Byzantine-Resilient Federated Learning at Edge

Yoming Tao, Student Member, IEEE, Sijia Cui, Wenlu Xu, Haofei Yin, Dongxiao Yu, Senior Member, IEEE, Weifa Liang Senior Member, IEEE, and Xiuzhen Cheng, Fellow, IEEE

Abstract—Both Byzantine resilience and communication efficiency have attracted tremendous attention recently for their significance in edge federated learning. However, most existing algorithms may fail when dealing with real-world irregular data that behaves in a heavy-tailed manner. To address this issue, we study the stochastic convex and non-convex optimization problem in the distributed setting at network edge and show how to handle heavy-tailed data while retaining the Byzantine resilience, communication efficiency and the optimal statistical error rates simultaneously. Specifically, we first present a Byzantine-resilient distributed gradient descent algorithm that can handle the heavy-tailed data and converge under the standard assumptions. Then we propose another algorithm that can further reduce the communication overhead in the learning process by adopting the technique of gradient compression. Theoretical analysis shows that our algorithms can achieve the statistical error rate that is order-optimal. Finally, we conduct extensive experiments on both synthetic and real-world datasets to evaluate our proposed algorithms and show the efficacy of our algorithms.

Index Terms—edge intelligent systems, federated learning, Byzantine resilience, communication efficiency

1 INTRODUCTION

Recent years have witnessed the proliferation of smart edge devices, which leads to an unprecedented amount of data generated at the network edge. Thanks to the significant increasing in computation power of edge devices and the ubiquitous deployment of communication infrastructures, the data can be processed locally and aggregated across devices efficiently. With these merits, it is natural to implement large-scale machine learning algorithms at the edge, which brings about the concept of edge intelligence and has empowered many emerging applications that benefit human lives, such as smart city and autonomous driving.

Due to the widespread concerns over data ownership and privacy, federated learning (FL), proposed by Google[1], has emerged as a popular paradigm for distributed ML model training, see, e.g., [2], [3], [4]. In edge FL, the data is retained in the edge and processed in parallel, thus much more real-time results can be provided for real-world applications. However, there remain some practical issues that hinder the successful implementation of edge FL.

One commonly encountered issue in such large-scale distributed systems arises from the potential unreliability of edge devices. Distributing the computation over multiple devices induces a higher risk of failures. In particular, some devices in the system may not follow the predefined protocol and exhibit abnormal behaviors, either actively or passively due to crashes, malfunctioning hardware, unreliable communication channels or attacks from adversaries. The inherently unpredictable behaviors of faulty devices are usually modeled as Byzantine fault [5], [6], [7]. It has been shown in [8] that even when the number of Byzantine devices is small (even only one) and the value sent by them is moderate and even difficult to detect, the performance can still be significantly degraded. Thus, Byzantine resilience has always been a main consideration in the design of FL frameworks, see, e.g., [9], [10], [11].

Communication overhead is also an important consideration in edge FL. Typically, in each update, edge devices need to upload local results (gradients or model parameters) to the server for global aggregation. Due to the inherent bandwidth limitation of the channels, which is especially the case at edge due to the large number of devices, the exchange of data between edge devices and the server will incur heavy communication load and cause the network congestion. Heavy communication overhead hinders the parallelization and scalability in practice, and is the major bottleneck that need addressing for edge FL [12].

In addition, data is a key ingredient in machine learning. Recent studies, e.g., [13], [14], [15], have shown that heavy-tailed noises exist widely in practical multi-sensor systems. Since most data stored in edge devices are collected via various sensors, it is natural for the data used for training learning models to be irregular and behave in a heavy-tailed manner. Furthermore, in many real-world applications, data have been observed to be heavy-tailed in themselves, especially those from biomedicine [16], [17] and finance [18], [19]. In a nutshell, heavy-tailed data can be widespread at edge. Heavy-tailed data could degrade the performance of learning algorithms (see, e.g., [20], [21]), and the presence of Byzantine devices could make things worse for federated training at edge. Unfortunately, existing works on Byzantine resilience in FL all make strong assumptions on the distribution of loss gradients, for example, sub-exponential gradients [22], [23], gradients...
with bounded skewness [8], or gradients with norm-wise bounded variance [24] (see Section 1.3 for more details). Hence, how to mitigate the impact of heavy-tailed data on edge FL is an urgent requirement and still lack investigation.

With the perceptions above, in this paper, we consider Byzantine resilience, communication efficiency and heavy-tailed data robustness simultaneously for the first time. In particular, we have the following natural question:

Is there any way to handle the heavy-tailed data for edge federated learning while retaining the Byzantine resilience, communication efficiency, and the optimal statistical error rates?

In this paper, we provide affirmative answer to the above question. We design an edge FL framework that is robust to heavy-tailed data as well as satisfying the requirement of Byzantine resilience and communication efficiency. Our main contributions and technical challenges are as follows.

1.1 Main Contributions

In the first part, we conduct a comprehensive study on the Byzantine-tolerant distributed gradient descent with heavy-tailed data under the standard assumptions. In particular, for heavy-tailed data, we assume that the distribution of loss gradients has only coordinate-wise bounded second-order raw moment. We establish the high-probability guarantees of statistical error rate for strongly convex, general convex and non-convex population risk functions respectively. Specifically, for all the cases, we show that our algorithm achieves the following statistical error rate:

$$\hat{O}\left(d^2\left[\frac{\alpha^2}{n} + \frac{1}{mn}\right]\right),$$

where $\alpha \in (0, \frac{1}{2})$ is the fraction of Byzantine devices, $n$ is the size of local dataset on each edge device and $m$ is the number of edge devices. The error rate above matches the error rate given in [8] and it has been shown in [8] that, for strongly-convex population risk functions and a fixed $d$, no algorithm can achieve an error lower than $\Omega\left(\frac{\alpha^2}{n} + \frac{1}{mn}\right)$, which implies that our algorithm still achieve order-wise optimality in terms of $(\alpha,n,m)$, even in the presence of heavy-tailed data.

In the second part, we study how to further retain the optimal statistical error rates under the requirement of both Byzantine resilience and communication efficiency. To achieve the communication efficiency, we adopt the technique of gradient compression and consider a generic class of compressors called $\delta$-approximate compressor. Based on this, we propose a communication-efficient and Byzantine resilient distributed gradient descent algorithm with heavy-tailed data. In this case, our statistical error rates become:

$$\hat{O}\left(d^2\left[\frac{\alpha^2}{n} + \frac{1 - \delta}{n} + \frac{1}{mn}\right]\right),$$

where $\delta$ is the compression factor, and when $\delta = 1$ (which implies no compression), the error rate becomes $\hat{O}\left(\frac{\alpha^2}{n} + \frac{1}{mn}\right)$, which means that the compression term has no order-wise contribution to the error rate.

1. Throughout this paper, the notations $\hat{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ hide logarithmic factors.

1.2 Technical Challenges

When only considering the Byzantine resilience, a natural and direct idea to address the problem under the heavy-tailed data setting is to replace the robust aggregator used by the server with some state-of-art robust mean estimators that can deal with the data with coordinate-wise bounded second moment, like [25], [26], [27]. Unfortunately, as far as we know, these robust mean estimators are not appropriate for the edge FL settings due to the following reasons. Firstly, they typically limit the corrupted data to a small fraction, which is usually not the case at edge. Secondly, to analyse the statistical error rate of the learning algorithm, it requires the estimators to have certain continuity property so that the uniform estimation error can be bounded. The reason why the uniform error bound is needed is due to the fact that the analysis of these robust mean estimators relies on the assumption of i.i.d. data. In FL, this means the gradients computed using the training dataset should be i.i.d., which is true for a given model parameter. However, the model parameter is updated iteratively based on the training dataset and thus the parameters across the iterations are highly dependent on each other. As a result, in any iteration $t > 1$, the gradients computed using the training dataset are no longer i.i.d.. Hence, we have to establish uniform concentration to bound the estimation error for all possible parameters simultaneously. However, it is still unclear whether these estimators can achieve the uniform convergence. In this paper, to solve the problem, we let the server and the devices jointly estimate the expected loss gradient in each iteration. At the device end, each device first performs a robust local mean estimator based on soft truncation and noise smoothness, which is motivated by [20]. Then the central machine aggregates the gradient estimates by coordinate-wise trimmed mean to rule out the outliers caused by Byzantine nodes. Based on this, we propose an efficient and more robust distributed gradient descent algorithm. The major challenge in analysis is to bound the uniform error when the local mean estimator is combined with the coordinate-wise trimmed mean. We overcome this by first analysing the point-wise error bound for each coordinate and then using the coordinate-wise continuity of the local mean estimator to obtain the uniform error bound for each coordinate via the covering arguments.

When further considering the gradient compression, we let the server perform a norm-based trimmed mean to aggregate the compressed local estimators. The key challenge here becomes to analyse the uniform error bound for the combination of the compressed local mean estimator and the norm-based trimmed mean. After introducing the gradient compression, the trimming process becomes norm-based and our previous “coordinate-wise” analysis no longer applies. Thus we have to adopt a different analysis. To tackle this problem, we build up on the techniques of [23] to directly bound the uniform error of the local estimator and then consider the impact of the compression.

1.3 Related Work

To cope with Byzantine attacks in distributed learning setting, most solutions rely on the outlier-robust estimators in statistics and use them to replace the
averaging of the updates from different devices, such as coordinate-wise median [8], coordinate-wise mean [8], geometric median [28], [22] and majority voting [29], [30]. Also, there are works developing robust aggregators by combining ideas of these robust estimators. For example, [31] proposed Krum based on the ideas of majority voting and geometric median, and [32] proposed Bulyan that ensures majority agreement on each coordinate of the aggregated gradients by combining Krum and coordinate-wise trimmed mean. A critical issue in these approaches is that their convergence guarantees rely on unrealistic assumptions on gradient distribution, which makes these approaches practically irrelevant. In particular, [24] provided some examples to show that these approaches do not obtain the true optimum especially when dealing with heavy-tailed distributions. To fix it, [24] proposed to utilize worker momentum and just assumed norm-wise bounded variance for loss gradients. Inspired by this, recent work [9] further investigated distributed momentum for Byzantine learning under the norm-wise bounded variance assumption. However, such an assumption implies that gradients are still well-behaved to some extent. According to recent empirical findings, e.g., [33], the gradient noise could be $\alpha$-stable random variable with extremely heavy tails. In this paper, we take the first step for dealing with more extremely heavy-tailed data (or gradients) under the assumption that the loss gradients have only coordinate-wise bounded second raw moment.

To the best of our knowledge, there are quite limited works focusing on Byzantine resilient learning and gradient compression simultaneously, except for a few notable exceptions of [34], [23], [11]. [34] assumed that all devices have access to the same data and their method can only tolerate blind multiplicative adversaries (i.e., adversaries that must determine how to corrupt the gradient before observing the true gradient and can only multiply each coordinate of the true gradient by arbitrary scalar). In contrast, we consider stronger adversaries and a more general setting where different devices have different local datasets. [23] proposed algorithms that combined the robust aggregators and gradient compression together, and introduced error feedback to further reduce the communication costs. However, their convergence guarantees rely on the sub-exponential gradients assumption, which makes their methods not applicable to our setting. Lately, [11] considered the Byzantine resilience and communication efficiency issues together in the heterogeneous data setting. Unfortunately, their results also rely on the norm-wise bounded variance assumption for loss gradients, while we relax this assumption in this work.

### 1.4 Road Map

The remaining part of the paper is organized as follows. The formal problem definition and system model are given in Section 2. In Section 3, we propose a distributed gradient descent algorithm that is robust to both Byzantine fault and heavy-tailed training data. We show that our proposed algorithm can achieve the optimal statistical error rates. In Section 4, we consider how to further reduce the communication overheads, and propose a modified algorithm by introducing the gradient compression schemes. The optimal statistical error rates are shown to be retained. In Section 5, we report the experimental results. Finally, we conclude the paper in Section 6.

## 2 Problem Setup and Preliminaries

### 2.1 Edge Federated Learning Problem

We consider the stochastic convex and non-convex optimization problem. Formally, let $W \subseteq \mathbb{R}^d$ be the parameter space containing all the possible model parameters and $D$ be an unknown distribution over the data universe $Z$. Given a loss function $\ell : W \times Z \rightarrow \mathbb{R}$, where $\ell(w, z)$ measures the risk induced by data $z$ under the model parameter choice $w$, and a dataset $D = \{z_1, z_2, \cdots, z_N\}$, where $z_i$’s are i.i.d. samples from the distribution $D$ over $Z$, the goal is to learn an optimal parameter choice $w^* \in W$ that minimizes the population risk $R_D(w)$, i.e.,

$$w^* \in \arg \min_{w \in W} R_D(w) \triangleq \mathbb{E}_{z \sim D}[\ell(w, z)]^2 \quad (1)$$

Note that since the data distributed $D$ is unknown, the population risk function $R_D(\cdot)$ is typically unknown in practice. Hence, we cannot compute $w^*$ straightforwardly by solving the minimization problem in (1) with gradient descent (GD).

We focus on solving the above stochastic optimization problem over an edge intelligent system via the federated learning framework. The edge intelligent system consists of an edge server and $m$ edge devices. We assume that the total $N$ training data are evenly distributed across the $m$ devices such that each worker machine holds $n = \frac{N}{m}$ data.

We consider a synchronous distributed system where the server can communicate with devices in each round. Among the $m$ devices, at most $\alpha$ ($\alpha < \frac{1}{2}$) fraction of devices are Byzantine and the rest $1 - \alpha$ fraction are normal/good. In each round, the good devices will follow the predefined protocols faithfully. While for Byzantine ones, we have the following assumptions. Firstly, we assume that the set of Byzantine devices can be dynamic throughout the learning process. We denote the Byzantine devices in round $t$ as $\mathcal{B}_t$ and the remaining good devices as $\mathcal{G}_t$. Secondly, we assume the Byzantine devices to be omniscient, i.e., they have complete knowledge of the system and the learning algorithm, and have access to the computations made by the rest good devices. Thirdly, the Byzantine devices need not obey any predefined protocols and can send arbitrary messages to the server (maybe send nothing at all) in each round. Moreover, Byzantine devices can even collude with each other. The only limit on Byzantine devices is that these devices cannot contaminate the local dataset.

1. We assume that $R_D(w) \triangleq \mathbb{E}_{z \sim D}[\ell(w, z)]$ is well-defined for every $w \in W$.
2. Although this is a simplified assumption on data balance over devices, our results can be easily extended to the heterogeneous data sizes setting provided the data sizes are of the same order. The same assumption has been adopted by many related works (e.g. [22], [8], [23]).
2.2 Preliminaries

We first review some key concepts in optimization.

Definition 1 (Lipschitz). A function $f : \mathcal{W} \to \mathbb{R}$ is $L$-Lipschitz if for $\forall w_1, w_2 \in \mathcal{W}$,

$$|f(w_1) - f(w_2)| \leq L\|w_1 - w_2\|_2.$$

Definition 2 (Strong Convexity). A function $f$ is $\alpha$-strongly convex on $\mathcal{W}$ if for $\forall w_1, w_2 \in \mathcal{W}$,

$$f(w_1) \geq f(w_2) + \langle \nabla f(w_2), w_1 - w_2 \rangle + \frac{\alpha}{2}\|w_1 - w_2\|_2^2.$$

Definition 3 (Smoothness). A function $f$ is $\beta$-smooth on $\mathcal{W}$ if for $\forall w_1, w_2 \in \mathcal{W}$,

$$f(w_1) \leq f(w_2) + \langle \nabla f(w_2), w_1 - w_2 \rangle + \frac{\beta}{2}\|w_1 - w_2\|_2^2.$$

Definition 4 (Projection). Given a convex set $\mathcal{W} \subseteq \mathbb{R}^d$, the projection of any $\theta \in \mathbb{R}^d$ to $\mathcal{W}$ is denoted by

$$\prod_{\mathcal{W}} \theta = \arg \min_{w \in \mathcal{W}} \|\theta - w\|.$$

Throughout this paper, we make the following assumptions for the learning.

Assumption 1. The parameter space $\mathcal{W}$ is closed, convex, and bounded with diameter $\Delta$, i.e., for $\forall w_1, w_2 \in \mathcal{W}$, $\|w_1 - w_2\|_2 \leq \Delta$.

Assumption 2. The population risk function $R_P(w)$ is $L_R$-smooth for $\forall w \in \mathcal{W}$, where $L_R$ is a known constant.

Assumption 3. For any given data $z \in \mathcal{Z}$, the loss gradient $\nabla \ell(w, z)$ satisfies that for each coordinate $k \in [d]$, $\nabla_k \ell(w, z)$ is $L_k$-Lipschitz. Let $\hat{L} \triangleq \sqrt{\sum_{k=1}^{d} L_k^2}$.

The above three assumptions are quite standard and has been commonly adopted in the previous works, e.g., [8], [23]. In this paper, we consider edge FL with heavy-tailed training data. Specifically, we assume that for any given parameter $w$, loss gradients have only coordinate-wise bounded second raw moment, which is formally defined as follows.

Assumption 4. For any given $w \in \mathcal{W}$ and each coordinate $k \in [d]$, $\mathbb{E}_{z \sim \mathcal{D}}[\nabla_k^2 \ell(w, x)] \leq \nu$, where $\nu$ is a known constant.

We note that this assumption is reasonable and has been used in some other learning problems with heavy-tailed data as well, e.g., [20], [35], [36], [37]. We provide a concrete example of classical linear regression to validate this.

Example: Consider a linear regression model $y = (w^\top + \xi) x + \xi$, where $x \in \mathbb{R}^d$ is the feature vector and $y \in \mathbb{R}$ is the label, and $\xi$ is the noise. For the noise, we assume that $\xi$ is independent of $x$, and satisfies: 1) $\mathbb{E}[\xi] = 0$; 2) $\mathbb{E}[\xi^2] \leq c_1$ for some constant $c_1$. For the feature vector, we assume that the coordinates of $x = (x_1, \ldots, x_d)$ are independent of each other, and $x$ satisfies: 1) $\mathbb{E}[x] = 0$; 2) $\mathbb{E}[|x|^2] \leq c_2$ for some constant $c_2$; 3) for $\forall k \in [d]$, $\mathbb{E}[x_k^2] \leq c_3$ for some constant $c_3$. We consider the quadratic loss function $\ell(w, x, y) = \frac{1}{2}(y - (w^\top + \xi)x)^T$. Then $\nabla_k \ell(w, x, y) = (y - (w^\top + \xi)x)_k$. By some simple computation, we have for all $w \in \mathcal{W}$ that

$$\mathbb{E}[\nabla_k^2 \ell(w, x, y)] = \mathbb{E}[(y - (w^\top + \xi)x)^2].$$

where the third equality is because $\xi$ is independent of $x$ and $\mathbb{E}[\xi] = 0$, and the second inequality is due to Assumption 1. It can be seen from above that the loss gradient only has coordinate-wise bounded second moment, which indicates that our Assumption 4 is reasonable. Furthermore, the norm-wise variance of the loss gradient in our example is bounded as $\mathbb{E}[\|\nabla \ell(w, x, y)\|^2] \leq d \cdot C$, which is proportional to the dimension $d$. In contrast, the previous bounded variance assumption only assumed it to be a universal constant, which indicates that our Assumption 4 is weaker and thus more general.

3 BYZANTINE-RESILIENT HEAVY-TAILED GRADIENT DESCENT

In this section, we study how to handle the heavy-tailed data while retaining the Byzantine resilience and the statistical error rate. We propose a Byzantine-resilient heavy-tailed gradient descent algorithm called BHGD.

3.1 Algorithm Design

Since $R_D(\cdot)$ is unknown, it is infeasible to apply gradient descent algorithm directly due to the impossibility to compute the exact population risk gradient $\nabla R_D(\cdot)$. An natural alternative way is to estimate $\nabla R_D(\cdot)$ using the data samples $D = \{z_1, z_2, \ldots, z_N\}$. In our algorithm, the estimation is done jointly by the devices and the server. Each device first computes a local estimate of $\nabla R_D(\cdot)$ (for simplicity, we denote the local estimator of device $i \in [m]$ by $g_i(\cdot)$) from its local dataset and sends the local estimation to the server. The server aggregates the received $\{g_1(z_1), g_2(z_2), \ldots, g_m(z_m)\}$ by coordinate-wise trimmed mean and then updates the parameter vector (see Algorithm 1 for details).

The local gradient estimator in our algorithm is inspired by the robust mean estimator for heavy-tailed distribution given in [20]. To be self-contained, we first review the estimator. For simplicity, we consider a one-dimensional random variable $x \sim \mathcal{X}$ and assume that $x_1, x_2, \ldots, x_n$ are i.i.d. samples of $x$. The robust estimator consists of three steps:

1) Scaling and Truncation. For each sample $x_i$, we re-scale it by dividing $s$ and apply a soft truncation function $\phi$ on the re-scaled one. Then we compute the empirical mean of the altered samples and put the mean back to the original scale. That is,

$$\frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \approx \mathbb{E}[x].$$

2) Noise Multiplication. Let $\epsilon_1, \ldots, \epsilon_2$ be independent random noise generated from a common distribution $\epsilon \sim \nu$ with $\mathbb{E}[\epsilon] = 0$. We multiply each sample $x_i$ by
Algorithm 1 Byzantine-Resilient Heavy-tailed Gradient Descent (BHGD)

**Input:** Initial parameter vector $w_0 \in W$, step size $\eta$, time horizon $T$.

**Initialize:** $\zeta \leftarrow \frac{1}{(\Delta n L)^{2(m+1)+d(m)n}}$, $s \leftarrow \sqrt{\frac{m n}{2 \log(1/\zeta)}}$, $\tau \leftarrow \sqrt{2 \log(1/\zeta)}$.

1. for $t \leftarrow 0, 1, \ldots, T-1$ do
   2. Server: Send $w_t$ to all the worker machines
   3. Each good device $i \in G_t$ do in parallel:
      1. Computes local estimate of gradient $g_i(w_t)$.
         Specifically, for $\forall k \in [d]$,
         $$g_{i,k}(w_t) \leftarrow \frac{1}{n} \sum_{j=1}^{n} \left[ g \left( 1 - \frac{g^2}{2 \sigma^2} \right) - \frac{g^3}{3 \sigma^3} \right] + \frac{s}{n} \sum_{j=1}^{n} C \left( \frac{g}{s}, \frac{|g|}{s \sqrt{\tau}} \right),$$
         where $g$ denotes $\nabla_k (w_t, x_i)$.
      2. Sends $g_i(w_t)$ to the central machine
   4. Server:
      1. For each $k \in [d]$:
         a. Sorts $g_{i,k}(w_t)$'s in a non-decreasing order.
         b. Removes the largest and smallest $\beta$ fraction of elements in $\{g_{i,k}(w_t)\}_{i=1}^{m}$ and denotes the indices of the remaining elements as $U_{k,t}$.
         c. Aggregates by
         $$g_k(w_t) = \frac{1}{|U_{k,t}|} \sum_{i \in U_{k,t}} g_{i,k}(w_t)$$
         and denotes $g(w_t) = (g_1(w_t), \ldots, g_d(w_t))$.
      2. Updates the parameter by
         $$w_{t+1} = \prod_{w_t - \eta \cdot g(w_t)}$$

end for

(1 + $\epsilon_t$), and then perform the scaling and truncation step on $x_t(1 + \epsilon_t)$. That is,
$$\tilde{x}(\epsilon) = \frac{s}{n} \sum_{i=1}^{n} \phi(\epsilon x_i + \epsilon_i x_i).$$

3) **Noise Smoothing** We smooth the multiplicative noise via taking the expectation with respect to the noise distribution $\nu$. That is,
$$\tilde{x} = E[\tilde{x}(\epsilon)] = \frac{s}{n} \sum_{i=1}^{n} \int \phi(\epsilon x_i + \epsilon_i x_i) d\nu(\epsilon_i).$$

We note that the final estimator in (2) is random only dependent on the original samples. The explicit form of the integral in (2) is dependent on the choice of the soft truncation function $\phi(\cdot)$ and the noise distribution $\nu$. The results in [38] shows that, if we set $\phi$ to be
$$\phi(x) = \begin{cases} \frac{2x^2}{3}, & x > \sqrt{2} \\ x - \frac{x^3}{3}, & -\sqrt{2} \leq x \leq \sqrt{2} \\ -\frac{2x^2}{3}, & x < -\sqrt{2} \end{cases}$$

and set $\nu = N(0, \frac{1}{2})$, then the integral in (2) has an explicit form such that it can be computed efficiently. Generally, for any $a$ and $b$, we have
$$\mathbb{E}_\nu[\phi(a + b \sqrt{\tau} \epsilon)] = a(1 - \frac{b^2}{2}) - \frac{a^3}{6} + C(a, |b|).$$

The term $C(a, |b|)$ in (4) is a correction form that is extremely simple. To give its explicit form, We first define some preparatory notations:
$$V_- \triangleq \frac{\sqrt{2} - a}{|b|}, \quad V_+ \triangleq \frac{\sqrt{2} + a}{|b|},$$
$$F_- \triangleq \Phi(-V_-), \quad F_+ \triangleq \Phi(-V_+),$$
$$E_- \triangleq \exp(-\frac{V_-^2}{2}), \quad E_+ \triangleq \exp(-\frac{V_+^2}{2}),$$
where $\Phi$ denotes the CDF of the standard Gaussian distribution. Then with these atomic elements, the explicit form of $C(a, |b|)$ can be described as follows:
$$C(a, |b|) = T_1 + T_2 + T_3 + T_4 + T_5,$$
where
$$T_1 \triangleq \frac{2\sqrt{2}}{3} (F_+ - F_-)$$
$$T_2 \triangleq -(a - \frac{a^3}{6})(F_+ + F_-)$$
$$T_3 \triangleq \frac{|b|}{\sqrt{2\pi}} (1 - \frac{a^2}{2})(E_+ - E_-)$$
$$T_4 \triangleq \frac{ab^2}{2} (F_+ + F_- + \frac{1}{\sqrt{2\pi}} (V_+ E_+ + V_- E_-))$$
$$T_5 \triangleq \frac{|b|^3}{6 \sqrt{2\pi}} ((2 + V_-^2) E_- - (2 + V_+^2) E_+).$$

The idea of Algorithm 1 is that, instead of using the empirical mean estimator which may be subject to the heavy-tailed outliers, we let each device apply the one-dimensional robust mean estimator described above to each coordinate of its local loss gradients so that a more accurate local estimator $g_i(\cdot)$ for $\nabla R_D(\cdot)$ can be obtained. Specifically, in our setting, the parameter $a, b$ in (4) should be $\frac{\sqrt{\nu(x, w_T)} - \sqrt{\nu(x, w^n)}}{\sqrt{2\pi}}$ respectively, and the final estimator of device $i \in [m]$ for each coordinate $k \in [d]$ is shown in step 3(1) in Algorithm 1. The server then uses the coordinate-wise trimmed mean to aggregate these local estimators and obtain a global estimator $g(\cdot)$ for $\nabla R_D(\cdot)$ (see step 4(1)). Note that since the trimming threshold $\beta$ is at least $\alpha_t$, the trimming operation ensures that the effect of Byzantine devices can be removed and hence the global estimator $g(\cdot)$ is close to $\nabla R_D(\cdot)$.

### 3.2 Theoretical Results

In this part, we analyse the performance of Algorithm 1. Specifically, we study the statistical error rates for strongly convex, general convex and non-convex population risk function respectively. For strongly-convex and general-convex case, we focus on the excess population risk, i.e., $R_D(w_T) - R_D(w^*)$. For non-convex case, we focus on the rate of convergence to a critical point of the population risk,
Lemma 1. For all $w \in W$, it holds with probability at least $1 - \frac{1}{(mn)^2}$ that

$$
g(w) - \nabla R_D(w) ||_2 \leq \mathcal{O} \left( ad \sqrt{\frac{\log(mn)}{n}} + d \sqrt{\frac{\log(mn)}{mn}} \right).$$

We now provide the results on the statistical error rates for our algorithm.

Strongly convex population risks: We first consider the case where the population risk function $R_D(\cdot)$ is strongly convex. Note that the loss function for each data point is strongly convex.

Theorem 1. Suppose Assumptions 1, 2, 3, and 4 hold, and $R_D(\cdot)$ is $\lambda_R$-strongly convex. Choose step size $\eta = \frac{1}{L_R}$ and run Algorithm 1 for $T$ rounds, then with probability at least $1 - \frac{1}{(mn)^2}$, we have the following bound on excess population risk,

$$
R_D(w_T) - R_D(w^*) \leq L_R(1 - \frac{\lambda_R}{L_R + \lambda_R})^{2T} \| w_0 - w^* \|_2^2 + 4L_R \epsilon^2,
$$

where $\epsilon \in \mathcal{O} \left( ad \sqrt{\frac{\log(mn)}{n}} + d \sqrt{\frac{\log(mn)}{mn}} \right)$.

General convex population risks: For the general convex population risk case, we need a mild technical assumption on the size of the parameter space $W$.

Assumption 5. The parameter space $W$ contains the following $\ell_2$ ball centered at $w^*$:

$$
\{ w \in \mathbb{R}^d : \| w - w^* \|_2 \leq 2 \| w_0 - w^* \|_2 \}.
$$

Then we have the following result on excess population risk function.

Theorem 2. Suppose Assumptions 1, 2, 3, 4 and 5 hold, and $R_D(\cdot)$ is convex. Choose step size $\eta = \frac{1}{L_R}$ and run Algorithm 1 for $T = \frac{L_R}{\lambda_R} \| w_0 - w^* \|_2^2$ rounds, then with probability at least $1 - \frac{1}{(mn)^2}$, we have the following bound on excess population risk,

$$
R_D(w_T) - R_D(w^*) \leq 16 \Delta \epsilon + \frac{1}{2L_R} \epsilon^2,
$$

where $\epsilon \in \mathcal{O} \left( ad \sqrt{\frac{\log(mn)}{n}} + d \sqrt{\frac{\log(mn)}{mn}} \right)$.

Non-convex population risks: For the non-convex population risk case, we need a slightly distinct technical assumption on the size of $W$.

Assumption 6. Suppose that for all $w \in W$, $\| \nabla R_D(w) \|_2 \leq G$. We assume that $W$ contains the $\ell_2$ ball centered at the initial parameter $w_0$:

$$
\{ w \in \mathbb{R}^d : \| w - w_0 \|_2 \leq 2G + \frac{G}{\epsilon^2} || R_D(w_0) - R_D(w^*) ||_2 \}.
$$

We have the following guarantee on the rate of convergence to a critical point of the population risk $R_D(\cdot)$.

Theorem 3. Suppose Assumptions 1, 2, 3, 4 and 6 hold. Choose step size $\eta = \frac{1}{L_R}$ and run Algorithm 1 for $T = \frac{L_R}{\lambda_R} \| R_D(w_0) - R_D(w^*) \|^{\frac{1}{2}}$ rounds, then with probability at least $1 - \frac{1}{(mn)^2}$, we have

$$
\min_{t=0, \ldots, T} || \nabla R_D(w_t) ||_2 \leq \sqrt{2} \epsilon,
$$

where $\epsilon \in \mathcal{O} \left( ad \sqrt{\frac{\log(mn)}{n}} + d \sqrt{\frac{\log(mn)}{mn}} \right)$.

It can be seen from the results above that, for all the cases, our algorithm achieves an error rate of $O \left( d \sqrt{\frac{\log(mn)}{n}} + \frac{1}{mn} \right)$.

The error rate we obtain matches the error rates given in [8]. It has been shown in [8] that, for strongly-convex population risk functions and a fixed $d$, no algorithm can achieve an error lower than $\Omega \left( \frac{a^2}{n} + \frac{1}{mn} \right)$, which implies that our algorithm is still order-wise optimal in terms of $(a,n,m)$, even when considering the heavy-tailed data.

3.3 Analysis of Algorithm 1

Notation: Recall that we denote the Byzantine devices and the good devices in round $t$ as $B_t$ and $G_t$ respectively. Moreover, we use $U_{k,t}$ and $T_{k,t}$ to denote the untrimmed devices and the trimmed devices with respect to coordinate $k$ in round $t$. For notation simplicity, we will drop the subscript $t$ when the context is clear.

3.3.1 Proof of Lemma 1

To prove Lemma 1, we require the following two lemmas related to local estimators $g_i(\cdot)$’s. The following two lemmas show that $g_i,k(w)$ is concentrated around $\nabla_k R_D(w)$ for all good devices $i \in G$, any fixed coordinate $k \in [d]$ and any fixed parameter $w \in W$.

Lemma 2. For any fixed $w \in W$ and coordinate $k \in [d]$, the following holds with probability at least $1 - m \cdot \zeta$,

$$
\max_{i \in G} | g_{i,k}(w) - \nabla_k R_D(w) | \leq \sqrt{\frac{2v \log(1/\zeta)}{n}} + \sqrt{\frac{v}{n}}. 
$$

Lemma 3. For any fixed $w \in W$ and coordinate $k \in [d]$, the following holds with probability at least $1 - \zeta$,

$$
\left| \frac{1}{|G|} \sum_{i \in G} g_{i,k}(w) - \nabla_k R_D(w) \right| \leq \sqrt{\frac{2v \log(1/\zeta)}{(1 - \alpha)mn}} + \sqrt{\frac{v}{(1 - \alpha)mn}}.
$$

Proof of Lemma 2. The one-dimension estimator defined in (2) has the following pointwise accuracy, which is given in [20]:

Lemma 4. [20, Lemma 2] Consider the dataset $\{ x_i \}_{i=1}^n$ where $x_i$ are i.i.d. samples drawn from distribution $\mathcal{X}$. Assume that $\mathcal{X}$ has finite second-order moment and $\mathbb{E}_X[|x|^2] \leq v$. Then with probability at least $1 - \zeta$, the estimator $\hat{x}$ defined in (2) under truncation function defined in (3), noise distribution $\nu = \mathcal{N}(0, \frac{1}{v})$ and scale $s = \sqrt{2 \log(1/\zeta)}$ satisfies

$$
| \hat{x} - \mathbb{E}_X[x] | \leq \sqrt{\frac{2v \log(1/\zeta)}{n}} + \sqrt{\frac{v}{n}}.
$$
According to Lemma 4, we have for any \( i \in G, k \in [d] \) and \( w \in W \) that, with probability at least \( 1 - \zeta \),
\[
|g_{i,k}(w) - \nabla_k R_D(w)| \leq \sqrt{\frac{2 \log(1/\zeta)}{n}} + \frac{\nu}{n}.
\]
Then by taking union bound over all good devices \( i \in G \), we have with probability at least \( 1 - m \cdot \zeta \) that,
\[
\max_{i \in G} |g_{i,k}(w) - \nabla_k R_D(w)| \leq \sqrt{\frac{2 \log(1/\zeta)}{n}} + \frac{\nu}{n},
\]
which concludes the proof.

**Proof of Lemma 3.** By Lemma 4, we have with probability at least \( 1 - \zeta \) that,
\[
\frac{1}{|G|} \sum_{i \in G} g_{i,k}(w) - \nabla_k R_D(w) \leq \sqrt{\frac{2 \log(1/\zeta)}{|G| n}} + \frac{\nu}{|G| n}.
\]
Since \( |G| \geq (1 - \alpha)m \), we obtain with probability at least \( 1 - \zeta \) that
\[
\frac{1}{|G|} \sum_{i \in G} g_{i,k}(w) - \nabla_k R_D(w) \leq \sqrt{\frac{2 \log(1/\zeta)}{(1 - \alpha)mn}} + \frac{\nu}{(1 - \alpha)mn},
\]
which concludes the proof.

With Lemma 2 and Lemma 3, we are now ready to prove Lemma 1.

Due to Lemma 2 and Lemma 3, we already have (5) and (6) hold for any fixed \( w \in W \) and \( k \in [d] \). Next, to extend the pointwise accuracy to the uniform accuracy that holds for all \( w \in W \), we need to utilize the standard covering net argument. Let \( W_\epsilon = \{ w^1, w^2, \ldots, w^{N_\epsilon} \} \) be a finite subset of \( W \) such that for any \( w \in W \), there exists some \( w^\epsilon \in W_\epsilon \) satisfying \( \| w - w^\epsilon \|_2 \leq \epsilon \). According to the basic property of covering numbers for compact subsets of Euclidean space [39], we know that \( N_\epsilon \leq (\frac{2\pi}{\epsilon})^d \). Take the union bound, we have both (5) and (6) hold for any \( k \in [d] \) and all \( w = w^\epsilon \in W_\epsilon \) with probability at least \( 1 - N_\epsilon(m + 1)\zeta \).

Then consider an arbitrary \( w \in W \). Suppose that \( \| w - w^\epsilon \|_2 \leq \epsilon \). Since we assume in Assumption 3 that for any \( k \in [d] \), \( \nabla_k(\ell(w, z)) \) is \( L_k \)-Lipschitz for \( z \), we know that
\[
|\nabla_k R_D(w) - \nabla_k R_D(w^\epsilon)| \leq L_k \epsilon.
\]
According to [20, Lemma 4], the one-dimension estimator defined in (2) satisfies that \( |\hat{x}(X) - \tilde{x}(X')| \leq \frac{C}{n} \| X - X' \|_1 \) where \( X, X' \) denote two datasets and \( c_v \) is a constant that equals \( 1 - 2\Phi(-\sqrt{\tau}) + \frac{2}{\sqrt{\pi}} \exp(-\frac{x^2}{2}) \). For each coordinate \( k \in [d] \) we have
\[
|\hat{g}_{i,k}(w) - \tilde{g}_{i,k}(w)| \leq \frac{c_v}{n} \sum_{j=1}^{n} |\nabla_k \ell(w, z_j) - \nabla_k \ell(w^\epsilon, z_j)| 
\leq \frac{c_v}{n} \cdot n \cdot L_k \| w - w^\epsilon \|_2 \leq c_v L_k \epsilon,
\]
where the second inequality is due to Assumption 3. Based on (7) and (8), we obtain for any \( k \in [d] \) and all \( w \in W \) that, with probability at least \( 1 - N_\epsilon(m + 1)\zeta \),
\[
\max_{i \in G} |g_{i,k}(w) - \nabla_k R_D(w)| \leq \sqrt{\frac{2 \log(1/\zeta)}{n}} + \frac{\nu}{n} + (1 + c_v)L_k \epsilon.
\]
We next move on to \( g(\cdot) \). We have the following for all \( w \in W \) and any coordinate \( k \in [d] \).
\[
|g_k(w) - \nabla_k R_D(w)| = \frac{1}{|U_k|} \sum_{i \in G} |g_{i,k}(w) - \nabla_k R_D(w)| 
\leq \frac{1}{|U_k|} \sum_{i \in G} (g_{i,k}(w) - \nabla_k R_D(w)) 
\leq \frac{1}{|U_k|} \sum_{i \in G} (g_{i,k}(w) - \nabla_k R_D(w)) 
+ \frac{1}{|U_k|} \sum_{i \in G \cap T} (g_{i,k}(w) - \nabla_k R_D(w)) 
+ \frac{1}{|U_k|} \sum_{i \in G \cap T} (g_{i,k}(w) - \nabla_k R_D(w)).
\]
We bound each term in (11) respectively. By (10) we have
\[
\frac{1}{|U_k|} \sum_{i \in G} |g_{i,k}(w) - \nabla_k R_D(w)| 
\leq \frac{|G|}{|U_k|} \frac{1}{|G|} \sum_{i \in G} |g_{i,k}(w) - \nabla_k R_D(w)| 
\leq \frac{1}{1 - 2\beta} \left( \frac{2 \log(1/\zeta)}{(1 - \alpha)mn} + \frac{\nu}{(1 - \alpha)mn} + (1 + c_v)L_k \epsilon \right).
\]
By (9), we have
\[
\frac{1}{|U_k|} \sum_{i \in G \cap T} (g_{i,k}(w) - \nabla_k R_D(w)) 
\leq \frac{2\beta}{1 - 2\beta} \max_{i \in G} |g_{i,k}(w) - \nabla_k R_D(w)| 
\leq \frac{2\beta}{1 - 2\beta} \left( \frac{2 \log(1/\zeta)}{n} + \frac{\nu}{n} + (1 + c_v)L_k \epsilon \right).
\]
Since \( \beta \geq \alpha \), w.l.o.g., we assume that \( G \cap T_k \neq \emptyset \). Then by (9) again, we have
\[
\frac{1}{|U_k|} \sum_{i \in G \cap T} (g_{i,k}(w) - \nabla_k R_D(w)) 
\leq \frac{\alpha}{1 - 2\beta} \left( \frac{2 \log(1/\zeta)}{n} + \frac{\nu}{n} + (1 + c_v)L_k \epsilon \right).
\]
It is worth noting that, all (12), (13) and (14) hold as long as both (9) and (10) hold, which is with probability at least \( 1 - N_\epsilon(m + 1)\zeta \). Hence, With the same probability, we have the following for all \( w \in W \) and any \( k \in [d] \):
\begin{align*}
& \leq \frac{\alpha + 2\beta}{1 - 2\beta} \left( \sqrt{\frac{2v \log(1/\zeta)}{n}} + \sqrt{\frac{v}{n}} + (1 + c_v)Lk\epsilon \right) \\
& \quad + \frac{1 - \alpha}{1 - 2\beta} \left( \sqrt{\frac{2v \log(1/\zeta)}{(1 - \alpha)mn}} + \frac{v}{(1 - \alpha)mn} + (1 + c_v)Lk\epsilon \right)
\end{align*}

Taking the union bound for all \( k \in [d] \) yields

\[ \| g(w) - \nabla R_D(w) \|_2 \leq \sqrt{d} \left( \frac{\alpha + 2\beta}{1 - 2\beta} \sqrt{\frac{2v \log(1/\zeta)}{n}} + \frac{\alpha + 2\beta}{1 - 2\beta} \sqrt{\frac{vd}{n}} \\
\quad + \frac{1 - \alpha}{1 - 2\beta} \sqrt{\frac{2v \log(1/\zeta)}{(1 - \alpha)mn}} + \frac{1 - \alpha}{1 - 2\beta} \sqrt{\frac{vd}{(1 - \alpha)mn}} \\
\quad + \frac{1 + 2\beta}{1 - 2\beta} (1 + c_v)Lk\epsilon \right) \],

which holds for all \( w \in \mathcal{W} \) with probability at least \( 1 - N_s \Delta(n-1)d \). Setting \( \epsilon = \frac{3}{2nL} \) and noting that \( N_s = (\Delta nL)^d \), \( \zeta = \frac{1}{(\Delta nL)^a d(n-1)d} \), we obtain for all \( w \in \mathcal{W} \) that, with probability at least \( 1 - \frac{1}{mn^2} \),

\[ \| g(w) - \nabla R_D(w) \|_2 \leq O \left( \alpha d \sqrt{\frac{\log(mn)}{n}} + d \sqrt{\frac{\log(mn)}{mn}} \right). \]

The proof of Theorem 1, Theorem 2 and Theorem 3 are mainly technical, hence we omit them here and put them in the full version of the paper [40].

\textbf{4 Byzantine-Resilient Heavy-tailed Gradient Descent with Compression}

In this section, we study how to further reduce the communication overhead as well as retaining the Byzantine resilience and the statistical error rate. By modifying the BHGD Algorithm, we propose the Byzantine-resilient heavy-tailed gradient descent algorithm with compression, which is called BHGD-C.

\textbf{4.1 Algorithm Design}

To make the communication efficient, we adopt the technique of gradient compression and propose Algorithm 2. For the compression scheme, we consider a generic class of compressors called \( \delta \)-approximate compressors, just as the recent work [23] did. The formal definition for the compressors is given below.

\textbf{Definition 5 (\( \delta \)-Approximate Compressor).} An operator \( \mathcal{Q}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d \) is said to be an \( \delta \)-approximate compressor on a set \( S \subseteq \mathbb{R}^d \) if \( \forall x \in S \),

\[ \| \mathcal{Q}(x) - x \|_2^2 \leq (1 - \delta) \| x \|_2^2, \]

where \( \delta \in (0, 1] \) is the compression factor.

The compression factor \( \delta \) measures the degree of compression and \( \delta = 1 \) implies \( \mathcal{Q}(x) = x \), which means no compression. There are many compressors satisfying the definition, such as Top-k Sparsification [41], k-PCA [42], Randomized Quantization [43], 1-bit Quantization [29], \( \ell_1 \)-norm Quantization [44], etc.

\textbf{Algorithm 2 Byzantine-Resilient Heavy-tailed Gradient Descent with Compression (BHGD-C)}

\textbf{Input:} Initial parameter vector \( w_0 \in \mathcal{W} \), compressor \( \mathcal{Q}(\cdot) \), step size \( \eta \), time horizon \( T \).

\textbf{Initialize:} \( \zeta = \frac{1}{2 \alpha \Delta \log(mn)^d (mn)^2}, \ s = \frac{\sqrt{n v}}{2 \alpha \log(1/\zeta)}, \ T = \frac{\sqrt{2 \alpha \log(1/\zeta)}}{s} \).

\textbf{for} \( t \leftarrow 0, 1, \ldots, T - 1 \) \textbf{do}

\begin{enumerate}
\item \textbf{Server:} Send \( w_t \) to all the worker machines.
\item Each good device \( i \in G_t \) do in parallel:
\begin{enumerate}
\item Computes local estimate of gradient \( g_i(w_t) \). Specifically, for \( \forall k \in [d] \),
\[ g_i(k)(w_t) = \frac{1}{n} \sum_{j=1}^{n} \left[ g(\frac{1 - g^2}{2s^2}) - g^3 \right] \]
\[ + \frac{2}{n} \sum_{j=1}^{n} C \left( \frac{g}{s}, \sqrt{\frac{|g_s|}{s^{1/2}}} \right) \],
\end{enumerate}
\item Sends \( \mathcal{Q}(g_i(w_t)) \) to the central machine.
\end{enumerate}

\textbf{4: Server:}

\begin{enumerate}
\item Sorts \( \mathcal{Q}(g_i(w_t)) \)’s in a non-decreasing order according to \( \| \mathcal{Q}(g_i(w_t)) \|_2 \).
\item Denotes the indices of the first \( 1 - \beta \) fraction of elements as \( U_t \).
\item Aggregates the gradients through the trimmed mean: \( g(w_t) = \frac{1}{|U_t|} \sum_{i \in U_t} \mathcal{Q}(g_i(w_t)) \).
\item Updates the parameter by \( w_{t+1} = \prod_{w_t} (w_t - \eta \cdot g(w_t)) \).
\end{enumerate}

\textbf{5: end for}

In Algorithm 2, we let each non-Byzantine device compress its estimate for loss gradient by a \( \delta \)-approximate compressor \( \mathcal{Q}(\cdot) \) before sending it to the server (step 3(2)). We note that no restriction is placed on Byzantine devices. Note that, in Algorithm 2, the aggregation rule used by the server is different from that of Algorithm 1. Now the server performs a norm-based trimmed mean (i.e., to trim the gradients according their norm values, see step 4(1)-(4(2))) to aggregate the compressed local estimators. By doing this, the server eliminates only \( \beta \) (\( \beta \geq \alpha \)) fraction of local gradient estimators instead of 2\( \alpha \) as in Algorithm 1. And we believe this will bring a more accurate estimation for the server.

\textbf{4.2 Theoretical Results}

In this part, we analyse the influence of the gradient compression on the learning performance. Throughout the analysis, we need an additional mild assumption on population risk function \( R_D(\cdot) \).

\textbf{Assumption 7.} For all \( w \in \mathcal{W} \), \( \| \nabla R_D(w) \|_2 \leq G \), where \( G \) is a constant.

Note that, while the loss gradients are unbound, it is realistic to assume that the expected gradient (i.e., the population risk gradient) is inside a ball with some radius \( G \).
Before presenting the statistical error rates, we first analyse the uniform accuracy of $g(w)$ for all $w \in \mathcal{W}$.

**Lemma 5.** With Assumption 7, for all $w \in \mathcal{W}$, it holds with probability at least $1 - \frac{1}{mn^2}$ that

$$\|g(w) - \nabla R_D(w)\|_2 \leq O\left(\left(\alpha + \sqrt{1 - \delta}\right)d\sqrt{\frac{\log(mn)}{n}} + d\sqrt{\frac{\log(mn)}{mn}}\right). \tag{17}$$

Next, we provide the main results on the error rates for our algorithm.

**Strongly convex population risks:** Note that the loss function for each data $f(., z)$ need not be strongly convex. We derive the upper bound on the excess population risk as follows.

**Theorem 4.** Suppose Assumption 1, 2, 3, 4 and 7 hold, and $R_D(\cdot)$ is $\lambda_R$-strongly convex. Choose step size $\eta = \frac{1}{\lambda_R}$ and run Algorithm 2 for $T$ rounds, then with probability at least $1 - \frac{1}{mn^2}$, we have the following bound on excess population risk,

$$R_D(w_T) - R_D(w^*) \leq L_R(1 - \frac{\lambda_R}{L_R + \lambda_R})2T\|w_0 - w^*\|^2 + \frac{4L_R \tilde{E}^2}{\lambda_R^2},$$

where $\tilde{E} \in O\left(\left(\alpha + \sqrt{1 - \delta}\right)d\sqrt{\frac{\log(mn)}{n}} + d\sqrt{\frac{\log(mn)}{mn}}\right)$.

**General convex population risks:** We have the following upper bound on excess population risk function.

**Theorem 5.** Suppose Assumption 1, 2, 3, 4 and 5 hold, and $R_D(\cdot)$ is convex. Choose step size $\eta = \frac{1}{L_R}$ and run Algorithm 1 for $T = \frac{L_R}{\eta^2}\|w_0 - w^*\|^2$ rounds, then with probability at least $1 - \frac{1}{mn^2}$, we have the following bound on excess population risk,

$$R_D(w_T) - R_D(w^*) \leq 16\tilde{E} + \frac{1}{2L_R^2}\tilde{E}^2,$$

where $\tilde{E} \in O\left(\left(\alpha + \sqrt{1 - \delta}\right)d\sqrt{\frac{\log(mn)}{n}} + d\sqrt{\frac{\log(mn)}{mn}}\right)$.

**Non-convex population risks:** For the non-convex population risk case, we need a slightly distinct technical assumption on the size of $\mathcal{W}$.

**Assumption 8.** The parameter space $\mathcal{W}$ contains the $\ell_2$-ball centered at $w_0$:

$$\{w \in \mathcal{R}^d : \|w - w_0\|_2 \leq 2\frac{G}{\tilde{E}^2}(R_D(w_0) - R_D(w^*))\}. \tag{18}$$

**Theorem 6.** Suppose Assumption 1, 2, 3, 4 and 7 hold, and $R_D(\cdot)$ is non-convex. Choose step size $\eta = \frac{1}{L_R}$ and run Algorithm 2 for $T = \frac{L_R}{\eta^2}(R_D(w_0) - R_D(w^*))$ rounds, then with probability at least $1 - \frac{1}{mn^2}$, we have

$$\min_{t=0,1,\ldots,T} \|\nabla R_D(w_t)\|^2 \leq \sqrt{2\tilde{E}},$$

where $\tilde{E} \in O\left(\left(\alpha + \sqrt{1 - \delta}\right)d\sqrt{\frac{\log(mn)}{n}} + d\sqrt{\frac{\log(mn)}{mn}}\right)$.

### 4.3 Analysis of Algorithm 2

**Notation:** We denote the Byzantine devices and the good devices in round $t$ as $B_t$ and $G_t$ respectively. Also, we use $\mathcal{U}_t$ and $\mathcal{T}_t$ to denote the untrimmed devices and the trimmed devices in round $t$.

The analysis of algorithm 2 takes Lemma 5 as the core. To prove Lemma 5, we require the following two lemmas, which show that the local estimators $g_i(\cdot)$’s are concentrated around $\nabla R_D(w)$ for all $w \in \mathcal{W}$.

**Lemma 6.** For all $w \in \mathcal{W}$, the following holds with probability at least $1 - m \cdot (\Delta\sqrt{mn})^d d\zeta$,

$$\max_{i \in \mathcal{G}} \|g_i(w) - \nabla R_D(w)\|_2 \leq \frac{3(c_L + L_R)}{2\sqrt{n}} + \sqrt{\frac{2vd log(\zeta^{-1})}{n}} + \sqrt{\frac{v}{mn}} \leq \mathcal{E}_1 \tag{19}$$

**Lemma 7.** For all $w \in \mathcal{W}$, the following holds with probability at least $1 - (\Delta\sqrt{mn})^d d\zeta$,

$$\left\| \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} g_i(w) - g(w) \right\|_2 \leq \frac{3(c_L + L_R)}{2\sqrt{n}} + \sqrt{\frac{2vd log(\zeta^{-1})}{mn}} + \sqrt{\frac{v}{mn}} \leq \mathcal{E}_2 \tag{19}$$

**Proof of Lemma 6.** According to [20, Lemma 5], we know that for any fixed $i \in \mathcal{G}$ and all $w \in \mathcal{W}$, the following holds with probability at least $1 - N_c d\zeta$,

$$\|g_i(w) - \nabla R_D(w)\|_2 \leq (c_L + L_R)\epsilon + \sqrt{\frac{2vd log(\zeta^{-1})}{n}} + \sqrt{\frac{v}{n}},$$

where $N_c \leq (\Delta\sqrt{mn})^d$. By setting $\epsilon = \frac{3}{2\sqrt{n}}$, we obtain that with probability at least $1 - (\Delta\sqrt{mn})^d d\zeta$,

$$\|g_i(w) - \nabla R_D(w)\|_2 \leq \frac{3(c_L + L_R)}{2\sqrt{n}} + \sqrt{\frac{2vd log(\zeta^{-1})}{n}} + \sqrt{\frac{v}{n}}.$$

Take the union bound, we know that with probability at least $1 - m \cdot (\Delta\sqrt{mn})^d d\zeta$, the following holds for all $w \in \mathcal{W},$

$$\max_{i \in \mathcal{G}} \|g_i(w) - \nabla R_D(w)\|_2 \leq \frac{3(c_L + L_R)}{2\sqrt{n}} + \sqrt{\frac{2vd log(\zeta^{-1})}{mn}} + \sqrt{\frac{v}{mn}}.$$

**Proof of Lemma 7.** Following the same argument in the proof of Lemma 6 (except for taking $\epsilon = \frac{3}{2\sqrt{mn}}$), we have that, with probability at least $1 - (\Delta\sqrt{mn})^d d\zeta$, the following holds for all $w \in \mathcal{W},$

$$\left\| \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} g_i(w) - \nabla R_D(w) \right\|_2 \leq \frac{3(c_L + L_R)}{2\sqrt{mn}} + \sqrt{\frac{2vd log(\zeta^{-1})}{mn}} + \sqrt{\frac{v}{mn}}.$$

With Lemma 6 and Lemma 7, we are now ready to prove Lemma 5. For notation simplicity, we denote $\|g(w) - \nabla R_D(w)\|_2$ by $\tilde{E}(w)$. First, we take union bound, then with
Similarly, we bound the second term in (20) as follows.

We control each term in (20) separately. For the first term in (20), we have

\[
\frac{1}{|U_t|} \left| \sum_{i \in G} (Q[g_i(w)] - \nabla R_D(w)) \right|_2 \leq \frac{1}{|U_t|} \left| \sum_{i \in G} (Q[g_i(w)] - g_i(w)) \right|_2 + \frac{1}{|U_t|} \left| \sum_{i \in G} (g_i(w) - \nabla R_D(w)) \right|_2 \\
\leq \frac{1}{|U_t|} \left| \sum_{i \in G} (||Q[g_i(w)] - g_i(w)||_2) + \frac{1 - \alpha}{1 - \beta} \mathcal{E}_2 \right|_2 \\
\leq \frac{\sqrt{1 - \delta}}{|U_t|} \sum_{i \in G} (||Q[g_i(w)] - g_i(w)||_2) + \frac{1 - \alpha}{1 - \beta} \mathcal{E}_2 \\
\leq \frac{\sqrt{1 - \delta}}{|U_t|} \sum_{i \in G} (||Q[g_i(w)] - g_i(w)||_2) + \frac{1 - \alpha}{1 - \beta} \mathcal{E}_2 \\
\leq \frac{\sqrt{1 - \delta}}{|U_t|} (1 - \alpha) G + \frac{\sqrt{1 - \delta} \alpha}{1 - \beta} \mathcal{E}_1 + \frac{1 - \alpha}{1 - \beta} \mathcal{E}_2.
\]

Similarly, we bound the second term in (20).

\[
\frac{1}{|U_t|} \left| \sum_{i \in G} (Q[g_i(w)] - \nabla R_D(w)) \right|_2 \\
\leq \frac{|T|}{|U_t|} \max_{i \in G} (Q[g_i(w)] - \nabla R_D(w))_2 \\
\leq \frac{\beta}{1 - \beta} \max_i (\|Q[g_i(w)] - g_i(w)\|_2 + \|g_i(w) - \nabla R_D(w)\|_2) \\
\leq \frac{\beta}{1 - \beta} \max_i (\|Q[g_i(w)] - g_i(w)\|_2 + \|g_i(w) - \nabla R_D(w)\|_2)
\]

Finally, we work on the third term in (20). Owing to the trimming threshold \( \beta > \alpha \), we have at least one good device machine in the set \( T_t \) for all \( t \in |T| \).

\[
\frac{1}{|U_t|} \left| \sum_{i \in T_t \cap U_t} (Q[g_i(w)] - \nabla R_D(w)) \right|_2 \\
\leq \frac{|T_t|}{|U_t|} \max_{i \in T_t \cap U_t} (Q[g_i(w)] - \nabla R_D(w))_2 \\
\leq \frac{\beta}{1 - \beta} \max_i (\|Q[g_i(w)] - g_i(w)\|_2 + \|g_i(w) - \nabla R_D(w)\|_2) \\
\leq \frac{\beta}{1 - \beta} \max_i (\|Q[g_i(w)] - g_i(w)\|_2 + \|g_i(w) - \nabla R_D(w)\|_2)
\]

Combining (21), (22), (23) and letting \( \mathcal{E} \) be the uniform upper bound on \( \mathcal{E}(w) \) over \( w \in \mathcal{W} \), we obtain that with probability at least \( 1 - 2(\Delta \sqrt{m})^d d \zeta \), (18) and (19) hold simultaneously. Conditioned on this, we next proceed to bound \( \tilde{\mathcal{E}}(w) \).

\[
\tilde{\mathcal{E}} \leq \frac{(1 + \beta) \sqrt{1 - \delta} + 2 \alpha}{1 - \beta} G + \frac{(1 + \beta) \sqrt{1 - \delta} + \alpha + \beta}{1 - \beta} \mathcal{E}_1 + \frac{1 - \alpha}{1 - \beta} \mathcal{E}_2.
\]

Note that \( \zeta = \frac{3}{2(\Delta \sqrt{m})^d (d \zeta)} \), hence Lemma 5 follows. With Lemma 5, we can prove Theorem 4, 5 and 6 by replacing \( \mathcal{E} \) with \( \tilde{\mathcal{E}} \) and then following almost the same arguments as in the proof of Theorem 1, 2 and 3. Thus the details are omitted here.

5 Experiments

In this section, we conduct experiments on both synthetic and real-world data to show the performance of our proposed algorithms.

5.1 Experiment Setup

We will study tasks of linear regression and logistic regression on both synthetic and real-world datasets. The synthetic data is generated as follows. For linear model, we generate each data point \( (x, y) \) by \( y = \langle x, w^* \rangle + \xi \), where \( \xi \in \mathbb{R}^d \) is zero-mean noise sampled from \( \text{LogNormal}(0, 0.55848) \) by default and each coordinate of \( x \in \mathbb{R}^d \) is sampled from \( \text{LogNormal}(0, 0.78) \) by default. For logistic model, we generate each data point \( (x, y) \) by \( y = \text{sign} \langle \text{sigmoid}(z) - \frac{1}{2} \rangle \), where \( \text{sigmoid}(z) = \frac{1}{1 + \exp(-z)} \) and \( z = \langle x, w^* \rangle + \xi \) where \( \xi \in \mathbb{R}^d \) is zero-mean noise sampled from \( \text{LogNormal}(0, 0.55848) \) by default and each coordinate of \( x \in \mathbb{R}^d \) is sampled from \( \text{LogNormal}(0, 3) \) by default. For real-world datasets, we will use the Adult dataset [45].
for a binary classification task and the Boston Housing Price dataset\textsuperscript{4} for a linear regression task. We use these datasets since they are representative and have been used in previous works on machine learning with heavy-tailed data e.g., \cite{32}. In particular, to test our algorithms for stochastic non-convex optimization, we train a fully-connected neural network with one hidden layer to solve the tasks on real-world datasets.

Since the precise evaluation of the population risk, i.e., $R_D(w) - R_D(w^*)$, is impossible, we will use the empirical risk to approximate the population risk in the experiments. Specifically, we measure the performance of different algorithm in terms of test loss, i.e., excess empirical risk on the test dataset. For all the experiments, we run each algorithm for at least 10 times and report the average results over all the repetitions.

5.2 Results and Discussion

5.2.1 Comparison with Baseline Methods

To show the Byzantine-resilience of BHGD (Algorithm 1), we compare the performance of BHGD with previous methods. Specifically, we consider the distributed gradient descent algorithms with averaging rules being empirical mean (E-Mean) \cite{1}, coordinate-wise trimmed mean (CWT-Mean) \cite{8}, coordinate-wise median (CW-Median) \cite{8}, geometric mean (G-Median) \cite{22}, Krum \cite{31}, Bulian \cite{32}, and momentum Krum (M-Krum) \cite{10} respectively. For experiments on synthetic datasets, we set $m = 10$, $n = 100$, $d = 10$, $\alpha = 0.2$ and generate 200 test data for each model. For experiments on real-world datasets, we set $m = 10$, $n = 40$, $\alpha = 0.2$ and $m = 5$, $n = 10$, $\alpha = 0.2$ for Boston dataset and Adult dataset respectively and pick 100 data uniformly at random as the test data for each. The experimental results are shown in Fig. 1. As we can see from Fig. 1, BHGD (Algorithm 1) achieves lower test loss than all the baseline methods on both synthetic and real-world datasets. In addition, the convergence of BHGD is stabler than the other methods, which indicates that BHGD is more robust to heavy-tailed data as well as Byzantine devices.

\textsuperscript{4} http://lib.stat.cmu.edu/datasets/boston

Fig. 1. Comparison with Baseline Methods

Fig. 2. Impact of Byzantine Devices

Fig. 3. Performance under Different Heavy-tailed Label Noises

Fig. 4. Performance under Different Tail Weights of Heavy-Tailed Feature Vectors

Fig. 5. Impact of the System Scale
and Pareto Distribution to generate the label noises. The main difference between these two heavy-tailed distributions is that LogNormal distribution is symmetric while Pareto distribution is one-sided. For Pareto label noise in each task, we set the scale parameter to be 1 and the shape parameter to be 3.26953 such that its second moment is identical to that of LogNormal(0, 0.55848). The experimental results are shown in Fig. 3. It can be seen that, BHGD converges under both types of noises, and performs better on Pareto noises. We think the main reason lies in that Pareto noise is one-sided and thus its influence is smaller.

Next, we vary the tail weight of heavy-tailed feature vector. We adopt the LogNormal distribution LogNormal(µ, σ), where µ is set to be 0 and σ varies in different settings. Note that, a larger σ corresponds to a larger variance. For linear model, we let σ vary in {0.1, 0.3, 0.5, 0.7, 0.9}, while for logistic model, we let σ vary in {1.0, 1.5, 2.0, 2.5, 3.0}. The experimental results are shown in Fig. 4. From these results, we can conclude that more test loss would be incurred when the training data become more heavy-tailed.

5.2.4 Impact of System Scale

We study the impact of system scale on the performance of BHGD. Specifically, we consider the scale of learning devices and the scale of training data samples. In this experiment, we let m (i.e., the number of learning devices) vary in {10, 20, 30, 40} and let N (i.e., the number of training data samples) vary in {1000, 2000, 4000, 6000, 8000}. We run the BHGD algorithm on synthetic dataset for logistic model under each combination of m and N. The results are shown in Fig. 5. In Fig. 5 (a), we set m and present the test loss after 200 training rounds for various N. Since N = mn, when m is fixed, a larger N means a larger size of local data sample n for each device. The results in Fig. 5 (a) shows that when each learning device has more training data, the test loss would get smaller. Intuitively, this means that when each device has more local data, it learns more accurately, which is quite reasonable. In Fig. 5 (b), we set N and present the test loss after 200 training rounds for various m. When N is fixed, a larger m means a smaller size of local data n for each device. Hence the test loss grows slower as m increases in Fig. 5 (b). Note that, these results corroborate our previous theoretical upper bounds on statistical error rates.

5.2.5 Communication Efficiency

We show the communication efficiency of our BHGD-C algorithm (Algorithm 2). In the first part, we study the impact of different gradient compression techniques. To this end, we equip BHGD-C with different compressor, and then compare their performances with BHGD on real-world datasets. Specifically, we adopt ℓ1 quantization [44], Topk sparsification [41] and randomized quantization [46]. In ℓ1 quantization, the compressor Q(·) can be summarized as $Q(x) = \{\frac{\|x\|_1}{\mu}, \text{sign}(x)\}$ for $\forall x \in \mathbb{R}^d$, where $\text{sign}(x)$ is the quantized vector with each coordinate $i \in [d]$ being either $+1$ (for positive $x_i$) or $-1$ (for negative $x_i$) and $\frac{\|x\|_1}{\mu}$ is the scaling factor. In Topk sparsification, for any $x \in \mathbb{R}^d$, the compressor compresses $x$ by retaining the top $k$ largest coordinates of $x$ and sets the others to zero. In randomized sparsification, for any $x \in \mathbb{R}^d$ and each coordinate $i \in [d]$, it is quite reasonable. In Fig. 5 (b), we set N and present the test loss after 200 training rounds for various m. When N is fixed, a larger m means a smaller size of local data n for each device. Hence the test loss grows slower as m increases in Fig. 5 (b). Note that, these results corroborate our previous theoretical upper bounds on statistical error rates.

5.2.5 Communication Efficiency

We show the communication efficiency of our BHGD-C algorithm (Algorithm 2). In the first part, we study the impact of different gradient compression techniques. To this end, we equip BHGD-C with different compressor, and then compare their performances with BHGD on real-world datasets. Specifically, we adopt ℓ1 quantization [44], Topk sparsification [41] and randomized quantization [46]. In ℓ1 quantization, the compressor Q(·) can be summarized as $Q(x) = \{\frac{\|x\|_1}{\mu}, \text{sign}(x)\}$ for $\forall x \in \mathbb{R}^d$, where $\text{sign}(x)$ is the quantized vector with each coordinate $i \in [d]$ being either $+1$ (for positive $x_i$) or $-1$ (for negative $x_i$) and $\frac{\|x\|_1}{\mu}$ is the scaling factor. In Topk sparsification, for any $x \in \mathbb{R}^d$, the compressor compresses $x$ by retaining the top $k$ largest coordinates of $x$ and sets the others to zero. In randomized sparsification, for any $x \in \mathbb{R}^d$ and each coordinate $i \in [d]$,
the compressor set $x_i$ to 1 with probability $p$ and to 0 with probability $1 - p$. The experimental results are shown in Fig. 6. Note that, in Fig. 6, we present how the test loss varies with respect to the total communication costs (in byte) during the training process. It can be seen that, to achieve the same test loss, BHGD-C entails fewer bytes to transmit from devices to the server and thus provide communication efficiency. Next, in the second part, we conduct experiment on a networked hardware prototype system for edge federated learning. Our system, as illustrated in Fig. 7, consists of $m = 18$ Teclast mini PCs serving as learning devices and a Mac Studio serving as the central server. All the machines are interconnected via a Wi-Fi router and the bandwidth is set to 40Mbit/s by default. We implement our BHGD and BHGD-C algorithms on the system, where BHGD-C algorithm use the Top$_k$ sparsification with $k$ being half of the model dimension. We present the average communication latency across all the learning devices and all the training rounds in Fig. 8. Overall, the BHGD-C incurs less latency during the learning process. From Fig. 8 (a), we can see that, with the model dimension increases, the latency saved by BHGD-C also grows. From Fig. 8 (b), we can see that, the bandwidth affects the communication latency a lot and BHGD-C achieves a much lower latency than BHGD even when the bandwidth is limited.

6 Conclusions

In this paper, we studied how to retain Byzantine resilience and communication efficiency when training machine learning model with heavy-tailed data in a distributed manner. Specifically, we first presented an algorithm that is robust against both Byzantine worker nodes and heavy-tailed data. Then by adopting the gradient compression technique, we further proposed a robust learning algorithm with reduced communication overhead. Our theoretical analysis demonstrated that our proposed algorithms achieve optimal error rates for strongly convex population risk case. We also conducted extensive experiments, which yield consistent results with the theoretical analysis. It is worth noting that, our results may not have optimal dependence on the dimension $d$. Some recent work also attempt to incorporate recent breakthroughs in robust high-dimensional aggregators into the distributed learning scenarios, e.g., [47], [48], [49], [50]. How to achieve Byzantine resilience, communication efficiency and heavy-tailed data robustness simultaneously in high dimensions is an interesting and significant problem and we leave it as the future work.

References


Yoiming Tao is currently a graduate student at Shandong University. He received his Bachelor degree (with distinction) in Computer Science from Shandong University in 2021. He is interested in reliable and trustworthy machine learning.

Sijia Cui is currently a Ph.D. student in Computer Science at University of Chinese Academy of Sciences. He obtained his B.E. degree in computer science from Shandong University in 2022. He is interested in reinforcement learning.

Wenlu Xu is currently a Ph.D. student in statistics at University of California, Los Angeles (UCLA). She obtained her B.S. degree in Mathematics and Applied Mathematics from Shandong University in 2022. She is interested in machine learning and computer vision.

Haofei Yin is currently a Master student in computer science at Shandong University. He obtained his B.E. degree in Computer Science from Shandong University in 2022. He is interested in computer network.

Dongxiao Yu received the B.Sc. degree from the School of Mathematics, Shandong University, in 2006, and the Ph.D. degree from the Department of Computer Science, The University of Hong Kong, in 2014. He became an Associate Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. He is currently a Professor with the School of Computer Science and Technology, Shandong University. His research interests include wireless networks, distributed computing, and graph algorithms.

Weilin Liang received the B.Sc. degree in computer science from Wuhan University, China, in 1984 and the M.E. and Ph.D. degrees in computer science from Australian National University, in 1989 and 1998, respectively. He is currently a professor with the Department of Computer Science, City University of Hong Kong. Prior to that, he was a professor with Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing (MEC), network function virtualization (NFV), Internet of Things, software-defined networking (SDN), design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is currently an associate editor for the editorial board of IEEE Transactions on Communications.

Xiuzhen Cheng received her M.S. and Ph.D. degrees in computer science from University of Minnesota, Twin Cities, in 2000 and 2002, respectively. She was a faculty member at the Department of Computer Science, The George Washington University, from 2002-2020. Currently she is a professor of computer science at Shandong University, Qingdao, China. Her research focuses on blockchain computing, security and privacy, and Internet of Things. She is a Fellow of IEEE.