RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets

Liping Li,[1] Wei Xu,[1] Tianyi Chen,[2] Georgios B. Giannakis,[2] Qing Ling[3]
[1]Department of Automation, University of Science and Technology of China, Hefei, Anhui, China
[2]Digital Technology Center, University of Minnesota, Twin Cities, Minneapolis, Minnesota, USA
[3]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, Guangdong, China

Abstract

In this paper, we propose a class of robust stochastic subgradient methods for distributed learning from heterogeneous datasets at presence of an unknown number of Byzantine workers. The Byzantine workers, during the learning process, may send arbitrary incorrect messages to the master due to data corruptions, communication failures or malicious attacks, and consequently bias the learned model. The key to the proposed methods is a regularization term incorporated with the objective function so as to robustify the learning task and mitigate the negative effects of Byzantine attacks. The resultant subgradient-based algorithms are termed *Byzantine-Robust Stochastic Aggregation* methods, justifying our acronym RSA used henceforth. In contrast to most of the existing algorithms, RSA does not rely on the assumption that the data are independent and identically distributed (i.i.d.) on the workers, and hence fits for a wider class of applications. Theoretically, we show that: i) RSA converges to a near-optimal solution with the learning error dependent on the number of Byzantine workers; ii) the convergence rate of RSA under Byzantine attacks is the same as that of the stochastic gradient descent method, which is free of Byzantine attacks. Numerically, experiments on real dataset corroborate the competitive performance of RSA and a complexity reduction compared to the state-of-the-art alternatives.

Introduction

The past decade has witnessed the proliferation of smart phones and Internet-of-Things (IoT) devices. They generate a huge amount of data every day, from which one can learn models of cyber-physical systems and make decisions to improve the welfare of human being. Nevertheless, standard machine learning approaches that require centralizing the training data on one machine or in a datacenter may not be suitable for such applications, as data collected from distributed devices and stored at clouds lead to significant privacy risks (Sicari et al. 2015). To alleviate user privacy concerns, a new distributed machine learning framework called *federated learning* has been proposed by Google and become popular recently (McMahan and Ramage 2017; Smith et al. 2017). Federated learning allows the training data to be kept locally on the owners' devices. Data samples and computation tasks are distributed across multiple workers such as Internet-of-Things (IoT) devices in a smart home, which are programmed to collaboratively learn a model. Parallel implementations of popular machine learning algorithms, such as stochastic gradient descent (SGD), are applied to learning from the distributed data (Bottou 2010).

However, federated learning still faces two significant challenges: high communication overhead and serious security risk. While several recent approaches have been developed to tackle the communication bottleneck of distributed learning (Li et al. 2014; Liu et al. 2017; Smith et al. 2017; Chen et al. 2018b), the security issue has not been adequately addressed. In federated learning applications, a number of devices may be highly unreliable or even easily compromised by hackers. We call these devices as Byzantine workers. In this scenario, the learner lacks secure training ability, which makes it vulnerable to failures, not mentioning adversarial attacks (Lynch 1996). For example, SGD, the workhorse of large-scale machine learning, is vulnerable to even one Byzantine worker (Chen, Su, and Xu 2017).

In this context, the present paper studies distributed machine learning under a general Byzantine failure model, where the Byzantine workers can arbitrarily modify the messages transmitted from themselves to the master. With such a model, it simply does not have any constraints on the communication failures or attacks. We aim to develop efficient distributed machine learning methods tailored for this setting with provable performance guarantee.

Related work

Byzantine-robust distributed learning has received increasing attention in recent years. Most of the existing algorithms extend SGD to incorporate the Byzantine-robust setting and assume that the data are independent and identically distributed (i.i.d.) on the workers. Under this assumption, stochastic gradients computed by regular workers are presumably distributed around the true gradient, while those sent from the Byzantine workers to the master could be arbitrary. Thus, the master is able to apply robust estimation techniques to aggregate the stochastic gradients. Typical gradient aggregation rules include geometric median (Chen, Su, and Xu 2017), marginal trimmed mean (Yin et al. 2018a; Xie, Koyejo, and Gupta 2018b), dimensional median (Xie, Koyejo, and Gupta 2018a; Alistarh, Allen-Zhu, and Li

2018), etc. A more sophisticated algorithm termed as Krum selects a gradient which has minimal summation of Euclidean distances from a given number of nearest gradients (Blanchard et al. 2017). Targeting high-dimensional learning, an iterative filtering algorithm is developed in (Su and Xu 2018), which achieves the optimal error rate in the high-dimensional regime. The main disadvantage of these existing algorithms comes from the i.i.d. assumption, which is arguably not the case in federated learning over heterogeneous computing units. Actually, generalizing these algorithms to the non-i.i.d. setting is not straightforward. In addition, some of these algorithms rely on sophisticated gradient selection subroutines, such as those in Krum and geometric median, which incur high computational complexity.

Other related work in this context includes (Yin et al. 2018b) that targets escaping saddle points of nonconvex optimization problems under Byzantine attacks, and (Chen et al. 2018a) that leverages a gradient-coding based algorithm for robust learning. However, the approach in (Chen et al. 2018a) needs to relocate the data points, which is not easy to implement in the federated learning paradigm. Leveraging additional data, (Xie, Koyejo, and Gupta 2018c) studies the trustworthy score-based schemes that guarantee efficient learning even when there is only one non-Byzantine worker, but additional data may not always be available in practice. Our algorithms are also related to robust decentralized optimization studied in, e.g., (Ben-Ameur, Bianchi, and Jakubowicz 2016; Xu, Li, and Ling 2018), which consider optimizing a static or dynamic cost function over a decentralized network with unreliable nodes. In contrast, the focus of this work is Byzantine-robust stochastic optimization.

Our contributions

The contributions of this paper are summarized as follows.

c1) We develop a class of robust stochastic methods abbreviated as RSA for distributed learning over heterogeneous datasets and under Byzantine attacks. RSA has several variants, each tailored for an ℓ_p-norm regularized robustifying objective function.

c2) Performance is rigorously established for the resultant RSA approaches, in terms of the convergence rate as well as the error caused by the Byzantine attacks.

c3) Extensive numerical tests using the MNIST dataset are conducted to corroborate the effectiveness of RSA in term of both classification accuracy under Byzantine attacks and runtime.

Distributed SGD

We consider a general distributed system, consisting of a master and m workers, among which q workers are Byzantine (behaving arbitrarily). The goal is to find the optimizer of the following problem:

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

Here $\tilde{x} \in \mathbb{R}^d$ is the optimization variable, $f_0(\tilde{x})$ is a regularization term, and $F(\tilde{x},\xi_i)$ is the loss function of worker i with respect to a random variable ξ_i. Unlike the previous work which assumes the distributed data across the workers are i.i.d., we consider a more practical situation: $\xi_i \sim \mathcal{D}_i$, where $\mathcal{D}_i$ is the data distribution on worker i and could be different to the distributions on other workers.

Algorithm 1 Distributed SGD

Master:

1: Input: $\tilde{x}^0$, α^k. At time $k+1$:
2: Broadcast its current iterate $\tilde{x}^k$ to all workers;
3: Receive all gradients $\nabla F(\tilde{x}^k,\xi_i^k)$ sent by workers;
4: Update the iterate via (2).

Worker i:

1: At time $k+1$:
2: Receive the master's current iterate $\tilde{x}^k$;
3: Compute a local stochastic gradient $\nabla F(\tilde{x}^k,\xi_i^k)$;
4: Send the local stochastic gradient to the master.

In the master-worker architecture, at time $k+1$ of the distributed SGD algorithm, every worker i receives the current model $\tilde{x}^k$ from the master, samples a data point from the distribution $\mathcal{D}_i$ with respect to a random variable ξ_i^k, and computes the gradient of the local empirical loss $\nabla F(\tilde{x}^k,\xi_i^k)$. Note that this sampling process can be easily generalized to the mini-batch setting, in which every worker samples multiple i.i.d. data points and computes the averaged gradient of the local empirical losses. The master collects and aggregates the gradients sent by the workers, and updates the model. Its update at time $k+1$ is:

$$\tilde{x}^{k+1} = \tilde{x}^k - \alpha^{k+1}\left(\nabla f_0(\tilde{x}^k) + \sum_{i=1}^{m} \nabla F(\tilde{x}^k,\xi_i^k)\right) \tag{2}$$

where α^{k+1} is a diminishing learning rate at time $k+1$. The distributed SGD is outlined in Algorithm 1.

SGD is vulnerable to Byzantine attacks. While SGD has well-documented performance in conventional large-scale machine learning settings, its performance will significantly degrade at the presence of Byzantine workers (Chen, Su, and Xu 2017). Suppose that some of the workers are Byzantine, they can report arbitrary messages or strategically send well-designed messages according to the information sent by other workers so as to bias the learning process. Specifically, if worker m is Byzantine, at time $k+1$, it can choose one of two following attacks:

a1) sending $\nabla F(\tilde{x}^k,\xi_m^k) = \infty$;

a2) sending $\nabla F(\tilde{x}^k,\xi_m^k) = -\sum_{i=1}^{m-1} \nabla F(\tilde{x}^k,\xi_i^k)$.

In any case, the aggregated gradient $\sum_{i=1}^{m} \nabla F(\tilde{x}^k,\xi_i^k)$ used in the SGD update (2) will be either infinite or null, and thus the learned model $\tilde{x}^k$ will either not converge or converge to an incorrect value. The operation of SGD under Byzantine attacks is illustrated in Figure 1.

Instead of using the simple averaging in (2), robust gradient aggregation rules have been incorporated with SGD in (Blanchard et al. 2017; Chen, Su, and Xu 2017; Xie, Koyejo, and Gupta 2018a; Yin et al. 2018b; 2018a; Xie, Koyejo, and Gupta 2018c). However, in the federated learning setting, these aggregation rules become less effective due to the difficulty of distinguishing the statistical heterogeneity from the

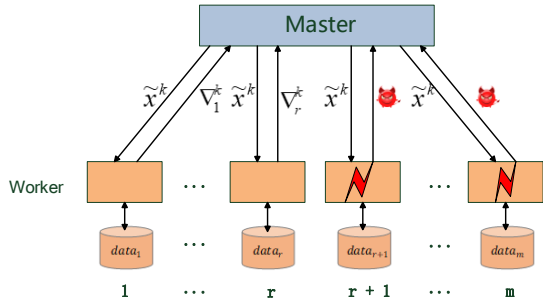


Figure 1: The operation of SGD. There are m workers, r being regular and the rest of $q = m - r$ being Byzantine. The master sends the current iterate to the workers, and the regular workers send back the stochastic gradients. The red devil marks denote the wrong messages that the Byzantine workers send to the master.

Byzantine attacks. In what follows, we develop a counterpart of SGD to address the issue of robust learning from distributed heterogeneous data.

RSA for Robust Distributed Learning

Observe that due to the presence of the Byzantine workers, it is meaningless to solve (1), which minimizes the summation of all workers' local expected losses without distinguishing regular and Byzantine workers, because the Byzantine workers can always prevent the learner from accessing their local data and finding the optimal solution. Instead, a less ambitious goal is to find a solution that minimizes the summation of the regular workers' local expected cost functions plus the regularization term:

$$\tilde{x}^* = \arg\min_{\tilde{x}\in\mathbb{R}^d} \sum_{i\in\mathcal{R}} \mathbb{E}[F(\tilde{x},\xi_i)] + f_0(\tilde{x}) \tag{3}$$

Here we denote $\mathcal{B}$ as the set of Byzantine workers and $\mathcal{R}$ as the set of regular workers, with $|\mathcal{B}| = q$ and $|\mathcal{R}| = m - q$. Letting each regular worker i have its local iterate x_i and the master have its local iterate x_0, we obtain an equivalent form to (3):

$$\min_{x:=[x_i;x_0]} \sum_{i\in\mathcal{R}} \mathbb{E}[F(x_i,\xi_i)] + f_0(x_0) \tag{4a}$$

$$\text{s.to} \quad x_0 = x_i,\ \forall i \in \mathcal{R} \tag{4b}$$

where $x := [x_i; x_0] \in \mathbb{R}^{(|\mathcal{R}|+1)d}$ is a vector that stacks the regular workers' local variables x_i and the master's variable x_0. The formulation (4) is aligned with the concept of consensus optimization in, e.g., (Shi et al. 2014).

ℓ_1-norm RSA

Directly solving (3) or (4) by iteratively updating $\tilde{x}$ or x is impossible since the identities of Byzantine workers are not available to the master. Therefore, we introduce an ℓ_1-norm regularized form of (4):

$$x^* := \arg\min_{x:=[x_i;x_0]} \sum_{i\in\mathcal{R}} \Big(\mathbb{E}[F(x_i,\xi_i)] + \lambda||x_i - x_0||_1\Big) + f_0(x_0) \tag{5}$$

where λ is a positive constant. The second term in the cost function (5) is the ℓ_1-norm penalty, whose minimization forces every x_i to be close to the master's variable x_0. We will show next that how this relaxed form brings the advantage of robust learning under Byzantine attacks.

In the ideal case that the identities of Byzantine workers are revealed, we can apply a stochastic subgradient method to solve (5). The optimization only involves the regular workers and the master. At time $k+1$, the updates of x_i^{k+1} at regular worker i and x_0^{k+1} at the master are given by:

$$x_i^{k+1} = x_i^k - \alpha^{k+1}\left(\nabla F(x_i^k,\xi_i^k) + \lambda \text{sign}(x_i^k - x_0^k)\right) \tag{6a}$$

$$x_0^{k+1} = x_0^k - \alpha^{k+1}\Big(\nabla f_0(x_0^k) + \lambda\Big(\sum_{i\in\mathcal{R}} \text{sign}(x_0^k - x_i^k)\Big)\Big) \tag{6b}$$

where $\text{sign}(\cdot)$ is the element-wise sign function. Given $a \in \mathbb{R}$, $\text{sign}(a)$ equals to 1 when $a > 0$, -1 when $a < 0$, and an arbitrary value within $[-1, 1]$ when $a = 0$. At time $k+1$, each worker i sends the local iterate x_i^k to the master, instead of sending its local stochastic gradient in distributed SGD. The master aggregates the models sent by the workers to update its own model x_0^{k+1}. In this sense, the updates in (6) are based on model aggregation, different to gradient aggregation in SGD.

Now let us consider how the updates in (6) behave at presence of Byzantine workers. The update of a regular worker i is the same as (6a), which is:

$$x_i^{k+1} = x_i^k - \alpha^{k+1}\left(\nabla F(x_i^k,\xi_i^k) + \lambda \text{sign}(x_i^k - x_0^k)\right). \tag{7}$$

If worker i is Byzantine, instead of sending the value x_i^k computed from (6a) to the master, it sends an arbitrary variable $z_i^k \in \mathbb{R}^d$. The master is unable to distinguish x_i^k sent by a regular worker or z_i^k sent by a Byzantine worker. Therefore, the update of the master at time $k+1$ is no longer (6b), but:

$$x_0^{k+1} = x_0^k - \alpha^{k+1}\Big(\nabla f_0(x_0^k) + \lambda\Big(\sum_{i\in\mathcal{R}} \text{sign}(x_0^k - x_i^k) + \sum_{j\in\mathcal{B}} \text{sign}(x_0^k - z_j^k)\Big)\Big). \tag{8}$$

We term this algorithm as ℓ_1-norm RSA (Byzantine-robust stochastic aggregation).

ℓ_1-norm RSA is robust to Byzantine attacks. ℓ_1-norm RSA is robust to Byzantine attacks due to the introduction of the ℓ_1-norm regularized term to (5). The regularization term allows every x_i to be different from x_0, and the bias is controlled by the parameter λ. This modification robustifies the objective function when any worker is Byzantine and behaves arbitrarily. From the algorithmic perspective, we can observe from the update (8) that the impacts of a regular worker and a Byzantine worker on x_0^{k+1} are similar, no matter the how different the values sent by them to the

master are. Therefore, only the number of Byzantine workers will influence the RSA update (8), rather than the magnitudes of malicious messages sent by the Byzantine workers. In this sense, ℓ_1-norm RSA is robust to arbitrary attacks from Byzantine workers. This is in sharp comparison with SGD, which is vulnerable to even a single Byzantine worker.

Generalization to ℓ_p-norm RSA

In addition to solving ℓ_1-norm regularized problem (5), we can also solve the following ℓ_p-norm regularized problem:

$$x^* := \arg\min_{x:=[x_i;x_0]} \sum_{i\in\mathcal{R}} \Big(\mathbb{E}[F(x_i,\xi_i)] + \lambda||x_i - x_0||_p\Big) + f_0(x_0) \tag{9}$$

where $p \geq 1$. Similar to the case of ℓ_1-regularized objective in (5), the ℓ_p norm penalty helps mitigate the negative influence of the Byzantine workers.

Akin to the ℓ_1-norm RSA, the ℓ_p-norm RSA still operates using subgradient recursions. For each regular worker i, its local update at time $k+1$ is:

$$x_i^{k+1} = x_i^k - \alpha^{k+1}\left(\nabla F(x_i^k,\xi_i^k) + \lambda \partial_{x_i}\|x_i^k - x_0^k\|_p\right) \tag{10}$$

where $\partial_{x_i}\|x_i^k - x_0^k\|_p$ is a subgradient of $\|x_i - x_0^k\|_p$ at $x_i = x_i^k$. Likewise, for the master, its update at time $k+1$ is:

$$x_0^{k+1} = x_0^k - \alpha^{k+1}\Big(\nabla f_0(x_0^k) + \lambda\Big(\sum_{i\in\mathcal{R}} \partial_{x_0}\|x_0^k - x_i^k\|_p + \sum_{j\in\mathcal{B}} \partial_{x_0}\|x_0^k - z_j^k\|_p\Big)\Big) \tag{11}$$

where $\partial_{x_0}\|x_0^k - x_i^k\|_p$ and $\partial_{x_0}\|x_0^k - z_j^k\|_p$ are subgradients of $\|x_0 - x_j^k\|_p$ and $\|x_0 - z_j^k\|_p$ at $x_0 = x_0^k$, respectively.

To compute the subgradient involved in ℓ_p-norm RSA, we will rely on the following proposition.

Proposition 1. *Let $p \geq 1$ and b satisfy $\frac{1}{b} + \frac{1}{p} = 1$. For $x \in \mathbb{R}^d$, we have the subdifferential $\partial_x\|x\|_p = \{z \in \mathbb{R}^d : \langle z, x\rangle = \|x\|_p,\ \|z\|_b \leq 1\}$.*

Here and thereafter, we slightly abuse the notation by using ∂ to denote both subgradient and subdifferential. The proof of Proposition 1 is in the supplementary document.

Together with ℓ_1-norm RSA, ℓ_p-norm RSA for robust distributed stochastic optimization under Byzantine attacks is summarized in Algorithm 2 and illustrated in Figure 2.

Remark 1 (Model vs. gradient aggregation)**.** *Most existing Byzantine-robust methods are based on gradient aggregation. Since each worker computes its gradient using the same iterate, these methods do not have the consensus issue (Blanchard et al. 2017; Chen, Su, and Xu 2017; Xie, Koyejo, and Gupta 2018a; Yin et al. 2018b; 2018a; Xie, Koyejo, and Gupta 2018c). However, to enable efficient gradient aggregation, these methods require the data stored in the workers are i.i.d., which is impractical in the federated learning setting. Under the assumption that data are i.i.d. on all the workers, the stochastic gradients computed by regular workers are also i.i.d. and their expectations are equal. Using this prior knowledge, the master is able to apply robust estimation techniques to aggregate the stochastic gradients collected from both regular and Byzantine workers. When the i.i.d. assumption does not hold, even the stochastic gradients computed by regular workers are different in expectation, such that the gradient aggregation-based methods no longer work. In comparison, the proposed RSA methods utilize model aggregation aiming at finding a consensual model, and do not rely on the i.i.d. assumption. On the other hand, the existing gradient aggregation methods generally require to design nontrivial subroutines to aggregate gradients, and hence incur relatively high complexities (Xie, Koyejo, and Gupta 2018a; Blanchard et al. 2017; Su and Xu 2018). In contrast, the proposed RSA methods enjoy much lower complexities, which are the same as that of the standard distributed SGD working in the Byzantine-free setting. We shall demonstrate the advantage of RSA in computational time in the numerical tests, in comparison with several state-of-the-art alternatives.*

Algorithm 2 RSA for Robust Distributed Learning

Master:

1: Input: x_0^0, $\lambda > 0$, α^k. At time $k+1$:
2: Broadcast its current iterates x_0^k to all workers;
3: Receive all local iterates x_i^k sent by regular workers or faulty values z_i^k sent by Byzantine workers;
4: Update the iterate via (8) or (11).

Regular Worker i:

1: Input: x_i^0, $\lambda > 0$, α^k. At time $k+1$:
2: Send the current local iterate x_i^k to the master;
3: Receive the master's local iterate x_0^k;
4: Update the local iterate via (7) or (10).

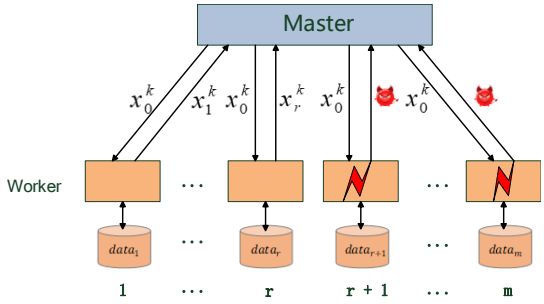


Figure 2: The operation of RSA. There are m workers, r being regular and the rest of $q = m-r$ being Byzantine. The master sends its local variable to the workers, and the regular workers send back their local variables. The red devil marks denote the wrong messages that the Byzantine workers send to the master.

Convergence Analysis

This section analyzes the performance of the proposed RSA methods, with proofs given in (Li et al. 2018). We make the following assumptions.

Assumption 1. (Strong convexity) The local cost functions $\mathbb{E}[F(\tilde{x}, \xi_i)]$ and the regularization term $f_0(\tilde{x})$ are strongly convex with constants μ_i and μ_0, respectively.

Assumption 2. (Lipschitz continuous gradients) The local cost functions $\mathbb{E}[F(\tilde{x}, \xi_i)]$ and the regularization term $f_0(\tilde{x})$ have Lipschitz continuous gradients with constants L_i and L_0, respectively.

Assumption 3. (Bounded variance) For every worker i, the data sampling is i.i.d. across time such that $\xi_i^k \sim \mathcal{D}_i$. The variance of $\nabla F(\tilde{x}, \xi_i)$ is upper bounded by δ_i^2, namely, $\mathbb{E}[||\nabla \mathbb{E}[F(\tilde{x}, \xi_i)] - \nabla F(\tilde{x}, \xi_i)||^2] \leq \delta_i^2$.

Note that Assumptions 1-3 are standard for performance analysis of stochastic gradient-based methods (Nemirovski et al. 2009), and they are satisfied in a wide range of machine learning problems such as ℓ_2-regularized least squares and logistic regression.

We start with investigating the ℓ_p-norm regularized problem (9), showing the condition under which the optimal solution of (9) is consensual and identical to that of (3).

Theorem 1. *Suppose that Assumptions 1 and 2 hold. If $\lambda \geq \lambda_0 := \max_{i \in \mathcal{R}} \|\nabla \mathbb{E}[F(\tilde{x}^*, \xi_i)]\|_b$ with $p \geq 1$ 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* (3) *and* (9)*, respectively.*

Theorem 1 asserts that if the penalty constant λ is selected to be large enough, the optimal solution of the regularized problem (9) is the same as that of (3). Next, we shall check the convergence properties of the RSA iterates with respect to the optimal solution of (9) under Byzantine attacks.

Theorem 2. *Suppose that Assumptions 1, 2 and 3 hold. Set the step size of ℓ_p-norm RSA ($p \geq 1$) as $\alpha^{k+1} = \min\{\underline{\alpha}, \frac{\overline{\alpha}}{k+1}\}$, where $\underline{\alpha}$ and $\overline{\alpha}$ depend on $\{\mu_0, \mu_i, L_0, L_i\}$. Then, for k_0 satisfying $\min\{k : \underline{\alpha} \geq \frac{\overline{\alpha}}{k+1}\}$, we have:*

$$\mathbb{E}\|x^{k+1} - x^*\|^2 \leq (1 - \eta\underline{\alpha})^k \|x^0 - x^*\|^2 + \frac{\underline{\alpha}\Delta_0 + \Delta_2}{\eta},\ k < k_0 \tag{12}$$

and

$$\mathbb{E}\|x^{k+1} - x^*\|^2 \leq \frac{\Delta_1}{k+1} + \overline{\alpha}\Delta_2,\ k \geq k_0 \tag{13}$$

where η, Δ_1 and $\Delta_2 = \mathcal{O}(\lambda^2 q^2)$ are certain positive constants.

Theorem 2 shows that the sequence of local iterates converge sublinearly to the near-optimal solution of the regularized problem (9). The asymptotic sub-optimality gap is quadratically dependent on the number of Byzantine workers q. Building upon Theorems 1 and 2, we can arrive at the following theorem.

Theorem 3. *Under the same assumptions as those in Theorem 2, if we choose $\lambda \geq \lambda_0$ according to Theorem 1, then for a sufficiently large $k \geq k_0$, we have:*

$$\mathbb{E}||x^k - [\tilde{x}^*]||^2 \leq \frac{\Delta_1}{k+1} + \overline{\alpha}\Delta_2 \tag{14}$$

If we choose $0 < \lambda < \lambda_0$, and suppose that the difference between the optimizer of (9) *and that of* (3) *is bounded by $||x^* - [\tilde{x}^*]||^2 \leq \Delta_3$, then for $k \geq k_0$ we have:*

$$\mathbb{E}||x^k - [\tilde{x}^*]||^2 \leq \frac{2\Delta_1}{k+1} + 2\overline{\alpha}\Delta_2 + 2\Delta_3 \tag{15}$$

Theorem 3 implies that the sequence of local iterates also converge sublinearly to the near-optimal solution of the original (3). Under a properly selected λ, the sub-optimality gap in the limit is proportional to the number of Byzantine workers. Note that since the $\mathcal{O}(1/k)$ step size is quite sensitive to its initial value (Nemirovski et al. 2009), we use the $\mathcal{O}(1/\sqrt{k})$ step size in our numerical tests. Its corresponding theoretical claim and convergence analysis are given in the supplementary document.

Regarding the optimal selection of the penalty constant λ and the ℓ_p norm, a remark follows next.

Remark 2 (Optimal selection of λ and p)**.** *Selecting different penalty constant λ and ℓ_p norms in RSA generally leads to distinct performance. For a fixed λ, if a norm ℓ_p with a small p is used, the dual norm ℓ_b has a large b and thus results in a small λ_0 in Theorem 1. Therefore, the local solutions are likely to be consensual. From the numerical tests, RSA with ℓ_∞ norm does not provide competitive performance, while those with ℓ_1 and ℓ_2 norms work well. On the other hand, for a fixed p, a small λ cannot guarantee consensus among local solutions, but it gives a small sub-optimality gap Δ_2. We recommend to use a λ that is relatively smaller than λ_0, slightly sacrificing consensus but reducing the sub-optimality gap.*

Numerical Tests

In this section, we evaluate the robustness of the proposed RSA methods to Byzantine attacks and compare them with several benchmark algorithms. We conduct experiments on the MNIST dataset, which has 60k training samples and 10k testing samples, and use softmax regression with an ℓ_2-norm regularization term $f_0(\tilde{x}) = \frac{0.01}{2}\|\tilde{x}\|^2$. We launch 20 worker processes and 1 master process on a computer with Intel i7-6700 CPU @ 3.40GHz. In the i.i.d. case, the training samples are randomly evenly assigned to the workers. In the heterogeneous case, every two workers evenly share the training samples of one digit. At every iteration, every regular worker estimates its local gradient on a mini-batch of 32 samples. The top-1 accuracy (evaluated with x_0 in RSA and $\tilde{x}$ in the benchmark algorithms) on the test dataset is used as the performance metric.

Benchmark algorithms

We use the SGD iteration (2) without attacks as the oracle, which is referred as **Ideal SGD**. Note that this method is not affected by q, the number of Byzantine workers. The other benchmark algorithms implement the following stochastic gradient aggregation recursion:

$$\tilde{x}^{k+1} = \tilde{x}^k - \alpha^{k+1}\tilde{\nabla}(\tilde{x}^k) \tag{16}$$

where $\tilde{\nabla}(\tilde{x}^k)$ is an algorithm-dependent aggregated stochastic gradient that approximates the gradient direction, at the point $\tilde{x}^k$ sent by the master to the workers. Let the message sent by worker i to the master be v_i^k, which is a stochastic gradient $\nabla F(\tilde{x}^k, \xi_i^k)$ if i is regular, while arbitrary if i is Byzantine. The benchmark algorithms use different rules to calculate the aggregated stochastic gradient.

GeoMed (Chen, Su, and Xu 2017). The geometric median of $\{v_i^k : i \in [m]\}$ is denoted by:

$$\text{GeoMed}(\{v_i^k\}) = \arg\min_{v \in \mathbb{R}^d} \sum_{i=1}^{m} ||v - v_i^k||_2. \quad (17)$$

We use a fast Weiszfeld's algorithm (Weiszfeld and Plastria 2009) to compute the geometric median in the experiments.

Krum (Blanchard et al. 2017). Krum calculates $\tilde{\nabla}(\tilde{x}^k)$ by:

$$\text{Krum}(\{v_i^k\}) = v_{i^*}^k,\ i^* = \arg\min_{i \in [m]} \sum_{i \to j} ||v_i^k - v_j^k||^2 \quad (18)$$

where $i \to j (i \neq j)$ selects the indexes j of the $m - q - 2$ nearest neighbors of v_i^k in $\{v_j^k : j \in [m]\}$, measured by Euclidean distances. Note that q, the number of Byzantine workers, must be known in advance in Krum.

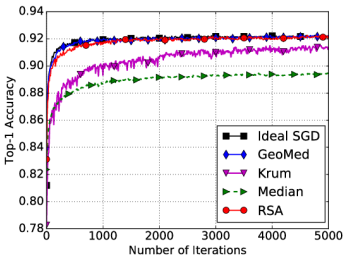


Figure 3: Top-1 accuracy without Byzantine attacks.

Median (Xie, Koyejo, and Gupta 2018a). The marginal median aggregation rule returns the element-wise median of the vectors $\{v_i^k : i \in [m]\}$.

SGD (Bottou 2010). The classical SGD aggregates $\{v_i^k : i \in [m]\}$ by returning the mean, and is hence not robust to Byzantine attacks.

In the following experiments, step sizes of the benchmark algorithms are all hand-tuned to the best.

Without Byzantine attacks

In this test, we consider learning without Byzantine workers, and show the performance of all algorithms in Figure 3. ℓ_1-norm RSA chooses the parameter $\lambda = 0.1$ and the step size $\alpha^k = 0.003/\sqrt{k}$. RSA and GeoMed are close to Ideal SGD, and significantly outperform Krum and Median. Therefore, robustifying the cost in RSA, though introduces bias, does not sacrifice performance in the regular case.

Same-value attacks

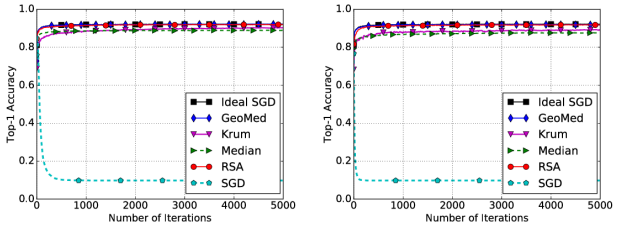
The same-value attacks set the message sent by a Byzantine worker i as $v_i^k = c\mathbf{1}$. Here $\mathbf{1} \in \mathbb{R}^d$ is an all-one vector and c is a constant, which we set as 100. We consider two different numbers of Byzantine workers, $q = 4$ and $q = 8$, and demonstrate the performance in Figure 4. ℓ_1-norm RSA chooses the regularization parameter $\lambda = 0.07$ and the step size $\alpha^k = 0.001/\sqrt{k}$. When $q = 4$, SGD fails, while RSA and GeoMed are still close to Ideal SGD and outperform Krum and Median. When q is increased to $q = 8$, Krum and Median perform worse than in $q = 4$, while RSA and GeoMed are almost the same as in $q = 4$.

Sign-flipping attacks

The sign-flipping attacks flip the signs of messages (gradients or local iterates) and enlarge the magnitudes. To be specific, a Byzantine worker i first calculates the true value $\hat{v}_i^k$, and then sends $v_i^k = \sigma \hat{v}_i^k$ to the master, where σ is a negative constant. We test $\sigma = -4$ while set $q = 4$ and $q = 8$, as shown in Figure 5. The parameters are $\lambda = 0.07$ and $\alpha = 0.001/\sqrt{k}$ for $q = 4$, while $\lambda = 0.01$ and $\alpha = 0.0003/\sqrt{k}$ for $q = 8$. Not surprisingly, SGD fails in both cases. GeoMed, Median and ℓ_1-norm RSA show similar performance, and Median is slightly worse than the other Byzantine-robust algorithms.

Runtime comparison

We show in Figure 6 the runtime of the algorithms under the same-value attacks with parameter $c = 100$ and $q = 8$ Byzantine workers. The total number of iterations for every algorithm is 5000. Though the algorithms are not implemented in a federated learning platform, the comparison clearly demonstrates the additional per-iteration computational costs incurred in handling Byzantine attacks. GeoMed has the largest per-iteration computational cost due to the difficulty of calculating the geometric median. ℓ_1-norm RSA and Median are both slightly slower than Ideal SGD, but faster than Krum. The only computational overhead of RSA than Ideal SDG lies in the computation of sign functions, which is light-weight. Therefore, RSA is advantageous in computational complexity comparing to other complicated gradient aggregation approaches.

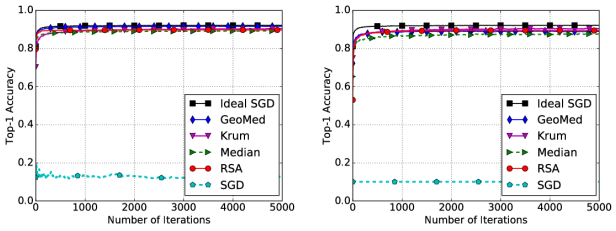


(a) (b)

Figure 4: Top-1 accuracy under same-value attacks: (a) $q = 4$ and $c = 100$; (b) $q = 8$ and $c = 100$.

Impact of hyper-parameter λ

We vary the hyper-parameter λ, and show how it affects the performance. We use the same value attacks with $c = 100$, vary λ, run RSA for 5000 iterations, and depict the final top-1 accuracy in Figure 7. The number of Byzantine workers is $q = 8$ and the step sizes are hand-tuned to the best. Observe that when λ is small, a regular worker tends to rely on its own data such that information fusion over the network is slow, which leads to slow convergence and large error. On the other hand, a large λ also incurs remarkable error, as we have investigated in the convergence analysis.



(a) (b)

Figure 5: Top-1 accuracy under sign-flipping attacks: (a) $q = 4$ and $\sigma = -4$; (b) $q = 8$ and $\sigma = -4$.

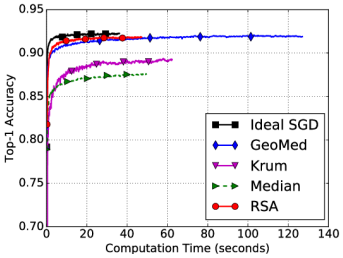


Figure 6: Runtime under same-value attacks with $q = 8$ and $c = 100$.

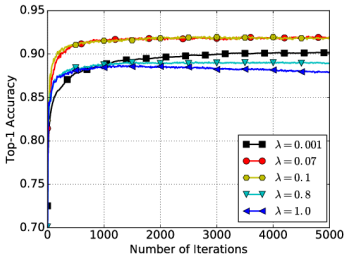
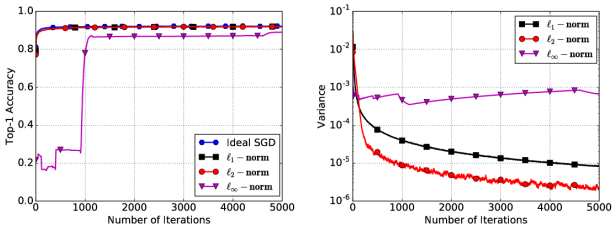


Figure 7: Top-1 accuracy with varying λ. We use same-value attacks with $q = 8$ and $c = 100$.



(a) (b)

Figure 8: RSA with different norms, without Byzantine attacks: (a) top-1 accuracy; (b) variance of regular workers' local iterates.

RSA with different norms

Now we compare RSA methods regularized with different norms. The results without Byzantine attacks and under the same-value attacks with $q = 8$ and $c = 100$ are demonstrated in Figures 8 and 9, respectively. We consider two performance metrics, top-1 accuracy and variance of the regular workers' local iterates. A small variance means that the regular workers reach similar solutions. Without the Byzantine attacks, ℓ_1 is with $\lambda = 0.1$ and $\alpha^k = 0.001/\sqrt{k}$, ℓ_2 is with $\lambda = 1.4$ and $\alpha^k = 0.001/\sqrt{k}$, while ℓ_∞ is with $\lambda = 51$ and $\alpha^k = 0.0001/\sqrt{k}$. Under the same-value attacks, ℓ_1 is with $\lambda = 0.07$ and $\alpha^k = 0.001/\sqrt{k}$, ℓ_2 is with $\lambda = 1.2$ and $\alpha^k = 0.001/\sqrt{k}$, while ℓ_∞ is with $\lambda = 20$ and $\alpha^k = 0.0001/\sqrt{k}$. In both cases, ℓ_1-norm RSA and ℓ_2-norm RSA are close in terms of top-1 accuracy, and both of them is better than ℓ_∞-norm RSA. This observation coincides with our convergence analysis, namely, ℓ_∞-norm RSA needs a large λ to ensure consensus, which in turn causes a large error. Indeed, we deliberately choose a not-too-large λ for ℓ_∞-norm RSA so as to reduce the error, but sacrificing the consensus property. Therefore, regarding the variance of the regular workers' local iterates, ℓ_∞-norm RSA is the largest, while ℓ_2-norm RSA is smaller than ℓ_1-norm RSA.

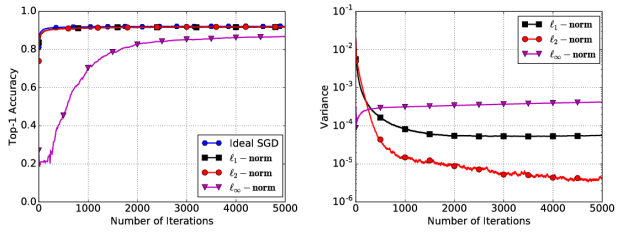
Heterogeneous Data

To show the robustness of RSA on heterogeneous dataset, we re-distribute the MNIST data in this way: each two workers associate with the data about the same handwriting digit. In experiment, each Byzantine worker i transmits $v_i^k = v_r^k$, where worker r is one of the regular workers. We set $r = 1$ in the experiment. The results are shown in Figure 10. When $q = 4$, two handwriting numbers' data are not available in the experiment, such that the best possible accuracy is around 0.8. When $q = 8$, the best possible accuracy is around 0.6. The parameters of ℓ_1-norm RSA are $\lambda = 0.5$ and $\alpha^k = 0.0005/\sqrt{k}$. Observe that when $q = 4$, Krum fails, while RSA outperforms GeoMed and Median. When q increases to 8, GeoMed, Krum and Median all fail, but RSA still performs well and reaches the near-optimal accuracy.

Conclusions

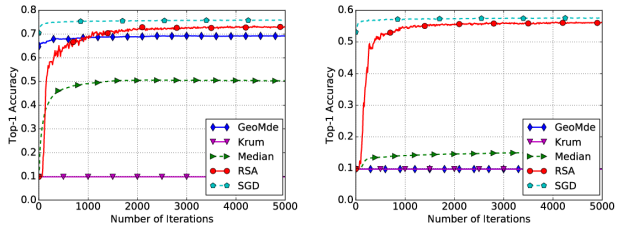
This paper dealt with distributed learning under Byzantine attacks. While the existing work mostly focuses on the case of i.i.d. data and relies on costly gradient aggregation rules, we developed an efficient variant of SGD for distributed learning from heterogeneous datasets under the Byzantine attacks. The resultant SGD-based methods that we term RSA converges to a near-optimal solution at an $O(1/k)$ convergence rate, where the optimality gap depends on the number of Byzantine workers. In the Byzantine-free settings, both SGD and RSA converge to the optimal solution at sublinear convergence rate. Numerically, experiments on real data corroborate the competitive performance of RSA compared to the state-of-the-art alternatives.

Acknowledgments. The work by T. Chen and G. Giannakis is supported in part by NSF 1500713 and 1711471. The work by Q. Ling is supported in part by NSF China 61573331 and Guangdong IIET 2017ZT07X355.



(a) (b)

Figure 9: RSA with different norms, under same-value attacks with $q = 8$ and $c = 100$: (a) top-1 accuracy; (b) variance of regular workers' local iterates.



(a) (b)

Figure 10: Top-1 accuracy under attacks on heterogeneous data: (a) $q = 4$; (b) $q = 8$.

References

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

Ben-Ameur, W.; Bianchi, P.; and Jakubowicz, J. 2016. Robust distributed consensus using total variation. *In IEEE Trans. Automat. Contr.* 61(6):1550–1564.

Blanchard, P.; Guerraoui, R.; Stainer, J.; and Mhamdi, E. M. E. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 119–129.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*. Physica-Verlag HD. 177–186.

Chen, L.; Wang, H.; Charles, Z.; and Papailiopoulos, D. 2018a. Draco: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, 902–911.

Chen, T.; Giannakis, G. B.; Sun, T.; and Yin, W. 2018b. LAG: Lazily aggregated gradient for communication-efficient distributed learning. *arXiv preprint arXiv:1805.09965*.

Chen, Y.; Su, L.; and Xu, J. 2017. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *In ACM Conference on Measurement and Analysis of Computing Systems* 1(2):44.

Li, M.; Andersen, D. G.; Smola, A. J.; and Yu, K. 2014. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, 19–27.

Li, L.; Xu, W.; Chen, T.; Giannakis, G. B.; and Ling, Q. 2018. *RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets*. http://home.ustc.edu.cn/~qingling/papers/C_AAAI2019_RSA.pdf.

Liu, Y.; Nowzari, C.; Tian, Z.; and Ling, Q. 2017. Asynchronous periodic event-triggered coordination of multi-agent systems. In *IEEE Conference Decision Control*, 6696–6701.

Lynch, N. A. 1996. *Distributed algorithms*. Elsevier.

McMahan, B., and Ramage, D. 2017. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*.

Nemirovski, A.; Juditsky, A.; Lan, G.; and Shapiro, A. 2009. Robust stochastic approximation approach to stochastic programming. *SIAM J. Optimization* 19(4):1574–1609.

Shi, W.; Ling, Q.; Yuan, K.; Wu, G.; and Yin, W. 2014. On the linear convergence of the ADMM in decentralized consensus optimization. *In IEEE Trans. Signal Process.* 62(7):1750–1761.

Sicari, S.; Rizzardi, A.; Grieco, L. A.; and Coen-Porisini, A. 2015. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks* 76:146–164.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. S. 2017. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, 4427–4437.

Su, L., and Xu, J. 2018. Securing distributed machine learning in high dimensions. *arXiv preprint arXiv:1804.10140*.

Weiszfeld, E., and Plastria, F. 2009. On the point for which the sum of the distances to n given points is minimum. *Annals of Operations Research* 167(1):7–41.

Xie, C.; Koyejo, O.; and Gupta, I. 2018a. Generalized Byzantine-tolerant SGD. *arXiv preprint arXiv:1802.10116*.

Xie, C.; Koyejo, O.; and Gupta, I. 2018b. Phocas: Dimensional byzantine-resilient stochastic gradient descent. *arXiv preprint arXiv:1805.09682*.

Xie, C.; Koyejo, O.; and Gupta, I. 2018c. Zeno: Byzantine-suspicious stochastic gradient descent. *arXiv preprint arXiv:1805.10032*.

Xu, W.; Li, Z.; and Ling, Q. 2018. Robust decentralized dynamic optimization at presence of malfunctioning agents. *Signal Processing* 153:24–33.

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

Yin, D.; Chen, Y.; Ramchandran, K.; and Bartlett, P. 2018b. Defending against saddle point attack in Byzantine-robust distributed learning. *arXiv preprint arXiv:1806.05358*.