ROBUST DISTRIBUTED GRADIENT DESCENT WITH ARBITRARY NUMBER OF BYZANTINE ATTACKERS

Xinyang Cao and Lifeng Lai

Dept. of Elec. and Comp. Engr., University of California, Davis, {xyacao,lflai}@ucdavis.edu

ABSTRACT

Due to the grow of modern dataset size and the desire to harness computing power of multiple machines, there is a recent surge of interest in the design of distributed machine learning algorithms. However, distributed algorithms are sensitive to Byzantine attackers who can send falsified data to prevent the convergence of algorithms or lead the algorithms to converge to value of the attackers' choice. Some recent work proposed interesting algorithms that can deal with the scenario when up to half of the workers are compromised. In this paper, we propose a novel algorithm that can deal with an arbitrary number of Byzantine attackers.

Index Terms— Byzantine attacker, convergence, distributed gradient descent.

1. INTRODUCTION

The design of distributed optimization algorithms has attracted significant recent research interests [1, 2, 3, 4, 5]. The surge of interest in this area is motivated by many factors. For example, as the size of modern data keeps growing, it may not be possible to fit all data in one machine.

In a typical distributed optimization setup, there are one parameter server and multiple working machines. Most of the existing work in this area assumes that these working machines are honest. However, in practice, there is a risk that some of the working machines might be compromised. Compromised machines may send falsified information to the server to prevent the convergence of the optimization algorithm or to lead the algorithm to converge to a value chosen by these attackers. For example, as shown in [6, 7], the presence of even a single Byzantine worker can prevent the convergence of distributed gradient descent algorithm.

There have been some interesting recent work to design distributed machine learning algorithm [6, 7] that can deal with Byzantine attacks. The main idea of these work is to compare information received from all workers, and compute a quantity that is robust to attackers for algorithm update. For example, the algorithm in [7] uses the geometric median of gradient information received from workers for parameter update. The algorithm in [6] chooses the gradient vector that is closest (in certain sense) to its $m-p$ neighbors, where m is the number of working machines and p is the number of compromised workers, to be the estimated gradient for parameter updating. [6, 7] show that their algorithms can successfully converge even if up to half of all workers are compromised. However, once more than half of the workers are compromised, the algorithms in these interesting work will not converge.

In this paper, we propose a new robust distributed gradient descent algorithm that can converge regardless of the number of compromised workers. The main idea is to ask the server to randomly select a *small* subset of data and compute a noisy gradient based on this small dataset. Even though the computed gradient is very noisy, it can be used as the ground truth to filter out information from attackers. In particular, once the server receives gradient information from workers, it compares the gradient information from each worker with the noisy gradient it has computed. If the distance between the gradient from worker and the noisy gradient computed by itself is small, the server accepts the gradient information from that worker as authentic. After the comparison step, the server then computes the average of all accepted gradient and its own noisy gradient as the final estimated gradient for update. We prove that the algorithm can converge to the neighborhood of the optimal value regardless of the number of compromised workers. We show this result by proving that the distance between the estimated gradient and the true gradient can be universally bounded.

The paper is organized as follows. In Section 2, we describe the model. In Section 3, we describe the proposed gradient descent algorithm and prove the convergence of the proposed algorithm. In Section 4, we provide numerical examples to validate the theoretic analysis. Finally, we offer several concluding remarks in Section 5. Due to space limitations, we only provide outline of proofs.

2. MODEL

We consider a system where the data $X \in \mathcal{X} \subset \mathbb{R}^n$ is generated randomly from a distribution parameterized by unknown vector θ taken value from a set $\Theta \subset \mathbb{R}^d$. Our goal is to infer the unknown parameter θ from data samples. In particular, consider a loss function $f : \mathcal{X} \times \Theta \to \mathbb{R}$, with $f(x, \theta)$ being the risk induced by data point x under the model parameter

θ. We aim to find the model parameter θ^* that minimizes the population risk $F(\theta)$:

$$\theta^* \in \arg\min_{\theta\in\Theta} F(\theta) \triangleq \mathbb{E}[f(X,\theta)]. \quad (1)$$

In this paper, we assume that $F(\theta)$ satisfies the following typical assumption.

Assumption 1. The population risk function $F : \Theta \to \mathbb{R}$ is L-strongly convex, and differentiable over Θ with M-Lipschitz gradient. That is for all $\theta, \theta' \in \Theta$,

$$F(\theta') \geq F(\theta) + < \nabla F(\theta), \theta' - \theta > + L \parallel \theta' - \theta \parallel^2 /2, \quad (2)$$

and $\parallel \nabla F(\theta') - \nabla F(\theta) \parallel \leq M \parallel \theta' - \theta \parallel$, in which $\parallel \cdot \parallel$ is the ℓ_2 norm.

When the distribution of X is known, the population risk can be evaluated exactly and θ^* can be computed by solving the above problem. However, as the distribution is unknown in our setup, it is typical to approximate the population risk $F(\theta)$ using the observed data samples. We assume that there exist N independently and identically distributed data samples X_i, with $i = 1, 2, \cdots, N$. Instead of minimizing the population risk (1) directly, we minimize the empirical risk

$$\min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} f(X_i, \theta). \quad (3)$$

In a typical distributed optimization setup, there are one server and m working machines in the system. These N data samples are distributed into these m working machines, and the server machine can communicate with all working machines synchronously. Let $\mathcal{S}_j$ be the set of data samples that the j-th worker has. In a system with data shuffling, $\mathcal{S}_j$ changes over iterations, while in a system without shuffling, $\mathcal{S}_j$ is fixed. Our algorithm and proof hold regardless whether there is data shuffling or not.

In the classic batch gradient descent, one solves (3) using distributed gradient descent. In particular, at iteration t, each worker $j \in [1, m]$ calculates $\nabla \overline{f}^{(j)}(\theta_{t-1})$ based on local data

$$\nabla \overline{f}^{(j)}(\theta_{t-1}) = \frac{1}{|\mathcal{S}_j|}\sum_{i\in\mathcal{S}_j} \nabla f(X_i, \theta_{t-1}), \quad (4)$$

and sends it back to the server. After receiving information from all machines, the server updates the parameter using

$$\theta_t = \theta_{t-1} - \eta \frac{1}{m}\sum_{i=1}^{m} \nabla \overline{f}^{(j)}(\theta_{t-1}) \quad (5)$$

and sends the updated parameter θ_t to working machines. Here η is the step size. This process continues until a certain stop criteria is satisfied.

In this paper, we consider an adversary setup, in which an unknown subset of working machines might be comprised. Furthermore, the set of compromised working machines might also change over time. If a machine is compromised, instead of the gradient calculated from local data, it can send arbitrary information to the server. In particular, let $\mathcal{B}_t$ denote the set of compromised machines at iteration t, the server receives data $g_t^{(j)}(\theta_{t-1})$ from j-th worker with

$$g_t^{(j)}(\theta_{t-1}) = \begin{cases} \nabla \overline{f}^{(j)}(\theta_{t-1}) & j \notin \mathcal{B}_t \\ \star & j \in \mathcal{B}_t \end{cases}, \quad (6)$$

in which $\star$ denotes an arbitrary vector chosen by the attacker.

In this case with Byzantine attackers, if one continues to use the classic batch gradient as in (5), the algorithm will fail to converge even if there is only one attacker [6, 7]. As discussed above, [6, 7] designed algorithm that converges if the number of compromised machines p is less than $m/2$ (i.e., more than half of the machines are not compromised). The goal of our paper is to design a robust batch gradient descent algorithm that can tolerate *any number* of Byzantine attackers.

3. ALGORITHM AND ANALYSIS

In this section, we describe our algorithm that can deal with an arbitrary number of Byzantine attackers, and analyze its convergence property.

A. Algorithm

The main idea of our algorithm is to ask the server to randomly select a small set of data points $\mathcal{S}_0$ at very beginning. Once $\mathcal{S}_0$ is selected, it is fixed throughout the algorithm. Then at each iteration t, the server calculates a noisy gradient using data points in $\mathcal{S}_0$:

$$\nabla \overline{f}^{(0)}(\theta_{t-1}) = \frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0} \nabla f(X_i, \theta_{t-1}).$$

Different choices of the size of $\mathcal{S}_0$ will strike a tradeoff between convergence speed and computational complexity.

The server then compares $g_t^{(j)}(\theta_{t-1})$ received from machine j with $\nabla \overline{f}_t^{(0)}(\theta_{t-1})$. The server will accept $g_t^{(j)}(\theta_{t-1})$ as authentic value and use it for further processing, if

$$\parallel g_t^{(j)}(\theta_{t-1}) - \nabla \overline{f}_t^{(0)}(\theta_{t-1}) \parallel \leq \xi_1 \parallel \nabla \overline{f}_t^{(0)}(\theta_{t-1}) \parallel, \quad (7)$$

where $\xi_1 < 1$ is a constant.

Assuming there are k values (which is a random variable) being accepted after the comparison step, we denote these k values as $q_t^{(1)}(\theta_{t-1})$, ..., $q_t^{(k)}(\theta_{t-1})$. Then the server updates the parameters as $\theta_t = \theta_{t-1} - \eta G(\theta_{t-1})$, where

$$G(\theta_{t-1}) = \frac{1}{k+1}\left(\sum_{l=1}^{k} q_t^{(l)}(\theta_{t-1}) + \nabla \overline{f}_t^{(0)}(\theta_{t-1})\right). \quad (8)$$

Main steps of the algorithms are listed in Table 1.

Table 1. Proposed algorithm

```
Algorithm Iteration $t \geq 1$
Parameter server:
Initialize: Randomly selects $\theta_0 \in \Theta$;randomly selects $\mathcal{S}_0$;
1: Broadcasts the current model parameter estimator $\theta_{t-1}$
to all working machines;
2: Computes $\nabla \overline{f}_t^{(0)}(\theta_{t-1})$ using $\mathcal{S}_0$;
3: Waits to receive all the gradients from the $m$ machines;
Let $g_t^{(j)}(\theta_{t-1})$ denote the value received from machine $j$;
4: Compares $g_t^{(j)}(\theta_{t-1})$ with $\nabla \overline{f}_t^{(0)}(\theta_{t-1})$;If
$\| g_t^{(j)}(\theta_{t-1}) - \nabla \overline{f}_t^{(0)}(\theta_{t-1}) \| \leq \xi_1 \| \nabla \overline{f}_t^{(0)}(\theta_{t-1}) \|$,
the server accepts it and sets it to be $q_t^{(l)}(\theta_{t-1})$;
5: Assume there are $k$ acceptable value, then
$G(\theta_{t-1}) \leftarrow \frac{1}{k+1}\left(\sum_{l=1}^{k} q_t^{(l)}(\theta_{t-1}) + \nabla \overline{f}_t^{(0)}(\theta_{t-1})\right)$;
6:Updates $\theta_t \leftarrow \theta_{t-1} - \eta G(\theta_{t-1})$;
Working machine $j$:
1: Computes the gradient $\nabla \overline{f}^{(j)}(\theta_{t-1})$;
2: If machine $j$ is honest,
it sends $\nabla \overline{f}^{(j)}(\theta_{t-1})$ back to the server;
If machine $j$ is compromised,
it sends the value determined by the attacker;
```

B. Convergence Analysis

In the following, we analyze the convergence property of the proposed algorithm. Before presenting detailed analysis, here we describe the high level ideas. It is well known that if $\nabla F(\theta)$ is available, then the gradient descent algorithm will converge to θ^* exponentially fast [7]. The main idea of our proof is to show that the distance between $G(\theta)$ and $\nabla F(\theta)$ is universally bounded in Θ regardless the number of attackers. Hence, $G(\theta)$ is a good estimate of $\nabla F(\theta)$. As the result, we can then show that the proposed algorithm converges.

Lemma 1. *For an arbitrary number of attackers, the distance between $G(\theta)$ and $\nabla F(\theta)$ is bounded as*

$$\|G(\theta) - \nabla F(\theta)\| \leq (1+\xi_1)\|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\| + \xi_1 \|\nabla F(\theta) - \nabla F(\theta^*)\|, \forall \theta. \tag{9}$$

Using the M-Lipschitz gradient assumption in Assumption 1, the term $\|\nabla F(\theta) - \nabla F(\theta^*)\|$ can be bounded. In the following, we show that the term $\|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\|$ can also be bounded. For this, we need to present several assumptions and intermediate results.

Assumption 2. There exist positive constants σ_1 and α_1 such that for any unit vector $v \in B$, $\langle \nabla f(X, \theta^*), v\rangle$ is sub-exponential with σ_1 and α_1, that is,

$$\sup_{v \in B} \mathbb{E}[\exp(\lambda \langle \nabla f(X,\theta^*), v\rangle)] \leq e^{\sigma_1^2 \lambda^2/2}, \forall |\lambda| \leq 1/\alpha_1,$$

where B denotes the unit sphere $\theta : \|\theta\|_2 = 1$.

With this assumption, we first have the following lemma that shows $\frac{1}{|\mathcal{S}_0|}\sum_{i \in \mathcal{S}_0} \nabla f(X_i, \theta^*)$ concentrates around $\nabla F(\theta^*)$.

Lemma 2. *Under Assumption 2, for any $\delta \in (0,1)$, let*

$$\Delta_1 = \sqrt{2}\sigma_1 \sqrt{(d \log 6 + \log(3/\delta))/|\mathcal{S}_0|}, \tag{10}$$

and if $\Delta_1 \leq \sigma_1^2/\alpha_1$, then

$$Pr\left\{ \left\| \frac{1}{|\mathcal{S}_0|}\sum_{i \in \mathcal{S}_0} \nabla f(X_i, \theta^*) - \nabla F(\theta^*) \right\| \geq 2\Delta_1 \right\} \leq \frac{\delta}{3}.$$

Second, we define gradient difference $h(x,\theta) \triangleq \nabla f(x,\theta) - \nabla f(x, \theta^*)$ and assume that for every θ, $h(x,\theta)$ normalized by $\| \theta - \theta^* \|$ is also sub-exponential.

Assumption 3. There exist positive constants σ_2 and α_2 such that for any $\theta \in \Theta$ with $\theta \neq \theta^*$ and any unit vector $v \in B$, $\langle h(X,\theta) - \mathbb{E}[h(X,\theta)], v\rangle / \| \theta - \theta^* \|$ is sub-exponential with σ_2 and α_2, that is,

$$\sup_{\theta \in \Theta, v \in B} \mathbb{E}\left[\exp\left(\frac{\lambda \langle h(X,\theta) - E[h(X,\theta)], v\rangle}{\|\theta - \theta^*\|}\right)\right] \leq e^{\sigma_2^2 \lambda^2/2}, \forall |\lambda| \leq \frac{1}{\alpha_2}.$$

This allows us to show that $\frac{1}{|\mathcal{S}_0|}\sum_{i \in \mathcal{S}_0} h(X_i, \theta)$ concentrates on $\mathbb{E}[h(X,\theta)]$ for every fixed θ:

Lemma 3. *Suppose Assumption 3 holds. For any $\delta \in (0,1)$ and fix any $\theta \in \Theta$, let $\Delta_1' = \sqrt{2}\sigma_2\sqrt{(d \log 6 + \log(3/\delta))/|\mathcal{S}_0|}$, and if $\Delta_1' \leq \sigma_2^2/\alpha_2$, then*

$$Pr\left\{ \left\| \frac{1}{|\mathcal{S}_0|}\sum_{i \in \mathcal{S}_0} h(X_i, \theta) - \mathbb{E}[h(X,\theta)] \right\| \geq 2\Delta_1' \|\theta - \theta^*\| \right\} \leq \frac{\delta}{3}.$$

Assumption 4. For any $\delta \in (0,1)$, there exists an $M' = M'(\delta)$ such that

$$\Pr\left\{ \sup_{\theta, \theta' \in \Theta : \theta \neq \theta'} \frac{\|\nabla f(X,\theta) - \nabla f(X,\theta')\|}{\|\theta - \theta'\|} \leq M' \right\} \geq 1 - \frac{\delta}{4}.$$

Assumption 4 ensures that $\nabla f(X,\theta)$ is M'-Lipschitz with high probability.

With these assumptions and intermediate lemmas, we are ready to state our universal bound for $\|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\|$.

Proposition 1. *Suppose Assumptions 2-4 hold, and $\Theta \subset \{\theta : \| \theta - \theta^* \| \leq r\sqrt{d}\}$ for some positive parameter r. For any $\delta \in (0,1)$,*

$$Pr\{\forall \theta : \|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\| \leq 4\Delta_2 \|\theta - \theta^*\| + 2\Delta_1\} \geq 1 - \delta, \tag{11}$$

in which Δ_1 is defined in (10) *and $\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1 + \tau_2)/|\mathcal{S}_0|}$, with $\tau_1 = d \log 6 + d \log((M \vee M')/\sigma_2)$, and*
$\tau_2 = 0.5 d \log(|\mathcal{S}_0|/d) + \log(3/\delta) + \log(r\sqrt{d})$.

Proof. (Outline): The proof relies on the typical ϵ-net argument. Let $\Theta_\epsilon = \{\theta_1, ..., \theta_{N_\epsilon}\}$ be an ϵ-cover of Θ. Then fix any $\theta \in \Theta$, there exists a $\theta_j \in \Theta_\epsilon$ such that $\| \theta - \theta_j \| \leq \epsilon$. By triangle inequality,

$$\left\|\nabla \overline{f}^{(0)}(\theta) - \nabla F(\theta)\right\| \leq \|\nabla F(\theta) - \nabla F(\theta_j)\|$$
$$+ \left\|\nabla \overline{f}^{(0)}(\theta) - \nabla \overline{f}^{(0)}(\theta_j)\right\| + \left\|\nabla \overline{f}^{(0)}(\theta_j) - \nabla F(\theta_j)\right\|.$$

These terms can be upper bounded using assumption 1, assumption 4 and Lemma 3. We can then employ union bound over Θ_ϵ to finish the argument. □

Combining Lemma 1 and Proposition 1, we know that $G(\theta)$ is a good approximation of $\nabla F(\theta)$. Using this fact, we have the following convergence result.

Theorem 1. *If Assumptions 1-4 hold, then regardless the number of attackers, we have probability at least* $1 - \delta$ *that*

$$\|\theta_t - \theta^*\| \leq (1-\rho)^t \|\theta_0 - \theta^*\| + (2\eta\Delta_1 + 2\eta\xi_1\Delta_1)/\rho,$$

in which $\rho = 1 - \left(\sqrt{1 - \frac{L^2}{4M^2}} + 4\Delta_2\eta + \eta\xi_1(4\Delta_2 + M)\right)$.

This theorem shows that under the event which would happen with highly probability, the estimated θ can converge to the neighborhood of θ^* exponentially fast.

4. NUMERICAL RESULTS

In this section, we provide a numerical example to illustrate the analytical results. In the example, we focus on linear regression, in which $Y_i = X_i^T\theta^* + \epsilon_i, i = 1, 2, \cdots, N$, where $X_i \in \mathbb{R}^d$, θ^* is a $d \times 1$ vector and ϵ_i is noise. In the simulation, we use synthesized data. We set the dimension $d = 20$, the total number of data $N = 10000$, the number of worker $m = 100$, and evenly distribute data among these machines. We set $|\mathcal{S}_0| = 50$. Furthermore, we set $\xi_1 = 0.975$ and let $\theta^* \overset{i.i.d.}{\sim} \mathcal{N}(0, 4)$. Here $\mathcal{N}(\mu, \sigma^2)$ denotes Gaussian variables with mean μ and variance σ^2. After θ^* is generated, we fix it. We then generate each entry of X_i using $\mathcal{N}(0, 16)$, and generate Y_i using the linear relationship mentioned above. We illustrate our results with two different attacks: 1) Inverse attack, in which the attackers inverse the correct gradient values; and 2) Random attack, in which the attackers randomly generate gradient value. In our simulation, we compare three algorithms: 1) Gradient descent using only data from $\mathcal{S}_0$; 2) Algorithm proposed in [7]; and 3) The proposed algorithm described in Table 1. Due to space limitation, we only show the result for the case with 95 attackers, i.e., most of the machines are compromised.

Figure 1 plots the l_2 norm of the difference between estimated θ_t and θ^* for $t = 1, \cdots, 250$ with the inverse attack. It is clear from the figure that the algorithm in [7] does not converge as the number of attackers is more than half of the

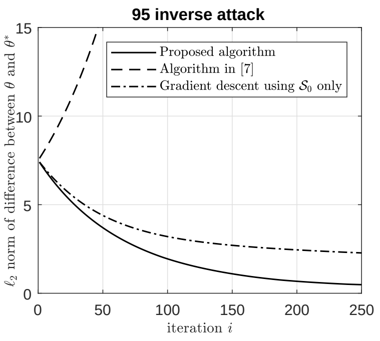

Fig. 1. Inverse attack.

total number of machines. The proposed algorithm, however, still converges. Furthermore, even though there are only 5 honest workers and the server does not know the identities of these honest workers, the proposed algorithm can still benefit from these workers, as the proposed algorithm outperforms the algorithm that only relies on information from $\mathcal{S}_0$.

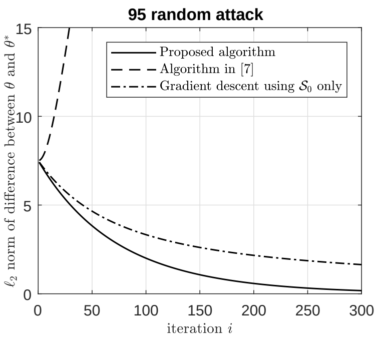

Fig. 2. Random attack.

Figure 2 shows the l_2 norm of the difference between estimated θ_t and θ^* for $t = 1, \cdots, 300$ for the case with random attack. Similar to the scenario with inverse attack, our algorithm outperforms the algorithm that uses $\mathcal{S}_0$ only, and both algorithms converges while the algorithm in [7] diverges.

5. CONCLUSION

In this paper, we have proposed a robust gradient descent algorithm that can tolerant an arbitrary number of Byzantine attackers. We have shown that the proposed algorithm converges to the true value. We have also provided numerical examples to illustrate the performance of the proposed algorithm and compared it with those of other algorithms.

6. ACKNOWLEDGMENT

The work of X. Cao and L. Lai was supported by National Science Foundation under grants CCF-1717943 and ECCS-1711468.

7. REFERENCES

[1] A. Crotty, A. Galakatos, and T. Kraska, "Tupleware: Distributed machine learning on small clusters.," *IEEE Data Eng. Bull.*, vol. 37, no. 3, pp. 63–76, Sep. 2014.

[2] M. Jordan, J. Lee, and Y. Yang, "Communication-efficient distributed statistical inference," *arXiv preprint arXiv:1605.07689*, Nov. 2016.

[3] F. Provost and D. Hennessy, "Scaling up: Distributed machine learning with cooperation," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, USA, Aug. 1996, vol. 1, pp. 74–79.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[5] P. Moritz, R. Nishihara, I. Stoica, and M. Jordan, "Sparknet: Training deep networks in spark," *arXiv preprint arXiv:1511.06051*, Feb. 2015.

[6] P. Blanchard, E. Mhamdi, R. Guerraoui, and J. Stainer, "Byzantine-tolerant machine learning," *arXiv preprint arXiv:1703.02757*, Mar. 2017.

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