Defending Against Saddle Point Attack in Byzantine-Robust Distributed Learning

Dong Yin [*1], Yudong Chen [†3], Kannan Ramchandran [‡1], and Peter Bartlett [§1,2]

[1]Department of Electrical Engineering and Computer Sciences, UC Berkeley
[2]Department of Statistics, UC Berkeley
[3]School of Operations Research and Information Engineering, Cornell University

January 28, 2019

Abstract

We study robust distributed learning that involves minimizing a non-convex loss function with saddle points. We consider the Byzantine setting where some worker machines have abnormal or even arbitrary and adversarial behavior. In this setting, the Byzantine machines may create fake local minima near a saddle point that is far away from any true local minimum, even when robust gradient estimators are used. We develop *ByzantinePGD*, a robust first-order algorithm that can provably escape saddle points and fake local minima, and converge to an approximate true local minimizer with low iteration complexity. As a by-product, we give a simpler algorithm and analysis for escaping saddle points in the usual non-Byzantine setting. We further discuss three robust gradient estimators that can be used in ByzantinePGD, including median, trimmed mean, and iterative filtering. We characterize their performance in concrete statistical settings, and argue for their near-optimality in low and high dimensional regimes.

1 Introduction

Distributed computing becomes increasingly important in modern data-intensive applications. In many applications, large-scale datasets are distributed over multiple machines for parallel processing in order to speed up computation. In other settings, the data sources are naturally distributed, and for privacy and efficiency considerations, the data are not transmitted to a central machine. An example is the recently proposed *Federated Learning* paradigm [49, 38, 37], in which the data are stored and processed locally in end users' cellphones and personal computers.

In a standard worker-server distributed computing framework, a single master machine is in charge of maintaining and updating the parameter of interest, and a set of worker machines store the data, perform local computation and communicate with the master. In this setting, messages received from worker machines are prone to errors due to data corruption, hardware/software malfunction, and communication delay and failure. These problems are only exacerbated in a decentralized distributed architecture such as Federated Learning, where some machines may be subjected to malicious and coordinated attack and manipulation. A well-established framework for studying such scenarios is the *Byzantine* setting [40], where a subset of machines behave completely arbitrarily—even in a way that depends on the algorithm used and the data on the other machines—thereby capturing the unpredictable nature of the errors. Developing distributed algorithms that are robust in the Byzantine setting has become increasingly critical.

In this paper we focus on robust distributed optimization for statistical learning problems. Here the data points are generated from some unknown distribution $\mathcal{D}$ and stored locally in m worker machines, each storing n data points; the goal is to minimize a population loss function

$F : \mathcal{W} \to \mathbb{R}$ defined as an expectation over $\mathcal{D}$, where $\mathcal{W} \subseteq \mathbb{R}^d$ is the parameter space. We assume that $\alpha \in (0, 1/2)$ fraction of the worker machines are Byzantine; that is, their behavior is arbitrary. This Byzantine-robust distributed learning problem has attracted attention in a recent line of work [3, 10, 17, 26, 61, 62, 69]. This body of work develops robust algorithms that are guaranteed to output an approximate minimizer of F when it is convex, or an approximate stationary point in the non-convex case.

However, fitting complicated machine learning models often requires finding a *local minimum* of *non-convex* functions, as exemplified by training deep neural networks and other high-capacity learning architectures [59, 28, 29]. It is well-known that many of the stationary points of these problems are in fact saddle points and far away from any local minimum [35, 29]. These tasks hence require algorithms capable of efficiently *escaping saddle points* and converging approximately to a local minimizer. In the centralized setting without Byzantine adversaries, this problem has been studied actively and recently [27, 32, 12, 33].

A main observation of this work is that the interplay between non-convexity and Byzantine errors makes escaping saddle points much more challenging. In particular, by orchestrating their messages sent to the master machine, *the Byzantine machines can create fake local minima near a saddle point of F that is far away from any true local minimizer.* Such a strategy, which may be referred to as **saddle point attack**, foils existing algorithms as we elaborate below:

- **Challenges due to non-convexity:** When F is convex, gradient descent (GD) equipped with a robust gradient estimator is guaranteed to find an approximate global minimizer (with accuracy depending on the fraction of Byzantine machines) [17, 69, 3]. However, when F is non-convex, such algorithms may be trapped in the neighborhood of a saddle point; see Example 1 in Appendix A.

- **Challenges due to Byzantine machines:** Without Byzantine machines, vanilla GD [42], as well as its more efficient variants such as perturbed gradient descent (PGD) [32], are known to converge to a local minimizer with high probability. However, Byzantine machines can manipulate PGD and GD (even robustified) into fake local minimum near a saddle point; see Example 2 in Appendix A.

We discuss and compare with existing work in more details in Section 2. The observations above show that existing robust and saddle-escaping algorithms, as well as their naive combination, are insufficient against saddle point attack. Addressing these challenges requires the development of new robust distributed optimization algorithms.

1.1 Our Contributions

In this paper, we develop ***ByzantinePGD***, a computation- and communication-efficient first-order algorithm that is able to escape saddle points and the fake local minima created by Byzantine machines, and converge to an approximate local minimizer of a non-convex loss. To the best of our knowledge, our algorithm is the first to achieve such guarantees under *adversarial* noise.

Specifically, ByzantinePGD aggregates the empirical gradients received from the normal and Byzantine machines, and computes a robust estimate $\widehat{\mathbf{g}}(\mathbf{w})$ of the true gradient $\nabla F(\mathbf{w})$ of the population loss F. Crucial to our algorithm is the injection of random perturbation to the iterates $\mathbf{w}$, which serves the dual purpose of escaping saddling point and fake local minima. Our use of perturbation thus plays a more signified role than in existing algorithms such as PGD [32], as it also serves to combat the effect of Byzantine errors. To achieve this goal, we incorporate two crucial innovations: (i) we use multiple rounds of larger, yet carefully calibrated, amount of perturbation that is necessary to survive saddle point attack, (ii) we use the moving distance in the parameter space as the criterion for successful escape, eliminating the need of (robustly) evaluating function values. Consequently, our analysis is significantly different, and arguably simpler, than that of PGD.

We develop our algorithmic and theoretical results in a flexible, two-part framework, decomposing the *optimization* and *statistical* components of the problem.

The optimization part: We consider a general problem of optimizing a population loss function F given an *inexact gradient oracle*. For each query point $\mathbf{w}$, the Δ-inexact gradient oracle returns a vector $\widehat{\mathbf{g}}(\mathbf{w})$ (possibly chosen adversarially) that satisfies $\|\widehat{\mathbf{g}}(\mathbf{w}) - \nabla F(\mathbf{w})\|_2 \leq \Delta$, where Δ

is non-zero but bounded. Given access to such an inexact oracle, we show that ByzantinePGD outputs an approximate local minimizer; moreover, *no* other algorithm can achieve significantly better performance in this setting in terms of the dependence on Δ:

Theorem 1 (Informal; see Sec. 4.2). *Within $\widetilde{\mathcal{O}}(\frac{1}{\Delta^2})$ iterations, ByzantinePGD outputs an approximate local minimizer $\widetilde{\mathbf{w}}$ that satisfies $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \lesssim \Delta$ and $\lambda_{\min}\big(\nabla^2 F(\widetilde{\mathbf{w}})\big) \gtrsim -\Delta^{2/5}$, where $\lambda_{\min}$ is the minimum eigenvalue. In addition, given only access to Δ-inexact gradient oracle, **no** algorithm is guaranteed to find a point $\widetilde{\mathbf{w}}$ with $\|\nabla F(\widetilde{\mathbf{w}})\|_2 < \Delta/2$ **or** $\lambda_{\min}\big(\nabla^2 F(\widetilde{\mathbf{w}})\big) > -\Delta^{1/2}/2$.*

Our algorithm is communication-efficient: it only sends gradients, and the number of parallel iterations in our algorithm *matches* the well-known iteration complexity of GD for non-convex problems in non-Byzantine setting [53] (up to log factors). In the exact gradient setting, a variant of the above result in fact matches the guarantees for PGD [32]—as mentioned, our proof is simpler.

Additionally, beyond Byzantine distributed learning, our results apply to any non-convex optimization problems (distributed or not) with inexact information for the gradients, including those with noisy but non-adversarial gradients. Thus, we believe our results are of independent interest in broader settings.

The statistical part: The optimization guarantee above can be applied whenever one has a robust aggregation procedure that serves as an inexact gradient oracle with a bounded error Δ. We consider three concrete examples of such robust procedures: median, trimmed mean, and iterative filtering [22, 23]. Under statistical settings for the data, we provide explicit bounds on their errors Δ as a function of the number of worker machines m, the number of data points on each worker machine n, the fraction of Byzantine machines α, and the dimension of the parameter space d. Combining these bounds with the optimization result above, we obtain concrete statistical guarantees on the output $\widetilde{\mathbf{w}}$. Furthermore, we argue that our first-order guarantees on $\|\nabla F(\widetilde{\mathbf{w}})\|_2$ are often nearly *optimal* when compared against a universal statistical lower bound. This is summarized below:

Theorem 2 (Informal; see Sec. 5). *When combined with each of following three robust aggregation procedures, ByzantinePGD achieves the statistical guarantees:*
(i) median/; $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \lesssim \frac{\alpha\sqrt{d}}{\sqrt{n}} + \frac{d}{\sqrt{nm}} + \frac{\sqrt{d}}{n}$;
(ii) trimmed mean: $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \lesssim \frac{\alpha d}{\sqrt{n}} + \frac{d}{\sqrt{nm}}$;
(iii) iterative filtering: $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \lesssim \frac{\sqrt{\alpha}}{\sqrt{n}} + \frac{\sqrt{d}}{\sqrt{nm}}$.
*Moreover, **no** algorithm can achieve $\|\nabla F(\widetilde{\mathbf{w}})\|_2 = o\big(\frac{\alpha}{\sqrt{n}} + \frac{\sqrt{d}}{\sqrt{nm}}\big)$.*

We emphasize that the above results are established under a very strong adversary model: the Byzantine machines are allowed to send messages that depend arbitrarily on each other and on the data on the normal machines; they may even behave adaptively during the iterations of our algorithm. Consequently, this setting requires robust *functional* estimation (of the gradient function), which is a much more challenging problem than the robust *mean* estimation setting considered by existing work on median, trimmed mean and iterative filtering. To overcome this difficulty, we make use of careful covering net arguments to establish certain error bounds that hold *uniformly* over the parameter space, regardless of the behavior of the Byzantine machines. Importantly, our inexact oracle framework allows such arguments to be implemented in a transparent and modular manner.

Notation For an integer $N > 0$, define the set $[N] := \{1, 2, \ldots, N\}$. For matrices, denote the operator norm by $\|\cdot\|_2$; for symmetric matrices, denote the largest and smallest eigenvalues by $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$, respectively. The d-dimensional ℓ_2 ball centered at $\mathbf{w}$ with radius r is denoted by $\mathbb{B}_{\mathbf{w}}^{(d)}(r)$, or $\mathbb{B}_{\mathbf{w}}(r)$ when it is clear from the context.

2 Related Work

Efficient first-order algorithms for escaping saddle points Our algorithm is related to a recent line of work which develops efficient first-order algorithms for escaping saddle points. Although vanilla GD converges to local minimizers almost surely [42, 43], achieving convergence

Algorithm	PGD	Neon+GD	Neon2+GD	**ByzantinePGD**
Byzantine-robust?	no	no	no	yes
Purpose of perturbation	escape SP	escape SP	escape SP	escape SP & robustness
Escaping method	GD	NC search	NC search	inexact GD
Termination criterion	decrease in F	decrease in F	distance in $\mathcal{W}$	distance in $\mathcal{W}$
Multiple rounds?	no	no	no	yes

Table 1: Comparison with PGD, Neon+GD, and Neon2+GD. SP = saddle point.

	Robust Aggregation Method	Non-convex Guarantee
Feng et al. [26]	geometric median	no
Chen et al. [17]	geometric median	no
Blanchard et al. [10]	Krum	first-order
Yin et al. [69]	median, trimmed mean	first-order
Xie et al. [67]	mean-around-median, marginal median	first-order
Alistarh et al. [3]	martingale-based	no
Su and Xu [63]	iterative filtering	no
This work	median, trimmed mean, iterative filtering	second-order

Table 2: Comparison with other Byzantine-robust distributed learning algorithms.

in polynomial time requires more a careful algorithmic design [25]. Such convergence guarantees are enjoyed by several GD-based algorithms; examples include PGD [32], Neon+GD [68], and Neon2+GD [5]. The general idea of these algorithms is to run GD and add perturbation to the iterate when the gradient is small. While our algorithm also uses this idea, the design and analysis techniques of our algorithm are significantly different from the work above in the following aspects (also summarized in Table 1).

- In our algorithm, besides helping with escaping saddle points, the random perturbation has the additional role of defending against adversarial errors.

- The perturbation used in our algorithm needs to be larger, yet carefully calibrated, in order to account for the influence of the inexactness of gradients across the iterations, especially iterations for escaping saddle points.

- We run inexact GD after the random perturbation, while Neon+GD and Neon2+GD use negative curvature (NC) search. It is not immediately clear whether NC search can be robustified against Byzantine failures. Compared to PGD, our analysis is arguably *simpler* and more *straightforward*.

- Our algorithm does not use the value of the loss function (hence no need for robust function value estimation); PGD and Neon+GD assume access to the (exact) function values.

- We employed *multiple* rounds of perturbation to boost the probability of escaping saddle points; this technique is not used in PGD, Neon+GD, or Neon2+GD.

Inexact oracles Optimization with an inexact oracle (e.g. noisy gradients) has been studied in various settings such as general convex optimization [7, 21], robust estimation [55], and structured non-convex problems [6, 16, 11, 71]. Particularly relevant to us is the recent work by Jin et al. [34], who consider the problem of minimizing F when only given access to the gradients of another *smooth* function $\widehat{F}$ satisfying $\|\nabla \widehat{F}(\mathbf{w}) - \nabla F(\mathbf{w})\|_\infty \leq \Delta/\sqrt{d}$, $\forall \mathbf{w}$. Their algorithm uses Gaussian smoothing on $\widehat{F}$. We emphasize that the inexact gradient setting considered by them is much more benign than our Byzantine setting, since (i) their inexactness is defined in terms of ℓ_∞ norm whereas the inexactness in our problem is in ℓ_2 norm, and (ii) we assume that the inexact gradient can be *any* vector within Δ error, and thus the smoothing technique is not applicable in our problem. Moreover, the iteration complexity obtained by Jin et al. [34] may be a high-degree polynomial of the problem parameters and thus not suitable for distributed implementation.

Byzantine-robust distributed learning Solving large scale learning problems in distributed systems has received much attention in recent years, where communication efficiency and Byzantine robustness are two important topics [58, 41, 70, 10, 15, 20]. Here, we compare with existing Byzantine-robust distributed learning algorithms that are most relevant to our work, and summarize the comparison in Table 2. A general idea of designing Byzantine-robust algorithms is to combine optimization algorithms with a robust aggregation (or outlier removal) subroutine. For convex losses, the aggregation subroutines analyzed in the literature include geometric median [26, 17], median and trimmed mean [69], iterative filtering for the high dimensional setting [63], and martingale-based methods for the SGD setting [3]. For non-convex losses, to the best of our knowledge, existing works only provide first-order convergence guarantee (i.e., small gradients), by using aggregation subroutines such as the Krum function [10], median and trimmed mean [69], mean-around-median and marginal median [67]. In this paper, we make use of subroutines based on median, trimmed mean, and iterative filtering. Our analysis of median and trimmed mean follows Yin et al. [69]. Our results based on the iterative filtering subroutine, on the other hand, are new:

- The problem that we tackle is harder than what is considered in the original iterative filtering papers [22, 23]. There they only consider robust estimation of a single mean parameter, where as we guarantee robust gradient estimation over the parameter space.

- Recent work by Su and Xu [63] also makes use of the iterative filtering subroutine for the Byzantine setting. They only study strongly convex loss functions, and assume that the gradients are sub-exponential and $d \leq \mathcal{O}(\sqrt{mn})$. Our results apply to the non-convex case and do not require the aforementioned condition on d (which may therefore scale, for example, linearly with the sample size mn), but we impose the stronger assumption of sub-Gaussian gradients.

Other non-convex optimization algorithms Besides first-order GD-based algorithms, many other non-convex optimization methods that can provably converge to approximate local minimum have received much attention in recent years. For specific problems such as phase retrieval [11], low-rank estimation [16, 72], and dictionary learning [1, 64], many algorithms are developed by leveraging the particular structure of the problems, and the either use a smart initialization [11, 65] or initialize randomly [18, 14]. Other algorithms are developed for general non-convex optimization, and they can be classified into gradient-based [27, 44, 68, 4, 5, 33], Hessian-vector-product-based [12, 2, 56, 57], and Hessian-based [54, 19] methods. While algorithms using Hessian information can usually achieve better convergence rates—for example, $\mathcal{O}(\frac{1}{\epsilon^{3/2}})$ by Curtis et al. [19], and $\mathcal{O}(\frac{1}{\epsilon^{7/4}})$ by Carmon et al. [12]— gradient-based methods are easier to implement in practice, especially in the distributed setting we are interested in.

Robust statistics Outlier-robust estimation is a classical topic in statistics [30]. The coordinate-wise median aggregation subroutine that we consider is related to the median-of-means estimator [52, 31], which has been applied to various robust inference problems [51, 47, 50].

A recent line of work develops efficient robust estimation algorithms in high-dimensional settings [8, 22, 39, 13, 60, 45, 9, 36, 46]. In the centralized setting, the recent work [24] proposes a scheme, similar to the iterative filtering procedure, that iteratively removes outliers for gradient-based optimization.

3 Problem Setup

We consider empirical risk minimization for a statistical learning problem where each data point $\mathbf{z}$ is sampled from an unknown distribution $\mathcal{D}$ over the sample space $\mathcal{Z}$. Let $f(\mathbf{w}; \mathbf{z})$ be the loss function of a parameter vector $\mathbf{w} \in \mathcal{W} \subseteq \mathbb{R}^d$, where $\mathcal{W}$ is the parameter space. The population loss function is therefore given by $F(\mathbf{w}) := \mathbb{E}_{\mathbf{z} \sim \mathcal{D}}[f(\mathbf{w}; \mathbf{z})]$.

We consider a distributed computing system with one *master* machine and m *worker* machines, αm of which are Byzantine machines and the other $(1-\alpha)m$ are normal. Each worker machine has n data points sampled i.i.d. from $\mathcal{D}$. Denote by $\mathbf{z}_{i,j}$ the j-th data point on the i-th worker machine, and let $F_i(\mathbf{w}) := \frac{1}{n} \sum_{j=1}^n f(\mathbf{w}; \mathbf{z}_{i,j})$ be the empirical loss function on the i-th machine. The master machine and worker machines can send and receive messages via the following communication protocol: In each parallel iteration, the master machine sends a parameter vector $\mathbf{w}$ to all the

worker machines, and then each *normal* worker machine computes the gradient of its empirical loss $F_i(\cdot)$ at $\mathbf{w}$ and sends the gradient to the master machine. The Byzantine machines may be jointly controlled by an adversary and send arbitrary or even malicious messages. We denote the unknown set of Byzantine machines by $\mathcal{B}$, where $|\mathcal{B}| = \alpha m$. With this notation, the gradient sent by the i-th worker machine is

$$\widehat{\mathbf{g}}_i(\mathbf{w}) = \begin{cases} \nabla F_i(\mathbf{w}) & i \in [m] \setminus \mathcal{B}, \\ * & i \in \mathcal{B}, \end{cases} \tag{1}$$

where the symbol $*$ denotes an arbitrary vector. As mentioned, the adversary is assumed to have complete knowledge of the algorithm used and the data stored on all machines, and the Byzantine machines may collude [48] and adapt to the output of the master and normal worker machines. We only make the mild assumption that the adversary cannot *predict* the random numbers generated by the master machine.

We consider the scenario where $F(\mathbf{w})$ is non-convex, and our goal to find an approximate local minimizer of $F(\mathbf{w})$. Note that a first-order stationary point (i.e., one with a small gradient) is not necessarily close to a local minimizer, since the point may be a *saddle point* whose Hessian matrix has a large negative eigenvalue. Accordingly, we seek to find a *second-order stationary point* $\widetilde{\mathbf{w}}$, namely, one with a small gradient and a nearly positive semidefinite Hessian:

Definition 1 (Second-order stationarity). *We say that $\widetilde{\mathbf{w}}$ is an (ϵ_g, ϵ_H)-second-order stationary point of a twice differentiable function $F(\cdot)$ if $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \leq \epsilon_g$ and $\lambda_{\min}\big(\nabla^2 F(\widetilde{\mathbf{w}})\big) \geq -\epsilon_H$.*

In the sequel, we make use of several standard concepts from continuous optimization.

Definition 2 (Smooth and Hessian-Lipschitz functions). *A function h is called L-smooth if $\sup_{\mathbf{w} \neq \mathbf{w}'} \frac{\|\nabla h(\mathbf{w}) - \nabla h(\mathbf{w}')\|_2}{\|\mathbf{w} - \mathbf{w}'\|_2} \leq L$, and ρ-Hessian Lipschitz if $\sup_{\mathbf{w} \neq \mathbf{w}'} \frac{\|\nabla^2 h(\mathbf{w}) - \nabla^2 h(\mathbf{w}')\|_2}{\|\mathbf{w} - \mathbf{w}'\|_2} \leq \rho$.*

Throughout this paper, the above properties are imposed on the *population* loss function $F(\cdot)$.

Assumption 1. *F is L_F-smooth, and ρ_F-Hessian Lipschitz on $\mathcal{W}$.*

4 Byzantine Perturbed Gradient Descent

In this section, we describe our algorithm, *Byzantine Perturbed Gradient Descent* (ByzantinePGD), which provably finds a second-order stationary point of the population loss $F(\cdot)$ in the distributed setting with Byzantine machines. As mentioned, ByzantinePGD robustly aggregates gradients from the worker machines, and performs multiple rounds of carefully calibrated perturbation to combat the effect of Byzantine machines. We now elaborate.

It is well-known that naively aggregating the workers' messages using standard averaging can be arbitrarily skewed in the presence of just a single Byzantine machine. In view of this, we introduce the subroutine $\mathsf{GradAGG}\{\widehat{\mathbf{g}}_i(\mathbf{w})\}_{i=1}^m$, which robustly aggregates the gradients $\{\widehat{\mathbf{g}}_i(\mathbf{w})\}_{i=1}^m$ collected from the m workers. We stipulate that $\mathsf{GradAGG}$ provides an estimate of the true population gradient $\nabla F(\cdot)$ with accuracy Δ, *uniformly* across $\mathcal{W}$. This property is formalized using the terminology of *inexact gradient oracle*.

Definition 3 (Inexact gradient oracle). *We say that $\mathsf{GradAGG}$ provides a Δ-inexact gradient oracle for the population loss $F(\cdot)$ if, for every $\mathbf{w} \in \mathcal{W}$, we have $\|\mathsf{GradAGG}\{\widehat{\mathbf{g}}_i(\mathbf{w})\}_{i=1}^m - \nabla F(\mathbf{w})\|_2 \leq \Delta$.*

Without loss of generality, we assume that $\Delta \leq 1$ throughout the paper. In this section, we treat $\mathsf{GradAGG}$ as a given black box; in Section 5, we discuss several robust aggregation algorithms and characterize their inexactness Δ. We emphasize that in the Byzantine setting, the output of $\mathsf{GradAGG}$ can take values adversarially within the error bounds; that is, $\mathsf{GradAGG}\{\widehat{\mathbf{g}}_i(\mathbf{w})\}_{i=1}^m$ may output an arbitrary vector in the ball $\mathbb{B}_{\nabla F(\mathbf{w})}(\Delta)$, and this vector can depend on the data in all the machines and all previous iterations of the algorithm.

The use of robust aggregation with bounded inexactness, however, is not yet sufficient to guarantee convergence to an approximate local minimizer. As mentioned, the Byzantine machines may create fake local minima that traps a vanilla gradient descent iteration. Our ByzantinePGD algorithm is designed to escape such fake minima as well as any existing saddle points of F.

```
ByzantinePGD(w₀, η, ε, r, Q, R, T_th)              Escape(w̃, η, r, Q, R, T_th)
  w ← w₀                                             for k = 1, 2, ..., Q do
  while true do                                        Master: sample p_k ~ Unif(𝔹₀(r)),
    Master: send w to worker machines.                 w' ← w̃ + p_k, w'₀ ← w'.
    for all i ∈ [m] do in parallel                     for t = 0, 1, ..., T_th do
      Worker i: compute ĝ_i(w)                           Master: send w' to worker machines.
      send to master machine.                            for all i ∈ [m] do in parallel
    end for                                                Worker i: compute ĝ_i(w')
    Master:                                                send to master machine.
    ĝ(w) ← GradAGG{ĝ_i(w)}ᵐᵢ₌₁.                         end for
    if ‖ĝ(w)‖₂ ≤ ε then                                 Master: ĝ(w') ← GradAGG{ĝ_i(w')}ᵐᵢ₌₁.
      Master: w̃ ← w,                                    if ‖w' − w'₀‖₂ ≥ R then
      (esc, w, ĝ(w)) ← Escape (w̃, η, r, Q, R,            return (true, w', ĝ(w')).
  T_th).                                                else
      if esc = false then                                 w' ← w' − ηĝ(w')
        return w̃.                                       end if
      end if                                           end for
    end if                                           end for
    Master: w ← w − ηĝ(w).                           return (false, w', ĝ(w')).
  end while
```

Algorithm 1: Byzantine Perturbed Gradient Descent (ByzantinePGD)

4.1 Algorithm

We now describe the details of our algorithm, given in the left panel of Algorithm 1. We focus on unconstrained optimization, i.e., $\mathcal{W} = \mathbb{R}^d$. In Section 5, we show that the iterates $\mathbf{w}$ during the algorithm actually stay in a bounded ℓ_2 ball centered at the initial iterate $\mathbf{w}_0$, and we will discuss the statistical error rates within the bounded space.

In each parallel iteration, the master machine sends the current iterate $\mathbf{w}$ to all the worker machines, and the worker machines send back $\{\widehat{\mathbf{g}}_i(\mathbf{w})\}$. The master machine aggregates the workers' gradients using GradAGG and computes a robust estimate $\widehat{\mathbf{g}}(\mathbf{w})$ of the population gradient $\nabla F(\mathbf{w})$. The master machine then performs a gradient descent step using $\widehat{\mathbf{g}}(\mathbf{w})$. This procedure is repeated until it reaches a point $\widetilde{\mathbf{w}}$ with $\|\widehat{\mathbf{g}}(\mathbf{w})\|_2 \leq \epsilon$ for a pre-specified threshold ϵ.

At this point, $\widetilde{\mathbf{w}}$ may lie near a saddle point whose Hessian has a large negative eigenvalue. To escape this potential saddle point, the algorithm invokes the Escape routine (right panel of Algorithm 1), which performs Q rounds of *perturbation-and-descent* operations. In each round, the master machine perturbs $\widetilde{\mathbf{w}}$ randomly and independently within the ball $\mathbb{B}_{\widetilde{\mathbf{w}}}(r)$. Let $\mathbf{w}'_0$ be the perturbed vector. Starting from the $\mathbf{w}'_0$, the algorithm conducts at most T_{th} parallel iterations of Δ-inexact gradient descent (using GradAGG as before):

$$\mathbf{w}'_t = \mathbf{w}'_{t-1} - \eta\widehat{\mathbf{g}}(\mathbf{w}'_{t-1}), \ t \leq T_{\mathrm{th}}. \tag{2}$$

During this process, once we observe that $\|\mathbf{w}'_t - \mathbf{w}'_0\|_2 \geq R$ for some pre-specified threshold R (this means the iterate moves by a sufficiently large distance in the parameter space), we claim that $\widetilde{\mathbf{w}}$ is a saddle point and the algorithm has escaped it; we then resume Δ-inexact gradient descent starting from $\mathbf{w}'_t$. If after Q rounds no sufficient move in the parameter space is ever observed, we claim that $\widetilde{\mathbf{w}}$ is a second-order stationary point of $F(\mathbf{w})$ and output $\widetilde{\mathbf{w}}$.

4.2 Convergence Guarantees

In this section, we provide the theoretical result guaranteeing that Algorithm 1 converges to a second-order stationary point. In Theorem 3, we let $F^* := \min_{\mathbf{w} \in \mathbb{R}^d} F(\mathbf{w})$, $\mathbf{w}_0$ be the initial iterate, and $F_0 := F(\mathbf{w}_0)$.

Theorem 3 (ByzantinePGD). *Suppose that Assumptions 1 holds, and assume that* GradAGG *provides a Δ-inexact gradient oracle for $F(\cdot)$ with $\Delta \leq 1$. Given any $\delta \in (0,1)$, choose the parameters for Algorithm 1 as follows: step-size $\eta = \frac{1}{L_F}$, $\epsilon = 3\Delta$, $r = 4\Delta^{3/5}d^{3/10}\rho_F^{-1/2}$, $R =$*

$\Delta^{2/5}d^{1/5}\rho_F^{-1/2}$,

$$Q = 2\log\left(\frac{\rho_F(F_0 - F^*)}{48L_F\delta(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10})}\right), \ and$$

$$T_{\text{th}} = \frac{L_F}{384(\rho_F^{1/2} + L_F)(\Delta^{2/5}d^{1/5} + \Delta^{3/5}d^{3/10})}.$$

Then, with probability at least $1 - \delta$, the output of Algorithm 1, denoted by $\widetilde{\mathbf{w}}$, satisfies the bounds

$$\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le 4\Delta,$$

$$\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \ge -1900(\rho_F^{1/2} + L_F)\Delta^{2/5}d^{1/5}\log\left(\frac{10}{\Delta}\right), \tag{3}$$

and the algorithm terminates within $\frac{2(F_0 - F^*)L_F}{3\Delta^2}Q$ parallel iterations.

We prove Theorem 3 in Appendix B.[1] Below let us parse the above theorem and discuss its implications.

Focusing on the scaling with Δ, we may read off from Theorem 3 the following result:

Observation 1. *Under the above setting, within $\widetilde{\mathcal{O}}(\frac{1}{\Delta^2})$ parallel iterations, ByzantinePGD outputs an $(\mathcal{O}(\Delta), \widetilde{\mathcal{O}}(\Delta^{2/5}))$-second-order stationary point $\widetilde{\mathbf{w}}$ of $F(\cdot)$;[2] that is,*

$$\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le 4\Delta \quad and \quad \lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \ge -\widetilde{\mathcal{O}}(\Delta^{2/5}).$$

In terms of the iteration complexity, it is well-known that for a smooth non-convex $F(\cdot)$, gradient descent requires at least $\frac{1}{\Delta^2}$ iterations to achieve $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le \mathcal{O}(\Delta)$ [53]; up to logarithmic factors, our result matches this complexity bound. In addition, our $\mathcal{O}(\Delta)$ first-order guarantee is clearly order-wise *optimal*, as the gradient oracle is Δ-inexact. It is currently unclear to us whether our $\widetilde{\mathcal{O}}(\Delta^{2/5})$ second-order guarantee is optimal. We provide a converse result showing that one cannot hope to achieve a second-order guarantee better than $\mathcal{O}(\Delta^{1/2})$.

Proposition 1. *There exists a class of real-valued 1-smooth and 1-Hessian Lipschitz differentiable functions $\mathcal{F}$ such that, for any algorithm that only uses a Δ-inexact gradient oracle, there exists $f \in \mathcal{F}$ such that the output of the algorithm $\widetilde{\mathbf{w}}$ must satisfy $\|\nabla F(\widetilde{\mathbf{w}})\|_2 > \Delta/2$ and $\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) < -\Delta^{1/2}/2$.*

We prove Proposition 1 in Appendix D. Again, we emphasize that our results above are in fact not restricted to the Byzantine distributed learning setting. They apply to any non-convex optimization problems (distributed or not) with inexact information for the gradients, including those with noisy but non-adversarial gradients; see Section 2 for comparison with related work in such settings.

As a byproduct, we can show that with a different choice of parameters, ByzantinePGD can be used in the standard (non-distributed) setting with access to the *exact* gradient $\nabla F(\mathbf{w})$, and the algorithm converges to an $(\epsilon, \widetilde{\mathcal{O}}(\sqrt{\epsilon}))$-second-order stationary point within $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations:

Theorem 4 (Exact gradient oracle). *Suppose that Assumptions 1 holds, and assume that for any query point $\mathbf{w}$ we can obtain exact gradient, i.e., $\widehat{\mathbf{g}}(\mathbf{w}) \equiv \nabla F(\mathbf{w})$. For any $\epsilon \in (0, \min\{\frac{1}{\rho_F}, \frac{4}{L_F^2\rho_F}\})$ and $\delta \in (0, 1)$, we choose the parameters in Algorithm 1 as follows: step-size $\eta = 1/L_F$, $Q = 1$, $r = \epsilon$, and $R = \sqrt{\epsilon/\rho_F}$, $T_{\text{th}} = \frac{L}{12\rho_F(R+r)}$. Then, with probability at least $1 - \delta$, Algorithm 1 outputs a $\widetilde{\mathbf{w}}$ satisfying the bounds*

$$\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le \epsilon,$$

$$\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \ge -60\sqrt{\rho_F\epsilon}\log\left(\frac{8\rho_F\sqrt{d}(F_0 - F^*)}{\delta\epsilon^2}\right),$$

and the algorithm terminates within $\frac{2L_F(F_0 - F^*)}{\epsilon^2}$ iterations.

We prove Theorem 4 in Appendix C. The convergence guarantee above matches that of the original PGD algorithm [32] up to logarithmic factors. Moreover, our proof is considerably simpler, and our algorithm only requires gradient information, whereas the original PGD algorithm also needs function values.

---

[1] We make no attempt in optimizing the multiplicative constants in Theorem 3.

[2] Here, by using the symbol $\widetilde{\mathcal{O}}$, we ignore logarithmic factors and only consider the dependence on Δ.

5 Robust Estimation of Gradients

The results in the previous section can be applied as long as one has a robust aggregation subroutine GradAGG that provides a Δ-inexact gradient oracle of the population loss F. In this section, we discuss three concrete examples of GradAGG: *median*, *trimmed mean*, and a high-dimension robust estimator based on the *iterative filtering* algorithm [22, 23, 60]. We characterize their inexactness Δ under the statistical setting in Section 3, where the data points are sampled independently according to an unknown distribution $\mathcal{D}$.

To describe our statistical results, we need the standard notions of sub-Gaussian/exponential random vectors.

Definition 4 (sub-Gaussianity and sub-exponentiality). *A random vector $\mathbf{x}$ with mean $\boldsymbol{\mu}$ is said to be ζ-sub-Gaussian if $\mathbb{E}[\exp(\lambda\langle\mathbf{x} - \boldsymbol{\mu}, \mathbf{u}\rangle)] \leq e^{\frac{1}{2}\zeta^2\lambda^2\|\mathbf{u}\|_2^2}, \forall \lambda, \mathbf{u}$. It is said to be ξ-sub-exponential if $\mathbb{E}[\exp(\lambda\langle\mathbf{x} - \boldsymbol{\mu}, \mathbf{u}\rangle)] \leq e^{\frac{1}{2}\xi^2\lambda^2\|\mathbf{u}\|_2^2}, \ \forall \ |\lambda| < \frac{1}{\xi}, \mathbf{u}$.*

We also need the following result (proved in Appendix E), which shows that the iterates of ByzantinePGD in fact stay in a bounded set around the initial iterate $\mathbf{w}_0$.

Proposition 2. *Under the choice of algorithm parameters in Theorem 3, all the iterates $\mathbf{w}$ in ByzantinePGD stay in the ℓ_2 ball $\mathbb{B}_{\mathbf{w}_0}(D/2)$ with $D := C\frac{F_0 - F^*}{\Delta}$, where $C > 0$ is a number that only depends on L_F and ρ_F.*

Consequently, for the convergence guarantees of ByzantinePGD to hold, we only need GradAGG to satisfy the inexact oracle property (Definition 3) within the bounded set $\mathcal{W} = \mathbb{B}_{\mathbf{w}_0}(D/2)$, with D given in Proposition 2. As shown below, the three aggregation procedures indeed satisfy this property, with their inexactness Δ depends mildly (logarithmically) on the radius D.

5.1 Iterative Filtering Algorithm

We start with a recently developed high-dimension robust estimation technique called the *iterative filtering* algorithm [22, 23, 60] and use it to build the subroutine GradAGG. As can be seen below, iterative filtering can tolerate a constant fraction of Byzantine machines even when the dimension grows—an advantage over simpler algorithms such as median and trimmed mean.

We relegate the details of the iterative filtering algorithm to Appendix F.1. Again, we emphasize that the original iterative filtering algorithm is proposed to robustly estimate a single parameter vector, whereas in our setting, since the Byzantine machines may produce unspecified probabilistic dependency across the iterations, we need to prove an error bound for robust gradient estimation uniformly across the parameter space $\mathcal{W}$. We prove such a bound for iterative filtering under the following two assumptions on the gradients and the smoothness of each loss function $f(\cdot; \mathbf{z})$.

Assumption 2. *For each $\mathbf{w} \in \mathcal{W}$, $\nabla f(\mathbf{w}; \mathbf{z})$ is ζ-sub-Gaussian.*

Assumption 3. *For each $\mathbf{z} \in \mathcal{Z}$, $f(\cdot; \mathbf{z})$ is L-smooth.*

Let $\boldsymbol{\Sigma}(\mathbf{w})$ be the covariance matrix of $\nabla f(\mathbf{w}; \mathbf{z})$, and define $\sigma := \sup_{\mathbf{w}\in\mathcal{W}} \|\boldsymbol{\Sigma}(\mathbf{w})\|_2^{1/2}$. We have the following bounds on the inexactness parameter of iterative filtering.

Theorem 5 (Iterative Filtering). *Suppose that Assumptions 2 and 3 hold. Use the iterative filtering algorithm described in Appendix F.1 for GradAGG, and assume that $\alpha \leq \frac{1}{4}$. With probability $1 - o(1)$, GradAGG provides a Δ_{ftr}-inexact gradient oracle with*

$$\Delta_{\mathsf{ftr}} \leq c\left((\sigma + \zeta)\sqrt{\frac{\alpha}{n}} + \zeta\sqrt{\frac{d}{nm}}\right)\sqrt{\log(nmDL)},$$

where c is an absolute constant.

The proof of Theorem 5 is given in Appendix F.2. Assuming bounded σ and ζ, we see that iterative filtering provides an $\widetilde{\mathcal{O}}(\sqrt{\frac{\alpha}{n}} + \sqrt{\frac{d}{nm}})$-inexact gradient oracle.

5.2 Median and Trimmed Mean

The median and trimmed mean operations are two widely used robust estimation methods. While the dependence of their performance on d is not optimal, they are conceptually simple and computationally fast, and still have good performance in low dimensional settings. We apply these operations in a coordinate-wise fashion to build GradAGG.

Formally, for a set of vectors $\mathbf{x}^i \in \mathbb{R}^d$, $i \in [m]$, their coordinate-wise median $\mathbf{u} := \mathsf{med}\{\mathbf{x}^i\}_{i=1}^m$ is a vector with its k-th coordinate being $u_k = \mathsf{med}\{x_k^i\}_{i=1}^m$ for each $k \in [d]$, where med is the usual (one-dimensional) median. The coordinate-wise β-trimmed mean $\mathbf{u} := \mathsf{trmean}_\beta\{\mathbf{x}^i\}_{i=1}^m$ is a vector with $u_k = \frac{1}{(1-2\beta)m} \sum_{x \in \mathcal{U}_k} x$ for each $k \in [d]$, where $\mathcal{U}_k$ is a subset of $\{x_k^1, \ldots, x_k^m\}$ obtained by removing the largest and smallest β fraction of its elements.

For robust estimation of the gradient in the Byzantine setting, the error bounds of median and trimmed mean have been studied by Yin et al. [69]. For completeness, we record their results below as an informal theorem; details are relegated to Appendix F.3.

Theorem 6 (Informal). *[69] Under appropriate smoothness and probabilistic assumptions,[3] with high probability, the median operation provides a Δ_{med}-inexact gradient oracle with $\Delta_{\mathsf{med}} \lesssim \frac{\alpha\sqrt{d}}{\sqrt{n}} + \frac{d}{\sqrt{nm}} + \frac{\sqrt{d}}{n}$, and the trimmed mean operation provides a Δ_{tm}-inexact gradient oracle with $\Delta_{\mathsf{tm}} \lesssim \frac{\alpha d}{\sqrt{n}} + \frac{d}{\sqrt{nm}}$.*

5.3 Comparison and Optimality

In Table 3, we compare the above three algorithms in terms of the dependence of their gradient inexactness Δ on the problem parameters α, n, m, and d. We see that when $d = \mathcal{O}(1)$, the median and trimmed mean algorithms have better inexactness due to a better scaling with α. When d is large, iterative filtering becomes preferable.

	Gradient inexactness Δ
median	$\widetilde{\mathcal{O}}\left(\frac{\alpha\sqrt{d}}{\sqrt{n}} + \frac{d}{\sqrt{nm}} + \frac{\sqrt{d}}{n}\right)$
trimmed mean	$\widetilde{\mathcal{O}}\left(\frac{\alpha d}{\sqrt{n}} + \frac{d}{\sqrt{nm}}\right)$
iterative filtering	$\widetilde{\mathcal{O}}\left(\frac{\sqrt{\alpha}}{\sqrt{n}} + \frac{\sqrt{d}}{\sqrt{nm}}\right)$

Table 3: Statistical bounds on gradient inexactness Δ.

Recall that according to Observation 1, with Δ-inexact gradients the ByzantinePGD algorithm converges to an $(\mathcal{O}(\Delta), \widetilde{\mathcal{O}}(\Delta^{2/5}))$-second-order stationary point. Combining this general result with the bounds in Table 3, we obtain explicit statistical guarantees on the output of ByzantinePGD. To understand the statistical optimality of these guarantees, we provide a converse result below.

Observation 2. *There exists a statistical learning problem in the Byzantine setting such that the output $\widetilde{\mathbf{w}}$ of any algorithm must satisfy $\|\nabla F(\widetilde{\mathbf{w}})\|_2 = \Omega\left(\frac{\alpha}{\sqrt{n}} + \frac{\sqrt{d}}{\sqrt{nm}}\right)$ with a constant probability.*

We prove Observation 2 in Appendix F.4. In view of this observation, we see that in terms of the first-order guarantee (i.e., on $\|\nabla F(\widetilde{\mathbf{w}})\|_2$) and up to logarithmic factors, trimmed mean is optimal if $d = \mathcal{O}(1)$, the median is optimal if $d = \mathcal{O}(1)$ and $n \gtrsim m$, and iterative filtering is optimal if $\alpha = \Theta(1)$. The statistical optimality of their second-order guarantees (i.e., on $\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}}))$) is currently unclear to us, and we believe this is an interesting problem for future investigation.

6 Conclusion

In this paper, we study security issues that arise in large-scale distributed learning because of the presence of saddle points in non-convex loss functions. We observe that in the presence of non-convexity and Byzantine machines, escaping saddle points becomes much more challenging. We develop ByzantinePGD, a computation- and communication-efficient algorithm that is able to provably escape saddle points and converge to a second-order stationary point, even in the

---

[3]Specifically, for median we assume that gradients have bounded skewness, and for trimmed mean we assume that the gradients are sub-exponentially distributed.

presence of Byzantine machines. We also discuss three different choices of the robust gradient and function value aggregation subroutines in ByzantinePGD—median, trimmed mean, and the iterative filtering algorithm. We characterize their performance in statistical settings, and argue for their near-optimality in different regimes including the high dimensional setting.

Acknowledgements

References

[1] Alekh Agarwal, Animashree Anandkumar, Prateek Jain, Praneeth Netrapalli, and Rashish Tandon. Learning sparsely used overcomplete dictionaries. In *COLT*, pages 123–137, 2014.

[2] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima for nonconvex optimization in linear time. *arXiv preprint arXiv:1611.01146*, 2016.

[3] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. *arXiv preprint arXiv:1803.08917*, 2018.

[4] Zeyuan Allen-Zhu. Natasha 2: Faster non-convex optimization than SGD. *arXiv preprint arXiv:1708.08694*, 2017.

[5] Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding local minima via first-order oracles. *arXiv preprint arXiv:1711.06673*, 2017.

[6] Sivaraman Balakrishnan, Martin J. Wainwright, and Bin Yu. Statistical guarantees for the EM algorithm: From population to sample-based analysis. *arXiv preprint arXiv:1408.2156*, 2014.

[7] Dimitri P. Bertsekas and John N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, 2000.

[8] Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding. In *Advances in Neural Information Processing Systems*, pages 721–729, 2015.

[9] Kush Bhatia, Prateek Jain, Parameswaran Kamalaruban, and Purushottam Kar. Consistent robust regression. In *Advances in Neural Information Processing Systems*, pages 2107–2116, 2017.

[10] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Byzantine-tolerant machine learning. *arXiv preprint arXiv:1703.02757*, 2017.

[11] Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval via Wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

[12] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Accelerated methods for non-convex optimization. *arXiv preprint arXiv:1611.00756*, 2016.

[13] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 47–60. ACM, 2017.

[14] Niladri Chatterji and Peter L Bartlett. Alternating minimization for dictionary learning with random initialization. In *Advances in Neural Information Processing Systems*, pages 1994–2003, 2017.

[15] Lingjiao Chen, Zachary Charles, Dimitris Papailiopoulos, et al. DRACO: Robust distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018.

[16] Yudong Chen and Martin J Wainwright. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*, 2015.

[17] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *arXiv preprint arXiv:1705.05491*, 2017.

[18] Yuxin Chen, Yuejie Chi, Jianqing Fan, and Cong Ma. Gradient descent with random initialization: Fast global convergence for nonconvex phase retrieval. *arXiv preprint arXiv:1803.07726*, 2018.

[19] Frank E Curtis, Daniel P Robinson, and Mohammadreza Samadi. A trust region algorithm with a worst-case iteration complexity of $\epsilon^{-3/2}$ for nonconvex optimization. *Mathematical Programming*, 162(1-2):1–32, 2017.

[20] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, and Mahsa Taziki. Asynchronous Byzantine machine learning. *arXiv preprint arXiv:1802.07928*, 2018.

[21] Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

[22] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Robust estimators in high dimensions without the computational intractability. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 655–664. IEEE, 2016.

[23] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. Being robust (in high dimensions) can be practical. *arXiv preprint arXiv:1703.00893*, 2017.

[24] Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018.

[25] Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. In *Advances in Neural Information Processing Systems*, pages 1067–1077, 2017.

[26] Jiashi Feng, Huan Xu, and Shie Mannor. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

[27] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *COLT*, pages 797–842, 2015.

[28] Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pages 2973–2981, 2016.

[29] Rong Ge, Chi Jin, and Yi Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. *arXiv preprint arXiv:1704.00708*, 2017.

[30] Peter J Huber. Robust statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer, 2011.

[31] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[32] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

[33] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*, 2017.

[34] Chi Jin, Lydia T Liu, Rong Ge, and Michael I Jordan. Minimizing nonconvex population risk from rough empirical risk. *arXiv preprint arXiv:1803.09357*, 2018.

[35] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

[36] Adam Klivans, Pravesh K Kothari, and Raghu Meka. Efficient algorithms for outlier-robust regression. *arXiv preprint arXiv:1803.03241*, 2018.

[37] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[38] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[39] Kevin A Lai, Anup B Rao, and Santosh Vempala. Agnostic estimation of mean and covariance. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 665–674. IEEE, 2016.

[40] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[41] Jason D Lee, Qihang Lin, Tengyu Ma, and Tianbao Yang. Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity. *arXiv preprint arXiv:1507.07595*, 2015.

[42] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

[43] Jason D Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I Jordan, and Benjamin Recht. First-order methods almost always avoid saddle points. *arXiv preprint arXiv:1710.07406*, 2017.

[44] Kfir Y Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

[45] Jerry Li. Robust sparse estimation tasks in high dimensions. *arXiv preprint arXiv:1702.05860*, 2017.

[46] Liu Liu, Yanyao Shen, Tianyang Li, and Constantine Caramanis. High dimensional robust sparse regression. *arXiv preprint arXiv:1805.11643*, 2018.

[47] Gabor Lugosi and Shahar Mendelson. Risk minimization by median-of-means tournaments. *arXiv preprint arXiv:1608.00757*, 2016.

[48] Nancy A. Lynch. *Distributed Algorithms*. Elsevier, 1996.

[49] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. https://research.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[50] Stanislav Minsker and Nate Strawn. Distributed statistical estimation and rates of convergence in normal approximation. *arXiv preprint arXiv:1704.02658*, 2017.

[51] Stanislav Minsker et al. Geometric median and robust estimation in banach spaces. *Bernoulli*, 21(4):2308–2335, 2015.

[52] Arkadii Nemirovskii, David Borisovich Yudin, and Edgar Ronald Dawson. *Problem complexity and method efficiency in optimization*. Wiley, 1983.

[53] Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 1998.

[54] Yurii Nesterov and Boris T Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[55] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

[56] Clément W Royer and Stephen J Wright. Complexity analysis of second-order line-search algorithms for smooth nonconvex optimization. *SIAM Journal on Optimization*, 28(2):1448–1477, 2018.

[57] Clément W Royer, Michael O'Neill, and Stephen J Wright. A newton-cg algorithm with complexity guarantees for smooth unconstrained optimization. *arXiv preprint arXiv:1803.02924*, 2018.

[58] Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International Conference on Machine Learning*, pages 1000–1008, 2014.

[59] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

[60] Jacob Steinhardt, Moses Charikar, and Gregory Valiant. Resilience: A criterion for learning in the presence of arbitrary outliers. *arXiv preprint arXiv:1703.04940*, 2017.

[61] Lili Su and Nitin H Vaidya. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 425–434. ACM, 2016.

[62] Lili Su and Nitin H Vaidya. Non-Bayesian learning in the presence of Byzantine agents. In *International Symposium on Distributed Computing*, pages 414–427. Springer, 2016.

[63] Lili Su and Jiaming Xu. Securing distributed machine learning in high dimensions. *arXiv preprint arXiv:1804.10140*, 2018.

[64] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery using nonconvex optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2351–2360, 2015.

[65] Stephen Tu, Ross Boczar, Max Simchowitz, Mahdi Soltanolkotabi, and Benjamin Recht. Low-rank solutions of linear matrix equations via procrustes flow. *arXiv preprint arXiv:1507.03566*, 2015.

[66] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[67] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized Byzantine-tolerant SGD. *arXiv preprint arXiv:1802.10116*, 2018.

[68] Yi Xu and Tianbao Yang. First-order stochastic algorithms for escaping from saddle points in almost linear time. *arXiv preprint arXiv:1711.01944*, 2017.

[69] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5650–5659, 2018.

[70] Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1998–2007, 2018.

[71] Huishuai Zhang, Yuejie Chi, and Yingbin Liang. Provable non-convex phase retrieval with outliers: Median-truncated wirtinger flow. In *International Conference on Machine Learning*, pages 1022–1031, 2016.

[72] Tuo Zhao, Zhaoran Wang, and Han Liu. A nonconvex optimization framework for low rank matrix estimation. In *Advances in Neural Information Processing Systems*, pages 559–567, 2015.

Appendix

A Challenges of Escaping Saddle Points in the Adversarial Setting

We provide two examples showing that in non-convex setting with saddle points, inexact oracle can lead to much worse sub-optimal solutions than in the convex setting, and that in the adversarial setting, escaping saddle points can be inherently harder than the adversary-free case.

Consider standard gradient descent using exact or Δ-inexact gradients. Our first example shows that Byzantine machines have a more severe impact in the non-convex case than in the convex case.

Example 1. Let $d = 1$ and consider the functions $F^{(1)}(w) = (w-1)^2$ and $F^{(2)}(w) = (w^2-1)^2/4$. Here $F^{(1)}$ is strongly convex with a unique local minimizer $w^* = 1$, whereas $F^{(2)}$ has two local (in fact, global) minimizers $w^* = \pm 1$ and a saddle point (in fact, a local maximum) $w = 0$. Proposition 3 below shows the following: for the convex $F^{(1)}$, gradient descent (GD) finds a near-optimal solution with sub-optimality proportional to Δ, regardless of initialization; for the nonconvex $F^{(2)}$, GD initialized near the saddle point $w = 0$ suffers from an $\Omega(1)$ sub-optimality gap.

Proposition 3. *Suppose that $\Delta \leq 1/2$. Under the setting above, the following holds.*
(i) For $F^{(1)}$, starting from any w_0, GD using a Δ-inexact gradient oracle finds w with $F^{(1)}(w) - F^{(1)}(w^*) \leq \mathcal{O}(\Delta)$.
(ii) For $F^{(2)}$, there exists an adversarial strategy such that starting from a w_0 sampled uniformly from $[-r, r]$, GD with a Δ-inexact gradient oracle outputs w with $F^{(2)}(w) - F^{(2)}(w^*) \geq \frac{9}{64}, \forall w^* = \pm 1$, with probability $\min\{1, \frac{\Delta}{r}\}$.

Proof. Since $F^{(2)}(w) = \frac{1}{4}(w^2 - 1)^2$, we have $\nabla F^{(2)}(w) = w^3 - w$. For any $w \in [-\Delta, \Delta]$, $|\nabla F^{(2)}(w)| \leq \Delta$ (since $\Delta \leq 1/2$). Thus, the adversarial oracle can always output $\widehat{g}(w) = 0$ when $w \in [-\Delta, \Delta]$, and we have $|\widehat{g}(w) - \nabla F^{(2)}(w)| \leq \Delta$. Thus, if $w \in [-\Delta, \Delta]$, the iterate can no longer move with this adversarial strategy. Then, we have $F^{(2)}(w) - F^{(2)}(w^*) \geq F^{(2)}(\Delta) - 0 \geq \frac{9}{64}$ (since $\Delta \leq 1/2$). The result for the convex function $F^{(1)}$ is a direct corollary of Theorem 1 in [69]. $\qquad\square$

Our second example shows that escaping saddle points is much harder in the Byzantine setting than in the non-Byzantine setting.

Example 2. Let $d = 2$, and assume that in the neighborhood $\mathbb{B}_0(b)$ of the origin, F takes the quadratic form $F(\mathbf{w}) \equiv \frac{1}{2}w_1^2 - \frac{\lambda}{2}w_2^2$, with $\lambda > \epsilon_H$.[4] The origin $\mathbf{w}_0 = 0$ is not an (ϵ_g, ϵ_H)-second-order stationary point, but rather a saddle point. Proposition 4 below shows that exact GD escapes the saddle point almost surely, while GD with an inexact oracle fails to do so.

Proposition 4. *Under the setting above, if one chooses $r < b$ and sample $\mathbf{w}$ from $\mathbb{B}_0(r)$ uniformly at random, then:*
(i) Using exact gradient descent, with probability 1, the iterate $\mathbf{w}$ eventually leaves $\mathbb{B}_0(r)$.
(ii) There exists an adversarial strategy such that, when we update $\mathbf{w}$ using Δ-inexact gradient oracle, if $\Delta \geq \lambda r$, with probability 1, the iterate $\mathbf{w}$ cannot leave $\mathbb{B}_0(r)$; otherwise with probability $\frac{2}{\pi}\left(\arcsin\left(\frac{\Delta}{\lambda r}\right) + \frac{\Delta}{\lambda r}\sqrt{1 - (\frac{\Delta}{\lambda r})^2}\right)$ the iterate $\mathbf{w}$ cannot leave $\mathbb{B}_0(r)$.

Proof. Since $F(\mathbf{w}) = \frac{1}{2}w_1^2 - \frac{1}{2}\lambda w_2^2$, $\forall \mathbf{w} \in \mathbb{B}_0(r)$, we have $\nabla F(\mathbf{w}) = [w_1, -\lambda w_2]^\top$. Sample $\mathbf{w}_0$ uniformly at random from $\mathbb{B}_0(r)$, and we know that with probability 1, $w_{0,2} \neq 0$. Then, by running exact gradient descent $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla F(\mathbf{w}_t)$, we can see that the second coordinate of $\mathbf{w}_t$ is $w_{t,2} = (1 + \eta\lambda)^t w_{0,2}$. When $w_{0,2}$, we know that as t gets large, we eventually have $w_{t,2} > r$, which implies that the iterate leaves $\mathbb{B}_0(r)$.

On the other hand, suppose that we run Δ-inexact gradient descent, i.e., $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\widehat{\mathbf{g}}(\mathbf{w}_t)$ with $\|\widehat{\mathbf{g}}(\mathbf{w}_t) - \nabla F(\mathbf{w}_t)\|_2 \leq \Delta$. In the first step, if $|w_{0,2}| \leq \frac{\Delta}{\lambda}$, the adversary can simply replace $\nabla F(\mathbf{w}_0)$ with $\widehat{\mathbf{g}}(\mathbf{w}_0) = [w_{0,1}, 0]^\top$ (one can check that here we have $\|\widehat{\mathbf{g}}(\mathbf{w}_0) - \nabla F(\mathbf{w}_0)\|_2 \leq \Delta$), and then the second coordinate of $\mathbf{w}_1$ does not change, i.e., $w_{1,2} = w_{0,2}$. In the following iterations, the

---

[4]$F(\mathbf{w}) \equiv \frac{1}{2}w_1^2 - \frac{\lambda}{2}w_2^2$ holds locally around the origin, not globally; otherwise $F(\mathbf{w})$ has no minimum.

adversary can keep using the same strategy and the second coordinate of $\mathbf{w}$ never changes, and then the iterates cannot escape $\mathbb{B}_0(r)$, since $F(\mathbf{w})$ is a strongly convex function in its first coordinate. To compute the probability of getting stuck at the saddle point, we only need to compute the area of the region $\{\mathbf{w} \in \mathbb{B}_0(r) : |w_2| \le \frac{\Delta}{\lambda}\}$, which can be done via simple geometry. $\qquad\square$

Remark. Even if we choose the largest possible perturbation in $\mathbb{B}_0(r)$, i.e., sample $\mathbf{w}$ from the circle $\{\mathbf{w} \in \mathbb{R}^2 : \|\mathbf{w}\|_2 = r\}$, the stuck region still exists. We can compute the length of the arc $\{\|\mathbf{w}\|_2 = r : |w_2| \le \frac{\Delta}{\lambda}\}$ and find the probability of stuck. One can find that when $\Delta \ge \lambda r$, the probability of being stuck in $\mathbb{B}_0(r)$ is still 1, otherwise, the probability of being stuck is $\frac{2}{\pi}(\arcsin(\frac{\Delta}{\lambda r}))$.

The above examples show that the adversary can significantly alter the landscape of the function near a saddle point. We counter this by exerting a large perturbation on the iterate so that it escapes this bad region. The amount of perturbation is carefully calibrated to ensure that the algorithm finds a descent direction "steep" enough to be preserved under Δ-corruption, while not compromising the accuracy. Multiple rounds of perturbation are performed, boosting the escape probability exponentially.

B Proof of Theorem 3

We first analyze the gradient descent step with Δ-inexact gradient oracle.

Lemma 1. *Suppose that $\eta = 1/L_F$. For any $\mathbf{w} \in \mathcal{W}$, if we run the following inexact gradient descent step:*

$$\mathbf{w}' = \mathbf{w} - \eta \widehat{\mathbf{g}}(\mathbf{w}), \tag{4}$$

with $\|\widehat{\mathbf{g}}(\mathbf{w}) - \nabla F(\mathbf{w})\|_2 \le \Delta$. Then, we have

$$F(\mathbf{w}') \le F(\mathbf{w}) - \frac{1}{2L_F}\|\nabla F(\mathbf{w})\|_2^2 + \frac{1}{2L_F}\Delta^2.$$

Proof. Since $F(\mathbf{w})$ is L_F smooth, we know that

$$
\begin{aligned}
F(\mathbf{w}') \le\; & F(\mathbf{w}) + \langle \nabla F(\mathbf{w}), \mathbf{w}' - \mathbf{w}\rangle + \frac{L_F}{2}\|\mathbf{w}' - \mathbf{w}\|_2^2 \\
=\; & F(\mathbf{w}) - \langle \nabla F(\mathbf{w}), \frac{1}{L_F}(\widehat{\mathbf{g}}(\mathbf{w}) - \nabla F(\mathbf{w}))\rangle - \langle \nabla F(\mathbf{w}), \frac{1}{L_F}\nabla F(\mathbf{w})\rangle \\
& + \frac{1}{2L_F}\|\widehat{\mathbf{g}}(\mathbf{w}) - \nabla F(\mathbf{w}) + \nabla F(\mathbf{w})\|_2^2 \\
\le\; & F(\mathbf{w}) - \frac{1}{2L_F}\|\nabla F(\mathbf{w})\|_2^2 + \frac{1}{2L_F}\Delta^2.
\end{aligned}
$$

$\qquad\square$

Let ϵ be the threshold on $\|\widehat{\mathbf{g}}(\widetilde{\mathbf{w}})\|_2$ that the algorithm uses to determine whether or not to add perturbation. Choose $\epsilon = 3\Delta$. Suppose that at a particular iterate $\widetilde{\mathbf{w}}$, we observe $\|\widehat{\mathbf{g}}(\widetilde{\mathbf{w}})\|_2 > \epsilon$. Then, we know that

$$\|\nabla F(\widetilde{\mathbf{w}})\|_2 \ge \|\widehat{\mathbf{g}}(\widetilde{\mathbf{w}})\|_2 - \Delta \ge 2\Delta.$$

According to Lemma 1, by running one iteration of the inexact gradient descent step, the decrease in function value is at least

$$\frac{1}{2L_F}\|\nabla F(\widetilde{\mathbf{w}})\|_2^2 - \frac{1}{2L_F}\Delta^2 \ge \frac{3\Delta^2}{2L_F}. \tag{5}$$

We proceed to analyze the perturbation step, which happens when the algorithm arrives at an iterate $\widetilde{\mathbf{w}}$ with $\|\widehat{\mathbf{g}}(\widetilde{\mathbf{w}})\|_2 \le \epsilon$. In this proof, we slightly abuse the notation. Recall that in equation (2) in Section 4.1 , we use $\mathbf{w}'_t$ ($0 \le t \le T_{\mathrm{th}}$) to denote the iterates of the algorithm in the saddle point escaping process. Here, we simply use $\mathbf{w}_t$ to denote these iterates. We start with the definition of *stuck region* at $\widetilde{\mathbf{w}} \in \mathcal{W}$.

Definition 5. *Given $\widetilde{\mathbf{w}} \in \mathcal{W}$, and parameters r, R, and T_{th}, the stuck region $\mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}}) \subseteq \mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ is a set of $\mathbf{w}_0 \in \mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ which satisfies the following property: there exists an adversarial strategy such that when we start with $\mathbf{w}_0$ and run T_{th} gradient descent steps with Δ-inexact gradient oracle $\widehat{\mathbf{g}}(\mathbf{w})$:*

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \widehat{\mathbf{g}}(\mathbf{w}_{t-1}), \ t = 1, 2, \ldots, T_{\text{th}}, \tag{6}$$

we observe $\|\mathbf{w}_t - \mathbf{w}_0\|_2 < R, \ \forall \ t \leq T_{\text{th}}$.

When it is clear from the context, we may simply use the terminology stuck region $\mathbb{W}_S$ at $\widetilde{\mathbf{w}}$. The following lemma shows that if $\nabla^2 F(\widetilde{\mathbf{w}})$ has a large negative eigenvalue, then the stuck region has a small width along the direction of the eigenvector associated with this negative eigenvalue.

Lemma 2. *Assume that the smallest eigenvalue of $\mathbf{H} := \nabla^2 F(\widetilde{\mathbf{w}})$ satisfies $\lambda_{\min}(\mathbf{H}) \leq -\gamma < 0$, and let the unit vector $\mathbf{e}$ be the eigenvector associated with $\lambda_{\min}(\mathbf{H})$. Let $\mathbf{u}_0, \mathbf{y}_0 \in \mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ be two points such that $\mathbf{y}_0 = \mathbf{u}_0 + \mu_0 \mathbf{e}$ with some $\mu_0 \geq \mu \in (0, r)$. Choose step size $\eta = \frac{1}{L_F}$, and consider the stuck region $\mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}})$. Suppose that r, R, T_{th}, and μ satisfy [5]*

$$T_{\text{th}} = \frac{2}{\eta\gamma} \log_{9/4}\left(\frac{2(R+r)}{\mu}\right), \tag{7}$$

$$R \geq \mu, \tag{8}$$

$$\rho_F(R+r)\mu \geq \Delta, \tag{9}$$

$$\gamma \geq 24\rho_F(R+r) \log_{9/4}\left(\frac{2(R+r)}{\mu}\right). \tag{10}$$

Then, there must be either $\mathbf{u}_0 \notin \mathbb{W}_S$ or $\mathbf{y}_0 \notin \mathbb{W}_S$.

We prove Lemma 2 in Appendix B.1. With this lemma, we proceed to analyze the probability that the algorithm escapes the saddle points. In particular, we bound the probability that $\mathbf{w}_0 \in \mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}})$ when $\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \leq -\gamma$ and $\mathbf{w}_0$ is drawn from $\mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ uniform at random.

Lemma 3. *Assume that $\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \leq -\gamma < 0$, and let the unit vector $\mathbf{e}$ be the eigenvector associated with $\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}}))$. Consider the stuck region $\mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}})$ at $\widetilde{\mathbf{w}}$, and suppose that r, R, T_{th}, and μ satisfy the conditions in (7)-(10). Then, when we sample $\mathbf{w}_0$ from $\mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ uniformly at random, the probability that $\mathbf{w}_0 \in \mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}})$ is at most $\frac{2\mu\sqrt{d}}{r}$.*

Proof. Since the starting point $\mathbf{w}_0$ is uniformly distributed in $\mathbb{B}_{\widetilde{\mathbf{w}}}(r)$, to bound the probability of getting stuck, it suffices to bound the volume of $\mathbb{W}_S$. Let $\mathbb{1}_{\mathbb{W}_S}(\mathbf{w})$ be the indicator function of the set $\mathbb{W}_S$. For any $\mathbf{w} \in \mathbb{R}^d$, let $w^{(1)}$ be the projection of $\mathbf{w}$ onto the $\mathbf{e}$ direction, and $\mathbf{w}^{(-1)} \in \mathbb{R}^{d-1}$ be the remaining component of $\mathbf{w}$. Then, we have

$$
\begin{aligned}
\text{Vol}(\mathbb{W}_S) &= \int_{\mathbb{B}_{\widetilde{\mathbf{w}}}^{(d)}(r)} \mathbb{1}_{\mathbb{W}_S}(\mathbf{w}) \mathrm{d}\mathbf{w} \\
&= \int_{\mathbb{B}_{\widetilde{\mathbf{w}}}^{(d-1)}(r)} \mathrm{d}\mathbf{w}^{(-1)} \int_{\widetilde{w}^{(1)} - \sqrt{r^2 - \|\widetilde{\mathbf{w}}^{(-1)} - \mathbf{w}^{(-1)}\|_2^2}}^{\widetilde{w}^{(1)} + \sqrt{r^2 - \|\widetilde{\mathbf{w}}^{(-1)} - \mathbf{w}^{(-1)}\|_2^2}} \mathbb{1}_{\mathbb{W}_S}(\mathbf{w}) \mathrm{d}\widetilde{w}^{(1)} \\
&\leq 2\mu \int_{\mathbb{B}_{\widetilde{\mathbf{w}}}^{(d-1)}(r)} \mathrm{d}\mathbf{w}^{(-1)} \\
&= 2\mu \text{Vol}(\mathbb{B}_0^{(d-1)}(r)),
\end{aligned}
$$

where the inequality is due to Lemma 2. Then, we know that the probability of getting stuck is

$$\frac{\text{Vol}(\mathbb{W}_S)}{\text{Vol}(\mathbb{B}_0^{(d)}(r))} \leq 2\mu \frac{\text{Vol}(\mathbb{B}_0^{(d-1)}(r))}{\text{Vol}(\mathbb{B}_0^{(d)}(r))} = \frac{2\mu}{\sqrt{\pi}r} \frac{\Gamma(\frac{d}{2}+1)}{\Gamma(\frac{d}{2}+\frac{1}{2})} \leq \frac{2\mu}{\sqrt{\pi}r} \sqrt{\frac{d}{2}+\frac{1}{2}} \leq \frac{2\mu\sqrt{d}}{r},$$

where we use the fact that $\frac{\Gamma(x+1)}{\Gamma(x+\frac{1}{2})} < \sqrt{x + \frac{1}{2}}$ for any $x \geq 0$. $\qquad\square$

We then analyze the decrease of value of the population loss function $F(\cdot)$ when we conduct the perturbation step. Assume that we successfully escape the saddle point, i.e., there exists $t \leq T_{\text{th}}$ such that $\|\mathbf{w}_t - \mathbf{w}_0\|_2 \geq R$. The following lemma provides the decrease of $F(\cdot)$.

---

[5]Without loss of generality, here we assume that $\frac{2}{\eta\gamma} \log_{9/4}(\frac{2(R+r)}{\mu})$ is an integer, so that T_{th} is an integer.

Lemma 4. *Suppose that $\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \leq -\gamma < 0$, and at $\widetilde{\mathbf{w}}$, we observe $\|\widehat{\mathbf{g}}(\widetilde{\mathbf{w}})\|_2 \leq \epsilon = 3\Delta$. Assume that $\mathbf{w}_0 \in \mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ and that $\mathbf{w}_0 \notin \mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\mathrm{th}})$. Let $t \leq T_{\mathrm{th}}$ be the step such that $\|\mathbf{w}_t - \mathbf{w}_0\|_2 \geq R$. Then, we have*

$$F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t) \geq \frac{L_F}{4T_{\mathrm{th}}}R^2 - \frac{\Delta^2 T_{\mathrm{th}}}{L_F} - 4\Delta r - \frac{L_F}{2}r^2. \tag{11}$$

We prove Lemma 4 in Appendix B.2.

Next, we choose the quantities μ, r, R, and γ such that (i) the conditions (7)-(10) in Lemma 2 are satisfied, (ii) the probability of escaping saddle point in Lemma 3 is at least a constant, and (iii) the decrease in function value in (11) is large enough. We first choose

$$\mu = \Delta^{3/5} d^{-1/5} \rho_F^{-1/2}, \tag{12}$$

$$r = 4\Delta^{3/5} d^{3/10} \rho_F^{-1/2}, \tag{13}$$

$$R = \Delta^{2/5} d^{1/5} \rho_F^{-1/2}. \tag{14}$$

One can simply check that, according to Lemma 3, when we drawn $\mathbf{w}_0$ from $\mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ uniformly at random, the probability that $\mathbf{w}_0 \in \mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\mathrm{th}})$ is at most $1/2$. Since we assume that $\Delta \leq 1$, one can also check that the condition (8) is satisfied. In addition, since $\rho_F R \mu = \Delta$, the condition (9) is also satisfied. According to (7), we have

$$T_{\mathrm{th}} = \frac{2L_F}{\gamma} \log_{9/4}\left(\frac{2d^{2/5}}{\Delta^{1/5}} + 8d^{1/2}\right). \tag{15}$$

In the following, we choose

$$\gamma = 768(\rho_F^{1/2} + L_F)(\Delta^{2/5}d^{1/5} + \Delta^{3/5}d^{3/10}) \log_{9/4}\left(\frac{2d^{2/5}}{\Delta^{1/5}} + 8d^{1/2}\right), \tag{16}$$

which implies

$$T_{\mathrm{th}} = \frac{L_F}{384(\rho_F^{1/2} + L_F)(\Delta^{2/5}d^{1/5} + \Delta^{3/5}d^{3/10})} \tag{17}$$

Then we check condition (10) holds. We have

$$24\rho_F(R + r) \log_{9/4}\left(\frac{2(R + r)}{\mu}\right) = 24\rho_F^{1/2}(\Delta^{2/5}d^{1/5} + 4\Delta^{3/5}d^{3/10}) \log_{9/4}\left(\frac{2d^{2/5}}{\Delta^{1/5}} + 8d^{1/2}\right) \leq \gamma.$$

Next, we consider the decrease in function value in (11). Using the equations (15) and (16), we can show the following three inequalities by direct algebra manipulation.

$$\frac{L_F}{4T_{\mathrm{th}}}R^2 \geq 6\frac{\Delta^2 T_{\mathrm{th}}}{L_F}, \tag{18}$$

$$\frac{L_F}{4T_{\mathrm{th}}}R^2 \geq 24\Delta r, \tag{19}$$

$$\frac{L_F}{4T_{\mathrm{th}}}R^2 \geq 3L_F r^2. \tag{20}$$

By adding up (18), (19), and (20), we obtain

$$\frac{L_F}{4T_{\mathrm{th}}}R^2 \geq 2\frac{\Delta^2 T_{\mathrm{th}}}{L_F} + 8\Delta r + L_F r^2,$$

which implies that when we successfully escape the saddle point, we have

$$F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t) \geq \frac{L_F}{8T_{\mathrm{th}}}R^2 = 48(\rho_F^{-1/2} + L_F\rho_F^{-1})(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10}). \tag{21}$$

Then, one can simply check that, the average decrease in function value during the successful round of the Escape process is

$$\frac{F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t)}{t} \geq \frac{F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t)}{T_{\mathrm{th}}} \geq \frac{2(\Delta^{8/5}d^{4/5} + \Delta^2 d)}{L_F} > \frac{3\Delta^2}{2L_F}. \tag{22}$$

Recall that according to (5), when the algorithm is not in the Escape process, the function value is decreased by at least $\frac{3\Delta^2}{2L_F}$ in each iteration. Therefore, if the algorithm successfully escapes the saddle point, during the Escape process, the average decrease in function value is *larger* than the iterations which are not in this process.

So far, we have chosen the algorithm parameters r, R, T_{th}, as well as the final second-order convergence guarantee γ. Now we proceed to analyze the total number of iterations and the failure probability of the algorithm. According to Lemma 3 and the choice of μ and r, we know that at each point with $\|\widehat{\mathbf{g}}(\widetilde{\mathbf{w}})\|_2 \leq \epsilon$, the algorithm can successfully escape this saddle point with probability at least $1/2$. To boost the probability of escaping saddle points, we need to repeat the process Q rounds in Escape, independently. Since for each successful round, the function value decrease is at least

$$48(\rho_F^{-1/2} + L_F\rho_F^{-1})(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10}) \geq 48L_F\rho_F^{-1}(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10}),$$

and the function value can decrease at most $F_0 - F^*$. Therefore, the total number of saddle points that we need to escape is at most

$$\frac{\rho_F(F_0 - F^*)}{48L_F(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10})}. \tag{23}$$

Therefore, by union bound, the failure probability of the algorithm is at most

$$\frac{\rho_F(F_0 - F^*)}{48L_F(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10})}\left(\frac{1}{2}\right)^Q,$$

and to make the failure probability at most δ, one can choose

$$Q \geq 2\log\left(\frac{\rho_F(F_0 - F^*)}{48L_F\delta(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10})}\right). \tag{24}$$

Again, due to the fact that the function value decrease is at most $F_0 - F^*$, and in each *effective* iteration, the function value is decreased by at least $\frac{3\Delta^2}{2L_F}$. (Here, the effective iterations are the iterations when the algorithm is not in the Escape process and the iterations when the algorithm successfully escapes the saddle points.) The total number of effective iterations is at most

$$\frac{2(F_0 - F^*)L_F}{3\Delta^2}. \tag{25}$$

Combing with (24), we know that the total number of parallel iterations is at most

$$\frac{4(F_0 - F^*)L_F}{3\Delta^2}\log\left(\frac{\rho_F(F_0 - F^*)}{48L_F\delta(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10})}\right).$$

When all the algorithm terminates, and the saddle point escaping process is successful, the output of the algorithm $\widetilde{\mathbf{w}}$ satisfies $\|\widehat{\mathbf{g}}(\widetilde{\mathbf{w}})\|_2 \leq \epsilon$, which implies that $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \leq 4\Delta$, and

$$\begin{aligned}
\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) &\geq -\gamma = -768(\rho_F^{1/2} + L_F)(\Delta^{2/5}d^{1/5} + \Delta^{3/5}d^{3/10})\log_{9/4}\left(\frac{2d^{2/5}}{\Delta^{1/5}} + 8d^{1/2}\right) \\
&\geq -950(\rho_F^{1/2} + L_F)(\Delta^{2/5}d^{1/5} + \Delta^{3/5}d^{3/10})\log\left(\frac{2d^{2/5}}{\Delta^{1/5}} + 8d^{1/2}\right).
\end{aligned} \tag{26}$$

We next show that we can simplify the guarantee as

$$\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \geq -1900(\rho_F^{1/2} + L_F)\Delta^{2/5}d^{1/5}\log\left(\frac{10}{\Delta}\right). \tag{27}$$

We can see that if $\Delta \leq \frac{1}{\sqrt{d}}$, then $\Delta^{3/5}d^{3/10} \leq \Delta^{2/5}d^{1/5}$ and $\frac{2d^{2/5}}{\Delta^{1/5}} + 8d^{1/2} \leq \frac{10}{\Delta}$. Thus, the bound (27) holds. On the other hand, when $\Delta > \frac{1}{\sqrt{d}}$, we have $\Delta^{2/5}d^{1/5} > 1$ and thus

$$1900(\rho_F^{1/2} + L_F)\Delta^{2/5}d^{1/5}\log\left(\frac{10}{\Delta}\right) > L_F.$$

By the smoothness of $F(\cdot)$, we know that $\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \geq -L_F$. Therefore, the bound (27) still holds, and this completes the proof.

B.1 Proof of Lemma 2

We prove by contradiction. Suppose that $\mathbf{u}_0, \mathbf{y}_0 \in \mathbb{W}_S$. Let $\{\mathbf{u}_t\}$ and $\{\mathbf{y}_t\}$ be two sequences generated by the following two iterations:

$$\mathbf{u}_t = \mathbf{u}_{t-1} - \eta \widehat{\mathbf{g}}(\mathbf{u}_{t-1}), \tag{28}$$

$$\mathbf{y}_t = \mathbf{y}_{t-1} - \eta \widehat{\mathbf{g}}(\mathbf{y}_{t-1}), \tag{29}$$

respectively, where $\|\widehat{\mathbf{g}}(\mathbf{w}) - \nabla F(\mathbf{w})\|_2 \leq \Delta$ for any $\mathbf{w} \in \mathcal{W}$. According to our assumption, we have $\forall\, t \leq T_{\text{th}}$, $\|\mathbf{u}_t - \mathbf{u}_0\|_2 < R$ and $\|\mathbf{y}_t - \mathbf{y}_0\|_2 < R$.

Define $\mathbf{v}_t := \mathbf{y}_t - \mathbf{u}_t$, $\boldsymbol{\delta}_t := \widehat{\mathbf{g}}(\mathbf{u}_t) - \nabla F(\mathbf{u}_t)$, and $\boldsymbol{\delta}_t' := \widehat{\mathbf{g}}(\mathbf{y}_t) - \nabla F(\mathbf{y}_t)$. Then we have

$$
\begin{aligned}
\mathbf{y}_{t+1} &= \mathbf{y}_t - \eta(\nabla F(\mathbf{y}_t) + \boldsymbol{\delta}_t') \\
&= \mathbf{u}_t + \mathbf{v}_t - \eta(\nabla F(\mathbf{u}_t + \mathbf{v}_t) + \boldsymbol{\delta}_t') \\
&= \mathbf{u}_t + \mathbf{v}_t - \eta \nabla F(\mathbf{u}_t) - \eta \left[\int_0^1 \nabla^2 F(\mathbf{u}_t + \theta \mathbf{v}_t) \right] \mathbf{v}_t - \eta \boldsymbol{\delta}_t' \\
&= \mathbf{u}_{t+1} + \eta \boldsymbol{\delta}_t + \mathbf{v}_t - \eta \left[\int_0^1 \nabla^2 F(\mathbf{u}_t + \theta \mathbf{v}_t) \mathrm{d}\theta \right] \mathbf{v}_t - \eta \boldsymbol{\delta}_t',
\end{aligned}
$$

which yields

$$\mathbf{v}_{t+1} = (\mathbf{I} - \eta \mathbf{H})\mathbf{v}_t - \eta \mathbf{Q}_t \mathbf{v}_t + \eta(\boldsymbol{\delta}_t - \boldsymbol{\delta}_t'), \tag{30}$$

where

$$\mathbf{Q}_t := \int_0^1 \nabla^2 F(\mathbf{u}_t + \theta \mathbf{v}_t) \mathrm{d}\theta - \mathbf{H}. \tag{31}$$

By the Hessian Lipschitz property, we know that

$$
\begin{aligned}
\|\mathbf{Q}_t\|_2 &\leq \rho_F(\|\mathbf{u}_t - \widetilde{\mathbf{w}}\|_2 + \|\mathbf{y}_t - \widetilde{\mathbf{w}}\|_2) \\
&\leq \rho_F(\|\mathbf{u}_t - \mathbf{u}_0\|_2 + \|\mathbf{u}_0 - \widetilde{\mathbf{w}}\|_2 + \|\mathbf{y}_t - \mathbf{y}_0\|_2 + \|\mathbf{y}_0 - \widetilde{\mathbf{w}}\|_2) \\
&\leq 2\rho_F(R + r).
\end{aligned}
\tag{32}
$$

We let ψ_t be the norm of the projection of $\mathbf{v}_t$ onto the $\mathbf{e}$ direction, and ϕ_t be the norm of the projection of $\mathbf{v}_t$ onto the remaining subspace. By definition, we have $\psi_0 = \mu_0 \geq \mu > 0$ and $\phi_0 = 0$. According to (30) and (32), we have

$$\psi_{t+1} \geq (1 + \eta\gamma)\psi_t - 2\eta\rho_F(R+r)\sqrt{\psi_t^2 + \phi_t^2} - 2\eta\Delta, \tag{33}$$

$$\phi_{t+1} \leq (1 + \eta\gamma)\phi_t + 2\eta\rho_F(R+r)\sqrt{\psi_t^2 + \phi_t^2} + 2\eta\Delta. \tag{34}$$

In the following, we use induction to prove that $\forall\, t \leq T_{\text{th}}$,

$$\psi_t \geq (1 + \frac{1}{2}\eta\gamma)\psi_{t-1} \quad \text{and} \quad \phi_t \leq \frac{t}{T_{\text{th}}}\psi_t \tag{35}$$

We know that (35) holds when $t = 0$ since we have $\phi_0 = 0$. Then, assume that for some $t < T_{\text{th}}$, we have $\forall\, \tau \leq t$, $\psi_\tau \geq (1 + \frac{1}{2}\eta\gamma)\psi_{\tau-1}$ and $\phi_\tau \leq \frac{\tau}{T_{\text{th}}}\psi_\tau$. We show that (35) holds for $t + 1$.

First, we show that $\psi_{t+1} \geq (1 + \frac{1}{2}\eta\gamma)\psi_t$. Since $\forall\, \tau \leq t$, $\psi_\tau \geq \psi_{\tau-1}$, we know that $\psi_t \geq \psi_0 \geq \mu$. Therefore, according to (9), we have

$$\Delta \leq \rho_F(R+r)\mu \leq \rho_F(R+r)\psi_t. \tag{36}$$

In addition, since $t < T_{\text{th}}$, we have

$$\phi_t \leq \psi_t. \tag{37}$$

Combining (36), (37) and (33), (34), we get

$$\psi_{t+1} \geq (1 + \eta\gamma)\psi_t - 2\eta\rho_F(R+r)\sqrt{2\psi_t^2} - 2\eta\rho_F(R+r)\psi_t > (1 + \eta\gamma)\psi_t - 6\eta\rho_F(R+r)\psi_t, \tag{38}$$

$$\phi_{t+1} \leq (1 + \eta\gamma)\phi_t + 2\eta\rho_F(R+r)\sqrt{2\psi_t^2} + 2\eta\rho_F(R+r)\psi_t < (1 + \eta\gamma)\phi_t + 6\eta\rho_F(R+r)\psi_t. \tag{39}$$

According to (10), we have $\gamma \geq 24\rho_F(R+r)\log_{9/4}(\frac{2(R+r)}{\mu}) > 12\rho_F(R+r)$. Combining with (38), we know that $\psi_{t+1} \geq (1 + \frac{1}{2}\eta\gamma)\psi_t$.

Next, we show that $\phi_{t+1} \leq \frac{t+1}{T_{\text{th}}}\psi_{t+1}$. Combining with (38) and (39), we know that to show $\phi_{t+1} \leq \frac{t+1}{T_{\text{th}}}\psi_{t+1}$, it suffices to show

$$(1+\eta\gamma)\phi_t + 6\eta\rho_F(R+r)\psi_t \leq \frac{t+1}{T_{\text{th}}}[1 + \eta\gamma - 6\eta\rho_F(R+r)]\psi_t. \tag{40}$$

According to the induction assumption, we have $\phi_t \leq \frac{t}{T_{\text{th}}}\psi_t$. Then, to show (40), it suffices to show that

$$(1+\eta\gamma)t + 6\eta\rho_F(R+r)T_{\text{th}} \leq (t+1)[1 + \eta\gamma - 6\eta\rho_F(R+r)] \tag{41}$$

Since $t+1 \leq T_{\text{th}}$, we know that to show (41), it suffices to show

$$12\eta\rho_F(R+r)T_{\text{th}} \leq 1. \tag{42}$$

Then, according to (7) and (10), we know that (42) holds, which completes the induction.

Next, according to (35), we know that

$$\|\mathbf{u}_{T_{\text{th}}} - \mathbf{y}_{T_{\text{th}}}\|_2 \geq \phi_{T_{\text{th}}} \geq (1 + \frac{1}{2}\eta\gamma)^{T_{\text{th}}}\mu_0$$

$$\geq (1 + \frac{1}{2}\eta\gamma)^{\frac{2}{\eta\gamma}\log_{9/4}(\frac{2(R+r)}{\mu})}\mu_0$$

$$\geq \frac{2(R+r)}{\mu} \cdot \mu_0 = 2(R+r),$$

where the last inequality is due to the fact that $\eta = \frac{1}{L_F}$ and thus $\eta\gamma \leq 1$. On the other hand, since we assume that $\mathbf{u}_0, \mathbf{y}_0 \in \mathbb{W}_S$, we know that

$$\|\mathbf{u}_{T_{\text{th}}} - \mathbf{y}_{T_{\text{th}}}\|_2 \leq \|\mathbf{u}_{T_{\text{th}}} - \mathbf{u}_0\|_2 + \|\mathbf{y}_{T_{\text{th}}} - \mathbf{y}_0\|_2 + \|\mathbf{u}_0 - \mathbf{y}_0\|_2 < 2(R+r),$$

which leads to contradiction and thus completes the proof.

B.2 Proof of Lemma 4

Recall that we have the iterations $\mathbf{w}_{\tau+1} = \mathbf{w}_\tau - \eta\widehat{\mathbf{g}}(\mathbf{w}_\tau)$ for all $\tau < t$. Let $\boldsymbol{\delta}_\tau = \nabla F(\mathbf{w}_\tau) - \widehat{\mathbf{g}}(\mathbf{w}_\tau)$, and then $\|\boldsymbol{\delta}_\tau\|_2 \leq \Delta$. By the smoothness of $F(\cdot)$ and the fact that $\eta = \frac{1}{L_F}$, we have

$$\begin{aligned} F(\mathbf{w}_\tau) - F(\mathbf{w}_{\tau+1}) &\geq \langle \nabla F(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_{\tau+1}\rangle - \frac{L_F}{2}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|_2^2 \\ &= \left\langle \frac{\mathbf{w}_\tau - \mathbf{w}_{\tau+1}}{\eta} + \boldsymbol{\delta}_\tau, \mathbf{w}_\tau - \mathbf{w}_{\tau+1}\right\rangle - \frac{L_F}{2}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|_2^2 \\ &= \frac{L_F}{2}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|_2^2 + \langle \boldsymbol{\delta}_\tau, \mathbf{w}_\tau - \mathbf{w}_{\tau+1}\rangle \\ &\geq \frac{L_F}{4}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|_2^2 - \frac{\|\boldsymbol{\delta}_\tau\|_2^2}{L_F} \\ &\geq \frac{L_F}{4}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|_2^2 - \frac{\Delta^2}{L_F}. \end{aligned} \tag{43}$$

By summing up (43) for $\tau = 0, 1, \ldots, t-1$, we get

$$F(\mathbf{w}_0) - F(\mathbf{w}_t) \geq \frac{L_F}{4}\sum_{\tau=0}^{t-1}\|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|_2^2 - \frac{\Delta^2 t}{L_F}. \tag{44}$$

Consider the k-th coordinate of $\mathbf{w}_\tau$ and $\mathbf{w}_{\tau+1}$, by Cauchy-Schwarz inequality, we have

$$\sum_{\tau=0}^{t-1}(w_{\tau,k} - w_{\tau+1,k})^2 \geq \frac{1}{t}(w_{0,k} - w_{t,k})^2,$$

which implies

$$\sum_{\tau=0}^{t-1} \|\mathbf{w}_\tau - \mathbf{w}_{\tau+1}\|_2^2 \geq \frac{1}{t}\|\mathbf{w}_0 - \mathbf{w}_t\|_2^2. \tag{45}$$

Combining (44) and (45), we obtain

$$F(\mathbf{w}_0) - F(\mathbf{w}_t) \geq \frac{L_F}{4t}\|\mathbf{w}_0 - \mathbf{w}_t\|_2^2 - \frac{\Delta^2 t}{L_F} \geq \frac{L_F}{4T_{\text{th}}}R^2 - \frac{\Delta^2 T_{\text{th}}}{L_F}. \tag{46}$$

On the other hand, by the smoothness of $F(\cdot)$, we have

$$F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_0) \geq \langle \nabla F(\widetilde{\mathbf{w}}), \widetilde{\mathbf{w}} - \mathbf{w}_0 \rangle - \frac{L_F}{2}\|\mathbf{w}_0 - \widetilde{\mathbf{w}}\|_2^2 \geq -(\epsilon + \Delta)r - \frac{L_F}{2}r^2. \tag{47}$$

Adding up (46) and (47), we obtain

$$F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t) \geq \frac{L_F}{4T_{\text{th}}}R^2 - \frac{\Delta^2 T_{\text{th}}}{L_F} - (\epsilon + \Delta)r - \frac{L_F}{2}r^2, \tag{48}$$

which completes the proof.

C Proof of Theorem 4

First, when we run gradient descent iterations $\mathbf{w}' = \mathbf{w} - \eta \nabla F(\mathbf{w})$, according to Lemma 1, we have

$$F(\mathbf{w}') \leq F(\mathbf{w}) - \frac{1}{2L_F}\|\nabla F(\mathbf{w})\|_2^2. \tag{49}$$

Suppose at $\widetilde{\mathbf{w}}$, we observe that $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \leq \epsilon$, and then we start the Escape process. When we have exact gradient oracle, we can still define the stuck region $\mathbb{W}_S$ at $\widetilde{\mathbf{w}}$ as in the definition of stuck region in Appendix B, by simply replacing the inexact gradient oracle with the exact oracle. Then, we can analyze the size of the stuck region according to Lemma 2. Assume that the smallest eigenvalue of $\mathbf{H} := \nabla^2 F(\widetilde{\mathbf{w}})$ satisfies $\lambda_{\min}(\mathbf{H}) \leq -\gamma < 0$, and let the unit vector $\mathbf{e}$ be the eigenvector associated with $\lambda_{\min}(\mathbf{H})$. Let $\mathbf{u}_0, \mathbf{y}_0 \in \mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ be two points such that $\mathbf{y}_0 = \mathbf{u}_0 + \mu_0 \mathbf{e}$ with some $\mu_0 \geq \mu \in (0, r)$. Consider the stuck region $\mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}})$. Suppose that r, R, T_{th}, and μ satisfy

$$T_{\text{th}} = \frac{2}{\eta\gamma} \log_{9/4}\left(\frac{2(R+r)}{\mu}\right), \tag{50}$$

$$R \geq \mu, \tag{51}$$

$$\gamma \geq 24\rho_F(R+r) \log_{9/4}\left(\frac{2(R+r)}{\mu}\right). \tag{52}$$

Then, there must be either $\mathbf{u}_0 \notin \mathbb{W}_S$ or $\mathbf{y}_0 \notin \mathbb{W}_S$. In addition, according to Lemma 3, if conditions (50)-(52) are satisfied, then, when we sample $\mathbf{w}_0$ from $\mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ uniformly at random, the probability that $\mathbf{w}_0 \in \mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}})$ is at most $\frac{2\mu\sqrt{d}}{r}$. In addition, according to (48) in the proof of Lemma 4, assume that $\mathbf{w}_0 \in \mathbb{B}_{\widetilde{\mathbf{w}}}(r)$ and that $\mathbf{w}_0 \notin \mathbb{W}_S(\widetilde{\mathbf{w}}, r, R, T_{\text{th}})$. Let $t \leq T_{\text{th}}$ be the step such that $\|\mathbf{w}_t - \mathbf{w}_0\|_2 \geq R$. Then, we have

$$F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t) \geq \frac{L_F}{4T_{\text{th}}}R^2 - \epsilon r - \frac{L_F}{2}r^2. \tag{53}$$

Combining (50) and (52), we know that the first term on the right hand side of (53) satisfies

$$\frac{L_F}{4T_{\text{th}}}R^2 \geq 3\rho_F R^3. \tag{54}$$

Choose $R = \sqrt{\epsilon/\rho_F}$ and $r = \epsilon$. Then, we know that when $\epsilon \leq \min\{\frac{1}{\rho_F}, \frac{4}{L_F^2\rho_F}\}$, we have $\epsilon r \leq \rho_F R^3$ and $\frac{1}{2}L_F r^2 \leq \rho_F R^3$. Combining these facts with (53) and (54), we know that, when the algorithm successfully escapes the saddle point, the decrease in function value satisfies

$$F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t) \geq \rho_F R^3. \tag{55}$$

Therefore, the average function value decrease during the Escape process is at least

$$\frac{F(\widetilde{\mathbf{w}}) - F(\mathbf{w}_t)}{T_{\text{th}}} \geq \frac{12}{L_F}\epsilon^2. \tag{56}$$

When we have exact gradient oracle, we choose $Q = 1$. According to (49) and (56), for the iterations that are not in the Escape process, the function value decrease in each iteration is at least $\frac{1}{2L_F}\epsilon^2$; for the iterations in the Escape process, the function value decrease on average is $\frac{12}{L_F}\epsilon^2$. Since the function value can decrease at most $F_0 - F^*$, the algorithm must terminate within $\frac{2L_F(F_0-F^*)}{\epsilon^2}$ iterations.

The we proceed to analyze the failure probability. We can see that the number of saddle points that the algorithm may need to escape is at most $\frac{F_0-F^*}{\rho_F R^3}$. Then, by union bound the probability that the algorithm fails to escape one of the saddle points is at most

$$\frac{2\mu\sqrt{d}}{r} \cdot \frac{F_0 - F^*}{\rho_F R^3}$$

By letting the above probability to be δ, we obtain

$$\mu = \frac{\delta\epsilon^{5/2}}{2\sqrt{\rho_F d}(F_0 - F^*)},$$

which completes the proof.

D   Proof of Proposition 1

We consider the following class of one-dimensional functions indexed by $s \in \mathbb{R}$:

$$\mathcal{F} = \{f_s(\cdot) : f_s(w) = \Delta^{3/2}\sin(\Delta^{-1/2}w + s), s \in \mathbb{R}\}.$$

Then, for each function $f_s(\cdot) \in \mathcal{F}$, we have

$$\nabla f_s(w) = \Delta\cos(\Delta^{-1/2}w + s),$$

and

$$\nabla^2 f_s(w) = -\Delta^{1/2}\sin(\Delta^{-1/2}w + s).$$

Thus, we always have $|\nabla f_s(w)| \leq \Delta, \forall w$. Therefore, the Δ-inexact gradient oracle can simply output 0 all the time. In addition, we verify that for all s and w, $|\nabla^2 f_s(w)| \leq \Delta^{1/2} \leq 1$ and $|\nabla^3 f_s(w)| = |-\cos(\Delta^{-1/2}w + s)| \leq 1$ under the assumption that $\Delta \leq 1$, so all the functions in $\mathcal{F}$ are 1-smooth and 1-Hessian Lipschitz as claimed.

In this case, the output of the algorithm does not depend on s, that is, the actual function that we aim to minimize. Consequently, for any output $\widetilde{w}$ of the algorithm, there exists $s \in \mathbb{R}$ such that $\Delta^{-1/2}\widetilde{w} + s = \pi/4$, and thus $|\nabla f_s(\widetilde{w})| = \Delta/\sqrt{2}$ and $\lambda_{\min}(\nabla^2 f_s(\widetilde{w})) = -\Delta^{1/2}/\sqrt{2}$.

E   Proof of Proposition 2

Suppose that during all the iterations, the Escape process is called $E + 1$ times. In the first E times, the algorithm escapes the saddle points, and in the last Escape process, the algorithm does not escape and outputs $\widetilde{\mathbf{w}}$. For the first E processes, there might be up to Q rounds of perturb-and-descent operations, and we only consider the successful descent round. We can then partition the algorithm into $E + 1$ segments. We denote the starting and ending iterates of the t-th segment by $\mathbf{w}_t$ and $\widetilde{\mathbf{w}}_t$, respectively, and denote the length (number of inexact gradient descent iterations) by T_t. When the algorithm reaches $\widetilde{\mathbf{w}}_t$, we randomly perturb $\widetilde{\mathbf{w}}_t$ to $\mathbf{w}_{t+1}$, and thus we have $\|\widetilde{\mathbf{w}}_t - \mathbf{w}_{t+1}\|_2 \leq r$ for every $t = 0, 1, \ldots, E - 1$. According to (25), we know that

$$\sum_{t=0}^{E} T_t \leq \frac{2(F_0 - F^*)L_F}{3\Delta^2} := \widetilde{T},$$

and according to (23), we have

$$E \leq \frac{\rho_F(F_0 - F^*)}{48L_F(\Delta^{6/5}d^{3/5} + \Delta^{7/5}d^{7/10})}.$$

According to (46), we know that

$$F(\mathbf{w}_t) - F(\widetilde{\mathbf{w}}_t) \geq \frac{L_F}{4T_t}\|\mathbf{w}_t - \widetilde{\mathbf{w}}_t\|_2^2 - \frac{\Delta^2 T_t}{L_F},$$

which implies

$$\|\mathbf{w}_t - \widetilde{\mathbf{w}}_t\|_2 \leq \frac{2}{\sqrt{L_F}}\sqrt{T_t(F(\mathbf{w}_t) - F(\widetilde{\mathbf{w}}_t))} + \frac{2\Delta T_t}{L_F}.$$

Then, by Cauchy-Schwarz inequality, we have

$$\sum_{t=0}^{E}\|\mathbf{w}_t - \widetilde{\mathbf{w}}_t\|_2 \leq 2\sqrt{\frac{\widetilde{T}}{L_F}\sum_{t=0}^{E}(F(\mathbf{w}_t) - F(\widetilde{\mathbf{w}}_t))} + \frac{2\Delta\widetilde{T}}{L_F}. \tag{57}$$

On the other hand, we have

$$\sum_{t=0}^{E}(F(\mathbf{w}_t) - F(\widetilde{\mathbf{w}}_t)) + \sum_{t=0}^{E-1}(F(\widetilde{\mathbf{w}}_t) - F(\mathbf{w}_{t+1})) = F(\mathbf{w}_0) - F(\widetilde{\mathbf{w}}_E) \leq F(\mathbf{w}_0) - F^*.$$

According to (47), we have

$$F(\widetilde{\mathbf{w}}_t) - F(\mathbf{w}_{t+1}) \geq -4\Delta r - \frac{L_F}{2}r^2,$$

and thus

$$\sum_{t=0}^{E}(F(\mathbf{w}_t) - F(\widetilde{\mathbf{w}}_t)) \leq F(\mathbf{w}_0) - F^* + E(4\Delta r + \frac{L_F}{2}r^2) \tag{58}$$

Combining (57) and (59), and using the bounds for $\widetilde{T}$ and E, we obtain that

$$\sum_{t=0}^{E}\|\mathbf{w}_t - \widetilde{\mathbf{w}}_t\|_2 \leq C_1\frac{F(\mathbf{w}_0) - F^*}{\Delta}, \tag{59}$$

where $C_1 > 0$ is a quantity that only depends on L_F and ρ_F. In addition, we have

$$\sum_{t=0}^{E-1}\|\widetilde{\mathbf{w}}_t - \mathbf{w}_{t+1}\|_2 \leq Er \leq C_2\frac{F(\mathbf{w}_0) - F^*}{\Delta^{3/5}d^{3/10} + \Delta^{4/5}d^{2/5}}, \tag{60}$$

where $C_2 > 0$ is a quantity that only depends on L_F and ρ_F. Combining (59) and (60), and using triangle inequality, we know that

$$\|\widetilde{\mathbf{w}}_E - \mathbf{w}_0\|_2 \leq C_1\frac{F(\mathbf{w}_0) - F^*}{\Delta} + C_2\frac{F(\mathbf{w}_0) - F^*}{\Delta^{3/5}d^{3/10} + \Delta^{4/5}d^{2/5}} \leq C\frac{F(\mathbf{w}_0) - F^*}{\Delta}.$$

Here, the last inequality is due to the fact that we consider the regime where $\Delta \to 0$, and C is a quantity that only depends on L_F and ρ_F. As a final note, the analysis above also applies to any iterate prior to the final output, and thus, all the iterates during the algorithm stays in the ℓ_2 ball centered at $\mathbf{w}_0$ with radius $C\frac{F(\mathbf{w}_0) - F^*}{\Delta}$.

F Robust Estimation of Gradients

F.1 Iterative Filtering Algorithm

We describe an iterative filtering algorithm for robust mean estimation. The algorithm is originally proposed for robust mean estimation for Gaussian distribution in [22], and extended to sub-Gaussian distribution in [23]; then algorithm is reinterpreted in [60]. Here, we present the

algorithm using the interpretation in [60]. Suppose that m random vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathbb{R}^d$ are drawn i.i.d. from some distribution with mean $\boldsymbol{\mu}$. An adversary observes all these vectors and changes an α fraction of them in an arbitrary fashion, and we only have access to the corrupted data points $\widehat{\mathbf{x}}_1, \widehat{\mathbf{x}}_2, \ldots, \widehat{\mathbf{x}}_m$. The goal of the iterative filtering algorithm is to output an accurate estimate of the true mean $\boldsymbol{\mu}$ even when the dimension d is large. We provide the detailed procedure in Algorithm 2. Here, we note that the algorithm parameter σ needs to be chosen properly in order to achieve the best possible statistical error rate.

Algorithm 2 Iterative Filtering [22, 23, 60]

Require: corrupted data $\widehat{\mathbf{x}}_1, \widehat{\mathbf{x}}_2, \ldots, \widehat{\mathbf{x}}_m \in \mathbb{R}^d$, $\alpha \in [0, \frac{1}{4})$, and algorithm parameter $\sigma > 0$.

$\mathcal{A} \leftarrow [m]$, $c_i \leftarrow 1$, and $\tau_i \leftarrow 0$, $\forall\, i \in \mathcal{A}$.

while true **do**

Let $\mathbf{W} \in \mathbb{R}^{|\mathcal{A}| \times |\mathcal{A}|}$ be a minimizer of the convex optimization problem:

$$\min_{\substack{0 \le W_{ji} \le \frac{3+\alpha}{(1-\alpha)(3-\alpha)m} \\ \sum_{j \in \mathcal{A}} W_{ji} = 1}} \max_{\substack{\mathbf{U} \succeq 0 \\ \operatorname{tr}(\mathbf{U}) \le 1}} \sum_{i \in \mathcal{A}} c_i (\widehat{\mathbf{x}}_i - \sum_{j \in \mathcal{A}} \widehat{\mathbf{x}}_j W_{ji})^\top \mathbf{U} (\widehat{\mathbf{x}}_i - \sum_{j \in \mathcal{A}} \widehat{\mathbf{x}}_j W_{ji}),$$

and $\mathbf{U} \in \mathbb{R}^{d \times d}$ be a maximizer of the convex optimization problem:

$$\max_{\substack{\mathbf{U} \succeq 0 \\ \operatorname{tr}(\mathbf{U}) \le 1}} \min_{\substack{0 \le W_{ji} \le \frac{3+\alpha}{(1-\alpha)(3-\alpha)m} \\ \sum_{j \in \mathcal{A}} W_{ji} = 1}} \sum_{i \in \mathcal{A}} c_i (\widehat{\mathbf{x}}_i - \sum_{j \in \mathcal{A}} \widehat{\mathbf{x}}_j W_{ji})^\top \mathbf{U} (\widehat{\mathbf{x}}_i - \sum_{j \in \mathcal{A}} \widehat{\mathbf{x}}_j W_{ji}).$$

$\forall\, i \in \mathcal{A}$, $\tau_i \leftarrow (\widehat{\mathbf{x}}_i - \sum_{j \in \mathcal{A}} \widehat{\mathbf{x}}_j W_{ji})^\top \mathbf{U} (\widehat{\mathbf{x}}_i - \sum_{j \in \mathcal{A}} \widehat{\mathbf{x}}_j W_{ji})$.

if $\sum_{i \in \mathcal{A}} c_i \tau_i > 8m\sigma^2$ **then**

$\forall\, i \in \mathcal{A}$, $c_i \leftarrow (1 - \frac{\tau_i}{\tau_{\max}}) c_i$, where $\tau_{\max} = \max_{i \in \mathcal{A}} \tau_i$.

$\mathcal{A} \leftarrow \mathcal{A} \setminus \{i : c_i \le \frac{1}{2}\}$.

else

return $\widehat{\boldsymbol{\mu}} = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \widehat{\mathbf{x}}_i$

end if

end while

F.2 Proof of Theorem 5

To prove Theorem 5, we first state a result that bounds the error of the iterative filtering algorithm when the original data points $\{\mathbf{x}_i\}$ are deterministic. The following lemma is proved in [23, 60]; also see [63] for additional discussion.

Lemma 5. *[23, 60] Let $\mathcal{S} := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ be the set of original data points and $\boldsymbol{\mu}_{\mathcal{S}} := \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$ be their sample mean. Let $\widehat{\mathbf{x}}_1, \widehat{\mathbf{x}}_2, \ldots, \widehat{\mathbf{x}}_m$ be the corrupted data. If $\alpha \le \frac{1}{4}$, and the algorithm parameter σ is chosen such that*

$$\left\| \frac{1}{m} \sum_{i=1}^m (\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{S}})(\mathbf{x} - \boldsymbol{\mu}_{\mathcal{S}})^\top \right\|_2 \le \sigma^2, \tag{61}$$

then the output of the iterative filtering algorithm satisfies $\|\widehat{\boldsymbol{\mu}} - \boldsymbol{\mu}_{\mathcal{S}}\|_2 \le \mathcal{O}(\sigma \sqrt{\alpha})$.

By triangle inequality, we have

$$\|\widehat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2 \le \|\widehat{\boldsymbol{\mu}} - \boldsymbol{\mu}_{\mathcal{S}}\|_2 + \|\boldsymbol{\mu}_{\mathcal{S}} - \boldsymbol{\mu}\|_2, \tag{62}$$

and

$$
\begin{aligned}
\left\| \frac{1}{m} \sum_{i=1}^m (\mathbf{x}_i - \boldsymbol{\mu}_{\mathcal{S}})(\mathbf{x} - \boldsymbol{\mu}_{\mathcal{S}})^\top \right\|_2 &= \frac{1}{m} \left\| ([\mathbf{x}_1, \cdots, \mathbf{x}_m] - \boldsymbol{\mu}_{\mathcal{S}} \mathbf{1}^\top)([\mathbf{x}_1, \cdots, \mathbf{x}_m] - \boldsymbol{\mu}_{\mathcal{S}} \mathbf{1}^\top)^\top \right\|_2 \\
&= \frac{1}{m} \left\| [\mathbf{x}_1, \cdots, \mathbf{x}_m] - \boldsymbol{\mu}_{\mathcal{S}} \mathbf{1}^\top \right\|_2^2 \\
&\le \frac{1}{m} \Big(\left\| [\mathbf{x}_1, \cdots, \mathbf{x}_m] - \boldsymbol{\mu} \mathbf{1}^\top \right\|_2 + \sqrt{m} \|\boldsymbol{\mu} - \boldsymbol{\mu}_{\mathcal{S}}\|_2 \Big)^2, \tag{63}
\end{aligned}
$$

where $\mathbf{1}$ denotes the all-one vector.[6] By choosing

$$\sigma = \Theta(\frac{1}{\sqrt{m}}\|[\mathbf{x}_1, \cdots, \mathbf{x}_m] - \boldsymbol{\mu}\mathbf{1}^\top\|_2 + \|\boldsymbol{\mu} - \boldsymbol{\mu}_\mathcal{S}\|_2)$$

in Lemma 5 and combining with the bounds (62) and (63), we obtain that

$$\|\widehat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2 \lesssim \frac{\sqrt{\alpha}}{\sqrt{m}}\|[\mathbf{x}_1, \cdots, \mathbf{x}_m] - \boldsymbol{\mu}\mathbf{1}^\top\|_2 + \|\boldsymbol{\mu} - \boldsymbol{\mu}_\mathcal{S}\|_2. \tag{64}$$

With the above bound in hand, we now turn to the robust gradient estimation problem, where the data points are drawn i.i.d. from some unknown distribution. Let $\widehat{\mathbf{g}}(\mathbf{w}) := \mathsf{filter}\{\widehat{\mathbf{g}}_i(\mathbf{w})\}_{i=1}^m$, where filter represents the iterative filtering algorithm. In light of (64), we know that in order to bound the gradient estimation error $\sup_{\mathbf{w}\in\mathcal{W}}\|\widehat{\mathbf{g}}(\mathbf{w}) - \nabla F(\mathbf{w})\|_2$, it suffices to bound the quantities

$$\sup_{\mathbf{w}\in\mathcal{W}}\|[\nabla F_1(\mathbf{w}), \cdots, \nabla F_m(\mathbf{w})] - \nabla F(\mathbf{w})\mathbf{1}^\top\|_2$$

and

$$\sup_{\mathbf{w}\in\mathcal{W}}\|\frac{1}{m}\sum_{i=1}^m \nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})\|_2.$$

Here, we recall that $\nabla F_i(\mathbf{w})$ is the true gradient of the empirical loss function on the i-th machine, and $\widehat{\mathbf{g}}_i(\mathbf{w})$ is the (possibly) corrupted gradient.

We first bound $\sup_{\mathbf{w}\in\mathcal{W}}\|\frac{1}{m}\sum_{i=1}^m \nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})\|_2$. Note that we have $\frac{1}{m}\sum_{i=1}^m \nabla F_i(\mathbf{w}) = \frac{1}{nm}\sum_{i=1}^m \sum_{j=1}^n \nabla f(\mathbf{w}; \mathbf{z}_{i,j})$. Using the same method as in the proof of Lemma 6 in [17], we can show that for each fixed $\mathbf{w}$, with probability at least $1 - \delta$,

$$\|\frac{1}{m}\sum_{i=1}^m \nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})\|_2 \leq \frac{2\sqrt{2}\zeta}{\sqrt{nm}}\sqrt{d\log 6 + \log\left(\frac{1}{\delta}\right)}.$$

For some $\delta_0 > 0$ to be chosen later, let $\mathcal{W}_{\delta_0} = \{\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^{N_{\delta_0}}\}$ be a finite subset of $\mathcal{W}$ such that for any $\mathbf{w} \in \mathcal{W}$, there exists some $\mathbf{w}^\ell \in \mathcal{W}_{\delta_0}$ such that $\|\mathbf{w}^\ell - \mathbf{w}\|_2 \leq \delta_0$. Standard ϵ-net results from [66] ensure that $N_{\delta_0} \leq (1 + \frac{D}{\delta_0})^d$. Then, by the union bound, we have with probability $1 - \delta$, for all $\mathbf{w}^\ell \in \mathcal{W}_{\delta_0}$,

$$\|\frac{1}{m}\sum_{i=1}^m \nabla F_i(\mathbf{w}^\ell) - \nabla F(\mathbf{w}^\ell)\|_2 \leq \frac{2\sqrt{2}\zeta}{\sqrt{nm}}\sqrt{d\log 6 + \log\left(\frac{N_{\delta_0}}{\delta}\right)}. \tag{65}$$

When (65) holds, by the smoothness of $f(\cdot; \mathbf{z})$ we know that for all $\mathbf{w} \in \mathcal{W}$,

$$\|\frac{1}{m}\sum_{i=1}^m \nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})\|_2 \leq \frac{2\sqrt{2}\zeta}{\sqrt{nm}}\sqrt{d\log 6 + \log\left(\frac{N_{\delta_0}}{\delta}\right)} + 2L\delta_0.$$

By choosing $\delta_0 = \frac{1}{nmL}$ and $\delta = \frac{1}{(1+mnDL)^d}$, we obtain that with probability at least $1 - \frac{1}{(1+mnDL)^d}$, for all $\mathbf{w} \in \mathcal{W}$,

$$\|\frac{1}{m}\sum_{i=1}^m \nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})\|_2 \lesssim \frac{\zeta}{\sqrt{nm}}\sqrt{d\log(1 + nmDL)}. \tag{66}$$

We next bound $\sup_{\mathbf{w}\in\mathcal{W}}\|[\nabla F_1(\mathbf{w}), \cdots, \nabla F_m(\mathbf{w})] - \nabla F(\mathbf{w})\mathbf{1}^\top\|_2$. We note that when the gradients are sub-Gaussian distributed, similar results for the centralized setting have been established in [13]. One can check that for every i, $\nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})$ is $\frac{\zeta}{\sqrt{n}}$-sub-Gaussian. Define $\mathbf{G}(\mathbf{w}) := [\nabla F_1(\mathbf{w}), \cdots, \nabla F_m(\mathbf{w})] - \nabla F(\mathbf{w})\mathbf{1}^\top$. Using a standard concentration inequality for the norm of a matrix with independent sub-Gaussian columns [66], we obtain that for each fixed $\mathbf{w}$, with probability at least $1 - \delta$,

$$\|\frac{1}{m}\mathbf{G}(\mathbf{w})\mathbf{G}(\mathbf{w})^\top - \frac{1}{n}\boldsymbol{\Sigma}(\mathbf{w})\|_2 \lesssim \frac{\zeta^2}{n}\left(\sqrt{\frac{d}{m}} + \frac{d}{m} + \frac{1}{m}\log\left(\frac{1}{\delta}\right) + \sqrt{\frac{1}{m}\log\left(\frac{1}{\delta}\right)}\right),$$

---

[6]We note that similar derivation also appears in [63].

which implies that

$$\frac{1}{\sqrt{m}}\|\mathbf{G}(\mathbf{w})\|_2 \lesssim \frac{\sigma}{\sqrt{n}} + \frac{\zeta}{\sqrt{n}}\left(\sqrt{\frac{d}{m}} + \frac{d}{m} + \frac{1}{m}\log\left(\frac{1}{\delta}\right) + \sqrt{\frac{1}{m}\log\left(\frac{1}{\delta}\right)}\right)^{1/2}.$$

Recall the δ_0-net $\mathcal{W}_{\delta_0} = \{\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^{N_{\delta_0}}\}$ as defined above. Then, we have with probability at least $1 - \delta$, for all $\mathbf{w}^\ell \in \mathcal{W}_{\delta_0}$

$$\frac{1}{\sqrt{m}}\|\mathbf{G}(\mathbf{w}^\ell)\|_2 \lesssim \frac{\sigma}{\sqrt{n}} + \frac{\zeta}{\sqrt{n}}\left(\sqrt{\frac{d}{m}} + \frac{d}{m} + \frac{1}{m}\log\left(\frac{N_{\delta_0}}{\delta}\right) + \sqrt{\frac{1}{m}\log\left(\frac{N_{\delta_0}}{\delta}\right)}\right)^{1/2}. \tag{67}$$

For each $\mathbf{w}$ with $\|\mathbf{w}^\ell - \mathbf{w}\|_2 \le \delta_0$, we have

$$\|\mathbf{G}(\mathbf{w}^\ell) - \mathbf{G}(\mathbf{w})\|_2 \le \|\mathbf{G}(\mathbf{w}^\ell) - \mathbf{G}(\mathbf{w})\|_F$$
$$\le \left(\sum_{i=1}^m \|(\nabla F_i(\mathbf{w}^\ell) - \nabla F(\mathbf{w}^\ell)) - (\nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w}))\|_2^2\right)^{1/2}$$
$$\le 2L\delta_0\sqrt{m}.$$

This implies that when the bound (67) holds, we have for all $\mathbf{w} \in \mathcal{W}$,

$$\frac{1}{\sqrt{m}}\|\mathbf{G}(\mathbf{w})\|_2 \lesssim \frac{\sigma}{\sqrt{n}} + \frac{\zeta}{\sqrt{n}}\left(\sqrt{\frac{d}{m}} + \frac{d}{m} + \frac{1}{m}\log\left(\frac{N_{\delta_0}}{\delta}\right) + \sqrt{\frac{1}{m}\log\left(\frac{N_{\delta_0}}{\delta}\right)}\right)^{1/2} + 2L\delta_0. \tag{68}$$

Choose $\delta_0 = \frac{1}{nmL}$, in which case the last term above is a high order term. In this case, choosing $\delta = \frac{1}{(1+mnDL)^d}$, we have with probability at least $1 - \frac{1}{(1+mnDL)^d}$, for all $\mathbf{w} \in \mathcal{W}$,

$$\frac{1}{\sqrt{m}}\|\mathbf{G}(\mathbf{w})\|_2 \lesssim \frac{\sigma}{\sqrt{n}} + \frac{\zeta}{\sqrt{n}}\left(\left(\frac{d}{m} + \sqrt{\frac{d}{m}}\right)\log(1 + nmDL)\right)^{1/2}$$
$$\lesssim \frac{\sigma}{\sqrt{n}} + \frac{\zeta}{\sqrt{n}}\left(1 + \sqrt{\frac{d}{m}}\right)\sqrt{\log(1 + nmDL)}. \tag{69}$$

Combining the bounds (64), (66), and (69), we obtain that with probability at least $1 - \frac{2}{(1+mnDL)^d}$,

$$\sup_{\mathbf{w} \in \mathcal{W}} \|\widehat{\mathbf{g}}(\mathbf{w}) - \nabla F(\mathbf{w})\|_2 \lesssim \left((\sigma + \zeta)\sqrt{\frac{\alpha}{n}} + \zeta\sqrt{\frac{d}{nm}}\right)\sqrt{\log(1 + nmDL)},$$

which completes the proof.

F.3 Median and Trimmed Mean

In this section, we present the error bounds of median and trimmed mean operations in the Byzantine setting in [69] for completeness.

Assumption 4. *For any $\mathbf{z} \in \mathcal{Z}$, the k-th partial derivative $\partial_k f(\cdot; \mathbf{z})$ is L_k-Lipschitz for each $k \in [d]$. Let $\widehat{L} := (\sum_{k=1}^d L_k^2)^{1/2}$.*

For the median-based algorithm, one needs to use the notion of the *absolute skewness* of a one-dimensional random variable X, defined as $S(X) := \mathbb{E}[|X - \mathbb{E}[X]|^3]/\text{Var}(X)^{3/2}$. Define the following upper bounds on the standard deviation and absolute skewness of the gradients:

$$v := \sup_{\mathbf{w} \in \mathcal{W}} \left(\mathbb{E}[\|\nabla f(\mathbf{w}; \mathbf{z}) - \nabla F(\mathbf{w})\|_2^2]\right)^{1/2}, \quad s := \sup_{\mathbf{w} \in \mathcal{W}} \max_{k \in [d]} S\left(\partial f_k(\mathbf{w}; \mathbf{z})\right).$$

Then one has the following guarantee for the median-based algorithm.

Theorem 7 (median). *[69] Suppose that Assumption 4 holds. Assume that*

$$\alpha + \left(\frac{d\log(1 + nmD\widehat{L})}{m(1-\alpha)}\right)^{1/2} + c_1\frac{s}{\sqrt{n}} \le \frac{1}{2} - c_2$$

for some constant $c_1, c_2 > 0$. Then, with probability $1 - o(1)$, $\mathsf{GradAGG} \equiv \mathsf{med}$ provides a Δ_{med}-inexact gradient oracle with

$$\Delta_{\mathsf{med}} \le \frac{c_3}{\sqrt{n}}v\big(\alpha + (\frac{d\log(nmD\widehat{L})}{m})^{1/2} + \frac{s}{\sqrt{n}}\big) + \mathcal{O}(\frac{1}{nm}),$$

where c_3 is an absolute constant.

Therefore, the median operation provides a $\widetilde{\mathcal{O}}(v(\frac{\alpha}{\sqrt{n}} + \sqrt{\frac{d}{nm}} + \frac{s}{n}))$-inexact gradient oracle. If each partial derivative is of size $\mathcal{O}(1)$, the quantity v is of the order $\mathcal{O}(\sqrt{d})$ and thus one has $\Delta_{\mathsf{med}} \lesssim \frac{\alpha\sqrt{d}}{\sqrt{n}} + \frac{d}{\sqrt{nm}} + \frac{\sqrt{d}}{n}$.

For the trimmed mean algorithm, one needs to assume that the gradients of the loss functions are sub-exponential.

Assumption 5. *For any $\mathbf{w} \in \mathcal{W}$, $\nabla f(\mathbf{w}; \mathbf{z})$ is ξ-sub-exponential.*

In this setting, there is the following guarantee.

Theorem 8 (trimmed mean). *[69] Suppose that Assumptions 4 and 5 hold. Choose $\beta = c_4\alpha \le \frac{1}{2} - c_5$ with some constant $c_4 \ge 1$, $c_5 > 0$. Then, with probability $1 - o(1)$, $\mathsf{GradAGG} \equiv \mathsf{trmean}_\beta$ provides a Δ_{tm}-inexact gradient oracle with*

$$\Delta_{\mathsf{tm}} \le c_6\xi d\Big(\frac{\alpha}{\sqrt{n}} + \frac{1}{\sqrt{nm}}\Big)\sqrt{\log(nmD\widehat{L})},$$

where c_6 is an absolute constant.

Therefore, the trimmed mean operation provides a $\widetilde{\mathcal{O}}(\xi d(\frac{\alpha}{\sqrt{n}} + \frac{1}{\sqrt{nm}}))$-inexact gradient oracle.

F.4 Lower Bound for First-Order Guarantee

In this section we prove Observation 2. We consider the simple mean estimation problem with random vector $\mathbf{z}$ drawn from a distribution $\mathcal{D}$ with mean $\boldsymbol{\mu}$. The loss function associated with $\mathbf{z}$ is $f(\mathbf{w}; \mathbf{z}) = \frac{1}{2}\|\mathbf{w} - \mathbf{z}\|_2^2$. The population loss is $F(\mathbf{w}) = \frac{1}{2}(\|\mathbf{w}\|_2^2 - 2\boldsymbol{\mu}^\top\mathbf{w} + \mathbb{E}[\|\mathbf{z}\|_2^2])$, and $\nabla F(\mathbf{w}) = \mathbf{w} - \boldsymbol{\mu}$. We first provide a lower bound for distributed mean estimation in the Byzantine setting, which is proved in [69].

Lemma 6. *[69] Suppose that $\mathbf{z}$ is Gaussian distributed with mean $\boldsymbol{\mu}$ and covariance $\sigma^2\mathbf{I}$. Then, any algorithm that outputs an estimate $\widetilde{\mathbf{w}}$ of $\boldsymbol{\mu}$ has a constant probability such that*

$$\|\widetilde{\mathbf{w}} - \boldsymbol{\mu}\|_2 = \Omega(\frac{\alpha}{\sqrt{n}} + \sqrt{\frac{d}{nm}}).$$

Since $\nabla F(\widetilde{\mathbf{w}}) = \widetilde{\mathbf{w}} - \boldsymbol{\mu}$, the above bound directly implies the lower bound on $\|\nabla F(\widetilde{\mathbf{w}})\|_2$ in Observation 2.