Distributed Byzantine Tolerant Stochastic Gradient Descent in the Era of Big Data

Richeng Jin
Department of ECE
North Carolina State University
rjin2@ncsu.edu

Xiaofan He
Electronic Information School
Wuhan University
xiaofanhe@whu.edu.cn

Huaiyu Dai
Department of ECE
North Carolina State University
hdai@ncsu.edu

Abstract—**The recent advances in sensor technologies and smart devices enable the collaborative collection of a sheer volume of data from multiple information sources. As a promising tool to efficiently extract useful information from such big data, machine learning has been pushed to the forefront and seen great success in a wide range of relevant areas such as computer vision, health care, and financial market analysis. To accommodate the large volume of data, there is a surge of interest in the design of distributed machine learning, among which stochastic gradient descent (SGD) is one of the mostly adopted methods. Nonetheless, distributed machine learning methods may be vulnerable to Byzantine attack, in which the adversary can deliberately share falsified information to disrupt the intended machine learning procedures. In this work, two asynchronous Byzantine tolerant SGD algorithms are proposed, in which the honest collaborative workers are assumed to store the model parameters derived from their own local data and use them as the ground truth. The proposed algorithms can deal with an arbitrary number of Byzantine attackers and are provably convergent. Simulation results based on a real-world dataset are presented to verify the theoretical results and demonstrate the effectiveness of the proposed algorithms.**

I. INTRODUCTION

With the proliferation of sensors and smart devices, the past decade has witnessed the blowout growth in the size of the daily generated data. For example, according to [1], the world produces around 2.5 quintillion bytes of data per day in 2014. Also, as predicted by Cisco, there will be around 11.6 billion mobile devices by the year 2020 and a smartphone will generate 4.4 gigabytes data per month on average [2]. Facing such a data deluge, distributed machine learning is anticipated to play an essential role because of its ability to exploit the collective computation power of the local smart/sensing devices, thereby leading to enhanced big data analytics [3]. Specifically, distributed machine learning mechanisms have several advantages over its centralized counterpart in big data related applications. Firstly, decentralization offers better scalability, and thus facilitates large-scale machine learning applications in practice. Secondly, it eliminates the burdensome process of moving the large amount of data from the distributed devices to a central unit [4] as well as the difficulty of storing the excessive amount of data in a single machine [5].

In the existing literature of distributed machine learning, the workers are usually assumed to be honest and perform perfectly well (i.e., not making any mistake in calculation and information transmission). However, in practice, some of the workers may share wrong information due to system malfunction or software bugs. Also, some of them may even be compromised by an adversary and deliberately share falsified information to mislead the other co-workers. As is shown in [6] and [7], even a single Byzantine worker can severely disrupt the convergence of distributed gradient descent algorithms. This problem becomes more critical in big data applications, since the large number of data collection devices (i.e., the workers in machine learning applications) and the sheer volume of the collected data make it extremely challenging, if not impossible, to ensure perfect trustworthiness in data sharing and processing.

There have been some recent works [6]–[12] on Byzantine tolerant distributed machine learning algorithms, and most of them focus on stochastic gradient descent (SGD), which is one of the classic and widely adopted distributed machine learning algorithm with good scalability. However, most of them only consider the synchronous setting. This may lead to a waste of computation resources, since the workers with better computation capability have to wait for the other slower workers. In addition, most existing methods can deal with only a limited number of Byzantine workers. Moreover, they all assume a parameter server to coordinate the collaboration among the workers, which may be vulnerable to the single point of failure (SPOF).

To better accommodate the need of big data analytics, two asynchronous distributed Byzantine tolerant SGD algorithms that can deal with an arbitrary number of Byzantine workers are proposed in this work. Particularly, in the proposed algorithms, the workers are allowed to maintain their own local model parameters, which eliminates the need for a shared parameter server as in the existing literature. Also, in this setting, the workers do not need to wait for the latest broadcast model parameter from the parameter server and can proactively fetch the current learning results from the other (possibly Byzantine) co-workers at any time, thereby fulfilling asyn-

chronous learning.[1] The two proposed algorithms correspond to two different scenarios, respectively. In the first scenario, it is assumed that an *upper-bound* p of the number of Byzantine workers is known. To defend the Byzantine attack in this case, each worker takes an average of the $N - p$ model parameters that are closest to its own and then performs a gradient descent update step based on this average value. In the second scenario, no prior knowledge about the number of Byzantine workers is assumed. In this case, each worker first accepts the model parameters that potentially lead to lower empirical risk based on its evaluation over the local training samples. Then, the worker takes an average over the accepted model parameters and performs a gradient descent update step accordingly. Both of the proposed algorithms are provably convergent.

The remainder of this article is organized as follows. Section II reviews preliminaries and notations used in this work. The problem is formulated and presented in Section III. The proposed algorithms are presented in Section IV. The effectiveness of the proposed algorithms is examined through simulations in Section V. Related works are discussed in Section VI. Conclusions and future works are presented in Section VII.

II. PRELIMINARIES AND NOTATIONS

In this section, we start by reviewing some important definitions. Suppose that there is a training data set $\mathcal{S} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ with n training instances randomly sampled from a sample space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is a space of feature vectors and $\mathcal{Y}$ is a label space. Let $\mathcal{W} \subseteq \mathbb{R}^d$ be a hypothesis space of the model parameter equipped with the standard inner product and 2-norm $||\cdot||$. Given a prediction model $h(w) \in \mathcal{F} : \mathcal{X} \to \mathcal{Y}$ which is parameterized by $w \in \mathcal{W}$, the goal is to learn a good model parameter w. The prediction accuracy is measured by a loss function $f : \mathcal{W} \times \mathcal{Z} \to \mathbb{R}$. Given a hypothesis $w \in \mathcal{W}$ and a training sample $(x_i, y_i) \in \mathcal{S}$, we have a loss $f(w, (x_i, y_i))$. SGD [13] is a commonly used optimization algorithm, which aims to minimize the empirical risk $F(w) = \frac{1}{n} \sum_{i=1}^{n} f(w, (x_i, y_i))$ over the training data set $\mathcal{S}$ of n samples. For simplicity, let $f_i(w) = f(w, (x_i, y_i))$ for fixed $\mathcal{S}$. In each iteration, given a training sample (x_t, y_t), SGD updates the hypothesis w_t as follows:

$$w_{t+1} = G_{f_t, \eta_t} = w_t - \eta_t \nabla f_t(w_t), \tag{1}$$

in which η_t is the learning rate and $\nabla f_t(w_t) = \nabla f(w_t, (x_t, y_t))$ is the gradient.

To facilitate later discussion on convergence, some definitions related to the loss function are presented as follows.

Definition 1. *Let $g : \mathcal{W} \to \mathbb{R}$ be a function:*

- *g is convex if for any $u, v \in \mathcal{W}$,*
  $g(u) \geq g(v) + \ <\nabla g(v), u - v>$

- *g is L-Lipschitz if for any $u, v \in \mathcal{W}$,*
  $||g(u) - g(v)|| \leq L||u - v||$
- *g is $\lambda - strongly\ convex$ if for any $u, v \in \mathcal{W}$,*
  $g(u) \geq g(v) + \ <\nabla g(v), u - v> + \frac{\lambda}{2}||u - v||^2$

III. PROBLEM FORMULATION

In this work, a network $\mathcal{N} = \{1, \cdots, N\}$ consisting of N collaborative workers, each storing a portion of a dataset, is considered. It is assumed that each worker i stores and updates its own local model parameter w_t^i which will be used for their own classification tasks. In addition, asynchronous update is assumed in this work, where each worker can start its next update step immediately once the previous step finishes. Particularly, as is frequently done in the literature (e.g., [14], [15]), the time steps are modeled as the ticking of local clocks governed by Poisson processes. It is assumed that each worker has a clock that ticks with a rate 1 Poisson process. Thus, the inter-tick times at each worker are rate 1 exponentials, independent across workers and over time. In addition, there is a master clock which ticks whenever a local processor clock ticks and the time is discretized according to the master clock ticks (since these are the only time that the local models are updated). In this sense, the master clock ticks according to a rate N Poisson process and the local clock i that causes each master clock tick is an independently and identically distributed (i.i.d.) random variable drawn from $\mathcal{N}$. At iteration step t (when there have been $t-1$ total update steps for all the workers), a worker i sends requests and fetches the local model parameters from all the other workers (i.e., $w_t^j, \forall j \in \mathcal{N}/\{i\}$) and updates its local models based on the shared model parameters and its local dataset. In this work, it is assumed that up to p workers are Byzantine which behave arbitrarily and can share any information. Furthermore, the Byzantine workers are assumed to be aware of the local model parameters from the honest workers since they can also send requests to them. Let $\mathcal{B}$ and $\mathcal{H}$ denote the sets of Byzantine workers and honest workers, respectively. After fetching, worker i will receive

$$w_t^j = \begin{cases} w_t^j, & \text{if } j \in \mathcal{H}, \\ \text{any } w \in \mathcal{W}, & \text{if } j \in \mathcal{B}. \end{cases} \tag{2}$$

The goal of this work is to design robust SGD algorithms that can tolerant any number of Byzantine attackers (i.e., p can be any integer in $[0, N-1]$).

IV. PROPOSED ALGORITHMS

In this section, the proposed algorithms are presented. In particular, depending on whether the upper bound p of the number of Byzantine workers is known or not, two scenarios are considered. It is assumed that whenever a worker responds to a request and sends its model parameter to others, the shared information will arrive on time. In this case, all the Byzantine workers will choose to share something upon requests, since they can be easily identified if the others fail to receive information from them.

---

[1]We note that the proposed algorithms indeed introduce some communication overhead. However, it can also be implemented with a parameter server which maintains and updates the local models for the workers. In this case, the communication overhead is similar to the existing methods in the literature mentioned above. Nonetheless, it may be vulnerable to SPOF.

A. Scenario 1: p is known

The main steps of the proposed algorithm are given in Algorithm 1. The main idea is that the local model parameter stored by worker i (i.e., w_t^i) can serve as the ground truth, based on which the received shared model parameters (i.e., $w_t^j, \forall j \in \mathcal{N}/\{i\}$) can be filtered. Specifically, given the upper bound p of the number of Byzantine workers, accepting the $N-p-1$ model parameters which are closest to an honest worker's own model parameter will intuitively help filter out the wrong information shared by the Byzantine workers.

Algorithm 1 Byzantine Tolerant SGD Algorithm when p is known

---

1. Initialization: total number of workers: N, number of training data samples for each node: M, upper-bound of the number of Byzantine workers: p, each honest worker $i \in \mathcal{H}$ randomly initialize its model parameter w_0^i.
2. for iteration $t = 0, 1, \cdots, T$ do
3.   if worker i causes the master clock to tick:
4.   worker i sends requests and fetches the model parameters from all the other workers, then it accepts the $N-p-1$ model parameters which are closest to its own (i.e., the $N-p-1$ w_t^j with the smallest $||w_t^i - w_t^j||$). Then worker i takes an average over the accepted model parameters, and randomly samples a mini-batch of training samples $\mathcal{S}_t^i$ from its local dataset and performs one gradient descent step as follows:

$$w_{t+\frac{1}{2}}^i = \frac{w_t^i + \sum_{j \in \mathcal{A}_t^i} w_t^j}{N-p}, \quad (3)$$

$$w_{t+1}^i = w_{t+\frac{1}{2}}^i - \eta_t \nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i), \quad (4)$$

in which $w_{t+\frac{1}{2}}^i$ is the average of its own and the accepted model parameters for worker i, $\mathcal{A}_t^i$ is the set of accepted workers, $\nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i) = \frac{1}{|\mathcal{S}_t^i|}\sum_{m \in \mathcal{S}_t^i} \nabla f_m(w_{t+\frac{1}{2}}^i)$ is the average gradient and η_t is the learning rate at time t.
5.   worker i normalizes its own model parameter, i.e.,

$$w_{t+1}^i = \frac{w_{t+1}^i}{||w_{t+1}^i||} \quad (5)$$

6. end if
7. end for

Note that since each worker i compares the received information with its own local model parameter by measuring the distance between the model parameters, naturally one possible attack against the proposed mechanism for the Byzantine workers is to add some random noise to the model parameter (i.e., w_t^i) of the worker that sends the request and then send the resulting perturbed result back to the requester. Note that in order to pass the accepting condition in Algorithm 1 (i.e., step 4), the Byzantine workers tend to modify the requester's model parameter moderately (otherwise it will be filtered out). Therefore, in the following analysis on Algorithm 1, it is assumed that for any w_t^j from worker j accepted by worker

i at time t, it satisfies $w_t^j = w_t^i + \epsilon$, with $\mathbb{E}[||\epsilon||] = 0$ and $\mathbb{E}[||\epsilon||^2] \leq \sigma^2$, in which σ is a positive real number. Nonetheless, we note that the proposed algorithm works well on other types of attacks too.

Theorem 1. *Suppose that the loss function f is λ-strongly convex. At each iteration t, assume that worker i can sample a random gradient $\nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i)$ that satisfies $\mathbb{E}[\nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i)] = \mathbb{E}[\nabla F(w_{t+\frac{1}{2}}^i)]$ and $||\nabla f_{\mathcal{S}_t^i}(w)||^2 \leq G^2$ for any w and $\mathcal{S}_t^i$. Then running Algorithm 1 with the time model as described, with a constant step size η, we have*

$$\sum_{l=1}^{N-p} \mathbb{E}[||w_{t+1}^l - w^*||^2] \leq (1 - \frac{2\eta\lambda}{N-p})^t \sum_{l=1}^{N-p} \mathbb{E}[||w_0^l - w^*||^2]$$
$$+ \frac{(1 - 2\eta\lambda)\sigma^2 + \eta^2 G^2}{2\eta\lambda}. \quad (6)$$

Proof. Without loss of generality, we assume that the first $N-p$ workers are honest while the last p workers are Byzantine. Then we have

$$\sum_{l=1}^{N-p} \mathbb{E}[||w_{t+1}^l - w^*||^2] = \frac{N-p-1}{N-p}\Big[\sum_{l=1}^{N-p} \mathbb{E}[||w_t^l - w^*||^2]\Big] +$$
$$\frac{1}{N-p} \sum_{l=1}^{N-p} \Big[\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta \nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||^2]\Big]. \quad (7)$$

For any $l \in \mathcal{H}$, we have

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta \nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||^2]$$
$$= \mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2] + \eta^2 \mathbb{E}[||\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l)||^2] \quad (8)$$
$$- 2\eta \mathbb{E}[< w_{t+\frac{1}{2}}^l - w^*, \nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) >].$$

According to the strongly convexity of the loss function,

$$\mathbb{E}[< w_{t+\frac{1}{2}}^l - w^*, \nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) >]$$
$$= \mathbb{E}[< w_{t+\frac{1}{2}}^l - w^*, \nabla F(w_{t+\frac{1}{2}}^l) >] \geq \lambda \mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2]. \quad (9)$$

As a result,

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta \nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||^2]$$
$$\leq (1 - 2\eta\lambda)\mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2] + \eta^2 \mathbb{E}[||\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l)||^2]. \quad (10)$$

Note that $w_{t+\frac{1}{2}}^l = \frac{w_t^l + \sum_{m \in \mathcal{A}_t^l} w_t^m}{N-p}$, we have

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2] \leq \frac{1}{N-p}\Big[\mathbb{E}[||w_t^l - w^*||^2]$$
$$+ \sum_{m \in \mathcal{A}_t^l} \mathbb{E}[||w_t^m - w^*||^2]\Big] \leq \mathbb{E}[||w_t^l - w^*||^2] + \frac{N-p-1}{N-p}\sigma^2. \quad (11)$$

Plugging (10) and (11) into (15), we obtain

$$\sum_{l=1}^{N-p} \mathbb{E}[||w_{t+1}^l - w^*||^2] \leq \frac{N-p-1}{N-p} \Big[\sum_{l=1}^{N-p} \mathbb{E}[||w_t^l - w^*||^2]\Big]$$
$$+ \frac{1}{N-p} \sum_{l=1}^{N-p} \big[(1-2\eta\lambda)\mathbb{E}[||w_t^l - w^*||^2]$$
$$+ \frac{(1-2\eta\lambda)(N-p-1)}{N-p}\sigma^2 + \eta^2 G^2\big]$$
$$\leq \sum_{l=1}^{N-p}(1 - \frac{2\eta\lambda}{N-p})\mathbb{E}[||w_t^l - w^*||^2] + \frac{\eta^2 G^2}{N-p}$$
$$+ \frac{(1-2\eta\lambda)}{(N-p)}\sigma^2. \tag{12}$$

Therefore,

$$\sum_{l=1}^{N-p} \mathbb{E}[||w_{t+1}^l - w^*||^2] \leq (1 - \frac{2\eta\lambda}{N-p})^t \sum_{l=1}^{N-p} \mathbb{E}[||w_0^l - w^*||^2]$$
$$+ \frac{(1-2\eta\lambda)\sigma^2 + \eta^2 G^2}{2\eta\lambda}. \tag{13}$$

□

Remark 1. *Theorem 1 indicates that the local model parameters converge to a ball around the optimal solution, whose radius is upper bounded by a variable depending on the noise added to the shared model parameters from the Byzantine workers.*

B. Scenario 2: p is unknown

Note that in practice, the upper bound of the number of Byzantine workers may not be available for the honest workers, an algorithm that does not require any prior knowledge about p is developed in this subsection. The main steps of the proposed algorithm are given in Algorithm 2. The main difference with Algorithm 1 is the conditions of accepting a shared model parameter. In this case, the filtering criteria in Algorithm 1 cannot be used since p in unknown. Therefore, (24) is proposed to prevent the workers from accepting model parameters that are too far away from their local model parameters. However, the performance induced by condition (24) depends on the threshold parameter δ. If δ is too large, condition (24) cannot filter out the Byzantine workers when the total number of iterations is limited. If δ is too small, all the legit workers may be filtered, which renders the collaboration ineffective. With such consideration, condition (25) is proposed to further improve the performance of Algorithm 2, especially for large δ. In particular, (25) indicates that if worker i performs a stochastic gradient update based on the shared model parameter w_t^j from worker j, its own local parameter model w_t^i is not in the direction of this update and therefore w_t^j is supposed to be closer to the optimal w^*. In addition, (25) is the sufficient condition for $\sum_{k\in\mathcal{S}_t^i}[f_k(w_t^i) - f_k(w_t^j)] \geq 0$ when the loss function f is convex, which essentially means that the model

parameter shared by worker j is likely to be better than the local one.

The convergence of Algorithm 2 is given as follows.

Theorem 2. *Suppose that the loss function f is λ-strongly convex with L-Lipschitz gradients. At each iteration t, assume that worker i can sample a random gradient $\nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i)$ that satisfies $\mathbb{E}[\nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i)] = \mathbb{E}[\nabla F(w_{t+\frac{1}{2}}^i)]$. Then running Algorithm 2 with the time model as described, with a constant step size $0 \leq \eta \leq \frac{2}{\lambda+L}$, we have*

$$\sum_{l=1}^{N-p} \mathbb{E}[||w_{t+1}^l - w^*||]$$
$$\leq (1 - \frac{\eta\lambda L}{(N-p)(\lambda+L)})^t \sum_{l=1}^{N-p} \mathbb{E}[||w_0^l - w^*||] \tag{14}$$
$$+ (1 - \frac{\eta\lambda L}{\lambda+L})\frac{\delta}{(N-p)}\sum_{k=0}^{t} \frac{(1 - \frac{\eta\lambda L}{(N-p)(\lambda+L)})^{(t-k)}}{k+1}.$$

Proof. Without loss of generality, we assume that the first $N-p$ workers are honest while the last p workers are Byzantine. Then, we have

$$\sum_{l=1}^{N-p} \mathbb{E}[||w_{t+1}^l - w^*||] = \frac{N-p-1}{N-p} \Big[\sum_{l=1}^{N-p} \mathbb{E}[||w_t^l - w^*||]\Big] +$$
$$\frac{1}{N-p} \sum_{l=1}^{N-p} \big[\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||]\big]. \tag{15}$$

For any $l \in \mathcal{H}$, we have

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||^2]$$
$$= \mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2] + \eta^2 \mathbb{E}[||\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l)||^2] \tag{16}$$
$$- 2\eta\mathbb{E}[< w_{t+\frac{1}{2}}^l - w^*, \nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) >].$$

According to the strongly convexity of the loss function,

$$\mathbb{E}[< w_{t+\frac{1}{2}}^l - w^*, \nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) >]$$
$$\geq \frac{\lambda L}{\lambda+L}\mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2] + \frac{1}{\lambda+L}||\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l)||^2. \tag{17}$$

As a result,

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||^2]$$
$$\leq (1 - \frac{2\eta\lambda L}{\lambda+L})\mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2] \tag{18}$$
$$+ (\eta^2 - \frac{2\eta}{\lambda+L})\mathbb{E}[||\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l)||^2].$$

Note that when $\eta \leq \frac{2}{\lambda+L}$, the second term in (18) is negative and therefore,

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||^2]$$
$$\leq (1 - \frac{2\eta\lambda L}{\lambda+L})\mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||^2]. \tag{19}$$

Since $\sqrt{1-2x} \leq 1-x$ when $1-2x > 0$, we have

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - \eta\nabla f_{\mathcal{S}_t^l}(w_{t+\frac{1}{2}}^l) - w^*||]$$
$$\leq (1 - \frac{\eta\lambda L}{\lambda + L})\mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||]. \tag{20}$$

In addition, according to (24),

$$\mathbb{E}[||w_{t+\frac{1}{2}}^l - w^*||]$$
$$= \mathbb{E}[||\frac{w_t^l + \sum_{m\in\mathcal{A}_t^l} w_t^m}{|\mathcal{A}_t^l| + 1} - w^*||]$$
$$\leq \frac{1}{|\mathcal{A}_t^l| + 1}\left[\mathbb{E}[||w_t^l - w^*||] + \sum_{m\in\mathcal{A}_t^l}\mathbb{E}[||w_t^m - w^*||]\right] \tag{21}$$
$$\leq \mathbb{E}[||w_t^l - w^*||] + \frac{\delta}{t+1}.$$

Then,

$$\sum_{l=1}^{N-p}\mathbb{E}[||w_{t+1}^l - w^*||]$$
$$\leq \sum_{l=1}^{N-p}(1 - \frac{\eta\lambda L}{(N-p)(\lambda+L)})\mathbb{E}[||w_t^l - w^*||] \tag{22}$$
$$+ (1 - \frac{\eta\lambda L}{\lambda+L})\frac{\delta}{(N-p)(t+1)}.$$

As a result,

$$\sum_{l=1}^{N-p}\mathbb{E}[||w_{t+1}^l - w^*||]$$
$$\leq \sum_{l=1}^{N-p}(1 - \frac{\eta\lambda L}{(N-p)(\lambda+L)})^t\mathbb{E}[||w_0^l - w^*||] \tag{23}$$
$$+ (1 - \frac{\eta\lambda L}{\lambda+L})\frac{\delta}{(N-p)}\sum_{k=0}^{t}\frac{(1 - \frac{\eta\lambda L}{(N-p)(\lambda+L)})^{(t-k)}}{k+1}.$$

$\square$

Remark 2. *According to (14), the convergence of the local model parameters of the honest workers is immediate. In particular, the term induced by the bound given in (24) decreases as the number of iterations increases and will finally vanish to 0. In addition, we note that although condition (24) can guarantee the convergence of Algorithm 2, an appropriate δ should be determined for good performance. However, the choice of δ may depend on not only the specific dataset, but also the number of workers, which makes it hard to find a suitable δ in practice. This problem is solved by condition (25), which preserves good performance even when we set δ arbitrarily large.*

V. SIMULATION RESULTS

In this section, we present the simulation results to demonstrate the effectiveness of the proposed algorithms. In particular, the real public dataset MNIST [16] is used. MNIST is a widely used computer vision dataset which consists of 70,000 28×28

Algorithm 2 Byzantine Tolerant SGD Algorithm when p is unknown

1. Initialization: total number of workers: N, number of training data samples for each node: M, upper bound of the number of Byzantine workers: p, each honest worker randomly initialize their model parameters w_t^i's.
2. for iteration $t = 0, 1, \cdots, T$ do
3.   if worker i causes the master clock to tick:
4.   worker i sends requests and fetches the model parameters from all the other workers, and then accepts w_t^j if it satisfies the following conditions:

$$||w_t^i - w_t^j|| \leq \frac{\delta}{t+1}, \text{and} \tag{24}$$

$$< \nabla f_{\mathcal{S}_t^i}(w_t^j), w_t^i - w_t^j > \geq 0, \tag{25}$$

in which $\nabla f_{\mathcal{S}_t^i}(w_t^j) = \frac{1}{|\mathcal{S}_t^i|}\sum_{m\in\mathcal{S}_t^i}\nabla f_m(w_t^j)$ is the average gradient corresponding to the shared model parameter w_t^j and δ is subject to design and will be discussed in Section V.
5.   Then worker i takes an average over the accepted model parameters, randomly samples a mini-batch of training samples $\mathcal{S}_t^i$ from its local dataset and performs one gradient descent step as follows:

$$w_{t+\frac{1}{2}}^i = \frac{w_t^i + \sum_{j\in\mathcal{A}_t^i} w_t^j}{|\mathcal{A}_t^i| + 1}, \tag{26}$$

$$w_{t+1}^i = w_{t+\frac{1}{2}}^i - \eta_t\nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i), \tag{27}$$

in which $\mathcal{A}_t^i$ is the set of accepted workers, $\nabla f_{\mathcal{S}_t^i}(w_{t+\frac{1}{2}}^i) = \frac{1}{|\mathcal{S}_t^i|}\sum_{m\in\mathcal{S}_t^i}\nabla f_m(w_{t+\frac{1}{2}}^i)$ is the average gradient and η_t is the learning rate at time t.
5.   worker i normalizes its own model parameter, i.e.,

$$w_{t+1}^i = \frac{w_{t+1}^i}{||w_{t+1}^i||}. \tag{28}$$

6. end if
7. end for

---

pixel images of handwritten digits from 0 to 9. The dataset is divided into a training subset of size 60,000 and a testing subset of size 10,000. It is assumed that there are 50 workers in total (i.e., N=50) and all of them randomly select M training samples from the training dataset and test their local model parameters using the testing subset after training. It is assumed that every worker builds a softmax regression model locally and runs one epoch (and therefore a larger local dataset results in more training iterations). During the training process, the local clocks of the workers are governed by the asynchronous model discussed in Section III. In addition, we assumed that the Byzantine workers train their own local model parameters independently and send requests to others but never use the information from others when they perform gradient descent steps.
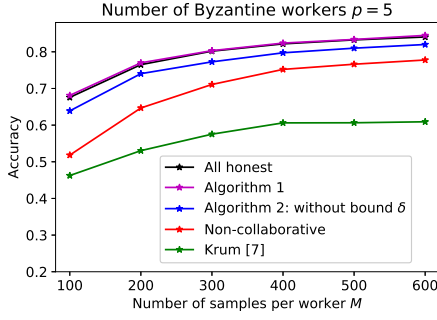
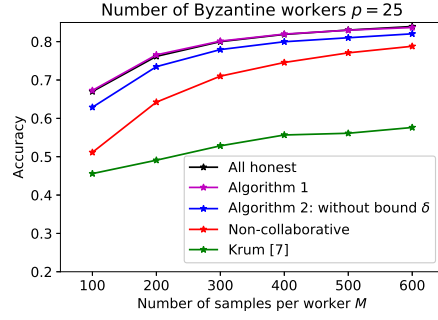Fig. 1: The performance against "Add noise" attack
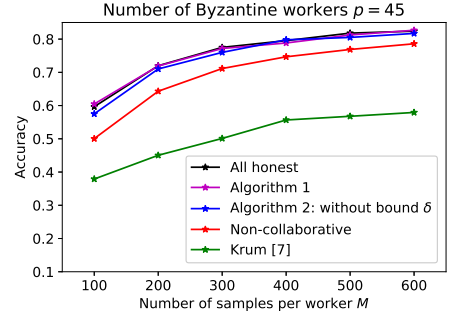

Fig. 2: The performance against "Add noise" attack

Fig. 3: The performance against "Add noise" attack
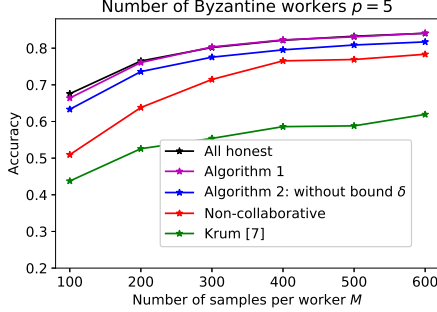

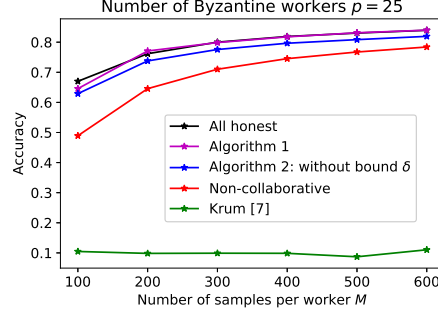Fig. 4: The performance against "Random" attack

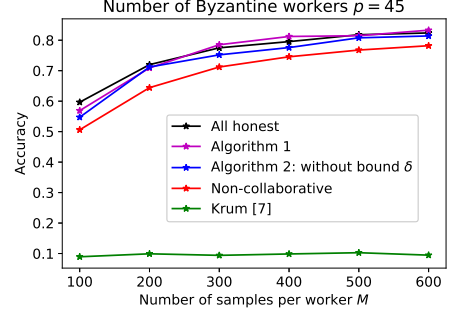Fig. 5: The performance against "Random" attack

Fig. 6: The performance against "Random" attack

To evaluate the effectiveness of the proposed algorithms, the average accuracy of the final local model parameters of all the legit workers is examined and compared with three baseline mechanisms. In the "Non-collaborative" case, the honest workers independently train their local model parameters and do not collaborate at all; in the "All honest" case, all the workers are supposed to be honest; for the baseline "Krum", we implement the algorithm proposed in [7]. In addition, three types of attacks are considered. In the "Add noise" attack, the Byzantine attackers add a random Gaussian noise with zero mean and a variance of 0.1 to the local model parameters of the workers that send requests; in the "Random" attack, the Byzantine attackers generate and share a random vector with each element drawn from a uniform distribution in [0,1]; in the "Inverse" attack, the Byzantine attackers share the opposite value of their own local model parameters.

A. The Performance of the Proposed Algorithms against Different Attacks

In this subsection, the performance of the proposed algorithms against different attacks is examined. In particular, in the "All honest" case, it is assumed that the total number of honest workers is the same as the other examined mechanisms (i.e., if $p = 5$, then there are $N - p = 45$ workers). For the implementation of Algorithm 2, we present the results ignoring (24) and the choice of δ will be discussed in Section V-C. It can be observed from Figs. 1-3 that both Algorithm 1 and Algorithm 2 perform better than the "Non-collaborative" case and the "Krum" counterpart against the "Add noise" attack. In particular, "Krum" performs even worse than the "Non-collaborative" case since it only utilizes one of the gradients shared by all the workers and therefore discards

useful information from most of the legit workers. In addition, for Algorithm 1, it is assumed that the exact number of Byzantine workers is known, and therefore it achieves almost the same performance as that in the "All honest" counterpart, which can be considered as the optimal case. When p is small, Fig. 1 and Fig. 2 show that Algorithm 2 is about 2% worse than Algorithm 1 in terms of testing accuracy. This is because the condition given in (25) may filter out some useful information from the honest workers. However, when the number of Byzantine workers is large (i.e., $p = 45$ in the simulation), the performance of Algorithm 2 is comparable to that of Algorithm 1 and the "All honest" counterpart.

Figs. 4-6 and Figs. 7-9 show the performance of the proposed algorithms against the "Random" attack and the "Inverse" attack, respectively. It can be observed the proposed algorithms outperform the "Non-collaborative" and "Krum" counterparts, which further verifies their effectiveness.

B. The Impact of Knowledge about p on Algorithm 1

Note that in the previous discussion, it is assumed that the exact number of Byzantine workers is known in the implementation of Algorithm 1. In practice, however, such an assumption is rarely valid. Fig. 10 shows the performance of Algorithm 1 under "Random" attack when the estimated number of Byzantine workers (i.e., the p used in Algorithm 1) is different from the actual one, in the case that each worker has 600 training samples. It can be seen that when the actual number of Byzantine workers exceeds the estimated one, the performance of Algorithm 1 degrades quickly. On the other hand, an accurate estimated p can lead to better performance. For example, Algorithm 1 with an estimated $p = 25$ performs better than the estimated $p = 45$ counterpart when there are
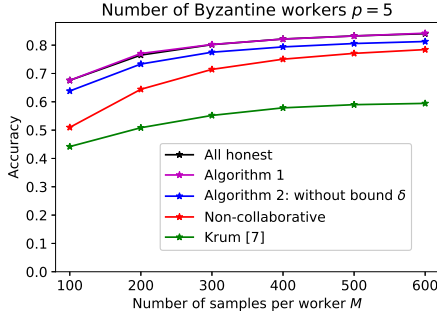
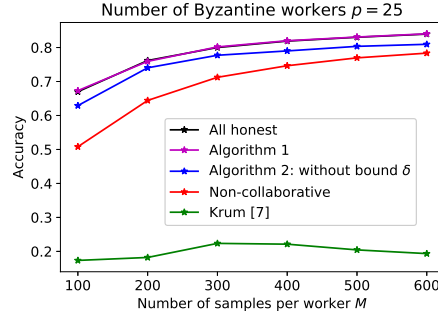**Fig. 7:** The performance against "Inverse" attack

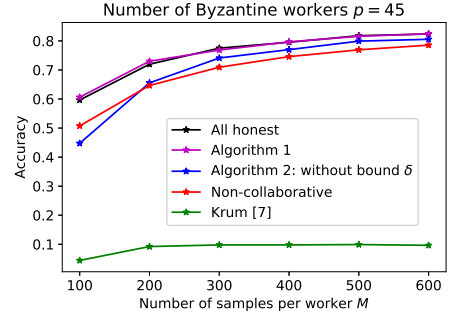**Fig. 8:** The performance against "Inverse" attack
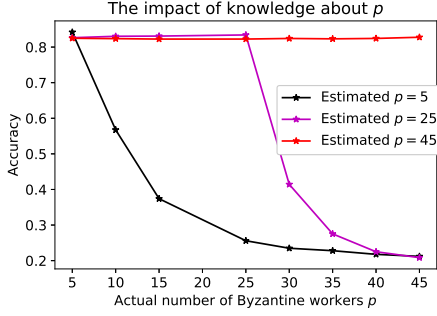

Fig. 9: The performance against "Inverse" attack

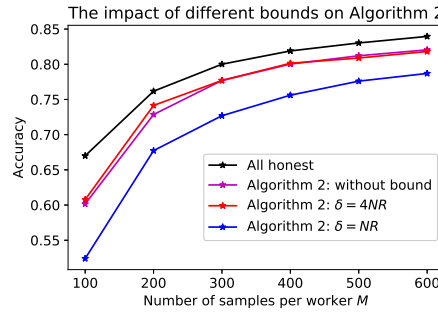**Fig. 10:** The impact of knowledge about p on Algorithm 1

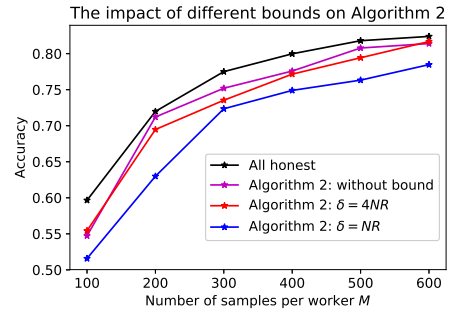**Fig. 11:** The impact of the bound δ on Algorithm 2

Fig. 12: The impact of the bound δ on Algorithm 2

less than 25 Byzantine workers. We note that when the number of local training samples is large enough, assuming a large p does not degrade the performance much since the workers can afford to discard useful information from some of the legit workers. However, an accurate estimate of p can be essential when the workers have only a limited number of training samples.

C. The Impact of Different Bound δ on Algorithm 2

Fig. 11 and Fig. 12 show the performance of Algorithm 2 under "Random" attack with different bound δ, in the case that there are 25 and 45 Byzantine attackers respectively. In particular, N is the total number of workers and R satisfies $||w|| \leq R, \forall w \in \mathcal{W}$. It can be seen that a smaller bound may lead to worse performance since more useful information may be filtered out. In fact, the optimal choice of δ may depend on the specific datasets, attacks and the number of Byzantine workers and it can be computationally expensive to obtain. However, Fig. 11 and Fig. 12 show that if we remove the bound δ (or equivalently set δ to arbitrarily large), the performance of Algorithm 2 is only around 2% worse than the "All honest" case.[2]

VI. RELATED WORKS

There have been many prior works on Byzantine tolerant SGD algorithms. In particular, [6] proposes a geometric median based aggregation rule to calculate the gradient used for parameter update, given all the gradients received from the workers. In [7], given the total number of workers N and the number of Byzantine workers p, for each worker

---

[2]Similar results can be observed for other scenarios and are omitted in the interest of space.

i and its gradient ∇f^i, the parameter server first selects a set V_i that contains the $N - p - 2$ closest gradients to ∇f^i. Then a score s_i is computed for each worker i, which measures how close its gradient is to the gradients in V_i (i.e., $s_i = \sum_{j \in V_i} ||\nabla f^i - \nabla f^j||^2$). Finally, the worker with the minimum score is selected and its gradient is used for parameter update. [8] considers generalized Byzantine attackers which attack certain elements of the gradient vectors instead of the whole gradient vectors and proposes modified median based aggregation rules. [9] proposes coordinate-wise median and coordinate-wise trimmed mean based aggregation rules for gradient selection. [10] tries to identify the good workers by comparing their shared gradients with the medians and use the gradient information from the good workers for parameter update. However, the algorithms proposed in [6]–[10] become incompetent when more than half of the workers are Byzantine. In addition, synchronous settings are assumed (i.e., the workers with better computation capability have to wait for the other slower workers) in these works, which leads to waste of computation resources. [11] proposes an asynchronous Byzantine tolerant SGD algorithm. Particularly, it consists of a Byzantine-resilient filter and a frequency filter to determine whether a (possibly outdated) gradient should be accepted or not. However, it can only deal with up to $\frac{1}{3}$ Byzantine workers. In [12], it is assumed that the parameter server has a small portion of dataset locally, which is used to compute a noisy version of the true gradient. After receiving the gradients from the workers, the parameter server compares them with the local noisy gradients and decides to accept them if the difference is within a threshold. In this sense, the algorithm proposed in [12] can deal with an arbitrary number of Byzantine workers and therefore is the most relevant one

to this work. However, it still requires a parameter server to collect the gradients and therefore may be vulnerable to the single point of failure. In addition, it requires to manually set the threshold, which depends on specific datasets. Finally, only synchronous scenarios are considered in [12].

VII. Conclusions and Future Works

Considering that most of the Byzantine tolerant SGD algorithms in the literature are either synchronous or robust against a limited number of Byzantine workers, two asynchronous distributed Byzantine tolerant SGD algorithms that can deal with an arbitrary number of Byzantine workers are proposed in this work. The convergence analysis for both algorithms is provided and the simulation results show that the proposed algorithms work well against all types of the examined Byzantine attacks. Since the proposed algorithms only consider the current shared information to decide whether to accept them or not, considering the usage of past information for performance improvement remains our future work.

References

[1] C. Dobre and F. Xhafa, "Intelligent services for big data science," *Future Generation Computer Systems*, vol. 37, pp. 267–281, 2014.

[2] C. V. N. Index, "Cisco visual networking index: Global mobile data traffic forecast update, 2015-2020 white paper," *Accessed date*, 2016.

[3] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, p. 67, 2016.

[4] A. Jochems, T. M. Deist, J. Van Soest, M. Eble, P. Bulens, P. Coucke, W. Dries, P. Lambin, and A. Dekker, "Distributed learning: developing a predictive model based on data from multiple hospitals without data leaving the hospital–a real life proof of concept," *Radiotherapy and Oncology*, vol. 121, no. 3, pp. 459–467, 2016.

[5] P. Richtárik and M. Takáč, "Distributed coordinate descent method for learning with big data," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2657–2681, 2016.

[6] 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, p. 44, 2017.

[7] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 119–129.

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

[9] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *arXiv preprint arXiv:1803.01498*, 2018.

[10] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," *arXiv preprint arXiv:1803.08917*, 2018.

[11] G. Damaskinos, E. M. E. Mhamdi, R. Guerraoui, R. Patra, and M. Taziki, "Asynchronous byzantine machine learning," *arXiv preprint arXiv:1802.07928*, 2018.

[12] X. Cao and L. Lai, "Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers," 2018.

[13] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[15] P. H. Jin, Q. Yuan, F. Iandola, and K. Keutzer, "How to scale distributed deep learning?" *arXiv preprint arXiv:1611.04581*, 2016.

[16] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.