STOCHASTIC ADMM FOR BYZANTINE-ROBUST DISTRIBUTED LEARNING

Feng Lin[*] *Qing Ling*[*] *Weiyu Li*[†] *Zhiwei Xiong*[‡]

[*]School of Data and Computer Science and Guangdong Province Key Laboratory of Computational Science, Sun Yat-Sen University
[†]School of Gifted Young, University of Science and Technology of China
[‡]School of Information Science and Technology, University of Science and Technology of China

ABSTRACT

In this paper, we aim at solving a distributed machine learning problem under Byzantine attacks. In the distributed system, a number of workers (termed as Byzantine workers) could send arbitrary messages to the master and bias the learning process, due to data corruptions, computation errors or malicious attacks. Prior work has considered a total variation (TV) norm-penalized approximation formulation to handle Byzantine attacks, where the TV norm penalty forces the regular workers' local variables to be close, and meanwhile, tolerates the outliers sent by the Byzantine workers. The stochastic subgradient method, which does not consider the problem structure, is shown to be able to solve the TV norm-penalized approximation formulation. In this paper, we propose a stochastic alternating direction method of multipliers (ADMM) that utilizes the special structure of the TV norm penalty. The stochastic ADMM iterates are further simplified, such that the iteration-wise communication and computation costs are the same as those of the stochastic subgradient method. Numerical experiments on the COVERTYPE and MNIST dataset demonstrate the resilience of the proposed stochastic ADMM to various Byzantine attacks.

***Index Terms*—** Distributed stochastic learning, alternating direction method of multipliers (ADMM), Byzantine attacks

1. INTRODUCTION

Most of the traditional machine learning algorithms need to collect training data from their owners to a single computer or data center, which is not only inefficient but also faces serious privacy risks [1]. With the explosive growth of the big data, federated learning has been proposed as a novel privacy-preserving distributed machine learning scheme, and received extensive research interest in recent years [2, 3]. In federated learning, the training data are stored at distributed workers, and the workers compute their local variables using local training data, under the coordination of a master. This scheme effectively reduces the risk of data leakage and protects privacy.

However, federated learning still faces significant security challenges. Some of the distributed workers could be unreliable and send wrong or even malicious messages to the master, due to data corruptions, computation errors or malicious attacks. To characterize the worse-case scenario, we adopt the Byzantine failure model, in which the Byzantine workers are aware of all information of the other workers, and able to send arbitrary messages to the master [4, 5]. In this paper, we aim at solving a distributed machine learning problem under Byzantine attacks.

Related Works. With the rapid popularization of federated learning, Byzantine-resilient distributed learning has become an attractive research topic in recent years. Most of the existing algorithms modify distributed stochastic gradient descent (SGD) to Byzantine-resilient versions. In the standard distributed SGD, at every iteration, all the workers send their local stochastic gradients, while the master averages all the received stochastic gradients and updates the optimization variable. When Byzantine workers are present, they can send faulty values other than true stochastic gradients to the master so as to bias the learning process. It is shown that the standard distributed SGD is vulnerable to Byzantine attacks [6].

When the training data are independently and identically distributed (i.i.d.) at the workers, the stochastic gradients of the regular workers are i.i.d. too. This fact motivates the introduction of robust aggregation rules to distributed SGD. The master can use geometric median, instead of mean, to aggregate the received messages [6, 7]. When the number of Byzantine workers is less than the number of regular worker, geometric median provides a reliable approximation to the average of regular worker's stochastic gradients. Other robust aggregation rules include marginal trimmed mean and dimensional median [8–10]. More sophisticated algorithms to achieve robust aggregation are Krum and h-Krum [11]. Krum selects the message with the smallest squared distance to a fixed number of nearest messages as the estimate to the average of regular worker's stochastic gradients, while h-Krum selects h messages using Krum and uses the average of these messages as the estimate.

When the training data are non-i.i.d. at the workers, robust aggregation of stochastic gradients is nontrivial. To address this issue, [12] investigates robust aggregation of optimization variables. To be specific, it considers a total variation (TV) norm-penalized approximation formulation to handle Byzantine attacks, where the TV norm penalty forces the regular workers' local variables to be close, and meanwhile, tolerates the outliers sent by the Byzantine workers. The stochastic subgradient method, which does not consider the problem structure, is shown to be able to solve the TV norm-penalized approximation formulation.

Our Contributions. This paper investigates the TV norm-penalized approximation formulation proposed in [12], and proposes a stochastic alternating direction method of multipliers (ADMM) that utilizes the special structure of the TV norm penalty. The stochastic ADMM iterates are further simplified, such that the iteration-wise communication and computation costs are the same as those of the stochastic subgradient method. Numerical experiments demonstrate the resilience of the proposed stochastic ADMM to Byzantine attacks.

2. PROBLEM FORMULATION

Let us consider a distributed network with a master and m workers, among which q workers are Byzantine. We are interested in solving the stochastic optimization problem in the form of

$$\min_{\tilde{x}} \sum_{i=1}^{m} \mathbb{E}[F(\tilde{x}, \xi_i)] + f_0(\tilde{x}), \tag{1}$$

where $\tilde{x} \in \mathbb{R}^d$ is the optimization variable, $f_0(\tilde{x})$ is the regularization term, and $F(\tilde{x}, \xi_i)$ is the loss function of worker i with respect to a random variable $\xi_i \sim \mathcal{D}_i$. Here we assume that the data distributions on the workers could be different.

Define $\mathcal{R}$ and $\mathcal{B}$ as the sets of regular workers and Byzantine workers, respectively, with $|\mathcal{B}| = q$ and $|\mathcal{R}| = m - q$. Because of the existence of Byzantine workers, directly solving (1) without distinguishing between regular and Byzantine workers is meaningless. A less ambitious alternative is to minimize the summation of the regular workers' local expected cost functions plus the regularization term, in the form of

$$\min_{\tilde{x}} \sum_{i\in\mathcal{R}} \mathbb{E}[F(\tilde{x}, \xi_i)] + f_0(\tilde{x}). \tag{2}$$

To enable aggregation of optimization variables instead of stochastic gradients, denote x_i as the local copy of $\tilde{x}$ at a regular worker $i \in \mathcal{R}$, and x_0 as the local copy at the master. Collecting the local copies in a vector $x = [x_0, \cdots, x_i, \cdots] \in \mathbb{R}^{(m-q+1)d}$, we know that (2) is equivalent to

$$\begin{aligned} &\min_{x} \sum_{i\in\mathcal{R}} \mathbb{E}[F(x_i, \xi_i)] + f_0(x_0), \\ &s.t.\ x_i - x_0 = 0,\ \forall i \in \mathcal{R}, \end{aligned} \tag{3}$$

where $x_i - x_0 = 0, \forall i \in \mathcal{R}$ are the consensus constraints to force the local copies to be the same.

The work of [12] considers a TV norm-penalized approximation formulation of (3), in the form of

$$x^* := \arg\min_{x} \sum_{i\in\mathcal{R}} (\mathbb{E}[F(x_i, \xi_i)] + \lambda\|x_i - x_0\|_1) + f_0(x_0), \tag{4}$$

where λ is a positive constant and $\sum_{i\in\mathcal{R}} \|x_i - x_0\|_1$ is the TV norm penalty for the constraints in (3). The TV norm penalty forces the regular workers' local optimization variables to be close to the master's, and meanwhile, tolerates the outliers when the Byzantine attackers are present [12].

3. ALGORITHM DEVELOPMENT

The challenge of solving (4) is that the Byzantine workers can send faulty messages during the optimization process. At this stage, we simply ignore the existence of Byzantine workers and develop an algorithm to solve (4). Then, we will consider the influence of Byzantine workers on the algorithm. Note that (4) can be solved by the stochastic subgradient method that, however, ignores the special structure of the TV norm penalty. In this section, we utilize the problem structure and propose a stochastic ADMM to solve (4).

Stochastic ADMM. To apply the stochastic ADMM, introduce variables $z(0, i) \in \mathbb{R}^d$ as the local copies on the directed edges $(0, i)$, as well as variables $z(i, 0) \in \mathbb{R}^d$ as the local copies on the directed edges $(i, 0)$. For the ease of presentation, we place these two sets of variables in a new variable z. Therefore, (4) is equivalent to

$$\begin{aligned} &\min_{x,z} \sum_{i\in\mathcal{R}} (\mathbb{E}[F(x_i, \xi_i)] + \lambda\|z(0, i) - z(i, 0)\|_1) + f_0(x_0), \\ &s.t.\ z(i, 0) - x_0 = 0,\ z(0, i) - x_i = 0,\ \forall i \in \mathcal{R}. \end{aligned} \tag{5}$$

As we shall see below, the introduction of z is to split the expectation minimization term $\sum_{i\in\mathcal{R}} \mathbb{E}[F(x_i, \xi_i)]$ and the TV norm penalty $\sum_{i\in\mathcal{R}} \|x_i - x_0\|_1$ so as to utilize the problem structure.

The Lagrangian function of (5) is

$$\begin{aligned} \mathcal{L}_\beta(x, z, \eta) = &\sum_{i\in\mathcal{R}} (\mathbb{E}[F(x_i, \xi_i)] + \lambda\|z(0, i) - z(i, 0)\|_1) + f_0(x_0) \\ &+ \sum_{i\in\mathcal{R}} \left(\langle \eta(i, 0), z(i, 0) - x_0 \rangle + \frac{\beta}{2}\|z(i, 0) - x_0\|^2 \right) \\ &+ \sum_{i\in\mathcal{R}} \left(\langle \eta(0, i), z(0, i) - x_i \rangle + \frac{\beta}{2}\|z(0, i) - x_i\|^2 \right), \end{aligned} \tag{6}$$

where β is a positive constant, while $\eta(i, 0) \in \mathbb{R}^d$ and $\eta(0, i) \in \mathbb{R}^d$ are the Lagrange multipliers attached to the consensus constraints $z(i, 0) - x_0$ and $z(0, i) - x_i$, respectively. For convenience, we also collect all the Lagrange multipliers in a new variable η.

With the Lagrangian function in (6), we can apply ADMM to update the variables x, z and η. However, the update of x contains the summation of expectations $\sum_{i\in\mathcal{R}} \mathbb{E}[F(x_i, \xi_i)]$, and is hence not applicable in the stochastic optimization setting. To address this issue, [13] proposes to consider the stochastic Lagrangian function. Suppose that at time k, the random variables at worker i are ξ_i^k and the optimization variables are x_i^k. Then, the stochastic Lagrangian function at time k is defined as

$$\begin{aligned} \mathcal{L}_\beta^k(x, z, \eta) = &\sum_{i\in\mathcal{R}} \lambda\|z(0, i) - z(i, 0)\|_1 \\ &+ f_0(x_0^k) + \langle f_0'(x_0^k), x_0 \rangle + \frac{\sigma^k\|x_0 - x_0^k\|^2}{2} \\ &+ \sum_{i\in\mathcal{R}} \left(F(x_i^k, \xi_i^k) + \langle F'(x_i^k, \xi_i^k), x_i \rangle + \frac{\sigma^k\|x_i - x_i^k\|^2}{2} \right) \\ &+ \sum_{i\in\mathcal{R}} \left(\langle \eta(i, 0), z(i, 0) - x_0 \rangle + \frac{\beta}{2}\|z(i, 0) - x_0\|^2 \right) \\ &+ \sum_{i\in\mathcal{R}} \left(\langle \eta(0, i), z(0, i) - x_i \rangle + \frac{\beta}{2}\|z(0, i) - x_i\|^2 \right), \end{aligned} \tag{7}$$

where σ^k is the positive stepsize. Observe that (7) is a stochastic first-order approximation to (6), in the sense that

$$\begin{aligned} \mathbb{E}[F(x_i, \xi_i)] &\sim F(x_i^k, \xi_i^k) + \langle F'(x_i^k, \xi_i^k), x_i \rangle + \frac{\sigma^k\|x_i - x_i^k\|^2}{2}, \\ f_0(x) &\sim f_0(x_0^k) + \langle f_0'(x_0^k), x_0 \rangle + \frac{\sigma^k\|x_0 - x_0^k\|^2}{2}, \end{aligned}$$

at the point $x_i = x_i^k$ and $x_0 = x_0^k$.

Similar to ADMM that operates on the Lagrangian function (6), the stochastic ADMM operates on the stochastic Lagrangian function and updates the variables in an alternating direction manner. The updates are as

$$x^{k+1} = \arg\min_{x} \mathcal{L}_\beta^k(x, z^k, \eta^k), \tag{8a}$$

$$z^{k+1} = \arg\min_{z} \mathcal{L}_\beta^k(x^{k+1}, z, \eta^k), \tag{8b}$$

$$\eta^{k+1}(i, 0) = \eta^k(i, 0) + \beta(z^{k+1}(i, 0) - x_0^{k+1}), \tag{8c}$$

$$\eta^{k+1}(0, i) = \eta^k(0, i) + \beta(z^{k+1}(0, i) - x_i^{k+1}). \tag{8d}$$

Simplification. Next, we will simplify the updates in (8) to obtain a compact algorithm. We will observe that in the resulting stochastic ADMM, the iteration-wise communication and computation costs are the same as those of the stochastic subgradient method. Note that [14] introduces the simplification of decentralized deterministic

ADMM. Our approach differs from that in [14] as we are considering the distributed stochastic ADMM.

First, observe that the updates of x_i^{k+1} and x_0^{k+1} are

$$\begin{aligned} x_i^{k+1} =& \arg\min_{x_i} \langle F'(x_i^k, \xi_i^k), x_i\rangle + \frac{\sigma^k\|x_i - x_i^k\|^2}{2} \\ &+ \frac{\beta}{2}\|z^k(0,i) - x_i\|^2 - \langle \eta^k(0,i), x_i\rangle, \\ x_0^{k+1} =& \arg\min_{x_0} \langle f_0'(x_0^k), x_0\rangle + \frac{\sigma^k\|x_0 - x_0^k\|^2}{2} \\ &+ \sum_{i\in\mathcal{R}}\left(\frac{\beta}{2}\|z^k(i,0) - x_0\|^2 - \langle \eta^k(i,0), x_0\rangle\right). \end{aligned}$$

Thus, their explicit solutions are

$$\begin{aligned} x_i^{k+1} =& \frac{1}{\sigma^k+\beta}\left(\sigma^k x_i^k + \beta z^k(0,i) + \eta^k(0,i) - F'(x_i^k,\xi_i^k)\right), \\ x_0^{k+1} =& \frac{1}{\sigma^k + \sum_{i\in\mathcal{R}}\beta}\Big(\sigma^k x_0^k \\ &+ \sum_{i\in\mathcal{R}}(\beta z^k(i,0) + \eta^k(i,0)) - f_0'(x_0^k)\Big). \end{aligned} \tag{9}$$

Second, to update z^{k+1}, we need to solve another optimization problem in the form of

$$\begin{aligned} z^{k+1} =& \arg\min_z \sum_{i\in\mathcal{R}} \lambda\|z(0,i) - z(i,0)\|_1 \\ &+ \sum_{i\in\mathcal{R}}\left(\langle \eta(i,0), z(i,0)\rangle + \frac{\beta}{2}\|z(i,0) - x_0^{k+1}\|^2\right) \\ &+ \sum_{i\in\mathcal{R}}\left(\langle \eta(0,i), z(0,i)\rangle + \frac{\beta}{2}\|z(0,i) - x_i^{k+1}\|^2\right), \end{aligned} \tag{10}$$

According to (10), we can obtain:

$$\frac{z^{k+1}(0,i) + z^{k+1}(i,0)}{2} = \frac{x_i^{k+1} + x_0^{k+1}}{2} - \frac{\eta^k(0,i) + \eta^k(i,0)}{2\beta}, \tag{11a}$$

$$\begin{aligned} &\frac{z^{k+1}(0,i) - z^{k+1}(i,0)}{2} \\ &= \mathrm{soft}_{\lambda/\beta}\left(\frac{x_i^{k+1} - x_0^{k+1}}{2} + \frac{\eta^k(i,0) - \eta^k(0,i)}{2\beta}\right), \end{aligned} \tag{11b}$$

where $\mathrm{soft}_{\lambda/\beta}(a) = \mathrm{sign}(a)\max(|a| - \lambda/\beta, 0)$ is the element-wise soft-thresholding function.

Last, we update η with (8c) and (8d).

With these preparations, we are ready to simplify the stochastic ADMM to eliminate z. According to (8c) and (8d), it holds

$$\begin{aligned} &\eta^{k+1}(0,i) + \eta^{k+1}(i,0) \\ =&\eta^k(0,i) + \eta^k(i,0) + \beta(z^{k+1}(0,i) + z^{k+1}(i,0) - x_i^{k+1} - x_0^{k+1}), \end{aligned}$$

which by (11a) leads to $\eta^{k+1}(i,0) + \eta^{k+1}(0,i) = 0$. Substituting this equality to (11a) implies that $z^{k+1}(i,0) + z^{k+1}(0,i) = x_0^{k+1} + x_i^{k+1}$. Assume that the algorithm is initialized such that $\eta^0(i,0) + \eta^0(0,i) = 0$ and $z^0(i,0) + z^0(0,i) = x_0^0 + x_i^0$. For any k we have

$$\eta^k(i,0) + \eta^k(0,i) = 0, \tag{12a}$$

$$z^k(i,0) + z^k(0,i) = x_0^k + x_i^k. \tag{12b}$$

From (12a), (11b) can be rewritten to

$$\begin{aligned} &\frac{z^{k+1}(0,i) - z^{k+1}(i,0)}{2} \\ =&\mathrm{soft}_{\lambda/\beta}\left(\frac{x_i^{k+1} - x_0^{k+1}}{2} - \frac{\eta^k(0,i)}{\beta}\right). \end{aligned} \tag{13}$$

From (12b), (8d) can be rewritten to

$$\begin{aligned} &\eta^{k+1}(0,i) \\ =&\eta^k(0,i) + \beta\left(\frac{z^{k+1}(0,i) - z^{k+1}(i,0)}{2} - \frac{x_i^{k+1} - x_0^{k+1}}{2}\right). \end{aligned} \tag{14}$$

Thus, we have

$$\eta^{k+1}(0,i) = \mathrm{proj}_{[-\lambda,\lambda]}\left(\eta^k(0,i) + \frac{\beta}{2}(x_0^{k+1} - x_i^{k+1})\right), \tag{15}$$

$$\eta^{k+1}(i,0) = \mathrm{proj}_{[-\lambda,\lambda]}\left(\eta^k(i,0) + \frac{\beta}{2}(x_i^{k+1} - x_0^{k+1})\right). \tag{16}$$

where $\mathrm{proj}_{[-\lambda,\lambda]}(a)$ is the element-wise projection function that projects each entry of a into the range of $[-\lambda, \lambda]$.

Now we simplify the update of x_i^{k+1} and x_0^{k+1} in (9). Because of (12b), we have

$$z^k(0,i) = \frac{x_i^k + x_0^k}{2} + \frac{z^k(0,i) - z^k(i,0)}{2}.$$

Substituting (12b) to the update of $\eta^k(0,i)$ in (8c) implies

$$\eta^k(0,i) = \eta^{k-1}(0,i) + \beta\left(\frac{z^k(0,i) - z^k(i,0)}{2} - \frac{x_i^k - x_0^k}{2}\right),$$

which also holds for $k = 0$ if $\eta^0(0,i)$ and $\eta^{-1}(0,i)$ are properly initialized. Combining these two equalities to cancel $\frac{1}{2}(z^{k+1}(0,i) - z^{k+1}(i,0))$, we have

$$z^k(0,i) = x_i^k + \frac{1}{\beta}(\eta^k(0,i) - \eta^{k-1}(0,i)). \tag{17}$$

Further combining (12a) and (12b) with (17), we have

$$z^k(i,0) = x_0^k + \frac{1}{\beta}(\eta^k(i,0) - \eta^{k-1}(i,0)). \tag{18}$$

Note that here we need initialization $\eta^{-1}(i,0) + \eta^{-1}(0,i) = 0$.

Substituting (17) into the update of x_i^{k+1} in (9) yields

$$x_i^{k+1} = x_i^k + \frac{1}{\sigma^k+\beta}\left(2\eta^k(0,i) - \eta^{k-1}(0,i) - F'(x_i^k,\xi_i^k)\right). \tag{19}$$

Substituting (18) into the update of x_0^{k+1} in (9) yields

$$\begin{aligned} &x_0^{k+1} \\ =&x_0^k + \frac{1}{\sigma^k + m\beta}\left(\sum_{i\in\mathcal{R}}(2\eta^k(i,0) - \eta^{k-1}(i,0)) - f_0'(x_0^k)\right). \end{aligned} \tag{20}$$

Note that here we still assume the Byzantine workers are absent, such that $\sum_{i\in\mathcal{R}}\beta = m\beta$, where m is the number of all workers.

To summarize, in the Byzantine-free case, the stochastic ADMM works as follows. Every worker i maintains x_i and $\eta(0,i)$, while the master maintains x_0 and $\eta(i,0)$. At time k, every worker i and

the master update x_i^{k+1} and x_0^{k+1} according to (19) and (20), respectively. Then, the master broadcasts x_0^{k+1} to all the workers, and every worker i uploads x_i^{k+1} to the master. Finally, every worker i and the master update $\eta^{k+1}(0,i)$ and $\eta^{k+1}(i,0)$ according to (15) and (16), respectively.

Presence of Byzantine Workers. Now we start to consider how the stochastic ADMM behaves when the Byzantine workers are present. At time k, every regular worker $i \in \mathcal{R}$ updates x_i^{k+1} with (19) and $\eta^{k+1}(0,i)$ with (15), as well as sends x_i^{k+1} to the master. However, every Byzantine worker $i \in \mathcal{B}$ can send an arbitrary $u_i^{k+1} \in \mathbb{R}^d$ to the master. At the master's side, the update of $\eta^{k+1}(i,0)$ for all $i \in \mathcal{R}$ is also the same as (16). But for all $i \in \mathcal{B}$, the update of $\eta^{k+1}(i,0)$ in (16) is biased and becomes

$$\eta^{k+1}(i,0) = \mathrm{proj}_{[-\lambda,\lambda]}\left(\eta^k(i,0) + \frac{\beta}{2}(u_i^{k+1} - x_0^{k+1})\right). \quad (21)$$

Then the update of x_0^{k+1} can be rewritten to

$$x_0^{k+1} = x_0^k + \frac{1}{\sigma^k + m\beta}\Big(\sum_{i\in\mathcal{R}}(2\eta^k(i,0) - \eta^{k-1}(i,0)) + \sum_{i\in\mathcal{B}}(2\eta^k(i,0) - \eta^{k-1}(i,0)) - f_0'(x_0^k)\Big). \quad (22)$$

The distributed stochastic ADMM for robust optimization under Byzantine attacks is outlined in Algorithm 1. The communication and computation costs are the same as those in the stochastic subgradient method in [12]. The only extra cost is that the master and every worker i must store the dual variables $\eta^k(i,0)$ and $\eta^k(0,i)$, respectively.

Algorithm 1 Distributed Stochastic ADMM

Master initializes x_0^0, $\eta^{-1}(i,0)$ and $\eta^0(i,0)$, for all $i = 1, \cdots, m$.

1: **while** not convergent **do**
2: Update x_0^{k+1} via (22);
3: Broadcast x_0^{k+1} to all workers;
4: Receive x_i^{k+1} or u_i^{k+1} from regular or Byzantine workers;
5: Update $\eta^{k+1}(i,0)$ via (16) or (21).

Regular Worker i initializes x_i^0, $\eta^{-1}(0,i)$ and $\eta^0(0,i)$.

1: **while** not convergent **do**
2: Update x_i^{k+1} via (19);
3: Send x_i^{k+1} to the master;
4: Receive x_0^{k+1} from master;
5: Update $\eta^{k+1}(0,i)$ via (15).

4. NUMERICAL EXPERIMENTS

In this section, we evaluate the robustness of the proposed distributed stochastic ADMM algorithm to Byzantine attacks. We compare it with several benchmark algorithms: (i) **Ideal SGD** without Byzantine attacks; (ii) **SGD** subject to Byzantine attacks; (iii) **Geometric Median** stochastic gradient aggregation [6, 7]; (iv) **Median** stochastic gradient aggregation [6, 7]; (v) **RSA** [12]. All the parameters in the benchmark algorithms are hand-tuned to the best. In the numerical experiments, we use two real world dataset: the MNIST dataset and COVERTYPE dataset for experiment. The statistics of these datasets are shown in Table 1. We launch one master and 20 workers. we use COVERTYPE in the first two numerical experiments, and use MNIST in the third experiment. The loss functions

Name	Training Samples	Testing Samples	Attributes
COVERTYPE	465264	115748	54
MNIST	60000	10000	784

Table 1. Specifications of the datasets.

of workers are softmax regressions, and the regularization term is $f_0(\tilde{x}) = \frac{0.01}{2}\|\tilde{x}\|^2$. Performance is evaluated by the top-1 accuracy.

Gaussian attacks. Under Gaussian attacks, every Byzantine worker sends a random vector following Gaussian distribution to the master. Here we set the standard variation as 100. In stochastic ADMM, we set λ = 0.5, β = 0.1 and $\sigma^k = 50\sqrt{k}$. As shown in Fig. 1, SGD performed worst in bost $q = 4$ and $q = 8$. When $q = 4$, Stochastic ADMM achieve the best accuracy than all the other Byzantine-robust methods. When $q = 8$, all the methods are close to Ideal SGD, and stochastic ADMM converges fastest.

Sign-flipping attacks. Under sign-flipping attacks, every Byzantine worker calculates its local variable, but flips the sign by multiplying every element with a constant $\varepsilon < 0$, and sends to the master. Here we set $\varepsilon = -3$. In stochastic ADMM, we set $\lambda = 0.5$, $\beta = 0.3$ and $\sigma^k = 100\sqrt{k}$. Fig. 2 shows that SGD also fails in both situation. When $q = 4$, stochastic ADMM are close to Ideal SGD, and RSA performe better than geometric median and median. When $q = 8$, stochastic ADMM, RSA and geometric median are close to ideal SGD, and achieve better accuracy than median.

Non-i.i.d. Data. To demonstrate the robustness of the proposed algorithm against Byzantine attacks on non-i.i.d. data, we redistribute the MNIST dataset by letting every two workers share one digit. All Byzantine workers i choose one regular worker denoted as r, and send $u_i^{k+1} = x_r^{k+1}$ to the master at every iteration k. In stochastic ADMM, we set $\lambda = 0.5$, $\beta = 5$ and $\sigma^k = 500000\sqrt{k}$. Since $q = 8$, four digits' data are not available and the best accuracy is around 0.6. As depicted in Fig. 3, geometric median and median fail, because stochastic gradients of regular worker r dominate, such that only one digit can be recognized. Stochastic ADMM and RSA both work well, and stochastic ADMM reaches better accuracy.

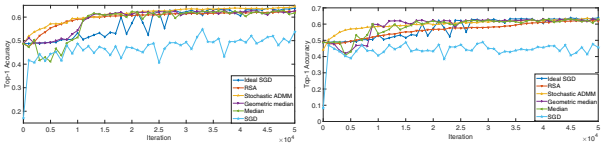

Fig. 1. Gaussian attacks in COVERTYE for $q = 4$ and $q = 8$.

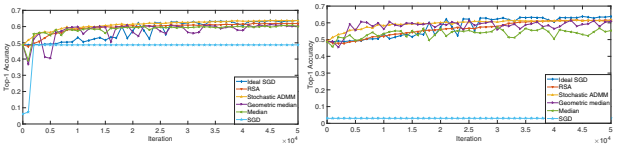

Fig. 2. Sign-flipping attacks in COVERTYPE for $q = 4$ and $q = 8$.

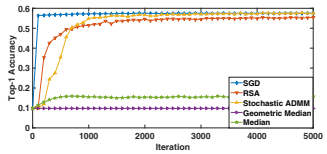

Fig. 3. Non-i.i.d. data.

Acknowledgement. Qing Ling is supported in part by NSF China Grants 61573331 and 61973324, and Fundamental Research Funds for the Central Universities.

5. REFERENCES

[1] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," Computer Networks, vol. 76, pp. 146–164, 2015

[2] J. Konecny, H. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," arXiv: 1511.03575, 2015

[3] J. Konecny, H. McMahan, F. Yu, P. Richtarik, A. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv: 1610.05492, 2016

[4] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," ACM Transactions on Programming Languages and Systems, vol. 4, no. 3, pp. 382–401, 1982

[5] N. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, San Francisco, USA, 1996

[6] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," arXiv: 1705.05491, 2017

[7] C. Xie, O. Koyejo, and I. Gupta, "Generalized Byzantine tolerant SGD," arXiv: 1802.10116, 2018

[8] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," In: Proceedings of ICML, 2018

[9] C. Xie, O. Koyejo, and I. Gupta, "Zeno: Byzantine-suspicious stochastic gradient descent," arXiv: 1805.10032, 2018

[10] C. Xie, O. Koyejo, and I. Gupta, "Phocas: Dimensional Byzantine-resilient stochastic gradient descent, arXiv: 1805.09682, 2018

[11] P. Blanchard, E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," In: Proceedings of NIPS, 2017

[12] L. Li, W. Xu, T. Chen, G. Giannakis, and Q. Ling. "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," In Proceedings of AAAI, 2019

[13] H. Ouyang, N. He, and A. Gray. "Stochastic ADMM for nonsmooth optimization," arXiv: 1211.0632, 2012

[14] 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, 2016