Defending Against Saddle Point Attack in Byzantine-Robust Distributed Learning

Dong Yin [1] **Yudong Chen** [2] **Kannan Ramchandran** [1] **Peter Bartlett** [1 3]

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, and 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 (McMahan & Ramage, 2017; Konečnỳ et al., 2016; 2015), 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 (Lamport et al., 1982), 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 (Alistarh et al., 2018; Blanchard et al., 2017; Chen et al., 2017; Feng et al., 2014; Su & Vaidya, 2016a;b; Yin et al., 2018a). 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 (Soudry & Carmon, 2016; Ge et al., 2016; 2017). 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 (Kawaguchi,

[1]Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, CA, USA [2]School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA [3]Department of Statistics, UC Berkeley, Berkeley, CA, USA. Correspondence to: Dong Yin <dongyin@eecs.berkeley.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

2016; Ge et al., 2017). 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 (Ge et al., 2015; Jin et al., 2017a; Carmon et al., 2016; Jin et al., 2017b).

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) (Chen et al., 2017; Yin et al., 2018a; Alistarh et al., 2018). However, when F is non-convex, such algorithms may be trapped in the neighborhood of a saddle point; see Example 1 in Appendix B.
- **Challenges due to Byzantine machines:** Without Byzantine machines, vanilla GD (Lee et al., 2016), as well as its more efficient variants such as perturbed gradient descent (PGD) (Jin et al., 2017a), 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 B.

We discuss and compare with existing work in more details in Section 5. 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 (Jin et al., 2017a), 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. 3.2)**.** *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 (Nesterov, 1998) (up to log factors). In the exact gradient setting, a variant of the above result in fact matches the guarantees for PGD (Jin et al., 2017a)—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 (Diakonikolas et al., 2016; 2017). 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. 4)**.** *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 $\mathbb{B}^{(d)}_{\mathbf{w}}(r)$, or $\mathbb{B}_{\mathbf{w}}(r)$ when it is clear from the context.

2. 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 (Lynch, 1996) 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* $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le \epsilon_g$ *and* $\lambda_{\min}\big(\nabla^2 F(\widetilde{\mathbf{w}})\big) \ge -\epsilon_H$.

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* $\sup_{\mathbf{w}\neq\mathbf{w}'} \frac{\|\nabla h(\mathbf{w}) - \nabla h(\mathbf{w}')\|_2}{\|\mathbf{w}-\mathbf{w}'\|_2} \le L$*, and* ρ*-Hessian Lipschitz if* $\sup_{\mathbf{w}\neq\mathbf{w}'} \frac{\|\nabla^2 h(\mathbf{w}) - \nabla^2 h(\mathbf{w}')\|_2}{\|\mathbf{w}-\mathbf{w}'\|_2} \le \rho$.

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 $\mathcal{W}$.*

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

Without loss of generality, we assume that $\Delta \le 1$ throughout the paper. In this section, we treat GradAGG as a given black box; in Section 4, we discuss several robust aggregation algorithms and characterize their inexactness Δ. We emphasize that in the Byzantine setting, the output of 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.

3.1. 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 4, 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 $\|\widehat{\mathbf{g}}(\mathbf{w})\|_2 \le \epsilon$ 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):

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

During this process, once we observe that $\|\mathbf{w}_t' - \mathbf{w}_0'\|_2 \ge R$ 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 $\widetilde{\mathbf{w}}$.

3.2. 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 $\Delta \le 1$. Given any $\delta \in (0,1)$, choose the parameters for Algorithm 1 as follows: stepsize $\eta = \frac{1}{L_F}$, $\epsilon = 3\Delta$, $r = 4\Delta^{3/5}d^{3/10}\rho_F^{-1/2}$, $R = \Delta^{2/5}d^{1/5}\rho_F^{-1/2}$,*

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

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

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

$$\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le 4\Delta,$$
$$\lambda_{\min}\big(\nabla^2 F(\widetilde{\mathbf{w}})\big) \ge -1900\big(\rho_F^{1/2} + L_F\big)\Delta^{2/5}d^{1/5}\log\Big(\frac{10}{\Delta}\Big), \tag{3}$$

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

We prove Theorem 3 in Appendix C.[1] Below let us parse

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

ByzantinePGD($\mathbf{w}_0, \eta, \epsilon, r, Q, R, T_{\rm th}$)

$\mathbf{w} \leftarrow \mathbf{w}_0$
while true **do**
  Master: send $\mathbf{w}$ to worker machines.
  for $i \in [m]$ in parallel **do**
    Worker i: compute $\widehat{\mathbf{g}}_i(\mathbf{w})$
    send to master machine.
  end for
  Master:
  $\widehat{\mathbf{g}}(\mathbf{w}) \leftarrow \mathsf{GradAGG}\{\widehat{\mathbf{g}}_i(\mathbf{w})\}_{i=1}^m$.
  if $\|\widehat{\mathbf{g}}(\mathbf{w})\|_2 \le \epsilon$ **then**
    Master: $\widetilde{\mathbf{w}} \leftarrow \mathbf{w}$,
    $(\mathsf{esc}, \mathbf{w}, \widehat{\mathbf{g}}(\mathbf{w})) \leftarrow$ Escape ($\widetilde{\mathbf{w}}, \eta, r, Q, R, T_{\rm th}$).
    if $\mathsf{esc} = \text{false}$ **then**
      return $\widetilde{\mathbf{w}}$.
    end if
  end if
  Master: $\mathbf{w} \leftarrow \mathbf{w} - \eta\widehat{\mathbf{g}}(\mathbf{w})$.
end while

Escape($\widetilde{\mathbf{w}}, \eta, r, Q, R, T_{\rm th}$)

for $k = 1, 2, \ldots, Q$ **do**
  Master: sample $\mathbf{p}_k \sim \mathrm{Unif}(\mathbb{B}_0(r))$,
  $\mathbf{w}' \leftarrow \widetilde{\mathbf{w}} + \mathbf{p}_k$, $\mathbf{w}'_0 \leftarrow \mathbf{w}'$.
  for $t = 0, 1, \ldots, T_{\rm th}$ **do**
    Master: send $\mathbf{w}'$ to worker machines.
    for $i \in [m]$ in parallel **do**
      Worker i: compute $\widehat{\mathbf{g}}_i(\mathbf{w}')$
      send to master machine.
    end for
    Master: $\widehat{\mathbf{g}}(\mathbf{w}') \leftarrow \mathsf{GradAGG}\{\widehat{\mathbf{g}}_i(\mathbf{w}')\}_{i=1}^m$.
    if $\|\mathbf{w}' - \mathbf{w}'_0\|_2 \ge R$ **then**
      return $(\text{true}, \mathbf{w}', \widehat{\mathbf{g}}(\mathbf{w}'))$.
    else
      $\mathbf{w}' \leftarrow \mathbf{w}' - \eta\widehat{\mathbf{g}}(\mathbf{w}')$
    end if
  end for
end for
return $(\text{false}, \mathbf{w}', \widehat{\mathbf{g}}(\mathbf{w}'))$.

Algorithm 1. Byzantine Perturbed Gradient Descent (ByzantinePGD)

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,*

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

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 $\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le \mathcal{O}(\Delta)$ (Nesterov, 1998); 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 E. 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 5 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 $R = \sqrt{\epsilon/\rho_F}$, $T_{\rm th} = \frac{L}{12\rho_F(R+r)}$. Then, with probability at least $1-\delta$, Algorithm 1 outputs a $\widetilde{\mathbf{w}}$ satisfying the bounds*

$$\|\nabla F(\widetilde{\mathbf{w}})\|_2 \le \epsilon,$$
$$\lambda_{\min}(\nabla^2 F(\widetilde{\mathbf{w}})) \ge -60\sqrt{\rho_F\epsilon}\log\Big(\frac{8\rho_F\sqrt{d}(F_0 - F^*)}{\delta\epsilon^2}\Big),$$

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

We prove Theorem 4 in Appendix D. The convergence guarantee above matches that of the original PGD algorithm (Jin et al., 2017a) 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.

in Theorem 3.

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

4. 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 (Diakonikolas et al., 2016; 2017; Steinhardt et al., 2017). We characterize their inexactness Δ under the statistical setting in Section 2, 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 $\mathbb{E}[\exp(\lambda\langle \mathbf{x}-\boldsymbol{\mu}, \mathbf{u}\rangle)] \le e^{\frac{1}{2}\zeta^2\lambda^2\|\mathbf{u}\|_2^2}, \forall\ \lambda, \mathbf{u}$. It is said to be ξ-sub-exponential if $\mathbb{E}[\exp(\lambda\langle \mathbf{x}-\boldsymbol{\mu}, \mathbf{u}\rangle)] \le e^{\frac{1}{2}\xi^2\lambda^2\|\mathbf{u}\|_2^2},\ \forall\ |\lambda| < \frac{1}{\xi}, \mathbf{u}$.*

We also need the following result (proved in Appendix F), 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.

4.1. Iterative Filtering Algorithm

We start with a recently developed high-dimension robust estimation technique called the *iterative filtering* algorithm (Diakonikolas et al., 2016; 2017; Steinhardt et al., 2017) 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 G.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 G.1 for* GradAGG*, and assume that $\alpha \le \frac{1}{4}$. With probability $1-o(1)$,* GradAGG *provides a Δ_{ftr}-inexact gradient oracle with*

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

where c is an absolute constant.

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

4.2. 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 $\{x_k^1,\dots,x_k^m\}$ 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. (2018a). For completeness, we record their results below as an informal theorem; details are relegated to Appendix G.4.

Theorem 6 (Informal)**.** *(Yin et al., 2018a) 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 $\Delta_{\mathsf{tm}} \lesssim \frac{\alpha d}{\sqrt{n}} + \frac{d}{\sqrt{nm}}$.*

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

4.3. Comparison and Optimality

In Table 1, 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	$\widetilde{\mathcal{O}}\big(\frac{\alpha\sqrt{d}}{\sqrt{n}} + \frac{d}{\sqrt{nm}} + \frac{\sqrt{d}}{n}\big)$
trimmed mean	$\widetilde{\mathcal{O}}\big(\frac{\alpha d}{\sqrt{n}} + \frac{d}{\sqrt{nm}}\big)$
iterative filtering	$\widetilde{\mathcal{O}}\big(\frac{\sqrt{\alpha}}{\sqrt{n}} + \frac{\sqrt{d}}{\sqrt{nm}}\big)$

Table 1. 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 1, 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 $\|\nabla F(\widetilde{\mathbf{w}})\|_2 = \Omega\big(\frac{\alpha}{\sqrt{n}} + \frac{\sqrt{d}}{\sqrt{nm}}\big)$ with a constant probability.*

We prove Observation 2 in Appendix G.3. 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.

5. 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 (Lee et al., 2016; 2017), achieving convergence in polynomial time requires more a careful algorithmic design (Du et al., 2017). Such convergence guarantees are enjoyed by several GD-based algorithms; examples include PGD (Jin et al., 2017a), Neon+GD (Xu & Yang, 2017), and Neon2+GD (Allen-Zhu & Li, 2017). 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 2).

- 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 (Bertsekas & Tsitsiklis, 2000; Devolder et al., 2014), robust estimation (Prasad et al., 2018), and structured non-convex problems (Balakrishnan et al., 2014; Chen & Wainwright, 2015; Candes et al., 2015; Zhang et al., 2016). Particularly relevant to us is the recent work by Jin et al. (2018), who consider the problem of minimizing F when only given access to the gradients of another *smooth* function $\widehat{F}$ satisfying $\|\nabla\widehat{F}(\mathbf{w}) - \nabla F(\mathbf{w})\|_\infty \le \Delta/\sqrt{d}$, $\forall \mathbf{w}$. 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. (2018) 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 (Shamir et al., 2014; Lee et al., 2015; Yin et al., 2018b; Blanchard et al., 2017; Chen et al., 2018a; Damaskinos et al., 2018; Lian et al., 2017; Jiang et al., 2017). Here, we compare with existing Byzantine-robust distributed learning algorithms that are most relevant to our work, and summarize the comparison in Table 3. A general idea of designing Byzantine-robust algorithms is to combine optimization algorithms with a robust aggregation (or outlier

Algorithm	PGD	Neon+GD	Neon2+GD	**ByzantinePGD**
Byzantine-robust?	no	no	no	yes
Purpose of perturbation	escape SP	escape SP	escape SP	escape SP + Byzantine 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 2. Comparison with PGD, Neon+GD, and Neon2+GD. SP = saddle point.

	Robust Aggregation Method	Non-convex Guarantee
Feng et al. (2014)	geometric median	no
Chen et al. (2017)	geometric median	no
Blanchard et al. (2017)	Krum	first-order
Yin et al. (2018a)	median, trimmed mean	first-order
Xie et al. (2018)	mean-around-median, marginal median	first-order
Alistarh et al. (2018)	martingale-based	no
Su & Xu (2018)	iterative filtering	no
This work	median, trimmed mean, iterative filtering	second-order

Table 3. Comparison with other Byzantine-robust distributed learning algorithms.

removal) subroutine. For convex losses, the aggregation subroutines analyzed in the literature include geometric median (Feng et al., 2014; Chen et al., 2017), median and trimmed mean (Yin et al., 2018a), iterative filtering for the high dimensional setting (Su & Xu, 2018), and martingale-based methods for the SGD setting (Alistarh et al., 2018). 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 (Blanchard et al., 2017), median and trimmed mean (Yin et al., 2018a), mean-around-median and marginal median (Xie et al., 2018). 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. (2018a). 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 (Diakonikolas et al., 2016; 2017). 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 & Xu (2018) 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 (Candes et al., 2015), low-rank estimation (Chen & Wainwright, 2015; Zhao et al., 2015), and dictionary learning (Agarwal et al., 2014; Sun et al., 2015), many algorithms are developed by leveraging the particular structure of the problems, and the either use a smart initialization (Candes et al., 2015; Tu et al., 2015) or initialize randomly (Chen et al., 2018b; Chatterji & Bartlett, 2017). Other algorithms are developed for general non-convex optimization, and they can be classified into gradient-based (Ge et al., 2015; Levy, 2016; Xu & Yang, 2017; Allen-Zhu, 2017; Allen-Zhu & Li, 2017; Jin et al., 2017b), Hessian-vector-product-based (Carmon et al., 2016; Agarwal et al., 2016; Royer & Wright, 2018; Royer et al., 2018), and Hessian-based (Nesterov & Polyak, 2006; Curtis et al., 2017) 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. (2017), and $\mathcal{O}(\frac{1}{\epsilon^{7/4}})$ by Carmon et al. (2016)— gradient-based methods are easier to implement in practice, especially in the distributed setting we are interested in.

We discuss additional related work on robust statistics in Appendix A.

Acknowledgements

D. Yin is partially supported by Berkeley DeepDrive Industry Consortium. Y. Chen is partially supported by NSF CRII award 1657420 and grant 1704828. K. Ramchandran is partially supported by NSF CIF award 1703678. P. Bartlett is partially supported by NSF grant IIS-1619362. The authors would like to thank Zeyuan Allen-Zhu for pointing out a potential way to improve our initial results, and Ilias Diakonikolas for discussing some related work. The authors would also like to thank anonymous reviewers for their comments.

References

Agarwal, A., Anandkumar, A., Jain, P., Netrapalli, P., and Tandon, R. Learning sparsely used overcomplete dictionaries. In *Conference on Learning Theory*, pp. 123–137, 2014.

Agarwal, N., Allen-Zhu, Z., Bullins, B., Hazan, E., and Ma, T. Finding approximate local minima for nonconvex optimization in linear time. *arXiv preprint arXiv:1611.01146*, 2016.

Alistarh, D., Allen-Zhu, Z., and Li, J. Byzantine stochastic gradient descent. *arXiv preprint arXiv:1803.08917*, 2018.

Allen-Zhu, Z. Natasha 2: Faster non-convex optimization than SGD. *arXiv preprint arXiv:1708.08694*, 2017.

Allen-Zhu, Z. and Li, Y. Neon2: Finding local minima via first-order oracles. *arXiv preprint arXiv:1711.06673*, 2017.

Balakrishnan, S., Wainwright, M. J., and Yu, B. Statistical guarantees for the EM algorithm: From population to sample-based analysis. *arXiv preprint arXiv:1408.2156*, 2014.

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

Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., and Stainer, J. Byzantine-tolerant machine learning. *arXiv preprint arXiv:1703.02757*, 2017.

Candes, E. J., Li, X., and Soltanolkotabi, M. Phase retrieval via Wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Accelerated methods for non-convex optimization. *arXiv preprint arXiv:1611.00756*, 2016.

Chatterji, N. and Bartlett, P. L. Alternating minimization for dictionary learning with random initialization. In *Advances in Neural Information Processing Systems*, pp. 1994–2003, 2017.

Chen, L., Charles, Z., Papailiopoulos, D., et al. DRACO: Robust distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, 2018a.

Chen, Y. and Wainwright, M. J. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. *arXiv preprint arXiv:1509.03025*, 2015.

Chen, Y., Su, L., and Xu, J. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *arXiv preprint arXiv:1705.05491*, 2017.

Chen, Y., Chi, Y., Fan, J., and Ma, C. Gradient descent with random initialization: Fast global convergence for nonconvex phase retrieval. *arXiv preprint arXiv:1803.07726*, 2018b.

Curtis, F. E., Robinson, D. P., and Samadi, M. 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.

Damaskinos, G., Mhamdi, E. M. E., Guerraoui, R., Patra, R., and Taziki, M. Asynchronous byzantine machine learning. *arXiv preprint arXiv:1802.07928*, 2018.

Devolder, O., Glineur, F., and Nesterov, Y. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Moitra, A., and Stewart, A. Robust estimators in high dimensions without the computational intractability. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pp. 655–664. IEEE, 2016.

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Moitra, A., and Stewart, A. Being robust (in high dimensions) can be practical. *arXiv preprint arXiv:1703.00893*, 2017.

Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Singh, A., and Poczos, B. Gradient descent can take exponential time to escape saddle points. In *Advances in Neural Information Processing Systems*, pp. 1067–1077, 2017.

Feng, J., Xu, H., and Mannor, S. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pp. 797–842, 2015.

Ge, R., Lee, J. D., and Ma, T. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pp. 2973–2981, 2016.

Ge, R., Jin, C., and Zheng, Y. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. *arXiv preprint arXiv:1704.00708*, 2017.

Jiang, Z., Balu, A., Hegde, C., and Sarkar, S. Collaborative deep learning in fixed topology networks. In *Advances in Neural Information Processing Systems*, pp. 5904–5914, 2017.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017a.

Jin, C., Netrapalli, P., and Jordan, M. I. Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*, 2017b.

Jin, C., Liu, L. T., Ge, R., and Jordan, M. I. Minimizing nonconvex population risk from rough empirical risk. *arXiv preprint arXiv:1803.09357*, 2018.

Kawaguchi, K. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2016.

Konečnỳ, J., McMahan, B., and Ramage, D. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

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

Lee, J. D., Lin, Q., Ma, T., and Yang, T. Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity. *arXiv preprint arXiv:1507.07595*, 2015.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

Lee, J. D., Panageas, I., Piliouras, G., Simchowitz, M., Jordan, M. I., and Recht, B. First-order methods almost always avoid saddle points. *arXiv preprint arXiv:1710.07406*, 2017.

Levy, K. Y. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5330–5340, 2017.

Lynch, N. A. *Distributed Algorithms*. Elsevier, 1996.

McMahan, B. and Ramage, D. Federated learning: Collaborative machine learning without centralized training data. https://research.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

Nesterov, Y. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 1998.

Nesterov, Y. and Polyak, B. T. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Prasad, A., Suggala, A. S., Balakrishnan, S., and Ravikumar, P. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485*, 2018.

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

Royer, C. W., O'Neill, M., and Wright, S. J. A newton-cg algorithm with complexity guarantees for smooth unconstrained optimization. *arXiv preprint arXiv:1803.02924*, 2018.

Shamir, O., Srebro, N., and Zhang, T. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pp. 1000–1008, 2014.

Soudry, D. and Carmon, Y. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

Steinhardt, J., Charikar, M., and Valiant, G. Resilience: A criterion for learning in the presence of arbitrary outliers. *arXiv preprint arXiv:1703.04940*, 2017.

Su, L. and Vaidya, N. H. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pp. 425–434. ACM, 2016a.

Su, L. and Vaidya, N. H. Non-Bayesian learning in the presence of Byzantine agents. In *International Symposium on Distributed Computing*, pp. 414–427. Springer, 2016b.

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

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

Tu, S., Boczar, R., Simchowitz, M., Soltanolkotabi, M., and Recht, B. Low-rank solutions of linear matrix equations via procrustes flow. *arXiv preprint arXiv:1507.03566*, 2015.

Xie, C., Koyejo, O., and Gupta, I. Generalized Byzantine-tolerant SGD. *arXiv preprint arXiv:1802.10116*, 2018.

Xu, Y. and Yang, T. First-order stochastic algorithms for escaping from saddle points in almost linear time. *arXiv preprint arXiv:1711.01944*, 2017.

Yin, D., Chen, Y., Kannan, R., and Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5650–5659, 2018a.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1998–2007, 2018b.

Zhang, H., Chi, Y., and Liang, Y. Provable non-convex phase retrieval with outliers: Median-truncated wirtinger flow. In *International conference on machine learning*, pp. 1022–1031, 2016.

Zhao, T., Wang, Z., and Liu, H. A nonconvex optimization framework for low rank matrix estimation. In *Advances in Neural Information Processing Systems*, pp. 559–567, 2015.