2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Asynchronous Byzantine-Robust Stochastic Aggregation with Variance Reduction for Distributed Learning

Zehan Zhu, Yan Huang, Chengcheng Zhao, Jinming Xu†

Abstract— **We consider Byzantine-robust distributed learning with asynchronous participation of clients at a certain probability, where Byzantine clients can send malicious messages to the server. Instead of relying on traditional robust aggregation rules, such as Krum and Median, that can only tolerate a limited proportion of Byzantine clients, we propose an asynchronous Byzantine-robust stochastic aggregation method that employs regularization-based techniques to mitigate Byzantine attacks, and adopts variance-reduced techniques to eliminate the effect of stochastic noise of gradient sampling. Leveraging a properly designed Lyapunov function, we show that the proposed algorithm converges linearly to an error ball that is independent of stochastic gradient variance. Extensive experiments are conducted to show its efficacy in dealing with Byzantine attacks compared to the existing counterparts.**

I. Introduction

Distributed learning [1]–[3] has attracted increasing attention from academia and industry due to its potential in getting rid of the dilemma of data island and protecting data privacy, and has been widely adopted in various application domains. We are particularly interested in distributed learning systems consisting of one central server and multiple clients, where updates of local variables such as stochastic gradients or model parameters are conducted based on local private data kept by each client, while the central server is responsible for aggregating these local variables and broadcasting the aggregated result to clients. However, the distributed nature makes it vulnerable to malicious attacks. Specifically, some clients may be compromised to become Byzantine clients in distributed scenarios, and they can transmit malicious messages to the central server to disrupt the learning process [4]–[6]. The final model learned by the attacked distributed learning system may be invalid [7] or backdoored [8], depending on the attacker's target. Thus, robustifying distributed learning against Byzantine attacks is vital for secure learning.

Most existing Byzantine-robust distributed algorithms primarily rely on two key strategies. The first strategy involves the central server performing robust estimation of the average of regular clients' inputs using robust aggregation rules, and the second strategy is based on regularization techniques. Typical examples using the first strategy include geometric median (Gm) [7], [9], [10], median (Med) [11], [12], trimmed mean [13], [14] and Krum [15], all of which are mainly based on the majority rule. Recently, Gupta et al. [16] proposed a simple yet efficient mechanism termed nearest neighbor mixing (NNM) which could be easily integrated with the aforementioned robust aggregation rules and boost their Byzantine resilience. The main limitation of robust distributed algorithms based on the above aggregation rules is that they require explicit assumptions on the maximum proportion of Byzantine clients.

To avoid the requirement on the proportion of Byzantine clients, regularization-based techniques are widely used to ensure the robustness of algorithms against Byzantine attacks. For instance, total variation (TV)-norm penalized approximation are proposed for robust decentralized optimization [17], where each node aggregates the signs of the differences between its own model and the neighbors' in order to limit the influence of Byzantine attacks. Motivated by [17], Peng et al. [18] applied stochastic subgradient descent to solve TV-norm penalized formulation, generating a Byzantine-robust decentralized stochastic optimization method. Likewise, for master-slaver architecture, l_p-norm penalty term used in [19] forces the clients' local models to be close to the central server's model, and Byzantine-robust distributed stochastic optimization method (RSA) is proposed by applying the stochastic subgradient method to that l_p-norm regularized approximation.

Recently, it has been shown that the stochastic error of gradient sampling has significant impact on the effectiveness of Byzantine-robust distributed optimization algorithms [20]. For instance, in the presence of large amounts of variance, algorithms relying on robust aggregation rules will struggle to distinguish malicious messages of Byzantine clients from the noisy stochastic gradients of regular clients. To address this issue, Wu et al. [20] combined the robust aggregation rule (geometric median) with a popular variance reduction method SAGA [21] which has been proven effective in finite-sum optimization in eliminating the impact of stochastic error of gradient sampling on Byzantine robustness. Guerraoui et al. [22] also showed that using momentum acceleration at the client side can reduce the stochastic sampling variance and strengthen Byzantine resilient aggregation rules such as Krum [15], median [12] and Bulyan [23]. For Byzantine-robust decentralized stochastic optimization problem, Ling et al. [24] introduced variance reduction methods to the algorithm in [18] to eliminate the stochastic error of gradient sampling. However, all the aforementioned Byzantine-robust algorithms only consider an ideal participation pattern where all nodes participate in update at each iteration (full participation), which may not be efficient in practical

This work has been supported in parts by National Natural Science Foundation of China under Grants 62373323, 62003302, 62088101 and in parts by the Young Elite Scientist Sponsorship Program by cast of China Association for Science and Technology under Grant YESS20210158.

†The authors are with the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China. Correspondence to jimmyxu@zju.edu.cn (Jinming Xu).

scenarios due to possible network congestion. There has been few Byzantine-robust methods taking into account asynchronous participation of nodes, such as BASGD [25] and Zeno++ [26]. BASGD combined the aforementioned robust aggregation rules with the introduction of buffers to account for asynchrony, while Zeno++ requires the central server to have access to a validation dataset to evaluate a descent score for each candidate gradient.

Our contributions. In this work, different from the existing literature, we consider Byzantine robustness of more practical asynchronous distributed learning where each client randomly participates in the interaction with the central server at certain probability. To this end, we develop an asynchronous Byzantine-robust method (termed AsynRSA-VR) using regularization techniques, and adopt variance reduction method to eliminate the impact of the gradient sampling variance on Byzantine robustness. The proposed AsynRSA-VR is proven to converge linearly to an error ball independent of stochastic gradient sampling variance. The theoretical analysis differs from that in [19] due to the introduction of asynchronous participation pattern and variance reduction technique, which calls for a careful design of the Lyapunov function. Extensive experiments are conducted to verify the effectiveness of AsynRSA-VR.

II. Problem Formulation and Algorithm Design

In this paper, we are interested in federated finite-sum optimization problem under Byzantine attacks. Consider a system with one central server and n clients, among which q clients are hidden Byzantine attackers. We use $\mathcal{R}$ and $\mathcal{B}$ to denote regular clients set and Byzantine clients set respectively, with $|\mathcal{R}| = r$ and $|\mathcal{B}| = q$. Without loss of generality, we assume that the total number of data samples held by each regular client is J, and the loss of the j-th data sample at the regular client $i \in \mathcal{R}$ with respect to the model parameter $\tilde{x} \in \mathbb{R}^d$ is denoted by $f_i(\tilde{x}; j)$. The goal is to find the optimal solution of the following problem

$$\tilde{x}^* = \arg\min_{\tilde{x}\in\mathbb{R}^d} \sum_{i\in\mathcal{R}} f_i(\tilde{x}) + f_0(\tilde{x}), \tag{1}$$

where $f_i(\tilde{x}) := \frac{1}{J}\sum_{j=1}^{J} f_i(\tilde{x}; j)$ is the loss function of regular client $i \in \mathcal{R}$ and $f_0(\tilde{x})$ is a regularization term. For participating rule of client, we consider more realistic asynchronous iterate pattern where each client participates in the interaction with the central server at certain probability γ in each iteration. Notably, solving (1) poses a significant challenge due to the presence of hidden Byzantine clients that can send arbitrary malicious messages to the central server, thereby disrupting the learning process.

Let the central server maintain a local model $x_0 \in \mathbb{R}^d$ and each regular client $i \in \mathcal{R}$ maintain a local model $x_i \in \mathbb{R}^d$. Therefore, (1) can be equivalently rewritten as follows:

$$\min_{x:=[x_i;x_0]} \sum_{i\in\mathcal{R}} f_i(x_i) + f_0(x_0) \tag{2a}$$

$$\text{s.t.} \quad x_0 = x_i, \forall i \in \mathcal{R}, \tag{2b}$$

where $x := [x_i; x_0] \in \mathbb{R}^{(|\mathcal{R}|+1)d} = \mathbb{R}^{(r+1)d}$ is a longer vector which stacks all regular clients' local models x_i and the central server's local model x_0.

Similar to [17], [19], we penalize the consensus constraints in (2) using a l_p-norm term:

$$x^* := \arg\min_{x:=[x_i;x_0]} \sum_{i\in\mathcal{R}} \Big(f_i(x_i) + \lambda \|x_i - x_0\|_p \Big) + f_0(x_0), \tag{3}$$

where λ is a positive penalty parameter and integer $p \geqslant 1$. Note that the minimization of the l_p-norm penalty term in (3) forces every model x_i to be close to the model x_0 of the central server. In absence of Byzantine clients, if regular clients participate in the update at each iteration, (3) can be solved by applying stochastic subgradient descent method [19]. In particular, at iteration k, each regular client $i \in \mathcal{R}$ updates x_i^{k+1} according to (4a) and the central server updates x_0^{k+1} according to (4b):

$$x_i^{k+1} = x_i^k - \alpha\Big(\nabla f_i(x_i^k; \xi_i^k) + \lambda \partial_{x_i} \|x_i^k - x_0^k\|_p \Big), \tag{4a}$$

$$x_0^{k+1} = x_0^k - \alpha\Big(\nabla f_0(x_0^k) + \lambda \sum_{i\in\mathcal{R}} \partial_{x_0} \|x_0^k - x_i^k\|_p \Big), \tag{4b}$$

where $\partial_{x_i} \|x_i^k - x_0^k\|_p$ is a subgradient of $\|x_i - x_0^k\|_p$ evaluated at $x_i = x_i^k$ and $\partial_{x_0} \|x_0^k - x_i^k\|_p$ is a subgradient of $\|x_0 - x_i^k\|_p$ evaluated at $x_0 = x_0^k$, while the constant $\alpha > 0$ is the learning rate, and ξ_i^k denotes the local sample selected by regular client i at iteration k.

Now, we consider a more practical setting where there exist Byzantine clients in the system with asynchronous participation of clients. We use $\mathcal{S}^k$ to denote the set of clients participating in aggregation at iteration k. For a regular client $i \in \mathcal{R}$, if it participates in aggregation at iteration k, i.e., $i \in \mathcal{R} \cap \mathcal{S}^k$, its update follows (4a). Otherwise, when $i \in \mathcal{R} \backslash \mathcal{S}^k$, its local model remains unchanged, i.e., $x_i^{k+1} = x_i^k$. If client $v \in \mathcal{B} \cap \mathcal{S}^k$ is Byzantine, it will send arbitrary malicious messages $z_v^k \in \mathbb{R}^d$ to the central server. Due to the fact that the identities of Byzantine clients are unknown to the central server, one cannot distinguish between normal message x_i^k from a regular client $i \in \mathcal{R} \cap \mathcal{S}^k$ and malicious message z_v^k from a Byzantine client $v \in \mathcal{B} \cap \mathcal{S}^k$. Thus, the update rule of the central server in (4b) becomes

$$\begin{aligned} x_0^{k+1} = & x_0^k - \alpha \Bigg(\nabla f_0(x_0^k) + \lambda \Bigg(\sum_{i\in\mathcal{R}\cap\mathcal{S}^k} \partial_{x_0} \|x_0^k - x_i^k\|_p \\ & + \sum_{v\in\mathcal{B}\cap\mathcal{S}^k} \partial_{x_0} \|x_0^k - z_v^k\|_p \Bigg) \Bigg), \end{aligned} \tag{5}$$

where $\partial_{x_0} \|x_0^k - z_v^k\|_p$ is a subgradient of $\|x_0 - z_v^k\|_p$ evaluated at $x_0 = x_0^k$.

To eliminate the effect of gradient sampling variance on Byzantine robustness, we introduce the variance reduction technique SAGA [21] to regular clients' local update. Each regular client $i \in \mathcal{R}$ keeps a stochastic gradient table for all its own local data samples. At iteration k, each regular

client $i \in \mathcal{R} \cap \mathcal{S}^k$ randomly selects a local data sample with index ξ_i^k and computes the stochastic gradient $\nabla f_i\left(x_i^k; \xi_i^k\right)$. However, it does not update x_i^{k+1} using $\nabla f_i\left(x_i^k; \xi_i^k\right)$ just like (4a). Instead, $\nabla f_i\left(x_i^k; \xi_i^k\right)$ is corrected by subtracting the previously stored stochastic gradient corresponding to the ξ_i^k-th data sample, and then adding the average of all stored stochastic gradients. With such a corrected stochastic gradient, regular client $i \in \mathcal{R} \cap \mathcal{S}^k$ updates x_i^{k+1} and replaces the stochastic gradient of the ξ_i^k-th data sample in the table with $\nabla f_i\left(x_i^k; \xi_i^k\right)$. In particular, let

$$\phi_{i,j}^{k+1} = \begin{cases} \phi_{i,j}^k, & i \in \mathcal{R} \backslash \mathcal{S}^k \text{ and } \forall j \\ x_i^k, & i \in \mathcal{R} \cap \mathcal{S}^k \text{ and } j = \xi_i^k \\ \phi_{i,j}^k, & i \in \mathcal{R} \cap \mathcal{S}^k \text{ and } j \neq \xi_i^k \end{cases}, \tag{6}$$

where $\phi_{i,j}^k$ refers to the most recent model parameter used for computing $\nabla f_i\left(\cdot; j\right)$ prior to iteration k. Therefore, $\nabla f_i\left(\phi_{i,j}^k; j\right)$ represents the previously stored stochastic gradient for the j-th data sample of regular client i prior to iteration k, and

$$g_i^k := \nabla f_i\left(x_i^k; \xi_i^k\right) - \nabla f_i\left(\phi_{i,\xi_i^k}^k; \xi_i^k\right) + \frac{1}{J}\sum_{j=1}^{J} \nabla f_i\left(\phi_{i,j}^k; j\right) \tag{7}$$

is the corrected stochastic gradient of regular client $i \in \mathcal{R} \cap \mathcal{S}^k$ at iteration k. Thus, we replace the original stochastic gradient $\nabla f_i\left(x_i^k; \xi_i^k\right)$ in (4a) by g_i^k, yielding a new update rule of x_i^{k+1} for each $i \in \mathcal{R} \cap \mathcal{S}^k$ at iteration k, i.e.,

$$x_i^{k+1} = x_i^k - \alpha\Big(g_i^k + \lambda \partial_{x_i} \left\|x_i^k - x_0^k\right\|_p\Big). \tag{8}$$

We term this new algorithm as AsynRSA-VR, and its complete Pseudocode is given in Algorithm 1.

III. Convergence Analysis

In this section, we analyze the convergence of the proposed AsynRSA-VR (c.f., Alg. 1), with detailed proofs provided in the Appendix. To this end, we first make the following common assumptions.

Assumption 1. *The local loss functions $f_i(\tilde{x})$ for regular client $i \in \mathcal{R}$ and the regularization term $f_0(\tilde{x})$ are μ-strongly convex.*

Assumption 2. *For every regular client $i \in \mathcal{R}$, any model $\tilde{x} \in \mathbb{R}^d$ and sample $\xi_i \in \{1, ..., J\}$, the local sample loss functions $f_i(\tilde{x}; \xi_i)$ and the regularization term $f_0(\tilde{x})$ have Lipschitz continuous gradients with constant L.*

Remark 1. *In Assumption 1 and 2, we assume that the regularization term has the same strong convexity and smoothness constants as the regular clients' local loss functions, just for simplicity. In fact, our conclusion (Theorem 1) can be easily extended to the case where the two constants for regularization term are different from those for regular clients' local loss functions.*

Assumption 3 (Unbiased stochastic gradient)**.** *For every regular client $i \in \mathcal{R}$ and any $\tilde{x} \in \mathbb{R}^d$, the expectation of stochastic gradient is its aggregated gradient, i.e.,*

$$\mathbb{E}\left[\nabla f_i\left(\tilde{x}; \xi_i\right)\right] = \nabla f_i\left(\tilde{x}\right). \tag{9}$$

Algorithm 1 Asynchronous Byzantine-Robust Stochastic Aggregation with Variance Reduction (**AsynRSA-VR**)

Regular Client i:

Initialization: model $x_i^0 \in \mathbb{R}^d$, penalty parameter $\lambda > 0$, learning rate $\alpha > 0$, participation probability γ.
for $j \in \{1, 2, ..., J\}$ **do**
  Initializes $\nabla f_i(\phi_{i,j}; j) = \nabla f_i(x_i^0; j)$
end for
for $k = 0, 1, 2, ...$ **do**
  if client i is activated (with probability γ) **then**
    Sends the current local model x_i^k to the server
    Receives the server's local model x_0^k
    Samples ξ_i^k from $\{1, 2, ..., J\}$ uniformly at random
    Computes the corrected gradient by: $g_i^k = \nabla f_i(x_i^k; \xi_i^k) - \nabla f_i(\phi_{i,\xi_i^k}; \xi_i^k) + \frac{1}{J}\sum_{j=1}^{J} \nabla f_i(\phi_{i,j}; j)$
    Stores gradient: $\nabla f_i(\phi_{i,\xi_i^k}; \xi_i^k) = \nabla f_i(x_i^k; \xi_i^k)$
    Updates x_i^{k+1} according to (8)
  else
    Updates x_i^{k+1} by: $x_i^{k+1} = x_i^k$
  end if
end for

Central Server:

Initialization: model $x_0^0 \in \mathbb{R}^d$, penalty parameter $\lambda > 0$, learning rate $\alpha > 0$.
for $k = 0, 1, 2, ...$ **do**
  Receives local model x_i^k from regular client $i \in \mathcal{R} \cap \mathcal{S}^k$ and malicious model z_v^k from Byzantine client $v \in \mathcal{B} \cap \mathcal{S}^k$.
  Sends its current local model x_0^k to all participating clients in the current iteration k.
  Updates x_0^{k+1} according to (5)
end for

Assumption 4 (Bounded inner variation)**.** *For every regular client $i \in \mathcal{R}$ and any $\tilde{x} \in \mathbb{R}^d$, the variation of its stochastic gradients is upper bounded by*

$$\mathbb{E}\left[\left\|\nabla f_i\left(\tilde{x}; \xi_i\right) - \nabla f_i\left(\tilde{x}\right)\right\|^2\right] \leqslant \sigma_i^2. \tag{10}$$

The following lemma guarantees that the optimal solution of l_p-norm regularized problem (3) is, indeed, the same as that of the original problem (1) as long as the penalty parameter λ is sufficiently large.

Lemma 1 (Theorem 1 in [19])**.** *Suppose that Assumptions 1 and 2 hold. If $\lambda \geqslant \max_{i \in \mathcal{R}} \left\|\nabla f_i(\tilde{x}^*)\right\|_b$ with $p \geqslant 1$ and b satisfying $\frac{1}{b} + \frac{1}{p} = 1$, then we have $x^* = [...; \tilde{x}^*; ...]$, where $\tilde{x}^*$ and x^* are the optimal solutions of* (1) *and* (3)*, respectively.*

Next, we consider the case of large λ and focus on the convergence to (3). The following theorem establishes the convergence of AsynRSA-VR.

Theorem 1. *Let the step size $\alpha \leqslant \frac{\mu}{12J(\mu+L)L}$. Suppose*

Assumptions 1, 2 and 3 hold. Then, we have

$$\mathbb{E}\left[T^k\right] \leqslant \left(1-\frac{\gamma\mu L}{\mu+L}\alpha\right)^k T^0 + \triangle_1, \tag{11}$$

where the Lyapunov function $T^k := \left\|x^k - x^*\right\|^2 + \frac{2J\mu L\alpha}{3(\mu+L)} \cdot S^k$ *with* $S^k := \sum_{i\in\mathcal{R}} \frac{1}{J} \sum_{j=1}^{J} \left\|x_i^* - \phi_{i,j}^k\right\|^2$*, while the steady-state error*

$$\begin{aligned}\triangle_1 =& \frac{\mu+L}{\gamma\mu L}\alpha\left(\gamma\cdot 16\lambda^2 rd + 16\lambda^2 r^2 d + 2\lambda^2 q^2 d\right) \\ &+ \frac{(\mu+L)^2\lambda^2 q^2 d}{\gamma\mu^2 L^2} + \frac{(1-\gamma)(\mu+L)^2\lambda^2 r^2 d}{\gamma\mu^2 L^2}.\end{aligned} \tag{12}$$

Proof. See Appendix I. □

Remark 2. *Theorem 1 shows that AsynRSA-VR can linearly converge to a neighborhood of the optimal solution of* (3) *and the size of the neighborhood* $\triangle_1$ *is determined by the penalty parameter* λ*, the number of Byzantine clients* q*, the problem dimension* d *and participating probability* γ*.*

Remark 3. *It follows from Theorem 1 that a larger* γ *will lead to faster convergence rate and smaller steady-state error, but it may lead to severe network congestion in practical scenarios, which calls for a careful choice of* γ *for a better trade-off between convergence performance and communication efficiency. When* $\gamma = 1$ *which corresponds to full participation pattern, we observe that the learning error of AsynRSA-VR is smaller than that of RSA [19] due to the elimination of gradient sampling variance* σ_i^2*.*

IV. Experiments

In this section, we validate the robustness against Byzantine attacks for our proposed AsynRSA-VR and compare it with benchmark Byzantine-robust algorithms. All algorithms are implemented using distributed communication package *torch.distributed* in PyTorch [27], where a process serves as central server or client, and we use inter-process communication to mimic the communication between central server and client. We conduct all experiments on the MNIST dataset [28] using softmax regression with an l_2-norm regularization term $f_0(\tilde{x}) = \frac{0.01}{2}\|\tilde{x}\|^2$. The MNIST dataset consists of 10 handwritten digits from 0 to 9, with 60,000 training images and 10,000 testing images. We launch several processes in a high performance computer with Intel Xeon E5-2680 v4 CPU @ 2.40GHz, in which one serves as central server and the rest serve as clients. In order to simulate the non-IID distribution of the data, training samples of each handwritten digit are allocated to regular clients in unbalanced proportions. To verify the robustness of the proposed algorithm, we consider two types of Byzantine attacks as follows:

- **Same value attack.** Each Byzantine client $v \in \mathcal{B}$ sends malicious model $c \cdot \mathbf{1}$ to central server, where $\mathbf{1}$ is an all-one vector and c is a constant set as 10000.
- **Gaussian attack.** Each Byzantine client $v \in \mathcal{B}$ sends malicious model following multi-variate Gaussian distribution $\mathcal{N}(0, 100^2)$ to central server.

We set $p = 1$ (i.e., using l_1-norm penalty) in our AsynRSA-VR for simplicity.

A. Impact of Different Penalty Parameter

Before deploying our AsynRSA-VR to actual scenarios with Byzantine attacks, we conduct some experiments on a system with only a small number of regular clients to explore the influence of the value of penalty parameter λ on the performance of AsynRSA-VR. We launch a server process and 8 client processes without Byzantine attackers, and vary the value of λ in the range $\{0.0001, 0.05, 0.5, 0.8, 1, 3\}$ for AsynRSA-VR. The learning rate is set as $\alpha = 0.001$, and the participating probability of each client in each iteration is set as $\gamma = 1$. It can be seen from the testing accuracy comparison for different λ shown in Fig. 1 that: too large or too small λ will reduce the accuracy of AsynRSA-VR, and $\lambda = 0.05$ performs best. It should be noted that this result is consistent with the fact that small value of λ yields slow information fusion over the network which in turn leads to slow convergence and large value of λ incurs large steady-state-error according to the expression of $\triangle_1$ in (12). Therefore, we select $\lambda = 0.05$ for our AsynRSA-VR in the subsequent experiments with Byzantine attacks.

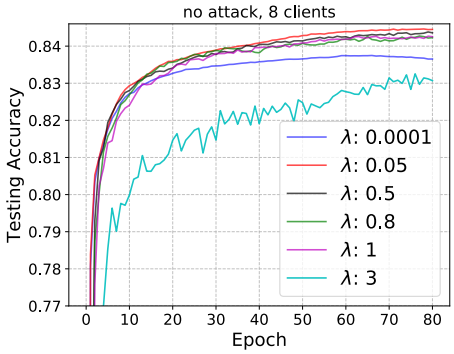


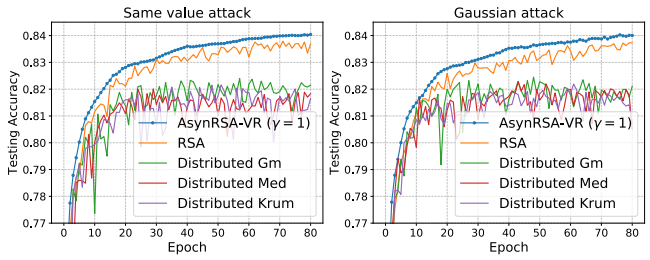
Fig. 1. Performance comparison for different λ.

B. Full Participation

Now we consider the actual scenario with Byzantine attacks, and launch 1 server process and 25 client processes in which 5 clients are Byzantine who will launch same value attack (gaussian attack). For the case where all clients participate in aggregation in each iteration (i.e., $\gamma = 1$), we compare our proposed AsynRSA-VR with several benchmarks including RSA [19], Distributed geometric median (Gm) [7], Distributed median (Med) [12] and Distributed Krum [15]. For all these 5 algorithms, we set the learning rate as $\alpha = 0.001$. It follows from the results shown in Fig. 2 that: for both two types of attacks, our AsynRSA-VR outperforms other Byzantine-robust algorithms, implying stronger robustness of AsynRSA-VR against Byzantine attacks.

C. Partial Participation

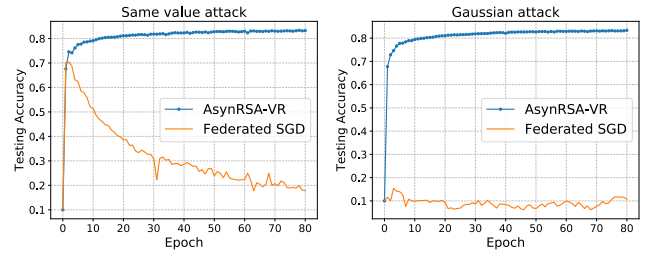
In this part, we consider a practical asynchronous participation pattern. We consider 1 central server and 25 clients, where 5 clients are Byzantine and the rest are regular, and



(a) (b)

Fig. 2. Performance comparison of AsynRSA-VR with RSA, Distributed Gm, Distributed Krum and Distributed Med under Byzantine attacks: (a) Same value attack; (b) Gaussian attack

each client participates in aggregation at a probability of $\gamma = 0.4$. We compare AsynRSA-VR with Federated SGD, which also works for asynchronous way of updating. For Federated SGD, the central server performs gradient descent based on the mean of all received messages $\{m_i^k : i \in \mathcal{S}^k\}$ (m_i^k is local stochastic gradient if i is regular and malicious message if i is Byzantine). The result shown in Fig. 3 demonstrates that Federated SGD fails under two types of Byzantine attacks while our AsynRSA-VR still achieves high testing accuracy, implying better robustness of AsynRSA-VR against Byzantine attacks.



(a) (b)

Fig. 3. Comparison of AsynRSA-VR with Federated SGD ($\gamma = 0.4$) under Byzantine attacks: (a) Same value attack; (b) Gaussian attack.

V. Conclusions

We have proposed a Byzantine-robust algorithm based on regularization techniques for distributed learning where nodes may participate for updating in an asynchronous pattern. Variance-reduced method is utilized in the proposed AsynRSA-VR algorithm to eliminate the stochastic error of gradient sampling for achieving better Byzantine robustness. Leveraging a properly designed Lyapunov function, we have showed that AsynRSA-VR converges linearly to an error ball that is independent of stochastic gradient variance. Extensive experiments have been conducted to demonstrate the effectiveness of AsynRSA-VR against Byzantine attacks. However, it is also important to extend our algorithm to more practical settings (e.g., non-convex), in the future work.

Appendix I
Proof of Theorem 1

Proof. According to (18) and (42) in [20], we have the following two useful properties with Assumption 2 and 3:

$$\mathbb{E}\left[g_i^k\right] = \nabla f_i\left(x_i^k\right), \tag{13a}$$

$$\mathbb{E}\left[\left\|g_i^k - \nabla f_i\left(x_i^k\right)\right\|^2\right] \leqslant L^2 \cdot \frac{1}{J}\sum_{j=1}^{J}\left\|x_i^k - \phi_{i,j}^k\right\|^2. \tag{13b}$$

Bounding $\mathbb{E}\left[\left\|x_i^{k+1} - x_i^*\right\|^2\right]$ for $\forall i \in \mathcal{R}$. According to the update of regular client in Algorithm 1, we have

$$\mathbb{E}\left[\left\|x_i^{k+1} - x_i^*\right\|^2\right] = (1-\gamma)\cdot\mathbb{E}\left[\left\|x_i^k - x_i^*\right\|^2\right] + \gamma\cdot\underbrace{\mathbb{E}\left[\left\|x_i^k - x_i^* - \alpha(g_i^k + \lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p)\right\|^2\right]}_{A_0}. \tag{14}$$

For A_0, we have

$$\begin{aligned}
A_0 &= \mathbb{E}\left[\left\|x_i^k - x_i^*\right\|^2\right] + \alpha^2\mathbb{E}\left[\left\|g_i^k + \lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p\right\|^2\right] \\
&\quad - 2\alpha\mathbb{E}\left[\left\langle g_i^k + \lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p, x_i^k - x_i^*\right\rangle\right] \\
&\overset{(a)}{=} \left\|x_i^k - x_i^*\right\|^2 + \underbrace{\alpha^2\mathbb{E}\left[\left\|g_i^k + \lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p\right\|^2\right]}_{A_1} \\
&\quad \underbrace{-2\alpha\left\langle\nabla f_i\left(x_i^k\right) - \nabla f_i\left(x_i^*\right), x_i^k - x_i^*\right\rangle}_{A_2} \\
&\quad -2\alpha\left\langle\lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p - \lambda\partial_{x_i}\left\|x_i^* - x_0^*\right\|_p, x_i^k - x_i^*\right\rangle,
\end{aligned} \tag{15}$$

where in (a) we have used (13a), and the optimality condition of (3) w.r.t. x_i, i.e.,

$$\nabla f_i\left(x_i^*\right) + \lambda\partial_{x_i}\left\|x_i^* - x_0^*\right\|_p = 0. \tag{16}$$

For A_1, we have

$$\begin{aligned}
A_1 &\leqslant 2\left\|\nabla f_i\left(x_i^k\right) + \lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p\right\|^2 \\
&\quad + 2\mathbb{E}\left[\left\|g_i^k - \nabla f_i\left(x_i^k\right)\right\|^2\right] \\
&\overset{(b)}{\leqslant} 4\lambda^2\left\|\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p - \partial_{x_i}\left\|x_i^* - x_0^*\right\|_p\right\|^2 \\
&\quad + 4\left\|\nabla f_i\left(x_i^k\right) - \nabla f_i\left(x_i^*\right)\right\|^2 + 2L^2\cdot\frac{1}{J}\sum_{j=1}^{J}\left\|x_i^k - \phi_{i,j}^k\right\|^2 \\
&\overset{(c)}{\leqslant} 4\left\|\nabla f_i\left(x_i^k\right) - \nabla f_i\left(x_i^*\right)\right\|^2 + 16\lambda^2 d \\
&\quad + 2L^2\cdot\frac{1}{J}\sum_{j=1}^{J}\left\|x_i^k - \phi_{i,j}^k\right\|^2,
\end{aligned} \tag{17}$$

where in (b) we plugged the optimality condition (16) and used (13b), while in (c) we used the property that the absolute value of every element of the $\partial_{x_i}\|x_i^k - x_0^k\|_p$(or $\partial_{x_i}\|x_i^* - x_0^*\|_p$) is no larger than 1.

For A_2, by Assumption1 and 2, we have

$$A_2 \geqslant \frac{\mu L}{\mu+L}\left\|x_i^k - x_i^*\right\|^2 + \frac{1}{\mu+L}\left\|\nabla f_i\left(x_i^k\right) - \nabla f_i\left(x_i^*\right)\right\|^2. \tag{18}$$

Substituting (17) and (18) into (15), and letting the step size

$$\alpha \leqslant \frac{1}{2\left(\mu+L\right)}, \tag{19}$$

(15) can be relaxed as

$$\begin{aligned} A_0 \leqslant & \left(1 - \frac{2\alpha\mu L}{\mu+L}\right)\left\|x_i^k - x_i^*\right\|^2 \\ & + \alpha^2\left(16\lambda^2 d + 2L^2 \cdot \frac{1}{J}\sum_{j=1}^{J}\left\|x_i^k - \phi_{i,j}^k\right\|^2\right) \\ & - 2\alpha\left\langle \lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p - \lambda\partial_{x_i}\left\|x_i^* - x_0^*\right\|_p, x_i^k - x_i^*\right\rangle. \end{aligned} \tag{20}$$

Substituting (20) into (14), we get

$$\begin{aligned} \mathbb{E}\left[\left\|x_i^{k+1} - x_i^*\right\|^2\right] \leqslant & \left(1 - \frac{2\gamma\alpha\mu L}{\mu+L}\right)\left\|x_i^k - x_i^*\right\|^2 \\ & + \gamma\cdot\alpha^2\left(16\lambda^2 d + 2L^2 \cdot \frac{1}{J}\sum_{j=1}^{J}\left\|x_i^k - \phi_{i,j}^k\right\|^2\right) \\ & - \gamma\cdot 2\alpha\left\langle \lambda\partial_{x_i}\left\|x_i^k - x_0^k\right\|_p - \lambda\partial_{x_i}\left\|x_i^* - x_0^*\right\|_p, x_i^k - x_i^*\right\rangle. \end{aligned} \tag{21}$$

Bounding $\mathbb{E}\left[\left\|x_0^{k+1} - x_0^*\right\|^2\right]$ for central server. According to the update of the server in (5), we have

$$\begin{aligned} \mathbb{E}\left[\left\|x_0^{k+1} - x_0^*\right\|^2\right] = & \left\|x_0^k - x_0^*\right\|^2 \\ & + \alpha^2\mathbb{E}\left[\left\|\nabla f_0(x_0^k) + \lambda\sum_{i\in\mathcal{R}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - x_i^k\right\|_p \right.\right. \\ & \left.\left. + \lambda\sum_{v\in\mathcal{B}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - z_v^k\right\|_p\right\|^2\right] \\ & \underbrace{- 2\alpha\mathbb{E}\left\langle \nabla f_0(x_0^k) + \lambda\sum_{i\in\mathcal{R}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - x_i^k\right\|_p, x_0^k - x_0^*\right\rangle}_{A_4} \\ & \underbrace{- 2\alpha\mathbb{E}\left\langle \lambda\sum_{v\in\mathcal{B}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - z_v^k\right\|_p, x_0^k - x_0^*\right\rangle}_{A_5}. \end{aligned} \tag{22}$$

Denoting the expectation term in the second term at the RHS of the above inequality by A_3, we have

$$\begin{aligned} A_3 \leqslant & 2\mathbb{E}\left[\left\|\nabla f_0(x_0^k) + \lambda\sum_{i\in\mathcal{R}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - x_i^k\right\|_p\right\|^2\right] \\ & + 2\lambda^2\mathbb{E}\left[\left\|\sum_{v\in\mathcal{B}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - z_v^k\right\|_p\right\|^2\right] \\ \overset{(d)}{\leqslant} & 4\mathbb{E}\left[\left\|\nabla f_0(x_0^k) - \nabla f_0(x_0^*)\right\|^2\right] + 2\lambda^2 q^2 d + 4\lambda^2\cdot \\ & \mathbb{E}\left[\left\|\sum_{i\in\mathcal{R}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - x_i^k\right\|_p - \sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^* - x_i^*\right\|_p\right\|^2\right] \\ \leqslant & 4\left\|\nabla f_0(x_0^k) - \nabla f_0(x_0^*)\right\|^2 + 16\lambda^2 r^2 d + 2\lambda^2 q^2 d, \end{aligned} \tag{23}$$

where in (d) we have plugged the optimality condition of (3) w.r.t. x_0, i.e., $\nabla f_0(x_0^*) + \lambda\sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^* - x_i^*\right\|_p = 0$.

For A_4, plugging the optimality condition $\nabla f_0(x_0^*) + \lambda\sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^* - x_i^*\right\|_p = 0$ again, we obtain

$$\begin{aligned} A_4 = & \mathbb{E}\left\langle \nabla f_0(x_0^k) - \nabla f_0(x_0^*), x_0^k - x_0^*\right\rangle + \mathbb{E}\left\langle x_0^k - x_0^*, \right. \\ & \left. \lambda\sum_{i\in\mathcal{R}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - x_i^k\right\|_p - \lambda\sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^* - x_i^*\right\|_p\right\rangle \\ \overset{(e)}{\geqslant} & \frac{\mu L}{\mu+L}\left\|x_0^k - x_0^*\right\|^2 + \frac{1}{\mu+L}\left\|\nabla f_0(x_0^k) - \nabla f_0(x_0^*)\right\|^2 \\ & + \left\langle \lambda\sum_{i\in\mathcal{R}}\left(\gamma\partial_{x_0}\left\|x_0^k - x_i^k\right\|_p - \partial_{x_0}\left\|x_0^* - x_i^*\right\|_p\right), x_0^k - x_0^*\right\rangle, \end{aligned} \tag{24}$$

where (e) is due to the fact that f_0 is strongly convex and has Lipschitz continuous gradients (c.f., Assumptions 1 and 2).

For A_5, using Young's inequality, we have

$$\begin{aligned} A_5 \leqslant & \alpha\beta\mathbb{E}\left[\left\|x_0^k - x_0^*\right\|^2\right] \\ & + \frac{\alpha\lambda^2}{\beta}\mathbb{E}\left[\left\|\sum_{v\in\mathcal{B}\cap\mathcal{S}^k}\partial_{x_0}\left\|x_0^k - z_v^k\right\|_p\right\|^2\right] \\ \leqslant & \alpha\beta\left\|x_0^k - x_0^*\right\|^2 + \frac{\alpha\lambda^2 q^2 d}{\beta}, \end{aligned} \tag{25}$$

where $\beta > 0$ is a tunable parameter.

Substituting (23), (24) and (25) into (22), and letting the step size α satisfy (19), we have

$$\begin{aligned} \mathbb{E}\left[\left\|x_0^{k+1} - x_0^*\right\|^2\right] \leqslant & \left(1 - \left(\frac{2\mu L}{\mu+L} - \beta\right)\alpha\right)\left\|x_0^k - x_0^*\right\|^2 \\ & + \alpha^2\left(16\lambda^2 r^2 d + 2\lambda^2 q^2 d\right) + \frac{\alpha\lambda^2 q^2 d}{\beta} - 2\alpha\cdot \\ & \left\langle \lambda\sum_{i\in\mathcal{R}}\left(\gamma\partial_{x_0}\left\|x_0^k - x_i^k\right\|_p - \partial_{x_0}\left\|x_0^* - x_i^*\right\|_p\right), x_0^k - x_0^*\right\rangle. \end{aligned} \tag{26}$$

Summing up (21) for all $i \in \mathcal{R}$ and adding (26), we have

$$\begin{aligned}
&\sum_{i\in\mathcal{R}} \mathbb{E}\left[\left\|x_i^{k+1}-x_i^*\right\|^2\right] + \mathbb{E}\left[\left\|x_0^{k+1}-x_0^*\right\|^2\right] \\
&\leqslant \left(1-\frac{2\gamma\alpha\mu L}{\mu+L}\right)\sum_{i\in\mathcal{R}}\left\|x_i^k-x_i^*\right\|^2 \\
&+ \left(1-\left(\frac{2\mu L}{\mu+L}-\beta\right)\alpha\right)\left\|x_0^k-x_0^*\right\|^2 \\
&+ 2\gamma\alpha^2L^2\sum_{i\in\mathcal{R}}\frac{1}{J}\sum_{j=1}^{J}\left\|x_i^k-\phi_{i,j}^k\right\|^2 + \frac{\alpha\lambda^2q^2d}{\beta} \\
&+ \alpha^2\left(16\lambda^2r^2d+2\lambda^2q^2d\right) + 16\alpha^2\gamma\lambda^2rd \\
&- 2\gamma\alpha\lambda\left(\sum_{i\in\mathcal{R}}\left\langle\partial_{x_i}\left\|x_i^k-x_0^k\right\|_p - \partial_{x_i}\left\|x_i^*-x_0^*\right\|_p, x_i^k-x_i^*\right\rangle\right. \\
&\left.+\left\langle\sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^k-x_i^k\right\|_p - \sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^*-x_i^*\right\|_p, x_0^k-x_0^*\right\rangle\right) \\
&+ \underbrace{(1-\gamma)\cdot 2\alpha\lambda\cdot\left\langle\sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^*-x_i^*\right\|_p, x_0^k-x_0^*\right\rangle}_{A_7}.
\end{aligned} \tag{27}$$

We denote the quantity in the bracket of the seventh term at the RHS of the above inequality by A_6, and using the convexity of $h(x) := \sum_{i\in\mathcal{R}}\|x_i - x_0\|_p$, we have that

$$A_6 = \left\langle\partial_x h\left(x^k\right) - \partial_x h\left(x^*\right), x^k - x^*\right\rangle \geqslant 0. \tag{28}$$

For A_7, using Young's inequality, we can bound it by

$$\begin{aligned}
&A_7 \leqslant \\
&\alpha\left(\nu\left\|x_0^k-x_0^*\right\|^2 + \frac{(1-\gamma)^2\lambda^2}{\nu}\left\|\sum_{i\in\mathcal{R}}\partial_{x_0}\left\|x_0^*-x_i^*\right\|_p\right\|^2\right) \\
&\leqslant \alpha\nu\left\|x_0^k-x_0^*\right\|^2 + \frac{\alpha(1-\gamma)^2\lambda^2r^2d}{\nu},
\end{aligned} \tag{29}$$

where $\nu > 0$ is a tunable parameter.

Substituting (28) and (29) into (27), we obtain

$$\begin{aligned}
&\sum_{i\in\mathcal{R}} \mathbb{E}\left[\left\|x_i^{k+1}-x_i^*\right\|^2\right] + \mathbb{E}\left[\left\|x_0^{k+1}-x_0^*\right\|^2\right] \\
&\leqslant \left(1-\frac{2\gamma\alpha\mu L}{\mu+L}\right)\sum_{i\in\mathcal{R}}\left\|x_i^k-x_i^*\right\|^2 \\
&+ \left(1-\left(\frac{2\mu L}{\mu+L}-\beta-\nu\right)\alpha\right)\left\|x_0^k-x_0^*\right\|^2 + \frac{\alpha\lambda^2q^2d}{\beta} \\
&+ \gamma\cdot 2\alpha^2L^2\cdot\sum_{i\in\mathcal{R}}\frac{1}{J}\sum_{j=1}^{J}\left\|x_i^k-\phi_{i,j}^k\right\|^2 + \frac{\alpha(1-\gamma)^2\lambda^2r^2d}{\nu} \\
&+ \alpha^2\left(\gamma\cdot 16\lambda^2rd + 16\lambda^2r^2d + 2\lambda^2q^2d\right).
\end{aligned} \tag{30}$$

Knowing that $\left\|x_i^k-\phi_{i,j}^k\right\|^2 \leqslant 2\left\|x_i^k-x_i^*\right\|^2 + 2\left\|x_i^*-\phi_{i,j}^k\right\|^2$ and letting

$$S^k := \sum_{i\in\mathcal{R}}\frac{1}{J}\sum_{j=1}^{J}\left\|x_i^*-\phi_{i,j}^k\right\|^2, \tag{31}$$

we further have

$$\begin{aligned}
&\mathbb{E}\left[\left\|x^{k+1}-x^*\right\|^2\right] \leqslant \frac{\alpha\lambda^2q^2d}{\beta} + \frac{\alpha(1-\gamma)^2\lambda^2r^2d}{\nu} \\
&+ \left(1-\frac{2\alpha\gamma\mu L}{\mu+L}+4\alpha^2\gamma L^2\right)\sum_{i\in\mathcal{R}}\left\|x_i^k-x_i^*\right\|^2 \\
&+ 4\alpha^2\gamma L^2\cdot S^k + \alpha^2\left(\gamma\cdot 16\lambda^2rd + 16\lambda^2r^2d + 2\lambda^2q^2d\right) \\
&+ \left(1-\left(\frac{2\mu L}{\mu+L}-\beta-\nu\right)\alpha\right)\left\|x_0^k-x_0^*\right\|^2.
\end{aligned} \tag{32}$$

By the definition of S^k in (31), we have

$$\begin{aligned}
\mathbb{E}\left[S^{k+1}\right] &= \sum_{i\in\mathcal{R}}\frac{1}{J}\sum_{j=1}^{J}\mathbb{E}\left[\left\|x_i^*-\phi_{i,j}^{k+1}\right\|^2\right] \\
&= \sum_{i\in\mathcal{R}}\frac{1}{J}\sum_{j=1}^{J}\left(\left(1-\frac{\gamma}{J}\right)\left\|x_i^*-\phi_{i,j}^k\right\|^2 + \frac{\gamma}{J}\left\|x_i^*-x_i^k\right\|^2\right) \\
&= \left(1-\frac{\gamma}{J}\right)S^k + \frac{\gamma}{J}\sum_{i\in\mathcal{R}}\left\|x_i^k-x_i^*\right\|^2.
\end{aligned} \tag{33}$$

By computing (32) + (33)$\times c$ (c is a positive constant to be properly determined later), and choosing the value of β and ν as $\beta = \frac{\mu L}{\mu+L}$ and $\nu = \frac{(1-\gamma)\mu L}{\mu+L}$, we get

$$\begin{aligned}
&\mathbb{E}\left[\left\|x^{k+1}-x^*\right\|^2\right] + c\cdot\mathbb{E}\left[S^{k+1}\right] \leqslant \frac{\alpha(\mu+L)\lambda^2q^2d}{\mu L} \\
&+ \left(1-\frac{2\gamma\mu L}{\mu+L}\alpha+4\alpha^2\gamma L^2+\frac{\gamma c}{J}\right)\sum_{i\in\mathcal{R}}\left\|x_i^k-x_i^*\right\|^2 + \\
&\left(1-\frac{\gamma\mu L}{\mu+L}\alpha\right)\left\|x_0^k-x_0^*\right\|^2 + \left[c\left(1-\frac{\gamma}{J}\right)+4\alpha^2\gamma L^2\right]S^k \\
&+ \alpha^2\left(\gamma\cdot 16\lambda^2rd + 16\lambda^2r^2d + 2\lambda^2q^2d\right) \\
&+ \frac{\alpha(1-\gamma)(\mu+L)\lambda^2r^2d}{\mu L}.
\end{aligned} \tag{34}$$

If the step size α is chosen such that

$$4\alpha^2\gamma L^2 + \frac{\gamma c}{J} \leqslant \frac{\gamma\mu L}{\mu+L}\alpha, \tag{35}$$

the coefficient in front of $\sum_{i\in\mathcal{R}}\left\|x_i^k-x_i^*\right\|^2$ satisfies

$$1-\frac{2\gamma\mu L}{\mu+L}\alpha+4\alpha^2\gamma L^2+\frac{\gamma c}{J} \leqslant 1-\frac{\gamma\mu L}{\mu+L}\alpha, \tag{36}$$

and the coefficient in front of S^k satisfies

$$c\left(1-\frac{\gamma}{J}\right)+4\alpha^2\gamma L^2 \leqslant \left(1-\frac{2\gamma}{J}\right)c + \frac{\gamma\mu L}{\mu+L}\alpha. \tag{37}$$

Further, if α and c are chosen such that

$$\frac{\mu L}{\mu+L}\alpha \leqslant \frac{1}{2J} \tag{38}$$

and

$$c = \frac{2J\mu L}{3(\mu+L)}\alpha \geqslant \frac{\mu L}{(\mu+L)\left(\frac{2}{J}-\frac{\mu L}{\mu+L}\alpha\right)}\alpha, \tag{39}$$

the factor in (37) further satisfies

$$c\left(1-\frac{\gamma}{J}\right)+4\alpha^2\gamma L^2 \leqslant \left(1-\frac{2\gamma}{J}\right)c+\frac{\gamma\mu L}{\mu+L}\alpha \leqslant \left(1-\frac{2\gamma}{J}\right)c+\left(\frac{2\gamma}{J}-\frac{\gamma\mu L}{\mu+L}\alpha\right)c = \left(1-\frac{\gamma\mu L}{\mu+L}\alpha\right)c. \tag{40}$$

Therefore, (34) becomes

$$\begin{aligned}&\mathbb{E}\left[\left\|x^{k+1}-x^*\right\|^2\right]+c\cdot\mathbb{E}\left[S^{k+1}\right]\\ &\leqslant\left(1-\frac{\gamma\mu L}{\mu+L}\alpha\right)\left(\left\|x^k-x^*\right\|^2+c\cdot S^k\right)\\ &+\frac{\alpha\left(\mu+L\right)\lambda^2q^2d}{\mu L}+\alpha^2\left(\gamma\cdot16\lambda^2rd+16\lambda^2r^2d+2\lambda^2q^2d\right)\\ &+\frac{\alpha\left(1-\gamma\right)\left(\mu+L\right)\lambda^2r^2d}{\mu L}.\end{aligned} \tag{41}$$

According to the definition of T^k in Theorem 1, the above inequality (41) further implies that

$$\begin{aligned}\mathbb{E}\left[T^{k+1}\right]&\leqslant\left(1-\frac{\gamma\mu L}{\mu+L}\alpha\right)T^k+\frac{\alpha\left(1-\gamma\right)\left(\mu+L\right)\lambda^2r^2d}{\mu L}\\ &+\frac{\alpha\left(\mu+L\right)\lambda^2q^2d}{\mu L}+\alpha^2\left(\gamma\cdot16\lambda^2rd+16\lambda^2r^2d+2\lambda^2q^2d\right).\end{aligned} \tag{42}$$

Using telescopic cancellation on (42), we get (11) and (12) in Theorem 1. To sum up, the step size α must satisfy (19), (35) and (38), i.e.,

$$\alpha\leqslant\min\left\{\frac{1}{2\left(\mu+L\right)},\frac{\mu+L}{2J\mu L},\frac{\mu}{12\left(\mu+L\right)L}\right\}, \tag{43}$$

which can be satisfied with

$$\alpha\leqslant\frac{\mu}{12J\left(\mu+L\right)L}. \tag{44}$$

We thus complete the proof. □

REFERENCES

[1] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

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

[3] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[4] A. Vempaty, L. Tong, and P. K. Varshney, "Distributed inference with byzantine data: State-of-the-art review on data falsification attacks," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 65–75, 2013.

[5] Y. Chen, S. Kar, and J. M. Moura, "The internet of things: Secure distributed inference," *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 64–75, 2018.

[6] Z. Yang, A. Gang, and W. U. Bajwa, "Adversary-resilient inference and machine learning: From distributed to decentralized," *stat*, vol. 1050, p. 23, 2019.

[7] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[8] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[9] A. Acharya, A. Hashemi, P. Jain, S. Sanghavi, I. S. Dhillon, and U. Topcu, "Robust training in high dimensions via block coordinate geometric median descent," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 11 145–11 168.

[10] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[11] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[12] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant sgd," *arXiv preprint arXiv:1802.10116*, vol. 21, 2018.

[13] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[14] C. Xie, O. Koyejo, and I. Gupta, "Phocas: dimensional byzantine-resilient stochastic gradient descent," *arXiv preprint arXiv:1805.09682*, 2018.

[15] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[16] Y. Allouah, S. Farhadkhani, R. Guerraoui, N. Gupta, R. Pinot, and J. Stephan, "Fixing by mixing: A recipe for optimal byzantine ml under heterogeneity," *arXiv preprint arXiv:2302.01772*, 2023.

[17] W. Ben-Ameur, P. Bianchi, and J. Jakubowicz, "Robust distributed consensus using total variation," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1550–1564, 2015.

[18] J. Peng, W. Li, and Q. Ling, "Byzantine-robust decentralized stochastic optimization over static and time-varying networks," *Signal Processing*, vol. 183, p. 108020, 2021.

[19] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1544–1551.

[20] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.

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

[22] E.-M. El-Mhamdi, R. Guerraoui, and S. Rouault, "Distributed momentum for byzantine-resilient learning," *arXiv preprint arXiv:2003.00010*, 2020.

[23] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[24] J. Peng, W. Li, and Q. Ling, "Variance reduction-boosted byzantine robustness in decentralized stochastic optimization," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4283–4287.

[25] Y.-R. Yang and W.-J. Li, "Basgd: Buffered asynchronous sgd for byzantine learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 751–11 761.

[26] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous sgd," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 495–10 503.

[27] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, no. 3, p. 67, 2017.

[28] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.