Byzantine-Resilient High-Dimensional SGD with Local Iterations on Heterogeneous Data

Deepesh Data[1] **Suhas Diggavi**[1]

Abstract

We study stochastic gradient descent (SGD) with local iterations in the presence of Byzantine clients, motivated by the federated learning. The clients, instead of communicating with the server in every iteration, maintain their local models, which they update by taking several SGD iterations based on their own datasets and then communicate the net update with the server, thereby achieving communication-efficiency. Furthermore, only a subset of clients communicates with the server at synchronization times. The Byzantine clients may collude and send arbitrary vectors to the server to disrupt the learning process. To combat the adversary, we employ an efficient high-dimensional robust mean estimation algorithm at the server to filter-out corrupt vectors; and to analyze the outlier-filtering procedure, we develop a novel matrix concentration result that may be of independent interest. We provide convergence analyses for both strongly-convex and non-convex smooth objectives in the heterogeneous data setting. We believe that ours is the first Byzantine-resilient local SGD algorithm and analysis with non-trivial guarantees. We corroborate our theoretical results with preliminary experiments for neural network training.

1. Introduction

In the *federated learning* (FL) paradigm (Konecný, 2017; Konecný et al., 2016; McMahan et al., 2017; Mohri et al., 2019), several clients (e.g., mobiles devices, organizations, etc.) collaboratively learn a machine learning model, where the training process is facilitated by the data held by the participating clients (without data centralization) and is coordinated by a central server (e.g., the service provider). Due to its many advantages over the traditional centralized learning (Dean et al., 2012) (e.g., training a machine learning model without collecting the clients' data, which, in addition to reducing the communication load on the network, provides a basic level of privacy to clients' data), FL has emerged as an active area of research recently; see (Kairouz et al., 2019) for a detailed survey. Stochastic gradient descent (SGD) has become a de facto standard in optimization for training machine learning models at such a large scale (Bottou, 2010; Kairouz et al., 2019; McMahan et al., 2017), where clients iteratively communicate the gradient updates with the central server, which aggregates the gradients, updates the learning model, and sends the aggregated gradient back to the clients. The promise of FL comes with its own set of challenges (Kairouz et al., 2019): (i) optimizing with *heterogeneous* data at different clients – the local datasets at clients may be "non-i.i.d.", i.e., can be thought of as being generated from different underlying distributions; (ii) slow and unreliable network connections between server and clients, so communication in every iteration may not be feasible; (iii) availability of only a subset of clients for training at a given time (maybe due to low connectivity, as clients may be in different geographic locations); and (iv) robustness against malicious/Byzantine clients who may send incorrect gradient updates to the server to disrupt the training process. In this paper, we propose and analyze an SGD algorithm that *simultaneously* addresses all these challenges. First we setup the problem, put our work in context with the related work, and then summarize our contributions.

We consider an empirical risk minimization problem, where data is stored at R clients, each having a different dataset (with no probabilistic assumption on data generation); client $r \in [R]$ has dataset $\mathcal{D}_r$. Let $F_r : \mathbb{R}^d \to \mathbb{R}$ denote the local loss function associated with the dataset $\mathcal{D}_r$, which is defined as $F_r(\boldsymbol{x}) \triangleq \mathbb{E}_{i \in_U [n_r]}[F_{r,i}(\boldsymbol{x})]$, where $n_r = |\mathcal{D}_r|$, i is uniformly distributed over $[n_r] \triangleq \{1, 2, \ldots, n_r\}$, and $F_{r,i}(\boldsymbol{x})$ is the loss associated with the i'th data point at client r with respect to (w.r.t.) $\boldsymbol{x}$. Our goal is to solve the following minimization problem:

$$\arg\min_{\boldsymbol{x} \in \mathcal{C}} \left(F(\boldsymbol{x}) \triangleq \frac{1}{R} \sum_{r=1}^{R} \mathbb{E}_{i \in_U [n_r]}[F_{r,i}(\boldsymbol{x})] \right), \qquad (1)$$

where $\mathcal{C} \subseteq \mathbb{R}^d$ denotes the parameter space that is either equal to $\mathbb{R}^d$ or a compact and convex set.

[1]University of California, Los Angeles, USA. Correspondence to: Deepesh Data <deepesh.data@gmail.com>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

In the absence of the above-mentioned FL challenges, we can minimize (1) using distributed *vanilla* SGD, where in any iteration, server broadcasts the current model parameters to all clients, each of them then samples a stochastic gradient from its local dataset and sends it back to the server, who aggregates the received gradients and updates the global model. However, this simple solution does not satisfy the FL challenges, as *every* client communicates with the server (i.e., no sampling of clients) in *every* SGD iteration (i.e., no local iterations), and furthermore, this solution breaks down even with a single malicious client (Blanchard et al., 2017).

Related work. Recent work have proposed variants of the above-described vanilla SGD that address *some* of the FL challenges. The algorithms in (Basu et al., 2019; Haddadpour & Mahdavi, 2019; Haddadpour et al., 2019; Karimireddy et al., 2020; Khaled et al., 2020; Li et al., 2020; Sahu et al., 2020; Yu et al., 2019b) work under different heterogeneity assumptions but do not provide any robustness to malicious clients. On the other hand, (Alistarh et al., 2018; Blanchard et al., 2017; Chen et al., 2017; Data & Diggavi, 2020b; Su & Xu, 2019; Xie et al., 2019b; Yin et al., 2018; 2019) provide robustness, but with no local iterations or sampling of clients; furthermore, they assume homogeneous (either same or i.i.d.) data across all clients. A different line of work (Chen et al., 2018; Data & Diggavi, 2019; 2020a; Data et al., 2019; 2021; Ghosh et al., 2019; Li et al., 2019a; Rajput et al., 2019) provide robustness with heterogeneous data, but without local iterations or sampling of clients: Chen et al. (2018), Rajput et al. (2019), Data et al. (2019; 2021) use coding across datasets, which is hard to implement in FL; Li et al. (2019a) change the objective function and adds a regularizer term to combat the adversary; Ghosh et al. (2019) effectively reduce the heterogeneous problem to a homogeneous problem by clustering, and then learning happens within each cluster having homogeneous data; and Data & Diggavi (2020a) studied SGD with heterogeneous data under the same assumptions as ours, but without local iterations or client sampling. Incorporating local iterations with Byzantine adversaries makes it significantly more challenging as it requires deriving a new matrix concentration bound (see Theorem 2) and different convergence analyses.

Xie et al. (2019a) also analyzed SGD in the FL setting, but the approximation error (even in the Byzantine-free setting) of their solution could be as large as $\mathcal{O}(D^2 + G^2)$, where G is the gradient bound and D is the diameter of the parameter space that contains the optimal parameters $\boldsymbol{x}^*$ and all the local parameters $\boldsymbol{x}_r^t$ ever emerged at any client $r \in [R]$ in any iteration $t \in [T]$; this, in our opinion, makes their bound vacuous. In optimization, one would ideally like to have convergence rates depend on D with a factor that decays with the number of iterations, e.g., with $\frac{1}{T}$ or $\frac{1}{\sqrt{T}}$, as also in Theorem 1. In Section 4, we also empirically demonstrate the poor learning performance of their algorithm.

Our contributions. In this paper, we tackle heterogeneity assuming that the gradient dissimilarity among local datasets is bounded (see (6)), and propose and analyze a Byzantine-resilient SGD algorithm (Algorithm 1) with local iterations and client sampling under the bounded variance assumption for SGD (see (2)). We provide convergence analyses for strongly-convex and non-convex smooth objectives.

For strongly-convex objectives, our algorithm can find approximate optimal parameters exponentially (in $\frac{T}{H}$) fast, and for non-convex objectives, it can reach to an approximate stationary point with a speed of $\frac{1}{T/H}$. See Theorem 1 for convergence results. The approximation error in the optimization solution comprises of two terms, one is because to the stochasticity in gradients (due to SGD) and is equal to zero if we work with full-batch gradients, and the other term arises because of heterogeneity in local datasets. See a detailed discussion in Section 2.2 on the approximation error analysis and the convergence rates, and also for the reason behind obtaining rates that are off by a factor of H when compared to *vanilla* SGD – looking ahead, the reason is working with weak assumptions.

To tackle the malicious behavior of Byzantine clients, we borrow tools from recent advances in high-dimensional robust statistics (Diakonikolas & Kane, 2019; Diakonikolas et al., 2019; Lai et al., 2016; Steinhardt et al., 2018); in particular, we use the polynomial-time outlier-filtering procedure from (Diakonikolas et al., 2019), which was developed for robust mean estimation in high dimensions. In order to use their algorithm (described in Algorithm 2) in our setting that combines Byzantine resilience with local iterations, we develop a novel matrix concentration result (see Theorem 2), which may be of independent interest. As far as we know, this is the first concentration result for stochastic gradients with local iterations on heterogeneous data.

We believe that ours is the first work that combines *local iterations* with *Byzantine-resilience* for SGD and achieves non-trivial results. Not only that, we also analyze our algorithm on *heterogeneous* data and allow *sampling of clients*. Note that the earlier work that provide robustness (without local iterations or sampling of clients) either assume homogeneous data across clients (Alistarh et al., 2018; Blanchard et al., 2017; Chen et al., 2017; Data & Diggavi, 2020b; Su & Xu, 2019; Yin et al., 2018; 2019) or require strong assumptions, such as the bounded gradient assumption on local functions (Xie et al., 2019b); more on this on page 3.

Paper organization. We describe our algorithm and state the convergence results in Section 2. In Section 3, we describe our main technical tool, a new matrix concentration result for analyzing the robust accumulated gradient estimation procedure. We provide empirical evaluation of our method in Section 4. Omitted details/proofs are given in appendices, provided as part of the supplementary material.

2. Problem Setup and Our Results

In this section, we state our assumptions, describe the adversary model and our algorithm, and state our convergence results followed by important remarks about them.

Assumption 1 (Bounded local variances)**.** *The stochastic gradients sampled from any local dataset have uniformly bounded variance over $\mathcal{C}$ for all clients, i.e., there exists a finite σ, such that for all $\boldsymbol{x} \in \mathcal{C}, r \in [R]$, we have*

$$\mathbb{E}_{i \in_U [n_r]} \|\nabla F_{r,i}(\boldsymbol{x}) - \nabla F_r(\boldsymbol{x})\|^2 \leq \sigma^2. \tag{2}$$

It will be helpful to formally define mini-batch stochastic gradients, where instead of computing stochastic gradients based on just one data point, each client samples $b \geq 1$ data points (without replacement) from its local dataset and computes the average of b gradients. For any $\boldsymbol{x} \in \mathbb{R}^d, r \in [R], b \in [n_r]$, consider the following set

$$\mathcal{F}_r^{\otimes b}(\boldsymbol{x}) := \left\{ \frac{1}{b} \sum_{i \in \mathcal{H}_b} \nabla F_{r,i}(\boldsymbol{x}) : \mathcal{H}_b \in \binom{[n_r]}{b} \right\}. \tag{3}$$

Note that $\boldsymbol{g}_r(\boldsymbol{x}) \in_U \mathcal{F}_r^{\otimes b}(\boldsymbol{x})$ is a mini-batch stochastic gradient with batch size b at client r. It is not hard to see the following, which hold for all $\boldsymbol{x} \in \mathcal{C}, r \in [R]$:

$$\mathbb{E}\left[\boldsymbol{g}_r(\boldsymbol{x})\right] = \nabla F_r(\boldsymbol{x}), \tag{4}$$

$$\mathbb{E}\left\|\boldsymbol{g}_r(\boldsymbol{x}) - \nabla F_r(\boldsymbol{x})\right\|^2 \leq \sigma^2/b. \tag{5}$$

Assumption 2 (Bounded gradient dissimilarity)**.** *The difference of the local gradients $\nabla F_r(\boldsymbol{x}), r \in [R]$ and the global gradient $\nabla F(\boldsymbol{x}) = \frac{1}{R}\sum_{r=1}^{R} \nabla F_r(\boldsymbol{x})$ is uniformly bounded over $\mathbb{R}^d$ for all clients, i.e., there exists a finite κ, such that*

$$\|\nabla F_r(\boldsymbol{x}) - \nabla F(\boldsymbol{x})\|^2 \leq \kappa^2, \quad \forall \boldsymbol{x} \in \mathcal{C}, r \in [R]. \tag{6}$$

Assumption 1 has been standard in SGD literature. Assumption 2 has also been used earlier to bound heterogeneity in datasets; see, for example, (Li et al., 2019b; Yu et al., 2019a), which study decentralized SGD with momentum (without adversaries). Note that when clients compute full-batch gradients, we have $\sigma = 0$ in Assumption 1; similarly, when all clients have access to the same dataset as in (Alistarh et al., 2018; Blanchard et al., 2017), we have $\kappa = 0$ in Assumption 2. Note that (6) can be seen as a *deterministic* condition on local datasets, under which we derive our results.

A note on Assumption 2. In the presence of Byzantine adversaries, since we do not know which ϵR clients are corrupt, we have to make some structural assumption on the data that can provide relationships among gradients sampled at different nodes for reliable decoding, and Assumption 2 is a natural way to achieve that. There are many alternatives to establish this relationship, e.g., by assuming homogeneous (same or i.i.d.) data across clients (Alistarh et al., 2018; Blanchard et al., 2017; Chen et al., 2017; Data & Diggavi, 2020b; Su & Xu, 2019; Yin et al., 2018; 2019) or by explicitly introducing redundancy in the system via coding-theoretic solutions (Chen et al., 2018; Data et al., 2021; Rajput et al., 2019); however, these approaches fall short of in the FL setting.

Assuming bounded gradients of local functions (i.e., $\|\nabla F_r(\boldsymbol{x})\| \leq G$ for some finite G) is a common assumption in literature with heterogeneous data; see, for example, (Li et al., 2020; Yu et al., 2019b, without adversaries) and (Xie et al., 2019b, with adversaries). Note that under this assumption, we can trivially bound the heterogeneity among local datasets by $\|\nabla F_r(\boldsymbol{x}) - \nabla F_s(\boldsymbol{x})\| \leq 2G$. So, assuming bounded gradients not only simplifies the analysis but also obscures the effect of heterogeneity on the convergence bounds, which Assumption 2 clearly brings out.[1]

Bounds on σ^2 and κ^2 in the statistical heterogeneous model. Since all our results (matrix concentration and convergence) are given in terms of σ and κ, to show the clear dependence of our results on the dimensionality of the problem, we bound these quantities in the statistical *heterogeneous* data model under different distributional assumptions on local gradients; see Appendix E for more details, where we prove the following: For the SGD variance bound, we show that if local gradients have sub-Gaussian distribution, then $\sigma = \mathcal{O}(\sqrt{d\log(d)})$. For the gradient dissimilarity bound, we show that if either the local gradients have sub-exponential distribution and each worker has at least $n = \Omega(d\log(nd))$ data points or local gradients have sub-Gaussian distribution and $n \in \mathbb{N}$ is arbitrary, then $\kappa \leq \kappa_{\text{mean}} + \mathcal{O}(\sqrt{d\log(nd)/n})$, where κ_{mean} denotes the distance of the expected local gradients from the global gradient. Note that we make distributional assumptions on data generation *only* to derive bounds on σ, κ; otherwise, all our results hold for arbitrary datasets satisfying (5), (6).

Adversary model. Throughout the paper, we assume that ϵ denotes the fraction of the K *communicating* clients that are corrupt, i.e., at most ϵK (out of K) clients that communicate with the server at synchronization indices may be corrupt, where $K \leq R$ is the number of clients chosen at synchronization indices. This translates to, in the *worst case*, having $\frac{\epsilon K}{R}$ fraction (i.e., a total of ϵK) of corrupt nodes in the entire system, as in the worst-case, all the corrupt nodes can be selected in a communication round; however, in practice, due to several constraints, such as the unreliable network connection (for which the adversary has no control over), we cannot expect that the server will select all corrupt nodes in all iterations. The corrupt clients may collude and arbitrarily

[1]See (Khaled et al., 2020) for a detailed discussion on the inappropriateness of making bounded gradient assumption in heterogeneous data settings and how it obscures the effect of heterogeneity on convergence rates (even without robustness).

Algorithm 1 Byzantine-Resilient SGD with Local Iterations

```
1: Initialize. Set $t := 0$, $\boldsymbol{x}_r^0 := \boldsymbol{0}, \forall r \in [R]$, and $\boldsymbol{x} := \boldsymbol{0}$.
   Here, $\boldsymbol{x}$ denotes the global model and $\boldsymbol{x}_r^0$ denotes the
   local model at client $r$ at time 0. Fix a constant step-size
   $\eta$ and a mini-batch size $b$.
2: while $(t \leq T)$ do
3:    Server selects an arbitrary subset $\mathcal{K} \subseteq [R]$ of $|\mathcal{K}| =$
      $K$ clients and sends $\boldsymbol{x}$ to all clients in $\mathcal{K}$.
4:    All clients $r \in \mathcal{K}$ do in parallel:
5:    Set $\boldsymbol{x}_r^t = \boldsymbol{x}$.
6:    while (true) do
7:       Take a mini-batch stochastic gradient $\boldsymbol{g}_r(\boldsymbol{x}_r^t) \in_U$
         $\mathcal{F}^{\otimes b}(\boldsymbol{x}_r^t)$ and update the local model:
            $\boldsymbol{x}_r^{t+1} \leftarrow \boldsymbol{x}_r^t - \eta \boldsymbol{g}_r(\boldsymbol{x}_r^t));\quad t \leftarrow (t+1).$
8:       if $(t \in \mathcal{I}_T)$ then
9:          Let $\widetilde{\boldsymbol{x}}_r^t = \boldsymbol{x}_r^t$, if client $r$ is honest, otherwise can
            be an arbitrary vector in $\mathbb{R}^d$.
10:         Send $\widetilde{\boldsymbol{x}}_r^t$ to the server and break the inner while
            loop.
11:      end if
12:   end while
13:   At Server:
14:   Receive $\{\widetilde{\boldsymbol{x}}_r, r \in \mathcal{K}\}$ from the clients in $\mathcal{K}$.
15:   For every $r \in \mathcal{K}$, let $\widetilde{\boldsymbol{g}}_{r,\text{accu}} := (\widetilde{\boldsymbol{x}}_r - \boldsymbol{x})/\eta$.
16:   Apply the decoding algorithm RAGE (see Algorithm 2) on $\{\widetilde{\boldsymbol{g}}_{r,\text{accu}}, r \in \mathcal{K}\}$. Let
         $\widehat{\boldsymbol{g}}_{\text{accu}} := \text{RAGE}(\widetilde{\boldsymbol{g}}_{r,\text{accu}}, r \in \mathcal{K})$.
17:   Update the global model $\boldsymbol{x} \leftarrow \Pi_{\mathcal{C}}(\boldsymbol{x} - \eta \widehat{\boldsymbol{g}}_{\text{accu}})$, where
      $\Pi_{\mathcal{C}}$ denotes the projection operator onto the set $\mathcal{C}$.
18: end while
```

deviate from their pre-specified programs: at synchronization indices, instead of sending the true stochastic gradients (or local models), corrupt clients may send adversarially chosen vectors to the server.

2.1. Main Results

Let $\mathcal{I}_T = \{t_1, t_2, \ldots, t_k, \ldots\}$, with $t_1 = 0$, denote the set of synchronization indices (where $\max_{i \geq 1} |t_{i+1} - t_i| = H$) when the server *arbitrarily* selects a subset of $K \leq R$ clients (denoted by $\mathcal{K} \subseteq [R]$) and sends the global model (denoted by $\boldsymbol{x}$) to them; each client $r \in \mathcal{K}$ updates its local model $\boldsymbol{x}_r$ by taking SGD steps based on its local dataset until the next synchronization time, when all clients in $\mathcal{K}$ send their local models to the server. Note that some of these clients may be corrupt and may send arbitrary vectors.[2] Server employs a decoding RAGE and update the global model $\boldsymbol{x}$ based on that. We present our Byzantine-resilient SGD algorithm with local iterations in Algorithm 1.

Our convergence results are for both strongly-convex and non-convex smooth objectives, and we state them in the following theorem. Since our main focus in this paper is on combining Byzantine resilience with local iterations, to avoid the technical complications arising due to the projection operator (in line 17), we prove our results assuming that the parameter space $\mathcal{C}$ is equal to $\mathbb{R}^d$. The analysis involving the projection can be done using the techniques in (Yin et al., 2018).

Theorem 1 (Mini-Batch Local Stochastic Gradient Descent). *Let $\mathcal{K}_t$ denote the set of K clients that are active at any given time $t \in [0:T]$ and ϵ denote the fraction of corrupt clients in $\mathcal{K}_t$. For a global objective function $F : \mathbb{R}^d \to \mathbb{R}$, let Algorithm 1 generate a sequence of iterates $\{\boldsymbol{x}_r^t : t \in [0:T], r \in \mathcal{K}_t\}$ when running with a fixed step-size $\eta = \frac{1}{8HL}$. Fix any constant $\epsilon' > 0$. If $\epsilon \leq \frac{1}{3} - \epsilon'$, then with probability $1 - \frac{T}{H}\exp(-\frac{\epsilon'^2(1-\epsilon)K}{16})$, the sequence of average iterates $\{\boldsymbol{x}^t = \frac{1}{K}\sum_{r \in \mathcal{K}_t} \boldsymbol{x}_r^t : t \in [0:T]\}$ satisfy the following convergence guarantees:*

- **Strongly-convex:** *If F is L-smooth for $L \geq 0$,[3] and μ-strongly convex for $\mu > 0$,[4] we get:*

$$\mathbb{E}\left\|\boldsymbol{x}^T - \boldsymbol{x}^*\right\|^2 \leq \left(1 - \frac{\mu}{16HL}\right)^T \left\|\boldsymbol{x}^0 - \boldsymbol{x}^*\right\|^2 + \frac{13}{\mu^2}\Gamma.$$

- **Non-convex:** *If F is L-smooth for $L \geq 0$, we get:*

$$\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\left\|\nabla F(\boldsymbol{x}^t)\right\|^2 \leq \frac{\left[\mathbb{E}[F(\boldsymbol{x}^0)] - \mathbb{E}[F(\boldsymbol{x}^*)]\right]}{T/16HL} + \frac{9}{2}\Gamma.$$

In both the bounds above, $\Gamma = \left(\frac{3\Upsilon^2}{H} + \frac{11H\sigma^2}{b} + 36H\kappa^2\right)$ with $\Upsilon^2 = \mathcal{O}\left(\sigma_0^2(\epsilon + \epsilon')\right)$, where $\sigma_0^2 = \frac{25H^2\sigma^2}{b\epsilon'}\left(1 + \frac{3d}{2K}\right) + 28H^2\kappa^2$, and expectation is taken over the sampling of mini-batch stochastic gradients.

We prove the strongly-convex part of Theorem 1 in Appendix B and the non-convex part in Appendix C. In addition to other complications arising due to handling Byzantine clients together with local iterations, our proof deviates from the standard proofs for local SGD: We need to show two recurrences, which arise because at synchronization indices, server performs decoding to filter-out the corrupt clients, while at other indices there is no decoding, as there is no communication. The proof of the first recurrence is significantly more involved than that of the other one.

[2]Note that the only disruption that the corrupt clients can cause in the training process is during the gradient aggregation at synchronization indices by sending adversarially chosen vectors to the server, and we give unlimited power to the adversary for that. Because of this and for the purpose of analysis, we can assume, without loss of generality, that in between the synchronization indices, the corrupt clients sample stochastic gradients and update their local parameters honestly.

[3]$F(\boldsymbol{y}) \leq F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$.

[4]$F(\boldsymbol{y}) \geq F(\boldsymbol{x}) + \langle \nabla F(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$.

2.2. Important Remarks About Theorem 1

Failure probability. The failure probability of our algorithm is at most $\frac{T}{H}\exp(-\frac{\epsilon'^2(1-\epsilon)K}{16})$, which though scales linearly with T, also goes down exponentially with K. As a result, in settings such as federated learning, where number of clients could be large (e.g., in tens/hundreds of millions) and server samples tens of thousands of them, we can get a very small probability of error, even if run our algorithm for a long time.[5] Note that the error probability is due to the *stochastic* sampling of gradients, and if we want a "zero" probability of error, we can run full-batch GD (yielding an error of $\varGamma = \mathcal{O}(H\kappa^2)$); we analyze that in Appendix D with a much simplified analysis than that of Theorem 1.

Analysis of the approximation error. In Theorem 1, the approximation error $\varGamma$ essentially consists of two types of error terms: $\varGamma_1 = \mathcal{O}\left(\frac{H\sigma^2}{b\epsilon'}\left(1+\frac{3d}{2K}\right)(\epsilon+\epsilon')\right)$ and $\varGamma_2 = \mathcal{O}(H\kappa^2)$, where $\varGamma_1$ arises due to stochastic sampling of gradients and $\varGamma_2$ arises due to dissimilarity in the local datasets. Observe that $\varGamma_1$ decreases as we increase the batch size b of stochastic gradients and becomes zero if we take full-batch gradients (which implies $\sigma = 0$), as is the case in Theorem 4 in Appendix D. Note that even though the variance (and gradient dissimilarity) of accumulation of H gradients blows up by a factor of H^2, still both $\varGamma_1$ and $\varGamma_2$ have a *linear* dependence on the number of local iterations H. Observe that since we are working with heterogeneous datasets, the presence of gradient dissimilarity bound κ^2 (which captures the heterogeneity) in the approximation error is inevitable, and will always show up when bounding the deviation of the true "global" gradient from the decoded one in the presence of Byzantine clients, even when $H = 1$.

Convergence rates. In the strongly-convex case, Algorithm 1 approximately finds the optimal parameters $\boldsymbol{x}^*$ (within $\varGamma$ error) with $\left(1-\frac{\mu}{16HL}\right)^T$ speed. Note that $\left(1-\frac{\mu}{16HL}\right)^T \leq \exp^{-\frac{\mu}{16L}\frac{T}{H}}$, which implies an exponentially fast (in T/H) convergence rate. In the non-convex case, Algorithm 1 reaches to a stationary point (within $\varGamma$ error) with a speed of $\frac{1}{T/H}$. Note that the convergence rates of *vanilla* SGD (i.e., without local iterations and in Byzantine-free settings) are exponential (in T) and $\frac{1}{T}$ for strongly-convex and non-convex objectives, respectively; whereas, our convergence rates are affected by the number of local iterations H. The reason for this is precisely because we need $\eta \leq \frac{1}{8HL}$ to bound the drift in local parameters across clients; see Lemma 2. Instead, if we had assumed a stronger bounded gradient assumption (which trivially bound the heterogeneity, as explained on page 3), then Lemma 2 would hold for a constant step-size (e.g., $\eta = \frac{1}{2L}$ would suffice), which would lead to vanilla SGD like convergence rates.

[5]As a concrete scenario, say the total number of devices is $R = 10$ million and the server selects $K = 10,000$ of them. Then, even if we want robustness against one million malicious clients, the total probability of failure of our algorithm would still be less than $\frac{T}{H}e^{-30}$, which even if $T = 10^6$ and $H = 1$, would still be less than 10^{-7}. Note that the bound on probability of error in Theorem 1 is a worst-case bound, and in practice, our algorithm succeeds with moderate parameter values; see, for example, Section 4 for our experimental setup and the results.

3. Robust Accumulated Gradient Estimation

In this section, first we discuss the inadequacy of traditional methods (such as coordinate-wise median and trimmed-mean) for filtering corrupt gradients in our setting, and then we motivate and describe the robust accumulated gradient estimation (RAGE) procedure that we use in Algorithm 1 as a subroutine at every synchronization index. Then we prove our new matrix concentration result that is required to establish the performance guarantee of RAGE.

Inadequacy of median and trimmed-mean: Coordinate-wise median (med) and trimmed-mean (trimmean) are the two widely used robust estimation procedures that are easy to describe and implement, and they have been employed earlier for robust gradient aggregation in distributed optimization; see, for example, (Yin et al., 2018; 2019, i.i.d. data setting) and (Xie et al., 2019a, FL setting). Below we argue that these methods give poor performance in FL settings for learning high-dimensional models; we also validate this claim through experiments in Section 4.

- For the simple task of robust mean estimation with inputs coming a unit covariance distribution, med and trimmean have an error that scales with the dimension as $\sqrt{d}$ (Diakonikolas et al., 2019; Lai et al., 2016); when we apply these methods in each SGD iteration, this error translates to a large sub-optimality gap in the convergence rate.
- The adversary may corrupt samples in a way that they preserve the norm of the original uncorrupted samples, but have different adversarially chosen directions (these are called directional attacks); since the performance of these methods are based on the magnitude of the samples, they cannot distinguish between the corrupt and uncorrupt samples.
- When taking coordinate-wise median, for estimating each coordinate, we use only a *single* sample and discard the rest. This is not a good idea in large-scale settings with non-i.i.d. data, such as FL, where there are potentially millions of clients, and if we somehow are able to use samples from *all* (or most of the) honest clients, we could get a significant reduction in variance of stochastic gradients. In med, we do not take advantage of this variance reduction, which leads to a performance degradation, which may be detrimental for performance due to heterogeneity in data. The same reason also applies to the robust gradient aggregation method (KRUM) adopted in (Blanchard et al., 2017), which also uses only one of the input gradients and discards the rest, giving poor performance.

Robust mean estimation: The above limitations of traditional methods motivate us to employ modern tools from high-dimensional robust statistics (Diakonikolas & Kane, 2019; Diakonikolas et al., 2019; Lai et al., 2016). In particular, we use the polynomial-time outlier-filtering procedure for high-dimensional robust mean estimation (RME) from (Diakonikolas et al., 2019) for robust gradient aggregation in Algorithm 1. For clear exposition of the ideas behind their algorithm, we use a version of their algorithm as described in Algorithm 2, which is from (Li, 2019). The crucial observation in these RME algorithms is that if the empirical mean of the samples is far from their true mean, then the empirical covariance matrix has high largest eigenvalue. So, the idea is to iteratively filter out samples that have large projection on the principal eigenvector of the empirical covariance matrix, and keep on doing it until the largest eigenvalue of the empirical covariance matrix becomes sufficiently small (line 7). This is done via a soft-removal method, where we assign weights (confidence score) to the samples and down-weighting those that have large projection (line 10) – in each iteration t, at least one sample (whose projection $\tau_i^{(t)}$ is the maximum) gets 0 weight. In the end, take the weighted average of the surviving samples.[6]

The RME algorithms overcome most of the above-mentioned limitations of traditional methods, except for that their guarantees are not directly applicable to our setting. This is because the error guarantee of RME algorithms are given in terms of concentration of the good samples around their sample mean, which is easy to bound if good samples come from the *same* distribution. Note that our setup significantly deviates from this, where not only the input samples (which are accumulated gradients) come from *different* distributions (as clients have heterogeneous data), but each of them is also a sum of H stochastic gradients (due to local iterations). Since local iterations cause local parameters to *drift* from each other, bounding the concentration of good samples requires bounding this drift.

To this end, we develop a novel matrix concentration inequality that first shows an existence of a large subset of uncorrupted accumulated stochastic gradients and then bounds their concentration around the sample mean; see (7) in Theorem 2 below. As far as we know, this is the first matrix concentration result in an FL setting.

First we setup the notation. Let Algorithm 1 generate a sequence of iterates $\{\boldsymbol{x}_r^t : t \in [0:T], r \in \mathcal{K}_t\}$ when running with a fixed step-size $\eta \leq \frac{1}{8HL}$, where $\mathcal{K}_t$ denotes the set of K clients that are active at time $t \in [0:T]$. Take any two consecutive synchronization indices $t_k, t_{k+1} \in \mathcal{I}_T$. Note that $|t_{k+1} - t_k| \leq H$. For an honest client $r \in \mathcal{K}_{t_k}$, let $\boldsymbol{g}_{r,\text{accu}}^{t_k,t_{k+1}} := \sum_{t=t_k}^{t_{k+1}-1} \boldsymbol{g}_r(\boldsymbol{x}_r^t)$ denote the sum of local mini-batch stochastic gradients sampled by client r between time t_k and t_{k+1}, where $\boldsymbol{g}_r(\boldsymbol{x}_r^t) \in_U \mathcal{F}_r^{\otimes b}(\boldsymbol{x}_r^t)$ satisfies (4), (5). At iteration t_{k+1}, every honest client $r \in \mathcal{K}_{t_k}$ reports its local model $\boldsymbol{x}_r^{t_{k+1}}$ to the server, from which server computes $\boldsymbol{g}_{r,\text{accu}}^{t_k,t_{k+1}}$ (see line 15 of Algorithm 1), whereas, the corrupt clients may report arbitrary and adversarially chosen vectors in $\mathbb{R}^d$. Server does not know the identities of the corrupt clients, and its goal is to produce an estimate $\widehat{\boldsymbol{g}}_{\text{accu}}^{t_k,t_{k+1}}$ of the average accumulated gradients from honest clients.

[6]Note that the outlier-filtering procedure described in Algorithm 2 is intuitive and easy to understand. There are better algorithms that are also more efficient and can achieve better guarantees; see, for example, (Dong et al., 2019). All these algorithms require the same bounded matrix concentration assumption that we show in Theorem 2, thus making them applicable to use as a subroutine in Algorithm 1 without requiring any modification in our analysis.

Algorithm 2 Robust Accumulated Gradient Estimation (RAGE) (Diakonikolas et al., 2019; Li, 2019)

```
1: Input: $K$ vectors $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_K \in \mathbb{R}^d$ such that there is a subset of them $\mathcal{S} \subset [K]$ with $|\mathcal{S}| \geq \frac{2K}{3}$ having bounded covariance $\lambda_{\max}\left(\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}(\boldsymbol{g}_i - \boldsymbol{g}_{\mathcal{S}})(\boldsymbol{g}_i - \boldsymbol{g}_{\mathcal{S}})^T\right) \leq \sigma_0^2$, where $\boldsymbol{g}_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\boldsymbol{g}_i$.
2: For any $\boldsymbol{w} \in [0,1]^K$ with $\|\boldsymbol{w}\|_1 > 0$, define
   $$\boldsymbol{\mu}(\boldsymbol{w}) = \sum_{i=1}^K \frac{w_i}{\|\boldsymbol{w}\|_1}\boldsymbol{g}_i$$
   $$\boldsymbol{\Sigma}(\boldsymbol{w}) = \sum_{i=1}^K \frac{w_i}{\|\boldsymbol{w}\|_1}(\boldsymbol{g}_i - \boldsymbol{\mu}(\boldsymbol{w}))(\boldsymbol{g}_i - \boldsymbol{\mu}(\boldsymbol{w}))^T$$
3: Let $\boldsymbol{w}^{(0)} = [\frac{1}{K}, \ldots, \frac{1}{K}]$ be a length $K$ vector.
4: Let $C \geq 11$ be a universal constant.
5: Let $\boldsymbol{\Sigma}^{(0)} = \boldsymbol{\Sigma}(\boldsymbol{w}^{(0)})$.
6: Let $t = 0$.
7: while $\lambda_{\max}(\boldsymbol{\Sigma}(\boldsymbol{w}^{(t)})) > C\sigma_0^2$ do
8:     Let $\boldsymbol{v}^{(t)}$ be the principal eigenvector of $\boldsymbol{\Sigma}(\boldsymbol{w}^{(t)})$.
9:     For $i \in [K]$, define $\tau_i^{(t)} = \left\langle \boldsymbol{v}^{(t)}, \boldsymbol{g}_i - \boldsymbol{\mu}(\boldsymbol{w}^{(t)})\right\rangle^2$.
10:    For $i \in [K]$, compute $w_i^{(t+1)} = \left(1 - \frac{\tau_i^{(t)}}{\tau_{\max}^{(t)}}\right) w_i^{(t)}$, where $\tau_{\max}^{(t)} = \max_{i: w_i^{(t)} > 0} \tau_i^{(t)}$.
11:    $t = t + 1$
12: end while
13: return $\widehat{\boldsymbol{g}} = \sum_{i=1}^K \frac{w_i^{(t)}}{\|\boldsymbol{w}^{(t)}\|_1}\boldsymbol{g}_i$.
```

Theorem 2 (Matrix concentration)**.** *Suppose an ϵ fraction of K clients that communicate with the server are corrupt. In the setting described above, suppose we are given $K \leq R$ accumulated gradients $\widetilde{\boldsymbol{g}}_{r,\text{accu}}^{t_k,t_{k+1}}, r \in \mathcal{K}_{t_k}$ in $\mathbb{R}^d$, where $\widetilde{\boldsymbol{g}}_{r,\text{accu}}^{t_k,t_{k+1}} = \boldsymbol{g}_{r,\text{accu}}^{t_k,t_{k+1}}$ if r'th client is honest, otherwise can be arbitrary. For any $\epsilon' > 0$, if $(\epsilon + \epsilon') \leq \frac{1}{3}$, then with probability $1 - \exp(-\frac{\epsilon'^2(1-\epsilon)K}{16})$, there exists a subset $\mathcal{S} \subseteq$*

$\mathcal{K}_{t_k}$ *of* uncorrupted *gradients of size* $(1-(\epsilon+\epsilon'))K$ *s.t.*

$$\lambda_{\max}\Big(\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\left(\boldsymbol{g}_i-\boldsymbol{g}_{\mathcal{S}}\right)\left(\boldsymbol{g}_i-\boldsymbol{g}_{\mathcal{S}}\right)^T\Big) \le \frac{25H^2\sigma^2}{b\epsilon'}\Big(1+\frac{3d}{2K}\Big)+28H^2\kappa^2, \quad (7)$$

where, for $i\in\mathcal{S}$, $\boldsymbol{g}_i=\boldsymbol{g}_{i,\text{accu}}^{t_k,t_{k+1}}, \boldsymbol{g}_{\mathcal{S}}=\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\boldsymbol{g}_{i,\text{accu}}^{t_k,t_{k+1}}$, *and* $\lambda_{\max}$ *denotes the largest eigenvalue.*

Theorem 2 establishes the concentration results required for the RME algorithm (described in Algorithm 2) that we employ in Algorithm 1. This RME algorithm takes a collection of vectors as input, out of which an unknown large subset (at least a $\frac{2}{3}$-fraction) is promised to be well-concentrated around its sample mean, and outputs an estimate of the sample mean. The formal guarantee is given as follows:

Theorem 3 (Outlier-filtering algorithm (Diakonikolas et al., 2019))**.** *Under the same setting and notation of Theorem 2, we can find an estimate* $\widehat{\boldsymbol{g}}$ *of* $\boldsymbol{g}_{\mathcal{S}}$ *in polynomial-time with probability 1, such that* $\|\widehat{\boldsymbol{g}}-\boldsymbol{g}_{\mathcal{S}}\|\le\mathcal{O}\left(\sigma_0\sqrt{\epsilon+\epsilon'}\right)$, *where* $\sigma_0^2=\frac{25H^2\sigma^2}{b\epsilon'}\left(1+\frac{3d}{2K}\right)+28H^2\kappa^2$.

Note that, instead of the RME algorithm, if we use med or trimmean, we would get an extra multiplicative factor of $\sqrt{d}$ in the upper-bound on $\|\widehat{\boldsymbol{g}}-\boldsymbol{g}_{\mathcal{S}}\|$ above.

3.1. Proof-sketch of Theorem 2 – Matrix Concentration

In order to prove Theorem 2, we use the following result from (Data & Diggavi, 2020a, Lemma 1):

Lemma 1 ((Data & Diggavi, 2020a, Lemma 1))**.** *Suppose there are* m *independent distributions* $p_1,p_2,\ldots,p_m$ *in* $\mathbb{R}^d$ *such that* $\mathbb{E}_{\boldsymbol{y}\sim p_i}[\boldsymbol{y}]=\boldsymbol{\mu}_i, i\in[m]$ *and each* p_i *has a bounded variance in all directions, i.e.,* $\mathbb{E}_{\boldsymbol{y}\sim p_i}[\langle\boldsymbol{y}-\boldsymbol{\mu}_i,\boldsymbol{v}\rangle^2]\le\sigma_{p_i}^2,\forall\boldsymbol{v}\in\mathbb{R}^d,\|\boldsymbol{v}\|=1$. *Take any* $\epsilon'>0$. *Then, given* m *independent samples* $\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_m$, *where* $\boldsymbol{y}_i\sim p_i$, *with probability* $1-\exp(-\epsilon'^2m/16)$, *there is a subset* $\mathcal{S}$ *of* $(1-\epsilon')m$ *points such that* $\lambda_{\max}\left(\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\left(\boldsymbol{y}_i-\boldsymbol{\mu}_i\right)\left(\boldsymbol{y}_i-\boldsymbol{\mu}_i\right)^T\right)\le\frac{4\sigma_{p_{\max}}^2}{\epsilon'}\left(1+\frac{d}{(1-\epsilon')m}\right)$, *where* $\sigma_{p_{\max}}^2=\max_{i\in[m]}\sigma_{p_i}^2$.

Lemma 1 shows that if we have m independent distributions each having bounded variance, and we take one sample from each of them, then there exists a large subset of these samples that has bounded variance as well. The important thing to note here is that the m samples come from *different* distributions, which makes it distinct from existing results, such as (Charikar et al., 2017, Proposition B.1), which shows concentration of i.i.d. samples.

Now we give a proof-sketch of Theorem 2 with the help of Lemma 1. A complete proof is provided in Appendix A.

Let $t_k,t_{k+1}\in\mathcal{I}_T$ be any two consecutive synchronization indices. For $i\in\mathcal{K}_{t_k}$ corresponding to an honest client, let $Y_i^{t_k},Y_i^{t_k+1},\ldots,Y_i^{t_{k+1}-1}$ be a sequence of $(t_{k+1}-t_k)\le H$ (dependent) random variables, where for any $t\in[t_k:t_{k+1}-1]$, the random variable Y_i^t is distributed as

$$Y_i^t\sim\text{Unif}\Big(\mathcal{F}_i^{\otimes b}\big(\boldsymbol{x}_i^t\big(\boldsymbol{x}_i^{t_k},Y_i^{t_k},\ldots,Y_i^{t-1}\big)\big)\Big). \quad (8)$$

Here, Y_i^t corresponds to the mini-batch stochastic gradient sampled from the set $\mathcal{F}_i^{\otimes b}\big(\boldsymbol{x}_i^t\big(\boldsymbol{x}_i^{t_k},Y_i^{t_k},\ldots,Y_i^{t-1}\big)\big)$, which itself depends on the local parameters $\boldsymbol{x}_i^{t_k}$ (which is a deterministic quantity) at the last synchronization index and the past realizations of $Y_i^{t_k},\ldots,Y_i^{t-1}$. This is because the evolution of local parameters $\boldsymbol{x}_i^t$ depends on $\boldsymbol{x}_i^{t_k}$ and the choice of gradients in between time indices t_k and $t-1$. Now define $Y_i:=\sum_{t=t_k}^{t_{k+1}-1}Y_i^t$. Let p_i be the distribution of Y_i, which we will take when using Lemma 1.

It is not hard to show that for any honest client $i\in\mathcal{K}_{t_k}$, we have $\mathbb{E}\|Y_i-\mathbb{E}[Y_i]\|^2\le\frac{H^2\sigma^2}{b}$. It is also easy to see that the hypothesis of Lemma 1 is satisfied with $\boldsymbol{\mu}_i=\mathbb{E}[Y_i],\sigma_{p_i}^2=\frac{H^2\sigma^2}{b}$ for all honest clients $i\in\mathcal{K}_{t_k}$, i.e., we have $\mathbb{E}_{\boldsymbol{y}_i\sim p_i}[\langle\boldsymbol{y}_i-\mathbb{E}[\boldsymbol{y}_i],\boldsymbol{v}\rangle^2]\le\frac{H^2\sigma^2}{b},\forall\boldsymbol{v}\in\mathbb{R}^d,\|\boldsymbol{v}\|=1$.

We are given K different accumulated gradients (each is a summation of H gradients), out of which at least $(1-\epsilon)K$ are according to the correct distribution. By considering only the uncorrupted gradients (i.e., taking $m=(1-\epsilon)K$), we have from Lemma 1 that there exists a subset $\mathcal{S}\subseteq\mathcal{K}_{t_k}$ of size $(1-\epsilon')(1-\epsilon)K\ge(1-(\epsilon+\epsilon'))K\ge\frac{2K}{3}$ that satisfies (in the following, $\widetilde{\boldsymbol{y}}_i=\boldsymbol{y}_i-\mathbb{E}[\boldsymbol{y}_i]$)

$$\lambda_{\max}\Big(\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\widetilde{\boldsymbol{y}}_i\widetilde{\boldsymbol{y}}_i^T\Big)\le\widehat{\sigma}_0^2:=\frac{4H^2\sigma^2}{b\epsilon'}\Big(1+\frac{3d}{2K}\Big). \quad (9)$$

Note that (9) bounds the deviation of the points in $\mathcal{S}$ from their respective means $\mathbb{E}[\boldsymbol{y}_i]$. However, in (7), we need to bound the deviation of the points in $\mathcal{S}$ from their sample mean $\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\boldsymbol{y}_i$. As it turns out, due to heterogeneity in data and our use of local iterations, this extension is non-trivial and requires some technical work, given next.

From the alternate definition of the largest eigenvalue of symmetric matrices $\mathbf{A}\in\mathbb{R}^{d\times d}$, we have $\lambda_{\max}(\mathbf{A})=\sup_{\boldsymbol{v}\in\mathbb{R}^d,\|\boldsymbol{v}\|=1}\boldsymbol{v}^T\mathbf{A}\boldsymbol{v}$. With this, (9) is equivalent to

$$\sup_{\boldsymbol{v}\in\mathbb{R}^d:\|\boldsymbol{v}\|=1}\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\langle\boldsymbol{y}_i-\mathbb{E}[\boldsymbol{y}_i],\boldsymbol{v}\rangle^2\le\widehat{\sigma}_0^2. \quad (10)$$

Define $\boldsymbol{y}_{\mathcal{S}}:=\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\boldsymbol{y}_i$ to be the sample mean of points in $\mathcal{S}$. Take an arbitrary unit vector $\boldsymbol{v}\in\mathbb{R}^d$. Using some algebraic manipulations provided in Appendix A, we get

$$\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\langle\boldsymbol{y}_i-\boldsymbol{y}_{\mathcal{S}},\boldsymbol{v}\rangle^2\le6\widehat{\sigma}_0^2+ \frac{4}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}\left\|\mathbb{E}[\boldsymbol{y}_j]-\mathbb{E}[\boldsymbol{y}_i]\right\|^2 \quad (11)$$

Using the gradient dissimilarity bound and the L-smoothness of F, we can show that for honest clients $r, s \in \mathcal{K}_{t_k}$, we have $\|\mathbb{E}[\boldsymbol{y}_r] - \mathbb{E}[\boldsymbol{y}_s]\|^2 \leq H\sum_{t=t_k}^{t_{k+1}-1}\left(6\kappa^2 + 3L^2\mathbb{E}\|\boldsymbol{x}_r^t - \boldsymbol{x}_s^t\|^2\right)$. Using this bound in (11) together with some algebraic manipulations, we get

$$\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\langle \boldsymbol{y}_i - \boldsymbol{y}_{\mathcal{S}}, \boldsymbol{v}\rangle^2 \leq 6\widehat{\sigma}_0^2 + 24H^2\kappa^2 + \frac{12HL^2}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}\|\boldsymbol{x}_r^t - \boldsymbol{x}_s^t\|^2 \quad (12)$$

Now we bound the last term of (12), which is the drift in local parameters at different clients in between any two synchronization indices.

Lemma 2. *If* $\eta \leq \frac{1}{8HL}$, *we have* $\sum_{t=t_k}^{t_{k+1}-1}\mathbb{E}\left\|\boldsymbol{x}_r^t - \boldsymbol{x}_s^t\right\|^2 \leq 7H^3\eta^2\left(\frac{\sigma^2}{b} + 3\kappa^2\right)$.

Substituting this in (12) together with some algebraic manipulations provided in Appendix A, we get

$$\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\langle \boldsymbol{y}_i - \boldsymbol{y}_{\mathcal{S}}, \boldsymbol{v}\rangle^2 \leq \frac{25H^2\sigma^2}{b\epsilon'}\left(1 + \frac{3d}{2K}\right) + 28H^2\kappa^2.$$

Note that this bound holds for all unit vectors $\boldsymbol{v} \in \mathbb{R}^d$. Now substituting $\boldsymbol{g}_{i,\text{accu}}^{t_k,t_{k+1}} = \boldsymbol{y}_i, \boldsymbol{g}_{\mathcal{S},\text{accu}}^{t_k,t_{k+1}} = \boldsymbol{y}_{\mathcal{S}}$ and using the alternate definition of largest eigenvalue proves Theorem 2.

4. Experiments

In this section, we present preliminary numerical results on a non-convex objective. Additional implementation details can be found in Appendix F in the supplementary material.

Setup: We train a single layer neural network for image classification on the MNIST handwritten digit (from 0-9) dataset. The hidden layer has 25 nodes with ReLU activation function and the output has softmax function. The dimension of the model parameter vector is $19,885$.[7] All clients compute stochastic gradients on a batch-size of 128 in each iteration and communicate the local parameter vectors with the server after taking $H = 7$ local iterations. For all the defense mechanisms, we start with a step-size $\eta = 0.08$ and decrease its learning rate by a factor of 0.96 when the difference in the corresponding test accuracies in the last 2 consecutive epochs is less than 0.001.

Heterogeneous datasets: The MNIST dataset has $60,000$ training images (with 6000 images of each label) and $10,000$ test images (each having $28 \times 28 = 784$ pixels), and is distributed among the 200 clients in the following *heterogeneous* manner: Each client takes a random permutation of the probability vector $[0.8, 0.1, 0.1, 0, 0, 0, 0, 0, 0, 0]$. Suppose it obtains a vector $\boldsymbol{p}$ such that $p_i = 0.8, p_j = 0.1, p_k = 0.1$ for some distinct $i, j, k \in [0 : 9]$ and $p_l = 0$ for the rest of the indices, then it selects *uniformly at random* $800, 100, 100$ training images with label i, j, k, respectively.

Adversarial attacks: We have 12.5% adversarial clients, i.e., 25 out of 200 clients are corrupt, and the corrupt set of clients may change in every iteration. We implement six adversarial attacks: (i) the 'random gradient attack', where local gradients at clients are replaced by independent Gaussian random vectors having the same norm[8] as the corresponding gradients; (ii) the 'reverse average gradient attack', where corrupt clients send -ve of their average local gradients; (iii) the 'gradient shift attack', where local gradients of corrupt clients are shifted by a scaled (by factor of 50) Gaussian random vector (same for all); (iv) the 'all ones attack', where gradients of the corrupt clients are replaced by the all ones vector; (v) the 'Baruch attack', which was designed in (Baruch et al., 2019) specifically for coordinate-wise trimmed mean (trimmean) (Yin et al., 2018), Krum (Blanchard et al., 2017), and Bulyan (Mhamdi et al., 2018) defenses; and (vi) the 'reverse scaled average gradient attack', where corrupt clients compute the -ve of their average local gradients, scale it by the factor of 50, and then send it.

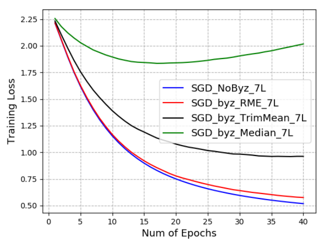
Performance: We train our neural network under all the above-described adversarial attacks, and demonstrate in Figure 1 the performance of our method (red color) against four other methods for robust gradient aggregation, namely, *coordinate-wise trimmed-mean* (black color) and *coordinate-wise median* (green color), which were used in (Xie et al., 2019a; Yin et al., 2018; 2019), Krum (magenta color), which was proposed in (Blanchard et al., 2017), and Bulyan (cyan color), which was proposed in (Mhamdi et al., 2018). For reference, we also plot (in blue color) the performance of Algorithm 1 with the same setup as above but without adversaries and with no decoding. For each attack, we plot two curves, one for training loss vs. number of epochs and the other for test accuracy vs. number of epochs.

It can be seen from the comparison in Figure 1 that our method consistently outperforms all these methods in all the attacks that we have implemented.[9] In particular, for attacks

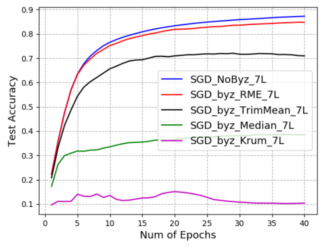
[7] $784 \times 25 = 19,600$ weights between the input and the first layer, 25 bias terms (one for each node in the hidden layer), $25 \times 10 = 250$ weights between the first layer and the output layer, and 10 bias terms (one for each node in the output layer).

[8] Note that changing the direction while keeping the norm same is among the worst attacks as the corrupt gradients cannot be filtered out just based on their norms.

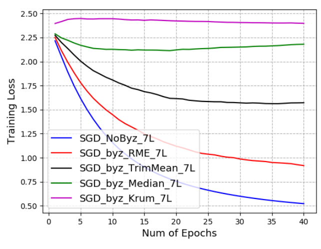
[9] We found out that the Bulyan defense mechanism is significantly slower than all other mechanisms. Due to this, we only implemented this for the Baruch-attack, which was specifically designed against Krum/Bulyan algorithms. Since a basic building block of Bulyan is Krum, and Krum performs the worst among all the mechanisms that we implemented, we do not expect Bulyan



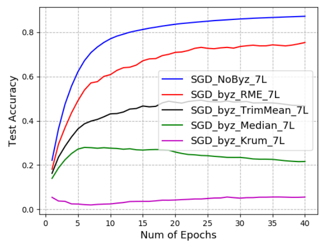
(a) Training loss vs. no. of epochs under the 'random gradient attack'



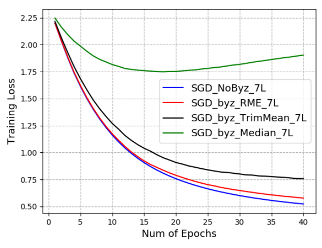
(b) Test accuracy vs. no. of epochs under the 'random gradient attack'



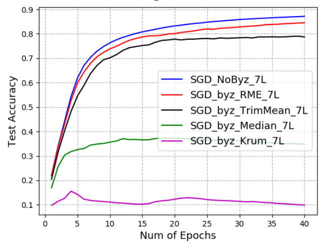
(c) Training loss vs. no. of epochs under the 'reverse average gradient attack'



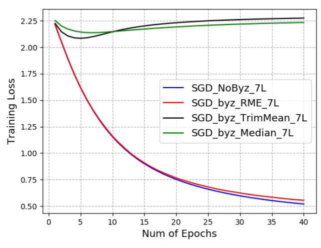
(d) Test accuracy vs. no. of epochs under the 'reverse average gradient attack'



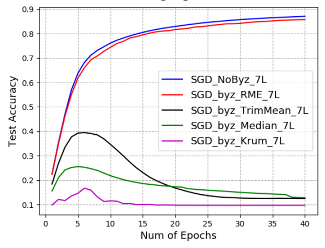
(e) Training loss vs. no. of epochs under the 'gradient shift attack'



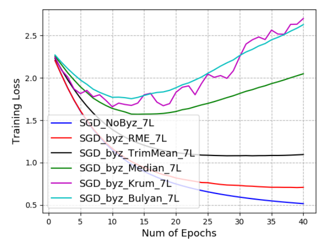
(f) Test accuracy vs. no. of epochs under the 'gradient shift attack'



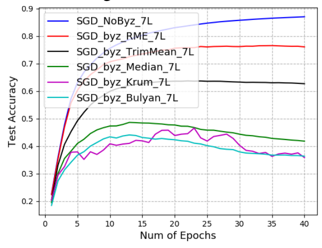
(g) Training loss vs. no. of epochs under the 'all ones attack'



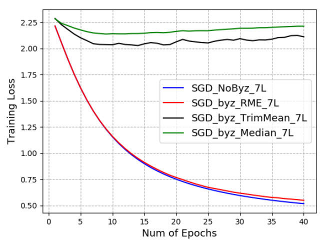
(h) Test accuracy vs. no. of epochs under the 'all ones attack'



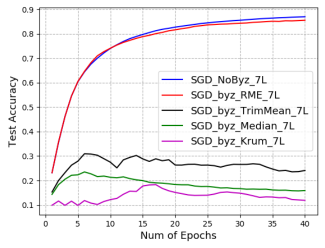
(i) Training loss vs. no. of epochs under the 'Baruch attack'



(j) Test accuracy vs. no. of epochs under the 'Baruch attack'



(k) Training loss vs. no. of epochs under 'reverse scaled average gradient attack'



(l) Test accuracy vs. no. of epochs under 'reverse scaled average gradient attack'

Figure 1. We compare the performance of our method (red) against four methods for robust gradient aggregation, namely, coordinate-wise trimmed-mean (black), coordinate-wise median (green), Krum (magenta), and Bulyan (cyan) under several adversarial attacks, and plot training loss and test accuracy against number of epochs. The plot in blue corresponds to running Algorithm 1 with no adversaries and no decoding. In the legends, $7L$ denotes that we are taking $H = 7$ local iterations. See also Footnotes 9, 10.

(i), (iii), (iv), (vi), our method (with adversaries) achieves *similar* performance for both training loss and test accuracy as that of running SGD with local iterations but *without* any adversaries and defense mechanism at the server; and for attacks (ii), (v), the performance difference (test accuracy) is around 0.1 at epoch 40, which is still significantly better than all other methods.[10] This conforms to the inadequacy of using these methods in our setting, as described in Section 3. Note that the experiments presented in (Xie et al., 2019a; Yin et al., 2018) only implemented a benign 'label-flipping' attack, which is a data poisoning attack. This is not a dynamic attack as, unlike gradient attacks, it does not adapt to the learning process over iterations. In contrast, in all our attacks, corrupt clients send adversarial gradients in *every iteration*, making them significantly more malicious than just flipping the labels. As we have mentioned in the related work (on page 2), and we want to emphasize again, that though (Xie et al., 2019a) also studied the same problem as ours, but employed 'coordinate-wise trimmed mean' for robust gradient aggregation, their convergence bound, in our opinion, are vacuous, as the sub-optimality gap in their bounds *always* scales linearly with the diameter of the parameter space. As far as we know, ours is the first theoretical result that combines Byzantine-resilience with local iterations for high-dimensional distributed training on heterogeneous datasets with good empirical performance.

to perform significantly better than Krum in other attacks as well – note that both Krum and Bulyan are the worst performing defense mechanisms against the Baruch-attack.

[10]We plot the Krum performance in the training loss vs. number of epochs figures only for the attacks (ii), (v); because in all other attacks, the Krum training loss became very high (above 100) even before epoch 40 and would have prevented observing other methods' performance if we had plotted it.

Acknowledgments

Deepesh Data would like to thank Navjot Singh for his generous help with setting up the experiments. This work was supported in part by NSF grants #1740047, #2007714, and UC-NL grant LFR-18-548554.

References

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. In *Neural Information Processing Systems (NeurIPS)*, pp. 4618–4628, 2018.

Baruch, G., Baruch, M., and Goldberg, Y. A little is enough: Circumventing defenses for distributed learning. In *Neural Information Processing Systems (NeurIPS)*, pp. 8632–8642, 2019.

Basu, D., Data, D., Karakus, C., and Diggavi, S. N. Qsparse-local-sgd: Distributed SGD with quantization, sparsification and local computations. In *NeurIPS*, pp. 14668–14679, 2019.

Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., and Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*, pp. 119–129, 2017.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, pp. 177–186, 2010.

Charikar, M., Steinhardt, J., and Valiant, G. Learning from untrusted data. In *STOC*, pp. 47–60, 2017.

Chen, L., Wang, H., Charles, Z. B., and Papailiopoulos, D. S. DRACO: byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning (ICML)*, pp. 902–911, 2018.

Chen, Y., Su, L., and Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *POMACS*, 1(2):44:1–44:25, 2017.

Data, D. and Diggavi, S. N. Byzantine-tolerant distributed coordinate descent. In *ISIT*, pp. 2724–2728, 2019.

Data, D. and Diggavi, S. N. Byzantine-resilient SGD in high dimensions on heterogeneous data. *CoRR*, abs/2005.07866, 2020a. URL https://arxiv.org/abs/2005.07866. Preliminary version appeared in IEEE ISIT.

Data, D. and Diggavi, S. N. On byzantine-resilient high-dimensional stochastic gradient descent. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 2628–2633. IEEE, 2020b.

Data, D., Song, L., and Diggavi, S. N. Data encoding methods for byzantine-resilient distributed optimization. In *ISIT*, pp. 2719–2723, 2019.

Data, D., Song, L., and Diggavi, S. N. Data encoding for byzantine-resilient distributed optimization. *IEEE Transactions on Information Theory*, 67(2):1117–1140, 2021. doi: 10.1109/TIT.2020.3035868.

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A. W., Tucker, P. A., Yang, K., and Ng, A. Y. Large scale distributed deep networks. In *Neural Information Processing Systems (NIPS)*, pp. 1232–1240, 2012.

Diakonikolas, I. and Kane, D. M. Recent advances in algorithmic high-dimensional robust statistics. *CoRR*, abs/1911.05911, 2019.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Moitra, A., and Stewart, A. Robust estimators in high-dimensions without the computational intractability. *SIAM J. Comput.*, 48(2):742–864, 2019.

Dong, Y., Hopkins, S. B., and Li, J. Quantum entropy scoring for fast robust mean estimation and improved outlier detection. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Neural Information Processing Systems (NeurIPS)*, pp. 6065–6075, 2019.

Ghosh, A., Hong, J., Yin, D., and Ramchandran, K. Robust federated learning in a heterogeneous environment. *CoRR*, abs/1906.06629, 2019. URL http://arxiv.org/abs/1906.06629.

Haddadpour, F. and Mahdavi, M. On the convergence of local descent methods in federated learning. *CoRR*, abs/1910.14425, 2019. URL http://arxiv.org/abs/1910.14425.

Haddadpour, F., Kamani, M. M., Mahdavi, M., and Cadambe, V. R. Local SGD with periodic averaging: Tighter analysis and adaptive synchronization. In *Neural Information Processing Systems (NeurIPS)*, pp. 11080–11092, 2019.

Kairouz, P. et al. Advances and open problems in federated learning. *CoRR*, abs/1912.04977, 2019.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. SCAFFOLD: stochastic controlled averaging for federated learning. In *International Conference on Machine Learning (ICML)*, pp. 5132–5143, 2020.

Khaled, A., Mishchenko, K., and Richtárik, P. Tighter theory for local SGD on identical and heterogeneous data. In Chiappa, S. and Calandra, R. (eds.), *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 4519–4529, 2020.

Konecný, J. Stochastic, distributed and federated optimization for machine learning. *CoRR*, abs/1707.01155, 2017.

Konecný, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, abs/1610.02527, 2016.

Lai, K. A., Rao, A. B., and Vempala, S. S. Agnostic estimation of mean and covariance. In *FOCS*, pp. 665–674, 2016.

Li, J. Robustness in Machine Learning (CSE 599-M); Lecture 5 - Efficient filtering from spectral signatures, 2019. URL https://jerryzli.github.io/robust-ml-fall19.html.

Li, L., Xu, W., Chen, T., Giannakis, G. B., and Ling, Q. RSA: byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Conference on Artificial Intelligence (AAAI)*, pp. 1544–1551, 2019a.

Li, X., Yang, W., Wang, S., and Zhang, Z. Communication efficient decentralized training with multiple local updates. *CoRR*, abs/1910.09126, 2019b.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=HJxNAnVtDS.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, 2017.

Mhamdi, E. M. E., Guerraoui, R., and Rouault, S. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning (ICML)*, pp. 3518–3527, 2018.

Mohri, M., Sivek, G., and Suresh, A. T. Agnostic federated learning. In *International Conference on Machine Learning (ICML)*, pp. 4615–4625, 2019.

Rajput, S., Wang, H., Charles, Z. B., and Papailiopoulos, D. S. DETOX: A redundancy-based framework for faster and more robust gradient aggregation. In *NeurIPS*, pp. 10320–10330, 2019.

Sahu, A. K., Li, T., Sanjabi, M., Zaheer, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *Conference on Machine Learning and Systems (MLSys)*, 2020. URL http://arxiv.org/abs/1812.06127.

Steinhardt, J., Charikar, M., and Valiant, G. Resilience: A criterion for learning in the presence of arbitrary outliers. In *ITCS*, pp. 45:1–45:21, 2018.

Su, L. and Xu, J. Securing distributed gradient descent in high dimensional statistical learning. *POMACS*, 3(1): 12:1–12:41, 2019.

Vershynin, R. Introduction to the non-asymptotic analysis of random matrices. *CoRR*, abs/1011.3027, 2010.

Xie, C., Koyejo, O., and Gupta, I. SLSGD: secure and efficient distributed on-device machine learning. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD, Proceedings, Part II*, pp. 213–228, 2019a.

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

Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. L. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, pp. 5636–5645, 2018.

Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. L. Defending against saddle point attack in byzantine-robust distributed learning. In *ICML*, pp. 7074–7084, 2019.

Yu, H., Jin, R., and Yang, S. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In *ICML*, pp. 7184–7193, 2019a.

Yu, H., Yang, S., and Zhu, S. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Conference on Artificial Intelligence (AAAI)*, pp. 5693–5700, 2019b.