High Dimensional Distributed Gradient Descent with Arbitrary Number of Byzantine Attackers

Wenyu Liu[1], **Tianqiang Huang** [1], **Pengfei Zhang** [2], **Zong Ke** [3], **Minghui Min** [4] , **Puning Zhao** [5*]

[1] College of Computer and Cyber Security, Fujian Normal University
[2] Anhui University of Science and Technology
[3] National University of Singapore
[4] China University of Mining Technology
[5] School of Cyber Science and Technology, Sun Yat-sen University
{liuwenyu, fjhtq}@fjnu.edu.cn, {zpf.bupt}@bupt.cn,{a0129009}@u.nus.edu
{minmh}@cumt.edu.cn,{zhaopn}@mail.sysu.edu.cn

Abstract

Adversarial attacks pose a major challenge to distributed learning systems, prompting the development of numerous robust learning methods. However, most existing approaches suffer from the curse of dimensionality, i.e. the error increases with the number of model parameters. In this paper, we make a progress towards high dimensional problems, under arbitrary number of Byzantine attackers. The cornerstone of our design is a direct high dimensional semi-verified mean estimation method. The idea is to identify a subspace with large variance. The components of the mean value perpendicular to this subspace are estimated using corrupted gradient vectors uploaded from worker machines, while the components within this subspace are estimated using auxiliary dataset. As a result, a combination of large corrupted dataset and small clean dataset yields significantly better performance than using them separately. We then apply this method as the aggregator for distributed learning problems. The theoretical analysis shows that compared with existing solutions, our method gets rid of $\sqrt{d}$ dependence on the dimensionality, and achieves minimax optimal statistical rates. Numerical results validate our theory as well as the effectiveness of the proposed method.

Introduction

Many modern machine learning tasks require distributed computing and storage. In such systems, there are many *worker machines*, which operate under the coordination of a centralized server, known as the *master machine*. During the whole training process, worker machines only need to compute an update to the current model, and send it to the master, while the local dataset remains private. This framework is called Federated Learning (FL) (McMahan et al. 2017), which has drawn considerable attention in recent years (Zhang et al. 2021; Kairouz et al. 2021). With advantages in both computational efficiency and privacy protection, FL has been widely applied in various areas, including smart devices, industrial engineering and healthcare (Li et al. 2020a,b).

*corresponding author

Despite significant advantages and widespread applications, FL is still facing many challenges (Kairouz et al. 2021). One critical issue is that not all machines are trustable (Lyu, Yu, and Yang 2020). In large scale FL systems, some worker machines may send wrong gradients to the master due to various reasons, including system crashes, faulty hardware, communication errors, and even malicious attacks (Bagdasaryan et al. 2020; Bhagoji et al. 2019; Fang et al. 2020; Sun et al. 2021; Luo et al. 2021). These faults are typically modeled as Byzantine failure (Lamport, Shostak, and Pease 1982), such that some workers are manipulated by an adversary, and send wrong gradient vectors to the master.

There have been extensive research on Byzantine robust distributed learning. Many popular methods (Blanchard et al. 2017; Guerraoui, Rouault et al. 2018; Chen, Su, and Xu 2017; Yin et al. 2018) have two issues that make them not perfect. Firstly, these methods only allow less than half worker machines to be Byzantine. Secondly, the statistical risk grows with dimension d, which is serious in modern large scale learning models. For the former issue, several new methods suitable for arbitrary number of Byzantine attackers have been proposed recently (Cao and Lai 2019; Regatti, Chen, and Gupta 2020; Xie, Koyejo, and Gupta 2019, 2020; Cao et al. 2020). These methods rely on the help of a small auxiliary clean dataset, which is necessary since the breakdown point of robust statistics can not exceed $1/2$ (Hampel 1971). The latter problem, i.e. curse of dimensionality, has been addressed by some recent works based on high dimensional robust statistics (Diakonikolas and Kane 2023), such as (Su and Xu 2018; Shejwalkar and Houmansadr 2021; Zhu, Jiao, and Jordan 2022), which have constant dependence on d. However, to the best of our knowledge, there are no previous methods that can solve both two issues mentioned above. Unfortunately, in modern federated learning problems, it may happen that models have large scale and most workers are not trusted.

In this paper, we tackle both problems mentioned above simultaneously. In particular, we propose a novel approach to high dimensional distributed learning under arbitrary number of Byzantine attackers, with the help of a small auxiliary dataset. The core of our new approach is a new al-

gorithm for semi-verified mean estimation, whose goal is to estimate the statistical mean by combining a small clean dataset and a large untrusted dataset (Charikar, Steinhardt, and Valiant 2017). The basic idea is to filter out some gradient vectors that are very likely to come from Byzantine workers, and then identify a linear subspace, in which the components in this subspace are hard to estimate using these corrupted gradient vectors. These components are then estimated using auxiliary clean dataset. For other components that are perpendicular to this subspace, we just use the projected sample average. Our design is partially motivated by (Diakonikolas et al. 2021). A theoretical analysis is provided, including the upper bound of our algorithm and the information-theoretic minimax lower bound. The new semi-verified mean estimation algorithm is then used as the gradient aggregator for robust distributed learning problem.

Preliminaries

Problem Statement of Byzantine Robust Distributed Learning

To begin with, we formalize the problem of distributed learning under Byzantine failures. Our formalization follows (Chen, Su, and Xu 2017; Yin et al. 2018; Zhu et al. 2023).

Suppose there is a master machine W_0 and m worker machines, $W_1, \ldots, W_m$. A clean dataset $A = \{\mathbf{Z}_{01}, \ldots, \mathbf{Z}_{0N_A}\}$ is stored in the master, with size $|A| = N_A$. Each worker machine W_i has n samples $\mathbf{Z}_{i1}, \ldots, \mathbf{Z}_{in}$. All samples stored in both master and worker machines are identically and independently distributed (i.i.d), following a common distribution Q.

Let $f(\mathbf{w}, \mathbf{z})$ be a loss function of a parameter vector $\mathbf{w} \in \Omega \subseteq \mathbb{R}^d$, in which Ω is the parameter space. The population risk function is defined as

$$F(\mathbf{w}) = \mathbb{E}[f(\mathbf{w}, \mathbf{Z})], \tag{1}$$

in which $\mathbf{Z} \sim Q$. Our goal is to learn the minimizer of F, i.e. $\mathbf{w}^* = \arg\min_{\mathbf{w}\in\Omega} F(\mathbf{w})$.

The FL framework runs as following. In each iteration, the master machine broadcasts current parameter $\mathbf{w}_t$ to all worker machines. Then these worker machines calculate the gradient using the local dataset:

$$\mathbf{X}_i(t) = \frac{1}{n}\sum_{j=1}^{n} \nabla f(\mathbf{w}_t, \mathbf{Z}_{ij}). \tag{2}$$

For any benign workers, the master can receive $\mathbf{X}_i(t)$ exactly. On the contrary, Byzantine machines can send arbitrary messages determined by the attacker. Denote $\mathbf{Y}_i(t) \in \mathbb{R}^d$ as the vector received by the master at t-th iteration, then

$$\mathbf{Y}_i(t) = \begin{cases} \mathbf{X}_i(t) & \text{if} \quad i \notin \mathcal{B} \\ \star & \text{if} \quad i \in \mathcal{B}. \end{cases} \tag{3}$$

Furthermore, we can calculate the gradient using the clean dataset stored in the master that is never corrupted, i.e.

$$\mathbf{X}_0(t) = \frac{1}{N_A}\sum_{j=1}^{N_A} \nabla f(\mathbf{w}_t, \mathbf{Z}_{0j}). \tag{4}$$

Algorithm 1: Robust Distributed Gradient Descent

Input: Master machine W_0, worker machines $W_1, \ldots, W_m$
Parameter: Initial weight parameter $\mathbf{w}_0 \in \Omega$, step length η, total number of iterations T
Output: Estimated weight $\hat{\mathbf{w}}$

for $t = 0, 1, \ldots, T-1$ **do**
  Master machine: broadcast current parameter $\mathbf{w}_t$ to all worker machines;
  for $i \in [m]$ **in parallel do**
    Worker machine i: compute local gradient $\mathbf{X}_i(t)$ using (2);
    if i is normal machine **then**
      send $\mathbf{X}_i(t)$ to master;
    else
      send arbitrary d dimensional vector to master;
    end if
  end for
  Master machine: Receive $\mathbf{Y}_i(t)$, $i = 1, \ldots, m$ from each worker machine;
  Calculate aggregated gradient $g(\mathbf{w}_t)$ using $\mathbf{Y}_i(t), i = 1, \ldots, m$ and clean dataset in W_0;
  Update parameter $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta g(\mathbf{w}_t)$;
end for

We would like to clarify that the adversary knows our algorithm, as well as m gradient vectors $\mathbf{X}_i(t)$, $i = 1, \ldots, m$. However, the adversary has no knowledge of the clean dataset A. This is practical since it is usually not hard to collect only a small set of auxiliary clean data from a reliable source that is not inspected by the attacker.

It remains to discuss how to calculate the aggregated gradient $g(\mathbf{w}_t)$. This problem can be formulated as *Semi-verified mean estimation* (see Definition 1), which was first defined in (Charikar, Steinhardt, and Valiant 2017).

Problem Statement of Semi-verified Mean Estimation

The aggregator $g(\mathbf{w}_t)$ estimates $\nabla F(\mathbf{w}_t)$ using $\mathbf{Y}_i(t)$, $i = 1, \ldots, m$ as well as $\mathbf{X}_0(t)$, which is calculated using (4). Among m worker machines, αm of them are ensured to be benign, while others are probably Byzantine. Therefore, we state semi-verified mean estimation problem as follows.

Definition 1. (Semi-verified mean estimation problem) Suppose $\mathbf{X}_1, \ldots, \mathbf{X}_m$ are i.i.d with mean μ^* and covariance matrix V^*. $\mathcal{B} \subset S_0 = [m]$ is the set of attacked samples. If $i \notin \mathcal{B}$, then $\mathbf{Y}_i = \mathbf{X}_i$, otherwise $\mathbf{Y}_i$ is an arbitrary vector determined by the attacker. In addition, we have a sample $\mathbf{X}_0$ with mean μ^* and covariance matrix V_A^*, which is guaranteed to be unattacked. The task is to estimate μ^* using untrusted samples $\mathbf{Y}_1, \ldots, \mathbf{Y}_m$ and the clean sample $\mathbf{X}_0$.

Consider that the semi-verified mean estimation need to be repeated for all iterations, we omit timestep t in Definition 1. Each sample $\mathbf{Y}_i$ corresponds to the gradient vector from a worker machine W_i. At iteration t, $\mu^* = \nabla F(\mathbf{w}_t)$. Denote V_0 as the covariance matrix of $\nabla f(\mathbf{w}_t, \mathbf{Z}_{ij})$, then

according to (2) and (4), V^* and V_A^* in Definition 1 become

$$V^* = \frac{V_0}{n}, V_A^* = \frac{V_0}{N_A}. \quad (5)$$

Finally, we discuss two attack models that are different on whether the attacked samples are randomly (Definition 2) or carefully selected (Definition 3).

Definition 2. (Additive Contamination Model) The adversary randomly pick at most $(1-\alpha)m$ samples from $\mathbf{X}_1, \ldots, \mathbf{X}_m$, and alter their values arbitrarily.

Definition 3. (Strong Contamination Model) The adversary picks $(1-\alpha)m$ samples from $\mathbf{X}_1, \ldots, \mathbf{X}_m$ according to their values, and alter their values arbitrarily.

These definitions follow (Diakonikolas et al. 2016; Diakonikolas, Kane, and Pensia 2020). Under additive contamination model, adversarial samples are randomly selected, so the distribution of remaining clean samples remains unchanged; equivalently, there are already αm samples, and the adversary injects the remaining $(1-\alpha)m$ corrupted samples with arbitrary values. Under strong contamination model, since adversarial samples are carefully selected by the attacker, and thus those remaining samples follow a different distribution. Even if good samples can be identified, estimation of μ^* is still not guaranteed.

Practical scenarios usually lie in between these two models. In distributed learning problems, the attacked worker machines are usually not randomly selected, thus additive contamination model underestimates the impact of Byzantine attacks. On the contrary, strong contamination model tends to overestimate such impact, since it is unlikely for the adversary to gather full information and design optimal attack strategy. In this work, we analyze both two models.

Semi-verified Mean Estimation

Our algorithm for semi-verified mean estimation is shown in Algorithm 2, which is partially motivated by the Subspace Isotropic Filtering (SIFT) algorithm proposed in (Diakonikolas et al. 2021). The idea is illustrated in Figure 1. The procedures are explained as follows.

(1) *Initialization and prefiltering.* Let $S_0 = \{1, \ldots, m\}$ be the indices of all untrusted samples and initialize $S = S_0$. We use a prefilter (step 2) to remove the samples whose norms are too large. This step is mainly used to simplify the theoretical analysis, and may not be necessary in practice.

(2) *Iterative filtering.* The algorithm then sanitizes the dataset by removing some samples that are highly likely to be attacked. In each iteration, sample mean and covariance matrix are calculated first:

$$\mu(S) = \frac{1}{|S|}\sum_{i\in S}\mathbf{Y}_i, \quad (6)$$

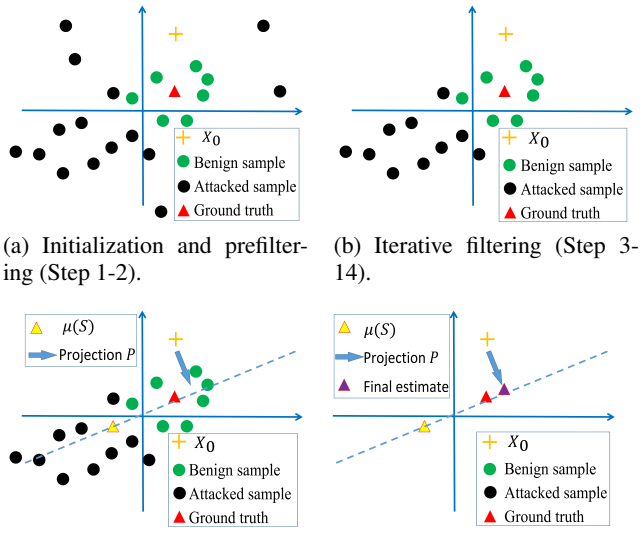
$$V(S) = \frac{1}{|S|}\sum_{i\in S}(\mathbf{Y}_i - \mu(S))(\mathbf{Y}_i - \mu(S))^T. \quad (7)$$

If some eigenvalues of $V(S)$ are too large, then there should be some attacked samples which are moved in the directions of corresponding eigenvectors by the adversary. Motivated by this intuition, we conduct spectral decomposition of $V(S)$, such that $V(S) = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, with $\mathbf{\Lambda} =$

Algorithm 2: Semi verified mean estimation

Input: Untrusted samples $\mathbf{Y}_1, \ldots, \mathbf{Y}_m$ (from (3)), and clean sample $\mathbf{X}_0$ (from (4))
Parameter: p, λ_c
Output: Estimated mean $\hat{\mu}$

1: Initialize $S = S_0$;
2: $S = S \setminus \{i | \|\mathbf{Y}_i\| > m^{1/3}\}$;
3: **while** True **do**
4: Calculate sample mean $\mu(S)$ and sample covariance $V(S)$ using (6) and (7);
5: Conduct spectral decomposition of $V(S)$, such that $V(S) = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, in which $\mathbf{U}$ is orthogonal, and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$, $\lambda_1 \geq \ldots \geq \lambda_d$;
6: **if** $\lambda_p \geq \lambda_c$ **then**
7: Let $\mathbf{P} = \mathbf{U}_p\mathbf{U}_p^T$, in which $\mathbf{U}_p \in \mathbb{R}^{d\times p}$ is the first p columns of $\mathbf{U}$;
8: For each $i \in S$, calculate τ_i using (8);
9: $\tau_{\max} = \max_{i\in S}\tau_i$;
10: For each $i \in S$, remove i from S with probability $\tau_i/\tau_{\max}$;
11: **else**
12: break;
13: **end if**
14: **end while**
15: Calculate $V(S)$, $\mathbf{P}$;
16: $\hat{\mu} = \mathbf{P}\mathbf{X}_0 + (\mathbf{I}-\mathbf{P})\mu(S)$;
17: **return** $\hat{\mu}$;



(a) Initialization and prefiltering (Step 1-2).
(b) Iterative filtering (Step 3-14).
(c) Calculate sample mean and projection matrix (Step 15).
(d) Calculate final estimate $\hat{\mu}$ (Step 17).

Figure 1: A two-dimensional illustration of the semi-verified mean estimation method shown in Algorithm 2. μ^* is represented by the red triangle. Benign and attacked samples correspond to green and black dots, respectively. The orange plus sign denotes $\mathbf{X}_0$.

$\mathrm{diag}(\lambda_1, \ldots, \lambda_d)$, $\lambda_1 \geq \ldots \geq \lambda_d$. If there are p eigenvalues larger than a certain threshold λ_c, i.e. $\lambda_p \geq \lambda_c$, then the algorithm removes some points to reduce the eigenvalue. In particular, each sample i is assigned a score

$$\tau_i = \|\mathbf{P}V^{-\frac{1}{2}}(S)(\mathbf{Y}_i - \mu(S))\|_2^2, \tag{8}$$

in which $\mathbf{P} = \mathbf{U}_p\mathbf{U}_p^T$, with $\mathbf{U}_p \in \mathbb{R}^{d\times p}$ being the matrix of first p columns of $\mathbf{U}$. $\mathbf{P}$ is a projection matrix to the space spanned by p principal components.

We now give an intuitive explanation of (8). We first regularize the dataset by linear transformation with $V^{-\frac{1}{2}}(S)$. It is easy to show that samples after transformation have an identity covariance matrix:

$$\begin{aligned}&\frac{1}{|S|}\sum_{i\in S}\left[V^{-\frac{1}{2}}(S)(\mathbf{Y}_i - \mu(S))\right]\left[V^{-\frac{1}{2}}(S)(\mathbf{Y}_i - \mu(S))\right]^T\\ =\;& V^{-\frac{1}{2}}(S)V(S)V^{-\frac{1}{2}}(S) = \mathbf{I}_d.\end{aligned} \tag{9}$$

After transformation, if a sample is far from the mean, i.e. $\|V^{-\frac{1}{2}}(S)(\mathbf{Y}_i - \mu(S))\|$ is large, then it is likely that this sample is attacked. As has been discussed earlier, it would be better to filter samples in directions corresponding to large eigenvalues. Therefore, (8) incorporates the projection matrix $\mathbf{P}$. Larger τ_i indicates higher confidence that sample i is attacked. With this intuition, sample i is removed with probability $\tau_i/\tau_{\max}$, where $\tau_{\max} = \max_{i\in S}\tau_i$. This randomized removal is mainly for theoretical convenience; in practice, one may simply remove the k samples with the largest τ values. The removal process continues until $\lambda_p < \lambda_c$.

Sometimes $V(S)$ has some very small eigenvalues, thus accurate numerical computation of $V^{-\frac{1}{2}}(S)$ is hard. In this case, the calculation can be simplified to $\tau_i = \|\sum_{j=1}^p \lambda_j^{-\frac{1}{2}}\mathbf{u}_j\mathbf{u}_j^T(\mathbf{Y}_i - \mu(S))\|_2^2$. This format can avoid the numerical problem of calculating (8) directly.

(3) *Calculate $V(S)$ and $\mathbf{P}$ again.* At step 15, after iterative removal, the algorithm calculates the projection matrix using step 7 in Algorithm 2. This is illustrated in Figure 1(c), in which the blue arrow represents the projection operation.

(4) *Calculate $\hat{\mu}$.* The semi-verified mean estimator is

$$\hat{\mu} = \mathbf{P}\mathbf{X}_0 + (\mathbf{I} - \mathbf{P})\mu(S). \tag{10}$$

As shown in Figure 1(d), both $\mathbf{X}_0$ and $V(S)$ are relatively far from the ground truth μ^*. However, with appropriate projection, the final estimate with (10) is much closer to μ^*.

Theoretical Analysis

This section provides theoretical analysis about the semi-verified mean estimation method in Algorithm 2. From now on, we use the following notations: For two matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \preceq \mathbf{B}$ if $\mathbf{B} - \mathbf{A}$ is positive semidefinite. $a \lesssim b$ if $a \leq Cb$ for some constant C that does not depend on N_A, m, n, d and α. $\mathrm{Cov}[\mathbf{U}]$ denotes the covariance matrix of random vector $\mathbf{U}$. The proofs are shown in the appendix.

Our analysis begins with the following assumption, which requires the boundedness of covariance matrix:

Assumption 1. $V_0 \preceq \sigma^2\mathbf{I}_d$, in which V_0 is the covariance matrix of $\nabla f(\mathbf{w}_t, \mathbf{Z}_{ij})$.

According to (5), we have $V^* \preceq (\sigma^2/n)\mathbf{I}_d$ and $V_A^* \preceq (\sigma^2/N_A)\mathbf{I}_d$.

Upper Bound

Theorem 1 provides a bound of the performance of Algorithm 2 under additive contamination model.

Theorem 1. *Under additive contamination model, if Assumption 1 hold, and parameters p, λ_c in Algorithm 2 satisfy*

$$p > \frac{8}{\alpha}, \tag{11}$$

$$\lambda_c > 32\frac{\sigma^2}{n}\left(1 + \frac{2d}{\alpha m}\right), \tag{12}$$

then

$$\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2^2\right] \leq \frac{3\sigma^2 p}{N_A} + \frac{15\lambda_c}{2\alpha} + \delta_m, \tag{13}$$

in which δ_m decays faster than any polynomial of m.

With $m \gtrsim d/\alpha$, from (12), we can let $\lambda_c \sim \sigma^2/n$, then (13) becomes

$$\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2^2\right] \lesssim \sigma^2\left(\frac{p}{N_A} + \frac{1}{\alpha n}\right). \tag{14}$$

We now discuss the selection of parameter p. Since (11) requires $p \gtrsim 1/\alpha$, we discuss two cases. If $N_A \lesssim n$, according to (14), let $p \sim 1/\alpha$, then $\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2^2\right] \lesssim \sigma^2/(\alpha N_A)$. If $N_A \gtrsim n$, according to (14), as long as p satisfies $1/\alpha \lesssim p \lesssim N_A/(\alpha n)$, then $\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2^2\right] \lesssim \sigma^2/(\alpha n)$. With this selection rule of p, (14) becomes

$$\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2\right] \lesssim \sigma\alpha^{-\frac{1}{2}}\left(\frac{1}{\sqrt{N_A}} + \frac{1}{\sqrt{n}}\right). \tag{15}$$

Theorem 2 shows the performance of Algorithm 2 under strong contamination model.

Theorem 2. *Under strong contamination model, if Assumption 1 hold, p satisfies* (11)*, and λ_c satisfies*

$$\lambda_c > \frac{8\sigma^2}{\alpha n}\left(1 + \sqrt{\frac{16d\ln^2 m}{3\alpha m}}\right)^2, \tag{16}$$

then

$$\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2^2\right] \leq \frac{3\sigma^2 p}{N_A} + \frac{15\lambda_c}{2\alpha} + \delta_m, \tag{17}$$

in which δ_m decays faster than any polynomial of m.

Despite that (17) appears to be the same as (13) for additive contamination model, the minimum value of λ_c is different between these two models. With $m/\ln^2 m \gtrsim d/\alpha$, from (16), we let $\lambda_c \sim \sigma^2/(\alpha n)$, then (17) becomes

$$\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2^2\right] \lesssim \sigma^2\left(\frac{p}{N_A} + \frac{1}{\alpha^2 n}\right). \tag{18}$$

If $N_A \lesssim \alpha n$, according to (18), let $p \sim 1/\alpha$. Otherwise, just need to ensure that $1/\alpha \lesssim p \lesssim N_A/(\alpha^2 n)$. Then (18) becomes

$$\mathbb{E}\left[\|\hat{\mu} - \mu^*\|_2\right] \lesssim \sigma\alpha^{-\frac{1}{2}}\left(\frac{1}{\sqrt{N_A}} + \frac{1}{\sqrt{\alpha n}}\right). \tag{19}$$

Traditional methods, such as Zeno (Xie, Koyejo, and Gupta 2019, 2020) has an error of $\sigma\alpha^{-1/2}(\sqrt{d/N_A} + \sqrt{d/n})$. In contrast, our approach—as shown in (15) and (19)—does not suffer from the $\sqrt{d}$ dependence, making it significantly more effective for high-dimensional mean estimation. Depending on whether we are assuming additive or strong contamination, the results are slightly different ($1/\sqrt{n}$ vs $1/\sqrt{\alpha n}$). As discussed earlier, strong contamination model gives the attacker more room for manipulation, since it allows the attacker to pick samples arbitrarily instead of randomly, thus the estimation error is larger.

Remark 1. (Diakonikolas et al. 2021) has proposed an efficient method for list-decodable mean estimation, which generates a list of $O(1/\alpha)$ hypotheses whose minimum distance to μ^* is $O(1/\sqrt{\alpha})$. As is discussed in (Charikar, Steinhardt, and Valiant 2017), list-decodable method can be used in semi-verified mean estimation. Compared with this indirect approach, our new method requires less number of auxiliary clean samples, and the parameter selection is more flexible. Detailed comparisons can be found in the appendix.

Minimax Lower Bound

Now we show the information theoretic lower bound of semi-verified mean estimation problem. Under additive contamination model, the minimax risk is defined as following:

$$R_A(\alpha) = \inf_{\hat{\mu}} \sup_{D\in\mathcal{F}_{\pi_A(\alpha)}} \sup \|\hat{\mu} - \mu^*\|_2, \tag{20}$$

in which $\mathcal{F}$ is the set of all distributions satisfying Assumption 1, as well as (5), which implies that $\mathrm{Cov}[\mathbf{X}_i] \preceq (\sigma^2/n)\mathbf{I}$ and $\mathrm{Cov}[\mathbf{X}_0] \preceq (\sigma^2/N_A)\mathbf{I}$. $\pi_A(\alpha)$ is the policy of attacker, which maps $\mathbf{X}_i$ for $i \in S_0$ to $\mathbf{Y}_i$, $i \in S_0$. In particular, it picks $\lceil \alpha N \rceil$ samples randomly, and let $\mathbf{Y}_i = \mathbf{X}_i$ for these samples. For other samples, $\mathbf{Y}_i$ are arbitrary. $\hat{\mu}$ is the estimator, which is a function of $\mathbf{Y}_1, \ldots, \mathbf{Y}_m, \mathbf{X}_0$.

Similarly, under strong contamination model, the minimax risk is defined as

$$R_S(\alpha) = \inf_{\hat{\mu}} \sup_{D\in\mathcal{F}_{\pi_S(\alpha)}} \sup \|\hat{\mu} - \mu^*\|_2, \tag{21}$$

in which $\pi_S(\alpha)$ is policy of strong contamination. It maps $\mathbf{X}_i$ to $\mathbf{Y}_i$ arbitrarily, as long as $\mathbf{Y}_i = \mathbf{X}_i$ for at least αN samples.

The results are shown in Theorem 3.

Theorem 3. *If* $\lceil N_A/n \rceil \le \ln\frac{d}{4} / \left(4\beta\left(\ln\frac{1}{\beta}+1\right)\right)$, *then with probability at least* $1/2 - \exp[-(\ln 2 - 1/2)m\alpha]$,

$$R_A(\alpha) \ge \frac{\sigma}{2\sqrt{2\alpha}}\left(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{N_A}}\right), \tag{22}$$

$$R_S(\alpha) \ge \frac{\sigma}{2\sqrt{2\alpha}}\left(\frac{1}{\sqrt{\alpha n}} + \frac{1}{\sqrt{N_A}}\right). \tag{23}$$

The upper bound matches the minimax lower bound up to constant factors. Such results indicate that the error rates of Algorithm 2 are optimal. In other words, it is impossible to further improve the bounds in Theorem 1 and 2 in general.

Application in Distributed Learning Under Byzantine Attack

Based on the analysis of semi-verified mean estimation in the previous section, we now analyze the distributed learning method in Algorithm 1, which uses our new semi-verified mean estimation method as the aggregator function. Our analysis is based on the following assumption.

Assumption 2. (a) For all $\mathbf{w} \in \Omega$, $\nabla f(\mathbf{w}, \mathbf{Z})$ is sub-exponential with parameter σ, i.e.

$$\sup_{\mathbf{v}:\|\mathbf{v}\|_2=1} \mathbb{E}\left[e^{\lambda \mathbf{v}^T(\nabla f(\mathbf{w},\mathbf{Z}) - \nabla F(\mathbf{w}))}\right] \le e^{\frac{1}{2}\sigma^2\lambda^2}, \tag{24}$$

for all λ with $|\lambda| \le 1/\sigma$;

(b) $F(\mathbf{w})$ is μ-strong convex and L-smooth in $\mathbf{w}$.

(a) is more restrictive than Assumption 1. (a) requires the distribution of gradient values to be sub-exponential, while the latter only requires bounded eigenvalues of covariance matrix. Such strengthened assumption is made only for theoretical completeness. For (b), despite that our theoretical results are derived under the assumption that F is strong convex, similar to (Yin et al. 2018), our analysis can be easily generalized to the case with non-strong convex and nonconvex functions.

Theorem 4 bounds the final error of $\hat{\mathbf{w}}$ under additive and strong contamination model.

Theorem 4. *Suppose that the following conditions are satisfied: (1) Assumption 2 hold; (2)* $p > 8/\alpha$; *(3)* λ_c *satisfies*

$$\lambda_c > 32\frac{\sigma^2}{n}\left(1 + \frac{2d}{\alpha m}\right) \tag{25}$$

under additive contamination model, or

$$\lambda_c > \frac{8\sigma^2}{\alpha n}\left(1 + \sqrt{\frac{16d\ln^2 m}{3\alpha m}}\right)^2 \tag{26}$$

under strong contamination model; (4) $\eta \le 1/L$; *(5) The size of auxiliary clean dataset satisfies* $N_A \ge 2\ln(mT/\delta)$.

Then with probability at least $1 - \delta - Te^{-\frac{1}{64}\alpha m} - 4Tme^{-\frac{1}{16}\lambda_c m\alpha^2 m^{\frac{1}{3}}\epsilon^2}$,

$$\|\hat{\mathbf{w}} - \mathbf{w}^*\| \le (1-\rho)^T\|\mathbf{w}_0 - \mathbf{w}^*\|_2 + \frac{2\Delta}{\mu}, \tag{27}$$

in which $\hat{\mathbf{w}} = \mathbf{w}_T$ *is the updated weight after* T *iterations, and* $\rho = \eta\mu/2$,

$$\Delta = \sqrt{\frac{6p}{N_A}\sigma^2\ln\frac{pT}{\delta} + \frac{15\lambda_c}{2\alpha}}. \tag{28}$$

If $m \gtrsim d/\alpha$, $p \sim 1/\alpha$, and T is large enough, then under additive contamination model, with $\lambda_c \sim \sigma^2/n$,

$$\|\mathbf{w}_T - \mathbf{w}^*\|_2 = \tilde{O}\left(\frac{\sigma}{\sqrt{\alpha}}\left(\frac{1}{\sqrt{N_A}} + \frac{1}{\sqrt{n}}\right)\right). \tag{29}$$

Under strong contamination model, with $\lambda_c \sim \sigma^2/(\alpha n)$,

$$\|\mathbf{w}_T - \mathbf{w}^*\|_2 = \tilde{O}\left(\frac{\sigma}{\sqrt{\alpha}}\left(\frac{1}{\sqrt{N_A}} + \frac{1}{\sqrt{\alpha n}}\right)\right). \tag{30}$$

The above results show that, as long as the number of worker machines m grows proportionally with d, and N_A grows slightly with d, then the learning error does not increase with dimensionality. Hence, compared with (Cao and Lai 2019; Xie, Koyejo, and Gupta 2019, 2020), our new method removes the $\sqrt{d}$ dependence and is thus more suitable to high dimensional problems.

Numerical Results

This section provides results of numerical simulation. We compare different gradient aggregators:

1. Master only. This means that we only use the gradient values from the auxiliary clean data stored in the master:

$$g_{Master}(\mathbf{w}) = \frac{1}{N_A}\sum_{j=1}^{N_A}\nabla f(\mathbf{w}, \mathbf{Z}_{ij}). \tag{31}$$

This method is used as a baseline, in order to show the benefits of combining clean samples with the untrusted gradient information from worker machines.

2. Distance based filtering. This method comes from (Cao and Lai 2019). Here we just assume that q is known. Among all m gradient vectors from worker machines, this method picks $m-q$ closest one, and then calculate the weighted average:

$$g_{DBF}(\mathbf{w}) = \frac{N_A g_{Master} + n \sum\limits_{i\in\mathcal{N}_{m-q}(g_{Master}(\mathbf{w}))} \mathbf{Y}_i}{N_A + n(m-q)}, \tag{32}$$

in which $\mathcal{N}_{m-q}(g_{Master}(\mathbf{w}))$ means the $m-q$ nearest neighbors of g_{Master} among corrupted gradient vectors from worker machines, $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_m\}$.

3. Zeno. This method was proposed in (Xie, Koyejo, and Gupta 2019). After that, an improved version was provided in (Xie, Koyejo, and Gupta 2020), called Zeno++. Here we use the first order expansion of stochastic descent score mentioned in (Xie, Koyejo, and Gupta 2020). In particular, this method assigns a score

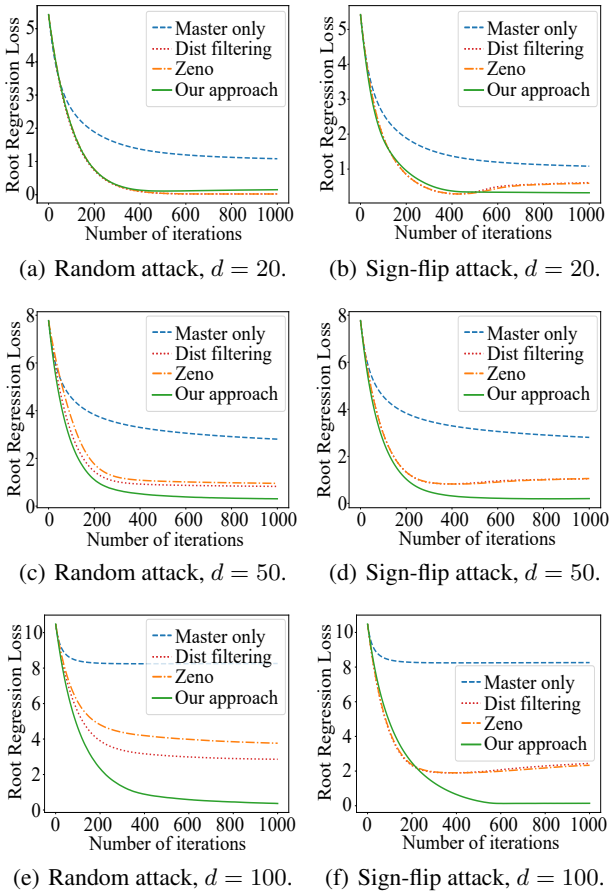
$$Score(\mathbf{Y}_i, g_{Master}) = \gamma\langle g_{Master}, \mathbf{Y}_i\rangle - \rho\|\mathbf{Y}_i\|_2^2, \tag{33}$$

and then use the average gradients from $m-q$ worker machines with highest scores: $g_{Zeno}(\mathbf{w}) = (1/(m-q))\sum_{i=1}^{m-q}\mathbf{Y}_{(i)}$, in which $\mathbf{Y}_{(i)}$ is the gradient vector with i-th highest score according to (33).

4. Our approach. This refers to our new approach in Algorithm 1.

For all these four methods, we implement two types of attack: random attack, in which $\mathbf{Y}_i \sim \mathcal{N}(0, \sigma_{attack}^2\mathbf{I})$, and sign-flip attack, which flips the sign of original gradient vector, such that $\mathbf{Y}_i = -\mathbf{X}_i$.

We would like to remark here that our method relies on auxiliary clean dataset, thus we only compare with previous methods that also use auxiliary samples. Other methods designed for $q < m/2$, such as Krum (Blanchard et al. 2017), geometric median-means (Chen, Su, and Xu 2017), coordinate-wise median or trimmed mean (Yin et al. 2018), and recent high dimensional methods (Zhu et al. 2023), are not shown here due to unfair comparison.



(a) Random attack, $d = 20$. (b) Sign-flip attack, $d = 20$.

(c) Random attack, $d = 50$. (d) Sign-flip attack, $d = 50$.

(e) Random attack, $d = 100$. (f) Sign-flip attack, $d = 100$.

Figure 2: Experiment results with synthesized data under linear model, with $q/m = 0.8$.

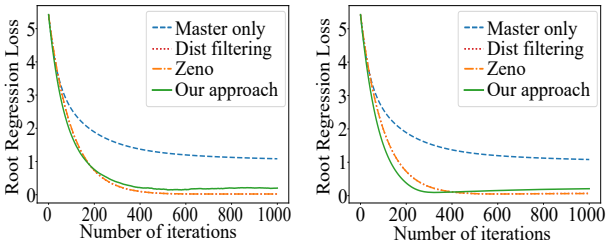
Synthesized Data

We first conduct experiments on distributed linear regression, where the model is

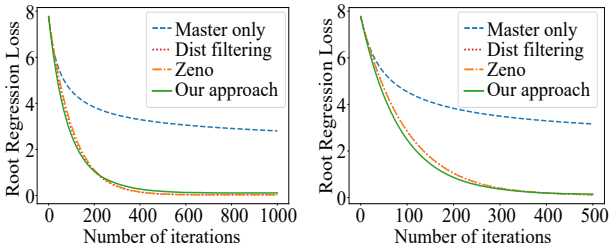
$$V_i = \langle \mathbf{U}_i, \mathbf{w}^*\rangle + W_i, \tag{34}$$

with $\mathbf{U}_i, \mathbf{w}^* \in \mathbb{R}^d$, and noise $W_i \sim \mathcal{N}(0,1)$. The true parameter vector $\mathbf{w}^*$ is randomly generated with entries drawn from $\mathcal{N}(0,2)$. We generate $N = 50,000$ samples. For each sample, all components of $\mathbf{U}_i$ are i.i.d following $\mathcal{N}(0,1)$, and V_i are calculated using (34). These samples are distributed into $m = 500$ worker machines. The learning rate η is 5×10^{-3}. Moreover, there are $N_A = 50$ auxiliary clean samples. The results with $q = 400$ (i.e. $\alpha = 0.2$) and $q = 100$ (i.e. $\alpha = 0.8$) Byzantine machines are shown in Figure 2 and Figure 3, respectively. We set $p = 5$ for $q = 400$, and $p = 2$ for $q = 100$. In both experiments, we show the results with different dimensionality $d = 20, 50, 100$. We run experiments with both random attack and sign-flip attack, in which $\sigma_{attack} = 1$ for the former one.

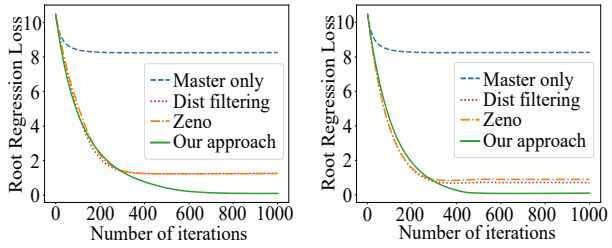
Figure 2 (a) and (b) show that with 80% Byzantine workers, if $d = 20$, then the loss curves of our method (green solid curves) are nearly the same as previous methods, including distance based filtering and Zeno. With $d = 50$,



(a) Random attack, $d = 20$. (b) Sign-flip attack, $d = 20$.



(c) Random attack, $d = 50$. (d) Sign-flip attack, $d = 50$.



(e) Random attack, $d = 100$. (f) Sign-flip attack, $d = 100$.

Figure 3: Experiment results with synthesized data under linear model, with $q/m = 0.2$.

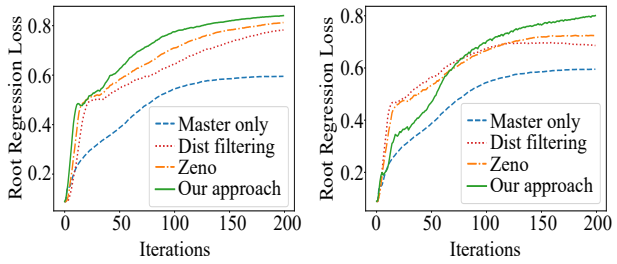
our method begins to outperform existing methods, which is shown in (c) and (d). If the dimensionality is further increased to $d = 100$, then the advantage of our method becomes clearer. This result agrees well with our theoretical analysis.

Now we move on to the case with only 20% worker machines being attacked. The results are shown in Figure 3. The result shows that even in the case with less than half machines being attacked, our method still performs comparable to or better than existing approaches.

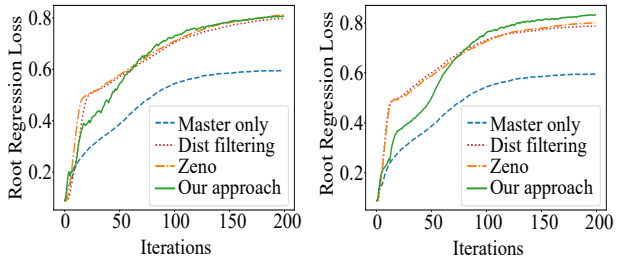
Real Data

Here we use MNIST dataset (LeCun 1998) to test the performance of robust gradient aggregators. In MNIST dataset, there are $60,000$ training images and $12,000$ testing images, with each image has a size of 28×28.

We use a neural network with a single hidden layer of size 32. In each experiment, we first randomly select $N_A = 50$ samples as the auxiliary clean dataset. The remaining samples are distributed into $m = 500$ worker machines evenly. The gradients are obtained by backpropagating cross entropy loss function. The results for $q = 400$ (i.e. $\alpha = 0.2$) and $q = 100$ (i.e. $\alpha = 0.8$) under both random and sign-flip attack are shown in Figure 4, respectively. For random attack, $\sigma_{attack} = 0.01$.



(a) Random attack, $q/m = 0.8$. (b) Sign-flip attack, $q/m = 0.8$.



(c) Random attack, $q/m = 0.2$. (d) Sign-flip attack, $q/m = 0.2$.

Figure 4: Experiment results with MNIST data.

From Figure 4 (a)-(d), our new method outperforms existing approaches in general, for both random attack and sign-flip attack, especially with $q/m = 0.8$. For the case with $q/m = 0.2$, our method is slightly better than others, but the advantage becomes less obvious. This result is natural since our design is primarily for the case with more than half workers being Byzantine. Furthermore, we would like to remark that Zeno appears to perform better than distance-based filtering. Our understanding is that the gradient descent score (33) actually selects gradient vectors whose directions are close to the gradient from auxiliary data. In neural networks with complex loss landscape, this rule may be better than filtering based entirely on distances.

Conclusion

This paper solves the problem of high dimensional distributed learning problem under arbitrary number of Byzantine attackers. Firstly, we have proposed a new method for semi-verified mean estimation, which combines a small clean dataset and a large corrupted dataset to estimate the statistical mean. We have also conducted theoretical analysis under both additive and strong contamination model. The results show that the new method is minimax rate optimal. We have then applied the semi-verified mean estimation method into the aggregator function in distributed learning. Compared with existing methods, the performance of our method is nearly the same as existing approaches for low dimensional problems. With higher dimensionality, our method performs significantly better. Finally, numerical results validate the effectiveness of our new method.

Acknowledgments

This work is supported by the Science Foundation for Youth of the Education Department of Fujian Province, China (Grant JZ250007), the National Natural Science Foundation of China under Grants (62072106), the Open Research Projects of the Key Laboratory of Blockchain Technology and Data Security of the Ministry of Industry and Information Technology for the year 2025 (Grant KT20250015), the Natural Science Foundation of China (General Program, No.62571529), and the Science and Technology Innovation Platform Project of Fujian Province, China (Grant 2023-P-003).

References

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, 2938–2948. PMLR.

Bhagoji, A. N.; Chakraborty, S.; Mittal, P.; and Calo, S. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, 634–643. PMLR.

Blanchard, P.; El Mhamdi, E. M.; Guerraoui, R.; and Stainer, J. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in neural information processing systems*, volume 30.

Cao, X.; Fang, M.; Liu, J.; and Gong, N. Z. 2020. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*.

Cao, X.; and Lai, L. 2019. Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers. *IEEE Transactions on Signal Processing*, 67(22): 5850–5864.

Charikar, M.; Steinhardt, J.; and Valiant, G. 2017. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 47–60.

Chen, Y.; Su, L.; and Xu, J. 2017. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2): 1–25.

Diakonikolas, I.; Kamath, G.; Kane, D. M.; Li, J.; Moitra, A.; and Stewart, A. 2016. Robust Estimators in High Dimensions without the Computational Intractability. In *57th Annual Symposium on Foundations of Computer Science*, 655–664.

Diakonikolas, I.; Kane, D.; Kongsgaard, D.; Li, J.; and Tian, K. 2021. List-decodable mean estimation in nearly-pca time. In *Advances in Neural Information Processing Systems*, volume 34, 10195–10208.

Diakonikolas, I.; and Kane, D. M. 2023. *Algorithmic high-dimensional robust statistics*. Cambridge University Press.

Diakonikolas, I.; Kane, D. M.; and Pensia, A. 2020. Outlier robust mean estimation with subgaussian rates via stability. In *Advances in Neural Information Processing Systems*, 1830–1840.

Fang, M.; Cao, X.; Jia, J.; and Gong, N. 2020. Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX security symposium (USENIX Security 20)*, 1605–1622.

Guerraoui, R.; Rouault, S.; et al. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, 3521–3530. PMLR.

Hampel, F. R. 1971. A general qualitative definition of robustness. *The annals of mathematical statistics*, 42(6): 1887–1896.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2): 1–210.

Lamport, L.; Shostak, R.; and Pease, M. 1982. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3): 382–401.

LeCun, Y. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Li, L.; Fan, Y.; Tse, M.; and Lin, K.-Y. 2020a. A review of applications in federated learning. *Computers & Industrial Engineering*, 149: 106854.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020b. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3): 50–60.

Luo, X.; Wu, Y.; Xiao, X.; and Ooi, B. C. 2021. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 181–192. IEEE.

Lyu, L.; Yu, H.; and Yang, Q. 2020. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Regatti, J.; Chen, H.; and Gupta, A. 2020. ByGARS: Byzantine SGD with arbitrary number of attackers. *arXiv preprint arXiv:2006.13421*.

Shejwalkar, V.; and Houmansadr, A. 2021. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*.

Su, L.; and Xu, J. 2018. Securing distributed machine learning in high dimensions. *arXiv preprint arXiv:1804.10140*, 1536–1233.

Sun, G.; Cong, Y.; Dong, J.; Wang, Q.; Lyu, L.; and Liu, J. 2021. Data poisoning attacks on federated machine learning. *IEEE Internet of Things Journal*, 9(13): 11365–11375.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, 6893–6901. PMLR.

Xie, C.; Koyejo, S.; and Gupta, I. 2020. Zeno++: Robust fully asynchronous SGD. In *International Conference on Machine Learning*, 10495–10503. PMLR.

Yin, D.; Chen, Y.; Kannan, R.; and Bartlett, P. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, 5650–5659. PMLR.

Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; and Gao, Y. 2021. A survey on federated learning. *Knowledge-Based Systems*, 216: 106775.

Zhu, B.; Jiao, J.; and Jordan, M. I. 2022. Robust Estimation for Non-parametric Families via Generative Adversarial Networks. In *2022 IEEE International Symposium on Information Theory (ISIT)*, 1100–1105. IEEE.

Zhu, B.; Wang, L.; Pang, Q.; Wang, S.; Jiao, J.; Song, D.; and Jordan, M. I. 2023. Byzantine-Robust Federated Learning with Optimal Statistical Rates. In *International Conference on Artificial Intelligence and Statistics*, 3151–3178. PMLR.