fig. 4, our asynchronous vertical algorithms consistently surpass their synchronous counterparts. The y-axis function suboptimality represents the error of objective function to the global optimal. The shape of the convergence curve is first determined by the optimization method that we choose, i.e., SGD, SVRG, and SAGA. The error precision of SGD is usually higher than SVRG, while that of SAGA is similar to SVRG. Then, the convergence speed is mostly influenced by the computation and communication complexity. In asynchronous settings, there is no inefficient idle time to wait for other workers, so the update frequency is much higher, which results in a faster convergence

[4]https://www.kaggle.com/datasets
[5]https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/

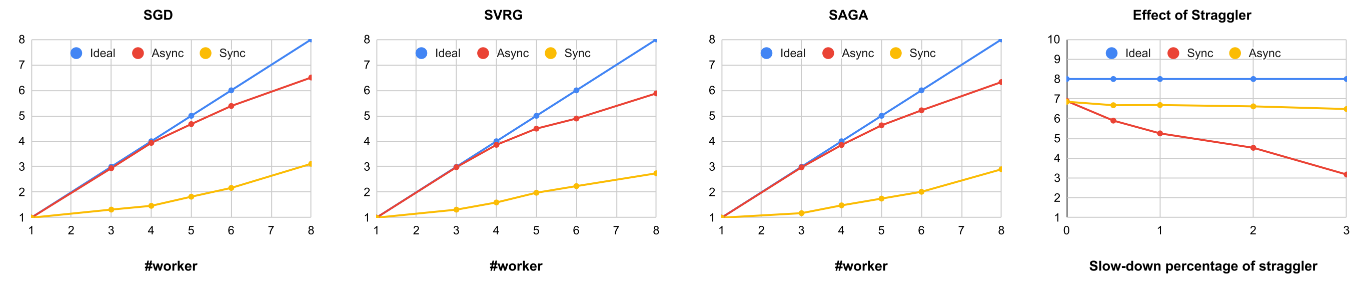

Fig. 8. Scalability (url dataset of binary classification task). The vertical axis represents the speedup.

speed of our asynchronous algorithm with regard to wall clock time.

Previous experiments show that our asynchronous federated learning algorithms could address real financial problems more efficiently. In this part, we will use large-scale benchmark datasets, i.e. a large number of data instances and high-dimensional features, for further validations. In our experiments, eight worker nodes are used for experiments on rcv1.binary, news20, and rcv1.multiclass datasets; 16 worker nodes are used for experiments on the url dataset. The results are visualized in Fig. 5. Our asynchronous SGD, SVRG, and SAGA still surpass their synchronous counterparts in the experiments on all four datasets.

2) Regression Tasks: To further illustrate that the advantages of asynchronous algorithms can scale to various tasks, we also conduct experiments on regression problems, as shown by the right two figures in Fig. 4. Both the E20060-tfidf with a smaller number of data instances but a larger number of features and the YearPredictionMSD with a larger number of instances but a smaller number of features are tested; four worker nodes are used in this experiment, and similar conclusions as previous can be reached.

3) Asynchronous Efficiency: The speedup results of asynchronous algorithms compared with synchronous ones are summarized in Table II. The speedup is computed based on the time when the algorithm reaches a certain precision of optimality ($1 \times e^{-4}$ for SVRG and SAGA; $1 \times e^{-2}$ or $1 \times e^{-3}$ for SGD based on different datasets).

To further analyze the efficiency of our asynchronous algorithms, we quantify the composition of the time consumption of asynchronous and synchronous algorithms, as shown in Fig. 7. The execution time and update frequency are scaled by those of the straggler in the synchronous algorithm. The computation time of stragglers is much higher than nonstragglers, which leads to a large amount of synchronization time for nonstragglers in synchronous algorithms, while, in our asynchronous algorithms, nonstragglers pull the update-to-date product information from stragglers without waiting for the straggler to finish its current iteration. As a result, the synchronization time is eliminated, and we can achieve a large gain in terms of the update frequency.

4) Scalability: The scalability in terms of the number of workers is shown in Fig. 8. The right figure shows that asynchronous methods are resilient to the straggler effect. Synchronous algorithms cannot address the problem of stragglers and behave poorly. Using a synchronization barrier keeps nonstragglers inefficiently waiting for the straggler. Our asynchronous algorithms behave like ideal in the beginning as they can address the straggler problem well and deviate from ideal when the number of workers continues to grow because the communication overheads will limit the speedup.

C. Comparison With FDML

We also compare our algorithm with FDML [34] on the multiclass classification task [see (21)], where a server is required to perform a *heuristic* prediction $p_i(\mathbf{w}, x_i)$ based on local predictions from workers with local data $x_i^\ell \in \mathbb{R}^{d_\ell}$ and submodel $\mathbf{w}^\ell \in \mathbb{R}^{d^{\ell \times c}}$ as follows:

$$p_i(\mathbf{w}, x_i) = \sigma\left(\sum_{\ell=1}^{q} a_\ell \alpha\left(\mathbf{w}^\ell, x_i^\ell\right)\right). \tag{23}$$

Specifically, we also let the submodel be the multinomial logistic regression, and $\alpha(\mathbf{w}^\ell, x_i^\ell)$ is the corresponding local prediction from this submodel. The coefficient is $a_\ell = (d_\ell / d)$, and the activation $\sigma(\cdot)$ is the identity function because it is a linear model. The cross-entropy loss as in multinomial logistic regression is used as the loss function $f_i(\cdot)$. Consequently, in FDML, we minimize

$$\min_{\mathbf{w} \in \mathbb{R}^{d^\ell \times c}} \; f(\mathbf{w}) = \frac{1}{l} \sum_{i=1}^{l} f_i(p_i(\mathbf{w}, x_i), y_i) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \tag{24}$$

which is the same as (21) when there is only one worker.

We note that FDML builds a local submodel with (21) for each worker, combines their local predictions at the server node with a theoretical value of bounded staleness, and jointly optimizes the whole model. Importantly, the convergence of FDML strongly depends on a theoretical value of bounded staleness. However, this value is unknown to a practical system. Thus, FDML inevitably leads to the loss of expressiveness in the practical implementation in the case of the inconsistency between the theoretical and actual values of bounded staleness, which is also shown in Fig. 6 by comparing the accuracy. We did not compare the loss because FDML is not exactly optimizing (21) but uses it as the submodel. FDML only proposes the SGD version, so we compare ours with FDML using asynchronous SGD as the training method. The two-worker settings as used in [34] of FDML exhibits a slightly degraded performance compared with the one-worker baseline. This has also been shown in [34] with the logistic regression model. However, when we gradually increase the number of workers, the performance of FDML deteriorates fast, while ours maintains the same. The loss of expressiveness becomes more severe as the number of workers increases, and each worker gets fewer features per sample. In comparison, ours is very scalable in terms of the number of workers as it is exactly optimizing (21).

VI. Conclusion

In this article, we proposed an AFSGD-VP algorithm and its SVRG and SAGA variants for VP data. To the best of our knowledge, these algorithms are the first asynchronous federated learning algorithms for VP data with theoretical guarantees. Importantly, we provided the convergence rates of AFSGD-VP and its SVRG and SAGA variants under the condition of strong convexity for the objective function and without any restrictions of staleness. We also proved the model privacy and data privacy. Extensive experimental results on a variety of VP datasets not only verify the theoretical results of AFSGD-VP and its SVRG and SAGA variants but also show that our algorithms have much better efficiency than the corresponding synchronous algorithms.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12, 2019.

[2] EU. (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council on the Protection of Natural Persons With Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation)*. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT

[3] N. Badshah. (2018). Facebook to contact 87 million users affected by data breach. The Guardian. [Online]. Available: https://www.theguardian.com/technology/2018/apr/08/facebook-to-contact-the-87-million-users-affected-by-data-breach

[4] W. Du and M. J. Atallah, "Privacy-preserving cooperative statistical analysis," in *Proc. 17th Annu. Comput. Secur. Appl. Conf.*, 2001, pp. 102–110.

[5] A. Gascoón *et al.*, "Privacy-preserving distributed linear regression on high-dimensional data," Cryptol. ePrint Arch., Tech. Rep. 2016/892, 2016. [Online]. Available: https://eprint.iacr.org/2016/892

[6] A. F. Karr, X. Lin, A. P. Sanil, and J. P. Reiter, "Privacy-preserving analysis of vertically partitioned data using secure matrix products," *J. Off. Statist.*, vol. 25, no. 1, p. 125, Jan. 2009.

[7] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter, "Privacy preserving regression modelling via distributed computation," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 677–682.

[8] A. Gascón *et al.*, "Privacy-preserving distributed linear regression on high-dimensional data," *Proc. Privacy Enhancing Technol.*, vol. 2017, no. 4, pp. 345–364, Oct. 2017.

[9] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2002, pp. 639–644.

[10] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2003, pp. 206–215.

[11] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv:1711.10677*. [Online]. Available: http://arxiv.org/abs/1711.10677

[12] R. Nock *et al.*, "Entity resolution and federated learning get a federated resolution," 2018, *arXiv:1803.04035*. [Online]. Available: http://arxiv.org/abs/1803.04035

[13] K. Cheng *et al.*, "SecureBoost: A lossless federated learning framework," 2019, *arXiv:1901.08755*. [Online]. Available: http://arxiv.org/abs/1901.08755

[14] Y. Liu, Y. Liu, Z. Liu, J. Zhang, C. Meng, and Y. Zheng, "Federated forest," 2019, *arXiv:1905.10053*. [Online]. Available: http://arxiv.org/abs/1905.10053

[15] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving SVM classification on vertically partitioned data," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*. Berlin, Germany: Springer, 2006, pp. 647–656.

[16] L. Wan, W. K. Ng, S. Han, and V. C. S. Lee, "Privacy-preservation for gradient descent methods," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2007, pp. 775–783.

[17] G.-D. Zhang, S.-Y. Zhao, H. Gao, and W.-J. Li, "Feature-distributed SVRG for high-dimensional linear classification," 2018, *arXiv:1802.03604*. [Online]. Available: http://arxiv.org/abs/1802.03604

[18] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*. Berlin, Germany: Springer, 2010, pp. 177–186.

[19] B. Gu, Z. Huo, C. Deng, and H. Huang, "Faster derivative-free stochastic algorithm for shared memory machines," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1807–1816.

[20] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1646–1654.

[21] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Math. Program.*, vol. 162, nos. 1–2, pp. 83–112, Mar. 2017.

[22] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 689–699.

[23] N. H. Pham, L. M. Nguyen, D. T. Phan, and Q. Tran-Dinh, "Prox-SARAH: An efficient algorithmic framework for stochastic composite nonconvex optimization," 2019, *arXiv:1902.05679*. [Online]. Available: http://arxiv.org/abs/1902.05679

[24] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2016.

[25] S.-Y. Zhao and W.-J. Li, "Fast asynchronous parallel stochastic gradient descent: A lock-free approach with convergence guarantee," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1–7.

[26] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, and M. I. Jordan, "Perturbed iterate analysis for asynchronous stochastic optimization," 2015, *arXiv:1507.06970*. [Online]. Available: http://arxiv.org/abs/1507.06970

[27] Z. Huo and H. Huang, "Asynchronous mini-batch gradient descent with variance reduction for non-convex optimization," in *Proc. AAAI*, 2017, pp. 2043–2049.

[28] R. Leblond, F. Pedregosa, and S. Lacoste-Julien, "ASAGA: Asynchronous parallel saga," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2017, pp. 46–54.

[29] Q. Meng, W. Chen, J. Yu, T. Wang, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic proximal optimization algorithms with variance reduction," 2016, *arXiv:1609.08435*. [Online]. Available: http://arxiv.org/abs/1609.08435

[30] B. Gu, Z. Huo, and H. Huang, "Asynchronous stochastic block coordinate descent with variance reduction," 2016, *arXiv:1610.09447*. [Online]. Available: http://arxiv.org/abs/1610.09447

[31] F. Pedregosa, R. Leblond, and S. Lacoste-Julien, "Breaking the non-smooth barrier: A scalable parallel method for composite optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 56–65.

[32] X. Li, R. Arora, H. Liu, J. Haupt, and T. Zhao, "Nonconvex sparse learning via stochastic optimization with progressive variance reduction," 2016, *arXiv:1605.02711*. [Online]. Available: http://arxiv.org/abs/1605.02711

[33] V. Kungurtsev, M. Egan, B. Chatterjee, and D. Alistarh, "Asynchronous optimization methods for efficient training of deep neural networks with guarantees," 2019, *arXiv:1905.11845*. [Online]. Available: http://arxiv.org/abs/1905.11845

[34] Y. Hu, D. Niu, J. Yang, and S. Zhou, "FDML: A collaborative machine learning framework for distributed features," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2232–2240.

[35] B. Conroy and P. Sajda, "Fast, exact model selection and permutation testing for l2-regularized logistic regression," in *Proc. Artif. Intell. Statist.*, 2012, pp. 246–254.

[36] X. Shen, M. Alam, F. Fikse, and L. Rönnegård, "A novel generalized ridge regression method for quantitative genetics," *Genetics*, vol. 193, no. 4, pp. 1255–1268, Apr. 2013.

[37] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.

[38] L. Xiao and T. Zhang, "A proximal stochastic gradient method with progressive variance reduction," *SIAM J. Optim.*, vol. 24, no. 4, pp. 2057–2075, Jan. 2014.

[39] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2037–2060, Jan. 2013.

[40] X. Li, T. Zhao, R. Arora, H. Liu, and M. Hong, "On faster convergence of cyclic block coordinate descent-type methods for strongly convex minimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 184:1–184:24, 2017.

[41] X. Li, T. Zhao, R. Arora, H. Liu, and M. Hong, "An improved convergence analysis of cyclic block coordinate descent-type methods for strongly convex minimization," in *Proc. 19th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Cadiz, Spain, May 2016, pp. 491–499.

[42] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, Jan. 2012.

[43] R. L. Graham, T. S. Woodall, and J. M. Squyres, "Open MPI: A flexible high performance MPI," in *Proc. Int. Conf. Parallel Process. Appl. Math.* Berlin, Germany: Springer, 2005, pp. 228–239.

[44] C. Sanderson and R. Curtin, "Armadillo: A template-based C++ library for linear algebra," *J. Open Source Softw.*, vol. 1, no. 2, p. 26, Jun. 2016.

[45] Y. Liu, Z. Ma, X. Liu, S. Ma, S. Nepal, and R. H. Deng, "Boosting privately: Privacy-preserving federated extreme boosting for mobile crowdsensing," *CoRR*, vol. abs/1907.10218, Jul. 2019.

[46] J. So, B. Guler, and A. S. Avestimehr, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 441–451, Mar. 2021.

[47] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2512–2520.

[48] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

Bin Gu received the B.S. and Ph.D. degrees in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2005 and 2011, respectively.

He joined the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, in 2010, as a Lecturer, where he was promoted to a Full Professor in 2018. He is currently an Assistant Professor with the Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates. His research interest focus on large-scale optimization in machine learning.

An Xu received the B.E. degree in electrical engineering from Tsinghua University, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Pittsburgh, Pittsburgh, PA, USA, under the supervision of Prof. Heng Huang.

His research interests include machine learning, large-scale optimization, federated learning, and computer vision.

Zhouyuan Huo received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 2014, and the Ph.D. degree in electrical and computer engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 2020, under the supervision of Prof. Heng Huang.

He is currently a Research Scientist with Google, Mountain View, CA, USA. His research interests include machine learning, large-scale optimization, computer vision, and bioinformatics.

Cheng Deng (Senior Member, IEEE) received the B.E., M.S., and Ph.D. degrees in signal and information processing from Xidian University, Xi'an, China, in 2001, 2006, and 2009, respectively.

He is currently a Full Professor with the School of Electronic Engineering, Xidian University. He is the author or a coauthor of more than 50 scientific articles at top venues, including TNNLS, IEEE TMM, IEEE TCYB, IEEE TSMC, IEEE TIP, ICCV, CVPR, IJCAI, and AAAI. His research interests include computer vision, multimedia processing and analysis, and information hiding.

Heng Huang received the B.S. and M.S. degrees from Shanghai Jiao Tong University, Shanghai, China, in 1997 and 2001, respectively, and the Ph.D. degree in computer science from Dartmouth College, Hanover, NH, USA, in 2006.

He is currently the John A. Jurenko Endowed Professor of Computer Engineering with the Electrical and Computer Engineering Department, University of Pittsburgh, Pittsburgh, PA, USA. He is also a Consulting Researcher with JD Finance America Corporation, Mountain View, CA, USA. His research interests include machine learning, data mining, computer vision, and biomedical data science.