Project Title: Defending Large-Scale Distributed Machine Learning Against Adversarial Attacks

Group Members: Lili Su, Vidyasagar Sadhu, Seyyed A. Fatemi, Rehana Mahfuz

1 Meetings in the last year:

- Meeting at the NSF Site Visit at Purdue University on December 6, 2016. Attendance: Vidyasagar Sadhu, Rehana Mahfuz and Lili Su.
- Skype meeting on Sep. 10th, 2016, 3pm EST, attended by Vidyasagar, Rehana and Lili.
- Rehana and Lili met in person at Purdue on Nov. 18th, 2016 and Nov. 23rd, 2016.
- Skype meeting on Nov. 27th, 2016, 8pm EST.
- Skype meeting on July 4, 2017, 5 pm UTC. This was attended by all four members of the team. The Byzantine Gradient Descent algorithm was discussed.
- Skype meeting on July 8, 2017, 2 am UTC. This was also attended by all four members. The discussion focused on implementation of Byzantine Gradient Descent.
- Communication via email has been going on since the project started.

2 Descriptions of Progress

We made progress on the following three aspects of the proposed research: (1) The practical impacts; (2) Theoretical abstraction and explicit problem formulation; (2) Algorithm implementation and testing.

Practical Impacts: Training machine learning models in adversarial settings is catching attentions among the industrial practitioners. This arises from the popularity of the implementation of *Federated Learning*, a modern machine learning paradigm proposed and tested by Google researchers.

Federated Learning: *Federated Learning* is originally proposed to address users' privacy concerns. Federated learning aims at learning an accurate model without collecting data from owners and storing the data in the cloud. The training data is kept locally on the owners' computing devices, which are recruited to participate directly in the model training process and hence function as working machines. Google has been intensively testing this new paradigm in their recent projects such as *Gboard*, the Google Keyboard.

Security Issues and Challenges: Compared to "training within cloud", Federated Learning has lower privacy risk, but inevitably becomes less secured. In particular, it faces the following three key challenges:

- Security: The devices of the recruited data owners can be easily reprogrammed and completely controlled by external attackers, and thus behave adversarially.
- Small local datasets versus high model complexity: While the total number of data samples over all data owners may be large, each individual owner may keep only a small amount of data, which by itself is insufficient for learning a complex model.
- Communication constraints: Data transmission between the recruited devices and the cloud may suffer from high latency and low-throughout. Communication between them is therefore a scarce resource, whose usage should be minimized.

2.1 Theoretical abstraction and explicit problem formulation

This project is multi-institution/multi-disciplinary. We divided the project into two stages:

1. theoretical abstraction and explicit problem formulation;
2. algorithm implementation.

Together with Yudong Chen (Cornell) and Jiaming Xu (Purdue), the project leader Lili Su formulated the machine learning problem as distributed statistical learning problems. Three of us submitted a manuscript for publication. The manuscript is titled "Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent" with arXiv 1705.05491.

Problem Formulation Let $X \in \mathcal{X}$ be the input data generated according to some distribution μ. Let $\Theta \subset \mathbb{R}^d$ be the set of all choices of model parameters. We consider a loss function $f : \mathcal{X} \times \Theta \to \mathbb{R}$, where $f(x, \theta)$ measures the risk induced by a realization x of the data under the model parameter choice θ. A classical example is linear regression, where $x = (w, y) \in \mathbb{R}^d \times \mathbb{R}$ is the feature-response pair and $f(x, \theta) = \frac{1}{2}(\langle w, \theta \rangle - y)^2$ is the usual squared loss.

We are interested in learning the model choice θ^* that minimizes the *population risk*, i.e.,

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

assuming that $\mathbb{E}[f(X, \theta)]$ is well defined over Θ.[1] The model choice θ^* is optimal in the sense that it minimizes the average risk to pay if the model chosen is used for prediction in the future with a fresh random data sample.

When μ—the distribution of X—is known, which is rarely the case in practice, the population risk can be evaluated exactly, and θ^* can be computed by solving the minimization problem in (1). We instead assume that μ is *unknown*, in which case the population risk function $F(\cdot) = \mathbb{E}[f(X, \cdot)]$ can only be approximated using the observed data samples generated from μ. In particular, we assume that there exist N independently and identically distributed data samples $X_i \overset{\text{i.i.d.}}{\sim} \mu$ for $i = 1, \ldots, N$. Note that estimating θ^* using finitely many data samples will always have a *statistical error* due to the randomness in the data, even in the centralized, failure-free setting. Our results account for this effect.

System Model We are interested in distributed solutions of the above statistical learning problem. Specifically, the system of interest consists of a parameter server[2] and m working machines. In the example of Federated Learning, the parameter server represents the cloud, and the m working machines correspond to m data owners' computing devices.

We assume that the N data samples are distributed evenly across the m working machines. In particular, each working machine i keeps a subset $\mathcal{S}_i$ of the data, where $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ and $|\mathcal{S}_i| = N/m$. We further assume that the parameter server communicates with all working machines in synchronous rounds. We leave the asynchronous setting as future directions.

Among the m working machines, we assume that up to q of them can suffer Byzantine failures and thus behave maliciously; for example, they may be reprogrammed and completely controlled by the system attacker. In a given execution, the set of Byzantine machines can even change between

[1] For example, if $\mathbb{E}[|f(X, \theta)|]$ is finite for every $\theta \in \Theta$, the population risk $\mathbb{E}[f(X, \theta)]$ is well defined.

[2] Note that, due to communication bandwidth constraints, practical systems use multiple networked parameter servers. In this chapter, for ease of explanation, we assume there is only one parameter server in the system. Fortunately, as can be seen from our algorithm descriptions and our detailed correctness analysis, the proposed algorithm also works for the aforementioned more practical setting.

communication rounds. Byzantine machines are assumed to have complete knowledge of the system, including the total number of working machines m, all N data samples over the whole system, the programs that the working machines are supposed to run, and the program run by the parameter server. Moreover, Byzantine machines can collaborate with each other. The only constraint is that these machines cannot corrupt the local data — but they can lie when communicating with the server. This arbitrary behavior of Byzantine machines creates unspecified dependency across communication rounds — a key challenge in our algorithm design and convergence analysis.

Results We proposed a Byzantine gradient descent method. Specifically, the parameter server aggregates the local gradients reported by the working machines in three steps: (1) it partitions all the received local gradients into k batches and computes the mean for each batch, (2) it computes the *geometric median* of the k batch means, and (3) it performs a gradient descent step using the geometric median.

We prove that the proposed algorithm can tolerate q Byzantine failures up to $2(1+\epsilon)q \leq m$ for an arbitrarily small but fixed constant $\epsilon > 0$, is applicable even in the scarce local data regime where $N/m \ll d$, and only requires $\log(N)$ communication rounds. However, as q increases, the estimation error also increases. In particular, the error in estimating the target model parameter θ^* converges exponentially fast to $\max\{\sqrt{dq/N},\ \sqrt{d/N}\}$, whereas the idealized estimation error rate in the centralized and failure-free setting is $\sqrt{d/N}$. The total computational complexity of our algorithm is of $O((N/m)d\log N)$ at each working machine and $O(md+qd\log^3 N)$ at the parameter server, and the total communication cost is of $O(md\log N)$.

Notably, our algorithm does *not* assume that at each iteration fresh samples are drawn, or that the data is split into multiple chunks beforehand and the gradient is computed using a new chunk at each round. This poses a significant challenge in our convergence proof: there exists complicated probabilistic dependency among the iterates and the aggregated gradients. Even worse, such dependency cannot be specified due to the arbitrary behavior of the Byzantine machines. We overcome this challenge by proving that the geometric median of means of gradients *uniformly* converges to the true gradient function $\nabla F(\theta)$.

2.2 Algorithm Implementation

Rehana Mahfuz (Purdue) and Vidyasagar Sadhu are in charge of the algorithm implementation and testing. Up to now, Rehana simulated Byzantine GD algorithm proposed in "Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent" (Y. Chen, L. Su, J.Xu). We will test the problem further using mobile phones.

Simulation Results Experiments were conducted to compare the performance of Gradient Descent with Byzantine Gradient Descent. Twelve machines, divided into four batches were subjected to both algorithms. The underlying truth was (3, 4), and the estimate was initialized to (9, 10). When one of the machines was Byzantine, it was found that in BGD, the estimate converged to the underlying truth much more accurately than in GD. Each algorithm was performed 30 times with 100 iterations, and the final estimate for each execution was recorded. The average value that BGD converged to was (2.70, 3.54), while GD converged to an average value of (1.62, -9.67). In contrast to the vulnerability of standard gradient descent, the BGD proved to be able to tolerate that fault.

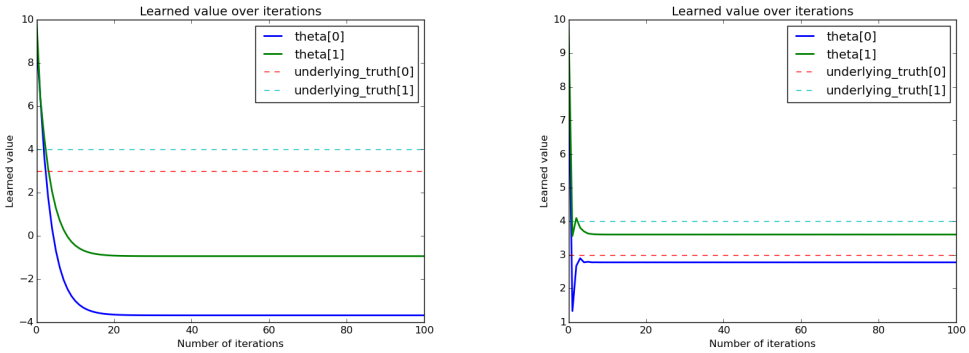

Fig. 1: The first figure is about the convergence of Standard Gradient Descent. The GD converges to (-3.68, -0.95) for an underlying truth of (3, 4) in the presence of a Byzantine machine. The second figure is abut the convergence of Byzantine Gradient Descent. The BGD converges to (2.78, 3.6) for an underlying truth of (3, 4) in the presence of a Byzantine machine.

3 Outputs

- Theoretical Manuscript: Yudong Chen, Lili Su and Jiaming Xu, "Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent." Online preprint: arXiv 1705.05491.
- Theoretical results presentation: Poster at 2017 North American School of Information Theory, GaTech, June 2017. Presented by Lili Su.
- Project overview poster: NSF Annual Site Visit, Purdue University, Dec. 2016. Presented by Rehana Mahfuz, Vidyasagar Sadhu and Lili Su.
- The Byzantine Gradient Descent algorithm was implemented by Rehana Mahfuz and Vidyasagar Sadhu. The codes can be found in the following GitHub repository: https://github.com/rmahfuz/Fault-tolerant-distributed-optimization/tree/Byzantine-Gradient-Descent

4 Future directions

- Theoretical directions: (1) Relax the asynchrony assumption; (2) Characterize how much privacy is lost; Improve algorithm performance.
- Implementation: To further explore and establish the capabilities of Byzantine Gradient Descent, we plan to experiment with more machines, a larger number of byzantine faults, data of higher dimensions, and more generally, vary more parameters and observe the results. Remaining balance from the seed grant and plans on spending: The remaining balance is . The plan is for the team members to meet (at a location yet to be decided) to explore the possibilities of testing the performance of Byzantine Gradient Descent.
- Texting on mobile devices, such as mobile phones.

5 Remaining balance from the seed grant and plans on spending

The remaining balance is $6000. The plan is for the team members to meet (at a location yet to be decided) to explore the possibilities of testing the performance of Byzantine Gradient Descent.