Distributed Optimization Methods Robust to Byzantine Attackers

By

XINYANG CAO

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

____

Lifeng Lai, Chair

____

Zhi Ding

____

Bernard C. Levy

Committee in Charge

2022

Abstract

Due to the grow of modern dataset size and the desire to harness computing power of multiple machines, there is a recent surge of interest in the design of distributed machine learning algorithms. There are many algorithms in this research area. First order methods use gradient information for update in each iteration. Second order methods employ second order information to harness the computer power of each worker and reduce the cost of communication. Zeroth order methods use estimated gradient information to solve the problem where the gradient is not available. However, since distributed algorithms require communication between workers and server, they are sensitive to Byzantine attackers who can send falsified data to prevent the convergence of algorithms or lead the algorithms to converge to value of the attackers' choice. Some recent work proposed interesting first order algorithms that can deal with the scenario when up to half of the workers are compromised.

In this thesis, we investigate different order algorithms that can deal with Byzantine attackers.

Firstly, we discuss the robust first order distributed algorithms in distributed network. A commonly used algorithm is distributed gradient descent algorithm. But most existing algorithms assume that there are no attack in the network. However, in practice, there is a risk that the some workers are compromised. We provide a novel first order algorithm that can deal with an arbitrary number of Byzantine attackers. The main idea is to ask the parameter server to randomly select a small clean dataset and compute noisy gradient using this small dataset. This noisy gradient will then be used as a ground truth to filter out information sent by compromised workers. We show that the proposed algorithm converges to the neighborhood of the population minimizer regardless the number of Byzantine attackers. We also proposed an algorithm that deal with arbitrary number of Byzantine attackers when we know the upper number of the Byzantine attackers. We show this algorithm can have a better convergence rate than the former one. We further provide numerical examples to show that the proposed algorithm can benefit from the presence of good workers and achieve better performance than existing algorithms.

Secondly, we discuss the robust second order distributed algorithms that can deal with Byzantine attackers. We propose two robust second-order algorithms. The main idea of the first algorithm, named median-based approximate Newton's method (MNM), is to ask the parameter server to aggregate gradient information and approximate Newton's direction from all workers by geometric median. We show that MNM can converge when up to half of the workers are Byzantine attackers. To deal with the case with an arbitrary number of attackers, we then propose a comparison-based approximate Newton's method (CNM). The main idea of CNM is to ask the server to randomly select a small clean dataset and compute noisy gradient and Newton's direction using this small dataset. These noisy information will then be used as an approximation of the ground truth to filter out bad information from Byzantine attackers. We show that CNM can converge to the neighborhood of the population minimizer even when more than half of the workers are Byzantine workers. We further provide numerical examples to illustrate the performance of the proposed algorithms.

Finally, we discuss the robust zeroth order distributed algorithm in decentralized distributed network. We propose a zeroth order adversarial robust alternating direction method of multipliers (ZOAR-ADMM) that can deal with Byzantine attackers for the zeroth-order methods in a consensus network. The main idea of the algorithm is to ask each worker store a local deviation statistics of distance between neighbor's model parameter and its own model parameter for every neighbor. These information will then be used to filter out bad model parameter from Byzantine attackers. We show that this algorithm can converge to the sample minimizer and the function can converge to the optimal value. We further provide numerical examples to illustrate the performance of the proposed algorithm.

Acknowledgement

I would like to thank all people who have helped, supported and encouraged me during my PhD life.

Foremost, I would like to express my sincere gratitude to my advisor Prof. Lifeng Lai for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. It was a great privilege and honor to work and study under his guidance.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Zhi Ding and Prof. Bernard C. Levy, for their effort spent in my research, their encouragement, insightful comments, and hard questions.

My sincere thanks also goes to Wenwen Tu and Zuoguan Wang, for offering me the summer internship opportunities in their groups and leading me working on diverse exciting projects in company, to help me become competitive in other field.

I thank my all my labmates in UC Davis: Wenwen Tu, Wenwen Zhao, Gaunlin Liu, Xinyi Ni, Minhui Huang, Yulu Jin, Xiaochuan Ma, Puning Zhao and Fuwei Li, for their help during my PhD and for all the fun we have had in the last many years.

Last but not the least, I would like to thank my family: my parents Limin Cao and Qi Chen, for giving birth to me at the first place and supporting me spiritually throughout my life. During the recent years, due to the COVID outbreak, I had very little time with my family. I would like to thank them for their understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In our daily life, machine learning is everywhere. Virtual personal assistants, like Siri, Alexa, will find information when we ask over voice. In virtual personal assitants, machine learning will help to collect and refine the information on the basis of your previous involvement. Traffic prediction is also an example of machine learning. When we use GPS navigation services, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. This helps in preventing the traffic jam.

In machine learning, optimization is the core part. Most machine learning problems reduce to optimization problems. In machine learning problem, when solving a problem for some set of data, people formulates the problem by selecting an appropriate family of models and massages the data into a format amenable to modeling. Then the model is typically trained by solving a core optimization problem that optimizes the variables or parameters of the model with respect to the selected loss function and possibly some regularization function.

1.1 Centralized and distributed optimization methods

In this section, we first introduce the optimization problem. Suppose that the data $X \in \mathcal{X} \subset \mathbb{R}^n$ is generated randomly from a unknown distribution $\mathcal{D}$ parameterized by unknown vector θ taken value from a set $\Theta \subset \mathbb{R}^d$. Our goal of optimization problem is to infer the unknown parameter θ

from data samples. In particular, consider a loss function $f : \mathcal{X} \times \Theta \to \mathbb{R}$, with $f(x, \theta)$ being the risk induced by data point x under the model parameter θ. We aim to find the model parameter θ^* that minimizes the population risk $F(\theta)$:

$$\theta^* \in \arg\min_{\theta \in \Theta} F(\theta) \triangleq \mathbb{E}[f(X, \theta)]. \tag{1.1}$$

When we know the distribution of X, the population risk can be evaluated exactly and θ^* can be computed by solving the above problem (1.1). However, in a typical machine learning problem, the distribution is unknown. To handle this, one normally approximates the population risk $F(\theta)$ from the observed data samples. In particular, we assume that there exist N independently and identically distributed (i.i.d.) data samples X_i, with $i = 1, 2, \cdots, N$, from the distribution $\mathcal{D}$. Instead of minimizing the population risk (1.1) directly, we minimize the empirical risk

$$\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} f(X_i, \theta). \tag{1.2}$$

In centralized optimization method, N data are all stored in central location, and computing is done at a central location. The central location stores all the data and controls all the processing when solving optimization problem (1.2).

Nowadays, a lot of challenges will arise when using centralized optimization method.

Firstly, as the amount of data keeps growing at a fast pace both in size and coordinate, it is challenging to fit all N data in one machine [36, 37, 55]. For example, automatic speech receoginition (ASR) can transcrible speech into correspoding text, which have been used in Apple Siri, Google Assistant. Using ASR, we need to solve an optimization problem to obtain high-performance models. The key is having access to lots of data: thousands of hours of speech. People always uses SWB2000 as the dataset when studying ASR, but SWB2000 has over 30 million training samples and 32,000 classes. The datasize so large that the data cannot be stored in a typical database, or even a single computer.

Secondly, in certain scenarios, data is naturally collected at different locations, and it is too costly to move all data to a centralized location [21]. For example, consider there is a model to treat patient, and the performance is roughly propotional to the number of patients trained on. Patient data are readily available at different hospitals, but unfortunately,sharing these data between hospitals is hampered by ethical, administrative, legal, and political barriers. In this example, we cannot share patient data, so we should consider to train model on local data in distributed method.

Thirdly, distributed optimization algorithms are useful to harness the computing power of multiple machines. When the data size is very large, even all data could be filled into one central location, the burden of computing is still very high, and it will slow the computing speed. Distributed optimization algorithm can reduce this burden by harnessing the computing power of multiple machines. Nowadays, there are many high performance computing systems built around multi-core processors, GPU-accelerators and computer clusters, which use multiple machines.

Due to these factors, decentralized optimization problem attract significant research interest [4, 6, 14, 15, 18, 22, 26, 29, 30, 32, 33, 42, 44, 53].

In a distributed optimization setup, there are two kinds of network: decentralized network and fully decentralized network.

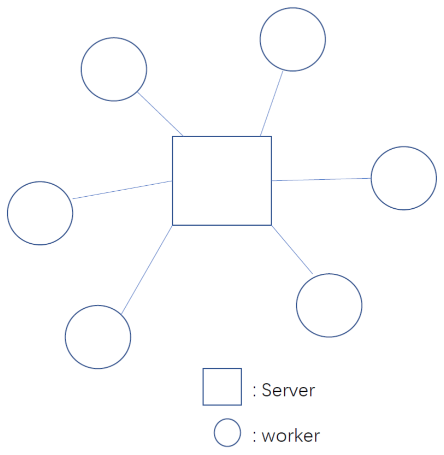

Figure 1.1: Dentralized network

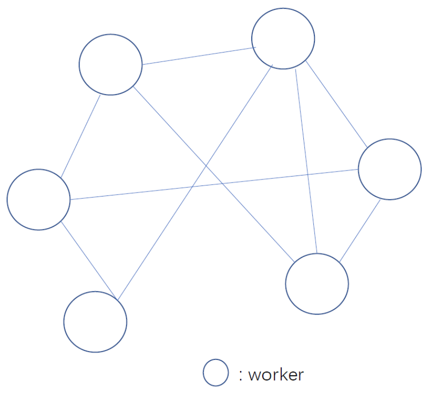

Figure 1.2: Fully decentralized network

In Figure 1.1, we can see the structure of a decentralized network in optimization problem. The decentralized network is built with one server and many workers. The information communication are happens between server and workers, and there are no information communication between workers directly. For example, server will broadcast parameter and order to all workers, then workers finish the work by themselves then respond to the server.

Figure 1.2 shows a fully decentralized network in optimization problem. The fully decentralized network works among several machines, instead of relying on a single central server. Workers will communicate with its neighbor workers. For example, a market economy is a decentralised economic system because it does not function via a central, economic plan but instead, acts through the distributed, local interactions in the market.

1.2 Zeroth, first and second order distributed optimization methods

When considering distributed optimization methods to solve empirical risk in (1.2), we will consider a decentralized optimization model in Figure 1.3, there are one server and m workers in the system. These N data samples are distributed into these m workers, and the server machine can communicate with all workers synchronously. Let $\mathcal{S}_j$ be the set of data samples that the j-th worker receives from the server. In a system with data shuffling, $\mathcal{S}_j$ changes over iterations, while in a system without shuffling, $\mathcal{S}_j$ is fixed. There are mainly three kinds of distributed optimization

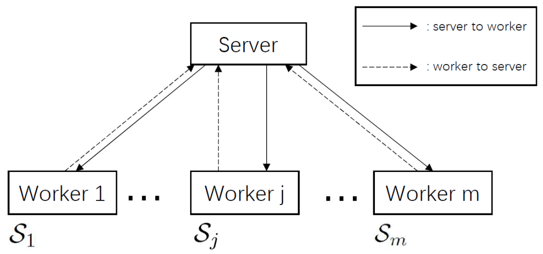

Figure 1.3: Distributed optimization model

methods: zeroth, first and second order distributed optimization methods.

1.2.1 First order distributed optimization methods

First order distributed optimizaiton methods employ the gradient information to find the local minimum of the loss function. Various distibuted first-order methods, which use gradient information and are often easy to implement, have been proposed in many existing works.

Distributed stochastic gradient descent (SGD) [4,32] is a drastic simplification when comparing with gradient descent. Instead of computing the gradient from all data exactly, each iteration the server estimates this gradient on the basis of a single randomly picked data. This stochastic process depends on the examples randomly picked at each iteration. Since the examples are randomly drawn from the ground truth distribution. The stochastic gradient descent directly optimizes the expected risk. However, because of the noisy approximation of the true gradient, the stepsize need to be decayed, then the convergence speed of stochastic gradient descent is limited. Therefore, SGD requires a large number of communication rounds which could be costly.

Distributed variance reduced SGD [11, 26, 34] reduces the variance of SGD, then the stepsize does not have to decay. In distributed variance reduced SGD, there is an update frequency m, the algorithm computes the average of all gradient based on model parameter every m iterations. When updating the model parameter in iteration t, the direction is computed from the gradient of a single random picked data at iteration t, the average recorded before t and the gradient of a single random picked data at iteration at iteration when the recorded happens.

Batch gradient descent (BGD) solves the convergence speed problem in SGD in another way. Each worker solves (1.2) using distributed gradient descent. In particular, at iteration t, each worker $j \in [1, m]$ calculates $\nabla \overline{f}^{(j)}(\theta_{t-1})$ based on local data

$$\nabla \overline{f}^{(j)}(\theta_{t-1}) = \frac{1}{|\mathcal{S}_j|} \sum_{i \in \mathcal{S}_j} \nabla f(X_i, \theta_{t-1}), \tag{1.3}$$

and sends it back to the server, where $|\mathcal{S}_j|$ is the size of data in j-th worker. After receiving information from all workers, the server updates the parameter using

$$\theta_t = \theta_{t-1} - \eta \sum_{i=1}^{m} w_i \nabla \overline{f}^{(j)}(\theta_{t-1}) \tag{1.4}$$

where $w_i = |\mathcal{S}_i|/N$ and sends the updated parameter θ_t to workers. Here η is the step size. This process continues until a certain stop criteria is satisfied. In this algorithm, the stepsize will not need to be decayed, which can improve the convergence speed.

Distributed coordinate descent method [36, 37] is also proposed to solve the problem when the coordinate is very large. In this method, they initially partition the coordinates $\{1, 2, ..., d\}$ into c sets and assign each set to a single worker. Each worker owns the coordinates belonging to its partition for the duration of the iterative process. Also, these coordinates are stored locally. The data matrix describing the problem is also partitioned in such a way. Now, at each iteration, each computer, independently from the others, chooses a random subset of τ coordinates from those they own, and computes and applies updates to these coordinates. Hence, once all computers are done, $c\tau$ coordinates will have been updated. The resulting vector, stored as c vectors of size $s = d/c$ each, in a distributed way, is the new iterate. This process is repeated until convergence.

In these methods, workers will compute gradient information and dual information from a small size of data and sends these information to its neighbor or server for updating. Since each worker only compute based on a small size of data, the first-order methods significantly reduce the amount of local computation.

1.2.2 Second order distributed optimization methods

In distributed optimization problems, workers must communicate with its neighbors or its server. These first order distributed optimization methods may require a far greater number of iterations for communication. Some algorithms also require synchronization in every iteration for parameter updating. In order to mitigate the negative impact of the large number of iterations for distributed optimization, communication-efficient second-order methods have also been proposed [17, 35, 38, 40, 48, 55].

Shamir et al. [38] proposed DANE algorithm to minimize a cost function consisting of local loss function, local gradient and global gradient on each worker. The method performs two distributed averaging computations per iteration, and outputs a predictor which, under suitable parameter choices, converges to the optimum. DANE maintains an agreed-upon iterate model parameter, which is synchronized among all machines at the end of each iteration. In each iteration, we first compute the gradient at the current iterate, by averaging the local gradients. Each machine then performs a separate local optimization, based on its own local objective function and the computed global gradient, to obtain a local iterate model parameter. These local iterates are averaged to obtain the centralized iterate model parameter.

DiSCO in [55] contains an outer-loop and an inner loop. The outer-loop employs an inexact damped Newton method to minimize loss function. But the number of steps needed to reach the superlinear convergence zone still depends on the condition number. To solve this problem, DiSCO uses the machinery of self-concordant functions, where the iteration complexity of the inexact damped Newton method has a much weaker dependence on the condition number. The inner loop, DiSCO employs a distributed preconditioned conjugate gradient (PCG) method to compute an inexact Newton step. In each iteration in inner loop, the server carries out the classical PCG algorithm and each worker computes the local gradients and Hessians and perform matrix-vector multiplication and send it to the server.

GIANT [48] requires the exact global gradient and approximate Newton direction.

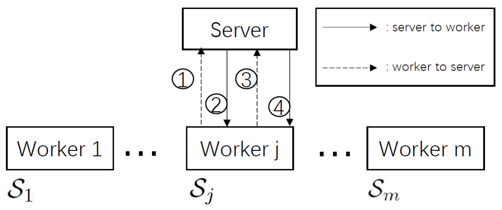

Figure 1.4: Information flow of GIANT algorithm in [48]. ①: $\nabla \overline{f}^{(j)}(\theta_{t-1})$; ②: $\nabla \overline{f}(\theta_{t-1})$; ③ : $H_{j,t-1}^{-1} \nabla \overline{f}(\theta_{t-1})$; ④ : θ_t. In this figure we only draw the information flow between machine j and the server, all other machines have similar information flow.

Global improved approximate Newton method (GIANT) solves (1.2) using approximate Newton's method by two-steps computing and communication between the server and workers. Figure 1.4 illustrates information flow between the server and workers during iteration t. In particular, at iteration t, each worker $j \in [1, m]$ first calculates $\nabla \overline{f}^{(j)}(\theta_{t-1})$ based on local data:

$$\nabla \overline{f}^{(j)}(\theta_{t-1}) = \frac{1}{|\mathcal{S}_j|} \sum_{i \in \mathcal{S}_j} \nabla f(X_i, \theta_{t-1}), \tag{1.5}$$

and sends it to the server, where $|\mathcal{S}_j|$ is the size of data in the j-th worker and we assume the size of data in each worker is equal. After receiving information from all workers, the server computes the gradient information using

$$\nabla \overline{f}(\theta_{t-1}) = \frac{1}{m} \sum_{j=1}^{m} \nabla \overline{f}^{(j)}(\theta_{t-1}), \tag{1.6}$$

and broadcasts the $\nabla \overline{f}(\theta_{t-1})$ to workers. After receiving $\nabla \overline{f}(\theta_{t-1})$, each worker $j \in [1, m]$ calculates $H_{j,t-1}^{-1} \nabla \overline{f}(\theta_{t-1})$ based on local data and $\nabla \overline{f}(\theta_{t-1})$ where

$$H_{j,t-1} = \nabla^2 \overline{f}^{(j)}(\theta_{t-1}) = \frac{1}{|\mathcal{S}_j|} \sum_{i \in \mathcal{S}_j} \nabla^2 f(X_i, \theta) \tag{1.7}$$

and sends it back to the server. After receiving Newton's direction information from all workers, the server updates model parameter by

$$\theta_t = \theta_{t-1} - \frac{1}{m} \sum_{j=1}^{m} H_{j,t-1}^{-1} \nabla \overline{f}(\theta_{t-1}), \tag{1.8}$$

where they use $\frac{1}{m} \sum_{j=1}^{m} H_{j,t-1}^{-1} \nabla \overline{f}(\theta_{t-1})$ to approximate the true Newton direction $(\frac{1}{N} \sum_{i=1}^{N} \nabla^2 f(X_i, \theta_{t-1}))^{-1} \nabla \overline{f}(\theta_{t-1})$, since the inverse of all data Hessian matrix cannot be computed through distributed setup. After updating, the server broadcasts the updated parameters to the workers. This process continues until a certain stop criteria is satisfied. Algorithm 1.1 summarizes steps involves in the GIANT algorithm.

Algorithm 1.1 GIANT algorithm [48]

```
Parameter server:
Initialize randomly selects $\theta_0 \in \Theta$.
repeat
    1: Broadcasts the current model parameter estimator $\theta_{t-1}$ to all workers;
    2: Waits to receive gradients from the $m$ workers;
    3: Computes $\nabla\overline{f}(\theta_{t-1}) = \frac{1}{m}\sum_{j=1}^{m}\nabla\overline{f}^{(j)}(\theta_{t-1})$;
    4: Broadcasts the current gradient estimator $\nabla\overline{f}(\theta_{t-1})$ to all workers;
    5: Waits to receive estimators from the $m$ workers;
    6:Updates $\theta_t = \theta_{t-1} - \eta\frac{1}{m}\sum_{j=1}^{m} H_{j,t-1}^{-1}\nabla\overline{f}(\theta_{t-1})$;
until $\|\theta_t - \theta^*\| \leq \epsilon$.
Worker $j$:
1: Receives model parameter estimator $\theta_{t-1}$, computes
the gradient $\nabla\overline{f}^{(j)}(\theta_{t-1})$, sends it back;
2: Receives gradient estimator $\nabla\overline{f}(\theta_{t-1})$, computes the parameter $H_{j,t-1}^{-1}\nabla\overline{f}(\theta_{t-1})$, sends it back;
```

ADN in [17] is built on an adaptive block-separable approximation of the objective function. Based on the block-separable approximation, the objective function is divided into local objective function model. Then each worker compute the direction locally parallel and server will average it to get a global direction. After each iteration, ADN will adjust parameter in model based on the agrrement between the model function and the ojective function for the current iteration. By this adjustment, the Hessian approximation will be more closer to the true Hessian matrix.

1.2.3 Zeroth order distributed optimization methods

In both first and second order distributed optimization methods, we assume the function is differentiable, but in some situations, for example, when only black-box procedures are available for computing, we do not have access to the gradient of the object loss function. Zeroth-order optimization use the approximate gardient to update the parameter.

Tang's method [43] cosider the distributed optimization methods in a two direction network. Each worker will compute the maps to approximate gradient by difference quotients along d orthogonal directions. During communication, workers will send model parameter information to its neighbors'. Then each worker will weights it neighbors' model information to updates the local variable.

ZOO-ADMM in [27] adopts a random gradient estimator to estimate the gradient. By replacing the gradient in the Online-ADMM, [27] proposes an extension of O-ADMM, which is call ZOO-ADMM.

[54] considered distributed zero-order methods for constrained convex optimization. It is somewhat different from Tang's method and ZOO-ADMM. Instead of establishing a gradient-free method based on Euclidean projection, [54] carries Bregman non-Euclidean structure for solving the distributed convex optimization over time-varying network. It uses Bregman divergence instead of using the classical Euclidean norm, leading to the non-Euclidean projection feature of the proposed algorithms. As a result, the classical distributed zeroth-order projection algorithm is generalized to the non-Euclidean circumstance.

1.3 Byzantine attack

Most of the existing works in distributed optimization assume that workers behave honestly and follow the protocol. However, in practice, by using distributed optimization methods, there is a risk that some of the workers might be compromised. Byzantine attack to distributed optimization was first studied in [24]. When Byzantine attack happens, compromised workers will send wrong information during communication with its neighbor or its server. Compromised workers can prevent the convergence of the optimization algorithms or lead the algorithms to converge to a value chosen by these attackers by modifying or falsifying intermediate results during the execution of optimization algorithms. In Figure 1.5, we use distributed gradient descent as an example to illustrate this. In this figure, we consider gradient descent in one iteration, the gradient information computed by honest workers are toward right, but a single attacker sends wrong information, which is toward left, to the server. When server averages the received information, it will go to the wrong direction, then the algorithm cannot converge.

There have been some interesting recent works to design distributed machine learning

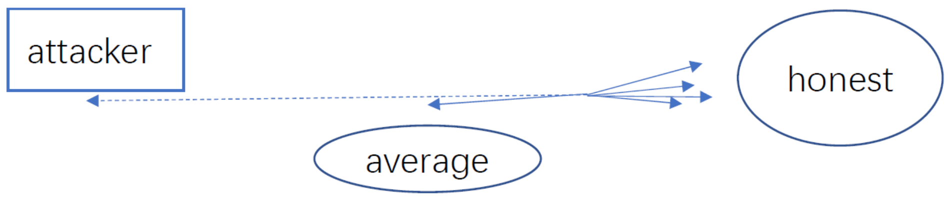

Figure 1.5: Example of attack in averagement.

algorithms [2, 3, 12, 13, 16, 18, 30, 41, 49–53] that can deal with Byzantine attacks. The main idea of these works is to compare information received from all workers, and compute a quantity that is robust to attackers for algorithm update.

The algorithm in [13] uses the geometric median mean of gradient information received from workers for parameter update. This algorithm employs the idea of Batch Gradient Descent. In Batch Gradient Descent, the parameter server sends the current model parameter estimate to all working machines, then each working machine computes the gradient based on all locally available data, and then sends the gradient back to the parameter server. Finally, the parameter server averages the received gradients and performs a gradient descent step. Since taking the average has no ability to defend against Byzantine attackers. The parameter server in this algorithm aggregates the local gradients reported by the working machines by three steps: (1) it partitions all the received local gradients into k batches, where k is smaller than the number of workers and computes the mean for each batch, (2) it computes the geometric median of the k batch means, and (3) it performs a gradient descent step using the geometric median. By adjusting value k, this method can trade the balance between the number of Byzantine attackers algorithm can tolerate and the accuracy of this algorithm. When $k = m$, the aggregate method is gometric meidan. When $k = 1$, the aggregate method is average, which has best accuray but cannot defend against Byzantine attacker at all.

The algorithm Krum in [3] uses a different way to aggregate gradient from all workers. In this algorithm, the server will collect all gradient vector from all workers, then for each received gradient g_i, it computes $s_i = \sum_{i \to j} \|g_i - g_j\|^2$, where g_j belongs to the $m - p - 2$ closest vector of

g_i and m is the number of workers and p is the number of compromised workers, finally the server choose g_i which has the smallest score s_i as the estimated gradient for parameter updating.

Alistarh et al. [2] proposed a Byzantine-resilient SGD algorithm, in which at each iteration the server combines the current and past gradient information from each worker to compute next update, to solve convex problem with high dimension. For worker i, the server will collect and store the received gradient information $g_i^{(k)}$, where k is the number of iteration, then the server will compute $B_i = \sum_{t=1}^{k} g_i^{(t)}/k$ and $A_i = \sum_{t=1}^{k} \langle g_i^{(t)}, \theta_t - \theta_1 \rangle /k$ in each iteration, and the server will use the median B_{med} and A_{med} as the ground truth to check B_i and A_i in each iteration. If the server discover some worker is Byzantine attacker, the the server will remove it from future consideration. By this algorithm, the server can find good worker, then update parameter based on the those.

Xie et al. [49] studied the dimensional Byzantine-resilient algorithms, where Byzantine values can happen in all workers but for each dimension, the number of Byzantine valued must be less than the number of correct ones. The server receives all gradients $\{g_1, ..., g_m\}$ and sorts values in dimension j, it compute $Trmean_j = \frac{1}{m-2p} \sum_{k=b+1}^{m-b} g_{k,j}$, then it sort the values by $\{|g_{1,j}/Trmean_j - Trmean_j|, ..., |g_{m,j}/Trmean_j - Trmean_j|\}$ and average the first $m-b$ number of $g_{1,j}/Trmean_j$ values as the update gradient value in dimension j. For high dimension, the server applies it in coordinate-wise manner. Then using it to update parameter.

Yin et al. [51] proposed a median-based algorithm that uses only one communication round to perform parameter updates. In each parallel iteration of the algorithms, the master machine broadcasts the current model parameter to all worker machines. The honest workers compute the gradients of their local loss functions and then send the gradients back to the master machine. The Byzantine workers may send any messages of their choices. The master server then performs a gradient descent update on the model parameter, using either the coordinate-wise median or trimmed mean of the received gradients, where the coordinate-wise median is a vector with its k-th coordinate being the one-dimensional median for each $k \in [d]$.

Chen et al. [12] proposed DRACO algorithm that uses ideas from coding theory to determine which machines are under attack. In DRACO, each worker is allocated a subset of the data set, which is redundant. Each workers computes redundant gradients, encodes them, and sends the resulting vector to the parameter server. These received vectors then pass through a decoder that detects where the adversaries are through the encoded redundant gradient information and removes their effects from the updates. The output of the decoder is the true sum of the gradients. The parameter applies the updates to the parameter model and we then continue to the next iteration.

Su et al. [41] proposed an approximate gradient descent algorithm that employs iterative filtering for robust gradient aggregation in high dimensional estimation. The parameter will collect all gradient information and using an iterative filtering algorithm to find the updating direction. The iterative filter will construct a cost function with parameter (W, U) by the received information and finding the saddle point (W^*, U^*), then using it iteratively to find the direction along which all data points are spread out the most, and filters away data points which have large residual errors projected along this direction. Finally, the server will use this direction to update parameter.

These algorithms in [2,3,12,13,16,41,49,51,52] can successfully converge to the neighborhood of the population minimizer even if up to half of all workers are compromised. However, once more than half of the workers are compromised, the algorithms in these interesting work will not converge.

As machine learning algorithms are increasingly deployed in security and safety critical applications, it is important to consider the robustness of these algorithms in adversarial environments where we need to make less or no assumption about the attackers (including the assumption that less than half of the workers are attackers in the distributed learning) [20]. These attacks may be achieved by a variety of ways including but not limited to: vulnerable communication channels, poisoned datasets, or virus. In 2006, 65% of companies surveyed in the CSI/FBI Computer Crime and Security Survey [19] reported that they had been attacked by virus. Once captured by virus, these devices can be used to attack the network from inside. For example,

Xie et al. [50] considered the case that the number of Byzantine worker is arbitrarily large and proposes an algorithm named Zeno. The server will have n_r number i.i.d. samples draw from the unknown distribution. Then we can denote $f_r(\theta) = \frac{1}{n_r} f(\theta; x_i)$, For any update gradient estimator u, based on the current parameter θ, learning rate eta, and a constant weight $\rho > 0$, Zeno defines its stochastic descendant score as follows:

$$Score(u; \theta) = f_r(\theta) - f_r(\theta - \rho u) - \|u\|. \tag{1.9}$$

This score composed of two parts: the estimated descendant of the loss function, and the magnitude of the update. The score increases when the estimated descendant of the loss function $f_r(\theta) - f_r(\theta - \rho u)$ increases. The score decreases when the magnitude of the update $\|u\|$ increases. Intuitively, the larger descendant suggests faster convergence, and the smaller magnitude suggests a smaller step size. Even if a gradient is Byzantine, a smaller step size makes it less harmful and easier to be cancelled by the correct gradients. With this score, the server will first sorts the gradient by a stochastic descendant score then averages the $m - b$ gradients with highest score, in which m is the total number of workers and b is an important parameter in the algorithm. The algorithm must have at least one good worker, it cannot solve the problem when server is isolated. Furthermore, in order to properly set the parameter b, Zeno must know an upper bound on the number of Byzantine workers. In addition, if b is selected to be larger than the true number of attackers, the algorithm may not benefit from all good workers.

1.4 Main contributions

In this research thesis, we propose robust distributed first, second order and zeroth order algorithms that can converge to the neighborhood of the population minimizer when there are compromised workers.

1.4.1 First order distributed optimization method

The main idea for our proposed robust distributed gradient descent algorthm is to ask the server to randomly select a *small* subset of clean data and compute a noisy gradient based on this small dataset. Even though the computed gradient is very noisy, it can be used as a proxy of the ground truth to filter out information from attackers. In particular, once the server receives gradient information from workers, it compares the gradient information from each worker with the noisy gradient it has computed. If the distance between the gradient from worker and the noisy gradient computed by itself is small, the server accepts the gradient information from that worker as authentic. After the comparison step, the server then computes the average of all accepted gradient and its own noisy gradient as the final estimated gradient for updating. We prove that the algorithm can converge to the neighborhood of the population minimizer regardless of the number of compromised workers. We show this result by proving that the distance between the estimated gradient and the true gradient can be universally bounded. In the analysis, we consider two different scenarios. In the first scenario, we do not assume any knowledge about the number of attackers. We provide a convergence proof in this case with minimal assumption about attackers. In the second scenario, we assume that the number of attackers is bounded from above by a constant p. We note that, here p is an upper bound of the number of attackers, it is not the exact number of attackers. Hence, this additional knowledge is not too restrictive. With this additional knowledge, we provide a modified algorithm that has a tighter convergence bound. The results in this part have been published in [8, 9].

1.4.2 Second order distributed optimization method

For robust distributed second order algorthm, we proposed two algorithms to defend against Byzantine workers. The first method, named median-based approximate Newton's method (MNM), can converge to the neighborhood of the population minimizer when less than half of the workers are Byzantine attackers. The main idea is to use geometric median to aggregate information from workers. The geometric median enables the server to mitigate the impact of attackers when up to half of the workers are Byzantine attackers. Using these, we prove that the algorithm can converge to the neighborhood of the population minimizer when q, the number of Byzantine attackers, is less than $m/2$ with m being the total number of workers. We show this result by proving that the distance between the approximate Newton's direction and true Newton's direction can be universally bounded. However, once $q > m/2$, MNM fail to converge.

The second method, named comparison-based approximate Newton's Method (CNM), can converge to the neighborhood of the population minimizer server regardless whether q is larger or smaller than $m/2$. Compared with MNM, CNM requires additional computation at the server. The main idea is similiar to proposed robust distributed gradient descent algorithm, instead of only compute noisy gradient, it also compute the noisy Newton direction as an approximation of the ground truth to filter out information from attackers during two times communication in each iteration. We prove that CNM can converge to the neighborhood of population minimizer regardless number of Byzantine attackers. The results in this part have been published in [10].

1.4.3 Zeroth order distributed optimization method

For robusted distributed zeroth order algorithm, we propose a new robust zeroth-order information based distributed optimization algorithm that is robust to Byzantine attacks. We name the method as zeroth-order adversarially robust alternating direction method of multipliers (ZOAR-ADMM). At each iteration, each worker will first receive model parameter from its neighbors. Then each worker will test received parameter information by computing the distance from the received parameter to the model parameter computed using local data, and then sum all such distances obtained in history to build a deviation statistic for all neighbor workers. If the deviation statistic computed for its neighbor worker is smaller than a specially designed threshold, the worker will accept the model parameter from that neighbor. If the deviation statistic is larger than the threshold, the worker will reject the model parameter and decide that worker to be an attacker. After testing, each worker will first update dual variable by using accepted model parameter, then compute temporary model parameter based on accepting parameter and using deterministic gradient approximation from its own data and update new model parameter then broadcast it to its neighbors. By this method, we prove that the algorithm can solve the optimization problem and the objective function can converge to the minimum value. We show this result by first investigating how the distance between model parameter and optimal value is affected by the attack vector generated by the attackers, and then carefully analyzing how the proposed testing method can mitigate these effects and eventually proving that the value of objective function of the proposed algorithm will converge to the optimal value despite the presence of Byzantine attackers. The results in this part have been submitted in [7].

Chapter 2

Distributed Gradient Descent Algorithm Robust to an Arbitrary Number of Byzantine Attackers

2.1 Introduction

In this chapter, we focus on first order methods. In particular, we aim to design gradient descent methods that are robust to Byzantine attackers, and analyze their convergence rates. We consider two different scenarios. In the first scenario, we do not assume any knowledge about the number of attackers. We provide a convergence proof in this case with minimal assumption about attackers. In the second scenario, we assume that the number of attackers is bounded from above by a constant p. We note that, here p is an upper bound of the number of attackers, it is not the exact number of attackers. Hence, this additional knowledge is not too restrictive. With this additional knowledge, we provide a modified algorithm that has a tighter convergence bound.

This chapter is organized as follows. In Section 2.2, we describe the model. In Section 2.3, we describe the proposed robust gradient descent algorithm. In Section 2.4, we analyze the convergence property of the proposed algorithm. In Section 2.5, we provide numerical examples to validate the theoretic analysis and show that we can benefit from the good workers to obtain a better convergence accuracy. Finally, we offer several concluding remarks in Section 2.6. The proofs are collected in Appendix.

2.2 Model

In this chapter, we assume that $F(\theta)$ discrible in introduction satisfies the following typical assumption.

Assumption 1. *The population risk function $F : \Theta \to \mathbb{R}$ is L-strongly convex, and differentiable over Θ with M-Lipschitz gradient. That is for all $\theta, \theta' \in \Theta$,*

$$F(\theta') \geq F(\theta) + \langle \nabla F(\theta), \theta' - \theta \rangle + h \parallel \theta' - \theta \parallel^2 /2, \tag{2.1}$$

and

$$\parallel \nabla F(\theta') - \nabla F(\theta) \parallel \leq M \parallel \theta' - \theta \parallel,$$

in which $\parallel \cdot \parallel$ is the ℓ_2 norm and $0 < h \leq M$.

We consider a system with Byzantine workers, in which an unknown subset of workers might be comprised. Furthermore, the set of compromised workers might change over time. If a worker is compromised, instead of the gradient calculated from local data, it can send arbitrary information to the server. Now, we will have a distributed optimization model similiar to figure 1.3, but the receiving information may not be correct. From figure 2.1, we let $\mathcal{B}_t$ denote the set of compromised workers at iteration t, the server receives data $g^{(j)}(\theta_{t-1})$ from j-th worker with

$$g^{(j)}(\theta_{t-1}) = \begin{cases} \nabla \overline{f}^{(j)}(\theta_{t-1}) & j \notin \mathcal{B}_t \\ \star & j \in \mathcal{B}_t \end{cases}, \tag{2.2}$$

in which $\star$ denotes an arbitrary vector chosen by the attacker.

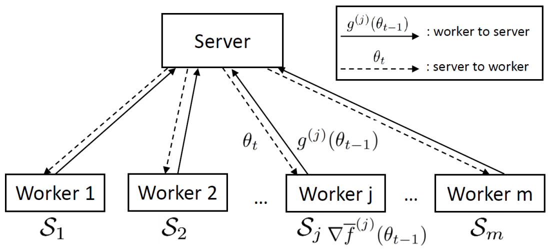
Server
$g^{(j)}(\theta_{t-1})$: worker to server
θ_t : server to worker
θ_t
$g^{(j)}(\theta_{t-1})$
Worker 1
Worker 2
...
Worker j
...
Worker m
$\mathcal{S}_1$
$\mathcal{S}_2$
$\mathcal{S}_j$ $\nabla \overline{f}^{(j)}(\theta_{t-1})$
$\mathcal{S}_m$

Figure 2.1: Distributed optimization model with Byantine attacker

We assume there are up to p Byzantine attackers in the system. Note that, in this chapter, p is not the exact number of attackers, it is merely an upper bound on the number of attackers. In this case with Byzantine attackers, if one continues to use the classic batch gradient as in (1.4), the algorithm will fail to converge even if there is only one attacker [3, 13]. As discussed above, [3, 13] designed algorithms that converge to the neighborhood of the population minimizer if the number of compromised machines p is less than $m/2$ (i.e., more than half of the machines are not compromised).

The goal of the chapter is to design a robust batch gradient descent algorithm that can tolerate *any number* of Byzantine attackers.

2.3 Algorithm

In this section, we describe our algorithm that can deal with an arbitrary number of Byzantine attackers under two scenarios: p being unknown and knowing the value of p.

2.3.1 Unknown p

In the first scenario, we do not have any knowledge about p. Main steps of the algorithm under the first scenario is listed in Table 2.1. The main idea of our algorithm is to ask the server to randomly select a small set of data points $\mathcal{S}_0$ at very beginning, where $|\mathcal{S}_0| \leq \min_{j\in[1,m]} |\mathcal{S}_j|$. Once $\mathcal{S}_0$ is selected, it is fixed throughout the algorithm. Then at each iteration t, the server calculates a

Algorithm 2.1 Proposed algorithm with unknown p

```
Parameter server:
Initialize: Randomly selects $\theta_0 \in \Theta$.
repeat
    randomly selects $\mathcal{S}_0$;
    1: Broadcasts the current model parameter estimator $\theta_{t-1}$ to all workers;
    2: Waits to receive gradients from the $m$ workers, where $g^{(j)}(\theta_{t-1})$ denote the value received
    from worker $j$;
    3: Computes $\nabla\overline{f}^{(0)}(\theta_{t-1})$ using $\mathcal{S}_0$;
    4: Compares $g^{(j)}(\theta_{t-1})$ with $\nabla\overline{f}^{(0)}(\theta_{t-1})$. If $\| g^{(j)}(\theta_{t-1}) - \nabla\overline{f}^{(0)}(\theta_{t-1}) \| \leq \xi \| \nabla\overline{f}^{(0)}(\theta_{t-1}) \|$,
    the server accepts it and sets it to be $q_t^{(l)}(\theta_{t-1})$;
    5: Assume the acceptable value are in $\mathcal{V}_t$, then $G(\theta_{t-1}) \leftarrow \sum_{l\in\mathcal{V}_t} w_l q_t^{(l)}(\theta_{t-1}) +$
    $w_0\nabla\overline{f}^{(0)}(\theta_{t-1})$;
    6:Updates $\theta_t \leftarrow \theta_{t-1} - \eta G(\theta_{t-1})$;
until $\|\theta_t - \theta^*\| \leq \epsilon$.
Worker $j$:
1: Receives model parameter estimator $\theta_{t-1}$, computes the gradient $\nabla\overline{f}^{(j)}(\theta_{t-1})$;
2: If worker $j$ is honest, it sends $\nabla\overline{f}^{(j)}(\theta_{t-1})$ back to the server; If worker $j$ is compromised, it
sends the value determined by the attacker;
```

noisy gradient using data points in $\mathcal{S}_0$:

$$\nabla\overline{f}^{(0)}(\theta_{t-1}) = \frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i, \theta_{t-1}).$$

Different choices of the size of $\mathcal{S}_0$ will strike a tradeoff between convergence speed and computational complexity.

The server then compares $g^{(j)}(\theta_{t-1})$ received from worker j with $\nabla\overline{f}^{(0)}(\theta_{t-1})$. The server will accept $g^{(j)}(\theta_{t-1})$ as authentic value and use it for further processing, if

$$\| g^{(j)}(\theta_{t-1}) - \nabla\overline{f}^{(0)}(\theta_{t-1}) \| \leq \xi \| \nabla\overline{f}^{(0)}(\theta_{t-1}) \|, \tag{2.3}$$

where ξ is a constant. The choice of ξ will impact the proposed scheme. Roughly speaking, choosing a smaller ξ can limit the effect of an attack, but it may also reject more correct information from honest workers. On the other hand, a larger ξ can increase the probability of data from honest workers being accepted, but it will also increase the probability of accepting information from attackers. We will discuss how to choose this parameter in the analysis.

Assuming there are $|\mathcal{V}_t|$ values (which is a random variable) being accepted after the comparison step at iteration t, we denote these values by $q_t^{(1)}(\theta_{t-1})$, ..., $q_t^{(|\mathcal{V}_t|)}(\theta_{t-1})$. Then the server updates the parameters as $\theta_t = \theta_{t-1} - \eta G(\theta_{t-1})$, where

$$G(\theta_{t-1}) = \sum_{l \in \mathcal{V}_t} w_l q_t^{(l)}(\theta_{t-1}) + w_0 \nabla \overline{f}^{(0)}(\theta_{t-1}), \tag{2.4}$$

where $w_l = \frac{|\mathcal{S}_l|}{\sum_{i\in\mathcal{V}_t}|\mathcal{S}_i| + |\mathcal{S}_0|}$.

2.3.2 Known p

In the second scenario, we assume that we know p. We again note that here p is an upper bound of the number of attackers, it is not the exact number of attackers. Hence, this additional knowledge is not too restrictive. With this additional knowledge, we modify the algorithm at the server side above slightly. The worker side remains the same. This modification will allow us to prove a tighter bound in the convergence analysis section. Main steps of the modified algorithm at the server side is listed in Table 2.2. The main difference with the algorithm in Table 2.1 is that we now sort the gradient information accepted by the server in an increasing order by $\|q^{(i)}(\theta_t) - \nabla \overline{f}^{(0)}(\theta_t)\|$, then keep the first $m - p$ gradient value in a set (as we know the number p). We call this set $\mathcal{U}_t$ at iteration t. Using this notation, the gradient used for updating at the server can be written as

$$G(\theta_{t-1}) = \sum_{j \in \mathcal{U}_t} w_j q_t^{(j)}(\theta_{t-1}) + w_0 \nabla \overline{f}^{(0)}(\theta_{t-1}),$$

where $w_j = \frac{|\mathcal{S}_j|}{\sum_{i\in\mathcal{U}_t}|\mathcal{S}_i| + |\mathcal{S}_0|}$.

Algorithm 2.2 Proposed algorithm with known p

```
Parameter server:
Initialize: Randomly selects $\theta_0 \in \Theta$.
repeat
    randomly selects $\mathcal{S}_0$;
    1: Broadcasts the current model parameter estimator $\theta_{t-1}$ to all workers;
    2: Waits to receive gradients from the $m$ workers, where $g^{(j)}(\theta_{t-1})$ denote the value received from worker $j$;
    3: Computes $\nabla\overline{f}^{(0)}(\theta_{t-1})$ using $\mathcal{S}_0$;
    4: Compares $g^{(j)}(\theta_{t-1})$ with $\nabla\overline{f}^{(0)}(\theta_{t-1})$. If $\| g^{(j)}(\theta_{t-1}) - \nabla\overline{f}^{(0)}(\theta_{t-1}) \| \leq \xi \| \nabla\overline{f}^{(0)}(\theta_{t-1}) \|$, the server accepts it and sets it to be $q_t^{(l)}(\theta_{t-1})$;
    5: After accepting, the server collects $m-p$ gradient information which are closest to its own. Then $G(\theta_{t-1}) \leftarrow \sum_{j\in\mathcal{U}_t} w_j q_t^{(j)}(\theta_{t-1}) + w_0\nabla\overline{f}^{(0)}(\theta_{t-1})$.
    6:Updates $\theta_t \leftarrow \theta_{t-1} - \eta G(\theta_{t-1})$.
until $\|\theta_t - \theta^*\| \leq \epsilon$.
```

2.4 Convergence analysis

In this section, we analyze the convergence property of the proposed algorithm. We consider two different scenarios: 1) In scenario 1, we do not have any knowledge about p; 2) In scenario 2, we assume that we know the value of p.

In this section, we will prove results that hold simultaneously for all $\theta \in \Theta$ with a high probability. Hence, in the following, we will drop subscript $t-1$. Before presenting detailed analysis, here we describe the high level ideas. It is well known that if $\nabla F(\theta)$ is available, then the gradient descent algorithm will converge to θ^* exponentially fast. The main idea of our proof is to show that, regardless of the number of attackers, the distance between $G(\theta)$ and $\nabla F(\theta)$ is universally bounded in Θ in both scenarios. Hence, $G(\theta)$ is a good estimate of $\nabla F(\theta)$. As the result, we can then show that the proposed algorithm converges to the neighborhood of the population minimizer.

2.4.1 Scenario 1: no assumption on p

In this scenario, we assume that we do not know even the upperbound on the number of bad workers. We first show that $\|G(\theta) - \nabla F(\theta)\|$ is universally bounded in Θ regardless the number of attackers.

Lemma 2.1. *For an arbitrary number of attackers, the distance between $G(\theta)$ and $\nabla F(\theta)$ is bounded as*

$$\|G(\theta) - \nabla F(\theta)\| \;\; \leq \;\; (1+\xi)\|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\| + \xi\|\nabla F(\theta)\|, \forall \theta. \tag{2.5}$$

Proof. Please see Appendix A.1. □

We next need to bound the two terms in the right hand side of (2.5). The term $\|\nabla F(\theta)\| = \|\nabla F(\theta) - \nabla F(\theta^*)\|$ can be bounded using the M-Lipschitz gradient assumption in Assumption 1. In the following, we show that the term $\|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\|$ can also be bounded. For this, we need to present several assumptions and intermediate results. These assumptions are similar to those used in [13], [41, 51], and proofs of some lemmas follow closely that of [13].

Assumption 2. *There exist positive constants σ_1 and α_1 such that for any unit vector $v \in B$, $\langle \nabla f(X, \theta^*), v\rangle$ is sub-exponential with σ_1 and α_1, that is,*

$$\sup_{v \in B} \mathbb{E}[\exp(\lambda \langle \nabla f(X, \theta^*), v\rangle)] \leq e^{\sigma_1^2 \lambda^2/2}, \forall |\lambda| \leq 1/\alpha_1,$$

where B denotes the unit sphere $\{v : \|v\|_2 = 1\}$.

With this assumption, we first have the following lemma that shows $\frac{1}{|\mathcal{S}_0|}\sum_{i \in \mathcal{S}_0} \nabla f(X_i, \theta^*)$ concentrates around $\nabla F(\theta^*)$.

Lemma 2.2. *Under Assumption 2, for any $\delta \in (0,1)$, let*

$$\Delta_1 = \sqrt{2}\sigma_1\sqrt{(d \log 6 + \log(3/\delta))/|\mathcal{S}_0|}, \tag{2.6}$$

and if $\Delta_1 \leq \sigma_1^2/\alpha_1$, then

$$Pr\left\{\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i \in \mathcal{S}_0} \nabla f(X_i, \theta^*) - \nabla F(\theta^*)\right\| \geq 2\Delta_1\right\} \leq \frac{\delta}{3}.$$

Proof. Please see Appendix A.2. $\square$

Second, we define gradient difference $h(x, \theta) \triangleq \nabla f(x, \theta) - \nabla f(x, \theta^*)$ and assume that for every θ, $h(x, \theta)$ normalized by $\| \theta - \theta^* \|$ is also sub-exponential.

Assumption 3. *There exist positive constants σ_2 and α_2 such that for any $\theta \in \Theta$ with $\theta \neq \theta^*$ and any unit vector $v \in B$, $\langle h(X, \theta) - \mathbb{E}[h(X, \theta)], v \rangle / \| \theta - \theta^* \|$ is sub-exponential with σ_2 and α_2, that is,*

$$\sup_{\theta \in \Theta, v \in B} \mathbb{E}\left[\exp\left(\frac{\lambda \langle h(X, \theta) - E[h(X, \theta)], v \rangle}{\|\theta - \theta^*\|}\right)\right] \leq e^{\sigma_2^2 \lambda^2 / 2}, \ \forall |\lambda| \leq \frac{1}{\alpha_2}.$$

This allows us to show that $\frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} h(X_i, \theta)$ concentrates on $\mathbb{E}[h(X, \theta)]$ for every fixed θ.

Assumption 2 and 3 ensure that random gradient $\nabla f(\theta)$ has good concentration properties, i.e., an average of $|\mathcal{S}_0|$ $i.i.d$ random gradients $\frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \nabla f(X_i, \theta)$ sharply concentrates on $\nabla F(\theta)$ for every fixed θ.

Lemma 2.3. *If Assumption 3 holds, for any $\delta \in (0, 1)$ and any fixed $\theta \in \Theta$, let $\Delta_1' = \sqrt{2}\sigma_2 \sqrt{(d \log 6 + \log(3/\delta))/|\mathcal{S}_0|}$, and if $\Delta_1' \leq \sigma_2^2/\alpha_2$, then*

$$Pr\left\{\left\|\frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} h(X_i, \theta) - \mathbb{E}[h(X, \theta)]\right\| \geq 2\Delta_1' \|\theta - \theta^*\|\right\} \leq \frac{\delta}{3}.$$

Proof. Please see Appendix A.3. $\square$

Assumption 4. *For any $\delta \in (0, 1)$, there exists an $M' = M'(\delta)$ such that*

$$Pr\left\{\sup_{\theta, \theta' \in \Theta : \theta \neq \theta'} \frac{\|\nabla f(X, \theta) - \nabla f(X, \theta')\|}{\|\theta - \theta'\|} \leq M'\right\} \geq 1 - \frac{\delta}{3}.$$

Assumption 4 ensures that $\nabla f(X, \theta)$ is M'-Lipschitz with high probability.

With these assumptions and intermediate lemmas, we are ready to state our universal bound for $\|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\|$.

Proposition 2.1. *Suppose Assumptions 2-4 hold, and* $\Theta \subset \{\theta :\| \theta - \theta^* \| \leq r\sqrt{d}\}$ *for some* $r > 0$. *For any* $\delta_1 \in (0, 1)$,

$$Pr\{\forall\theta : \|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\| \leq 8\Delta_2\|\theta - \theta^*\| + 4\Delta_1\} \geq 1 - \delta_1, \tag{2.7}$$

in which $\Delta_1 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(3/\delta_1))/|\mathcal{S}_0|}$ *and* $\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1 + \tau_2)/|\mathcal{S}_0|}$, *with* $\tau_1 = d\log 18 + d\log((M \vee M')/\sigma_2)$, *and* $\tau_2 = 0.5d\log(|\mathcal{S}_0|/d) + \log(3/\delta_1) + \log(\frac{2r\sigma_2^2\sqrt{|\mathcal{S}_0|}}{\alpha_2\sigma_1})$.

Proof. (Outline): The proof relies on the typical ϵ-net argument. Let $\Theta_\epsilon = \{\theta_1, ..., \theta_{N_\epsilon}\}$ be an ϵ-cover of Θ, i.e., for fix any $\theta \in \Theta$, there exists a $\theta_j \in \Theta_\epsilon$ such that $\| \theta - \theta_j \| \leq \epsilon$. By triangle inequality,

$$\begin{aligned}\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta) - \nabla F(\theta)\right\| &\leq \|\nabla F(\theta) - \nabla F(\theta_j)\| \\ + \quad &\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}(\nabla f(X_i,\theta) - \nabla f(X_i,\theta_j))\right\| \\ + \quad &\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta_j) - \nabla F(\theta_j)\right\|.\end{aligned}$$

Then first term can be upper bounded using Assumption 1. The second term can be bounded using Assumption 4, and the third term can be bounded using Lemma 2.3. We can then employ union bound over Θ_ϵ to finish the argument. Please see Appendix A.4 for details. □

Combining Lemma 2.1 and Proposition 2.1, we know that $G(\theta)$ is a good approximation of $\nabla F(\theta)$. Using this fact, we have the following convergence result.

Theorem 1. *If Assumptions 1-4 hold, and* $\Theta \subset \{\theta :\| \theta - \theta^* \| \leq r\sqrt{d}\}$ *for some* $r > 0$, *choose* $0 < \eta < L/M^2$, *then regardless of the number of attackers with probability at least* $1 - \delta_1$ *that*

$$\|\theta_t - \theta^*\| \leq (1 - \rho_1)^t\|\theta_0 - \theta^*\| + (4\eta\Delta_1 + 4\eta\xi\Delta_1)/\rho_1,$$

in which

$$\rho_1 = 1 - \left(\sqrt{1+\eta^2 M^2 - \eta h} + 8\Delta_2\eta + \eta\xi(8\Delta_2 + M)\right). \tag{2.8}$$

Proof. Please see Appendix A.5. □

This theorem shows that under an event that happens with a high probability, the estimated θ can converge to the neighborhood of θ^* exponentially fast. However, the convergence accuracy bound is not tighter than the bound one could obtain if the algorithm uses gradient descent calculated from $\mathcal{S}_0$ only. This is because we are working with an adversarial setup, for which we need to derive a bound that holds in the worst-case scenario. When there is no assumption on p, the worst-case scenario is when all workers are under attack, which corresponds to the case where the server can only trust the data from $\mathcal{S}_0$ only but it still using data from these Byzantine workers since these data pass the comparison test. Our numerical results in Section 3.5 will illustrate that the actual performance of the proposed algorithm is better than the case with using data from $\mathcal{S}_0$ only and it can benefit from the presence of honest workers even when more than half of the workers are Byzantine workers.

2.4.2 Scenario 2: known p.

In this section, we assume that we know an upper bound on the number of Byzantine workers. Note that p is not the exact number of Byzantine workers, it is merely an upper bound. Furthermore, p could be larger than $m/2$. Hence, this is not a too restrictive assumption. With this additional knowledge, we can derive a tighter convergence result. To proceed, we use $\mathcal{H}_t$ to denote the set of honest workers whose gradient information are accepted by the server at iteration t, and $\mathcal{A}_t$ denote the set of attackers whose information are accepted by the server at iteration t. The values of these two sets are unknown. We only know that $|\mathcal{A}_t| \leq p$. Let $k = |\mathcal{H}_t| + |\mathcal{A}_t|$. Using this notation, the gradient used for updating at the server can be written as

$$G(\theta_t) = \sum_{j\in\mathcal{H}_t\cap\mathcal{U}_t} w_j \nabla\overline{f}^{(j)}(\theta_t) + w_0\nabla\overline{f}^{(0)}(\theta_t) + \sum_{j\in\mathcal{A}_t\cap\mathcal{U}_t} w_j g^{(j)}(\theta_t),$$

in which $\mathcal{U}_t$ is defined in Section 3.3.

Similar to Section 2.4.1, we will prove results that hold simultaneously for all $\theta \in \Theta$ with a high probability. We will show that all gradient information from all honest workers have a high probability to be accepted, hence we will drop the subscript t from $\mathcal{H}_t$ to $\mathcal{H}$. By exploiting the knowledge of p, we provide a tighter bound on $\|G(\theta) - \nabla F(\theta)\|$ than the one presented in Lemma 2.1.

Lemma 2.4. *If there are up to p attackers, at iteration t, the distance between $G(\theta)$ and $\nabla F(\theta)$ is bounded as*

$$\begin{aligned}\|G(\theta) - \nabla F(\theta)\| \;\; &\leq \;\; \frac{\sum_{j\in\mathcal{H}\cap\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|}{\sum_{j\in\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|}\|C_t(\theta) - \nabla F(\theta)\| \\ &+ \sum_{j\in\mathcal{A}_t\cap\mathcal{U}_t} w_j \|g^{(j)}(\theta) - \nabla F(\theta)\|, \end{aligned} \tag{2.9}$$

where $C_t(\theta) = \sum_{j\in\mathcal{H}\cap\mathcal{U}_t} \beta_j \nabla \overline{f}^{(j)}(\theta) + \beta_0 \nabla \overline{f}^{(0)}(\theta)$, $\beta_j = \frac{|\mathcal{S}_j|}{\sum_{i\in\mathcal{H}\cap\mathcal{U}_t}|\mathcal{S}_i| + |\mathcal{S}_0|}$ *and* $w_j = \frac{|\mathcal{S}_j|}{\sum_{i\in\mathcal{U}_t}|\mathcal{S}_i|+|\mathcal{S}_0|}$.

Proof. Please see Appendix A.6. □

Before further simplifying (2.9), we present several supporting lemmas.

The following lemma shows that, by choosing ξ properly, the gradient information from an honest user will be accepted by the server with a high probability.

Lemma 2.5. *Suppose we set ξ as $c|\mathcal{S}_0|^{-1/4}$ and $|\mathcal{S}_0|$ sufficiently large, then for each honest worker j and $\delta_1 \in (0, 1)$*

$$\|\nabla \overline{f}^{(j)}(\theta) - \nabla \overline{f}^{(0)}(\theta)\| \leq \xi \|\nabla \overline{f}^{(0)}(\theta)\|, \forall \theta \in \Theta \tag{2.10}$$

holds with probability $(1 - \delta_1)^2 - \delta_1$.

Proof. Please refer to Appendix A.7. □

From Lemma 2.5, we can see that data sent by a honest worker has a high probability to pass the comparison test in the server. We can define event Υ_1 such that information from all $|\mathcal{H}|$

good workers satisfies (2.10). Using union bound, we know that information from these $|\mathcal{H}|$ honest workers will all be accepted with $\Pr\{\Upsilon_1\} \geq 1-\delta_3$, where $\delta_3 = 1-\{[(1-\delta_1)^2-\delta_1]|\mathcal{H}|-(|\mathcal{H}|-1)\}$.

We now bound the first term in (2.9), namely $\|C_t(\theta) - \nabla F(\theta)\|$ at iteration t. Towards this goal, consider a set $\mathcal{NC}_t = \{\mathcal{C}_t^1, \mathcal{C}_t^2, \mathcal{C}_t^3, ...\}$, each of which represents one possibility of choosing $|\mathcal{H} \cap \mathcal{U}_t|$ workers from $|\mathcal{H}|$ at iteration t. We have $|\mathcal{NC}_t| = \binom{|\mathcal{H}|}{|\mathcal{H}\cap\mathcal{U}_t|}$.

Proposition 2.2. *Suppose Assumptions 2-4 hold, and* $\Theta \subset \{\theta :\| \theta-\theta^* \|\leq r\sqrt{d}\}$ *for some positive parameter* r*, at iteration t, for any* $\delta_2 \in (0,1)$

$$Pr\{\forall\theta : \|C_t(\theta) - \nabla F(\theta)\| \leq 8\Delta_6\|\theta - \theta^*\| + 4\Delta_5\} \geq 1 - \delta_2,$$

in which

$$\Delta_5 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(3|\mathcal{NC}_t|/\delta_2))/|\mathcal{S}_t|}, \tag{2.11}$$

$$\Delta_6 = \sqrt{2}\sigma_2\sqrt{(\tau_1 + \tau_2)/|\mathcal{S}_t|}, \tag{2.12}$$

with $\tau_1 = d\log 18 + d\log((M \vee M')/\sigma_2)$, $\tau_2 = 0.5d\log\left(\frac{\max_{\mathcal{C}_t^l\in\mathcal{NC}_t}(\sum_{j\in\mathcal{C}_l}|\mathcal{S}_j|+|\mathcal{S}_0|)}{d}\right) + \log(3/\delta_2) + \log(\frac{2r\sigma_2^2\sqrt{|\mathcal{S}_t|}}{\alpha_2\sigma_1})$, *and* $|\mathcal{S}_t| = \min_{\mathcal{C}_t^l\in\mathcal{NC}_t}(\sum_{j\in\mathcal{C}_t^l}|\mathcal{S}_j| + |\mathcal{S}_0|)$.

Proof.

$$\|C_t(\theta) - \nabla F(\theta)\| \leq \sup_{\mathcal{C}_t^l\in\mathcal{NC}_t} \|Q_t^l(\theta) - \nabla F(\theta)\|, \tag{2.13}$$

where

$$Q_t^l(\theta) = \sum_{j\in\mathcal{C}_t^l} \beta_j \nabla\overline{f}^{(j)}(\theta) + \beta_0\nabla\overline{f}^{(0)}(\theta), \tag{2.14}$$

where $\beta_j = \frac{|\mathcal{S}_j|}{\sum_{i\in\mathcal{C}_t^l}|\mathcal{S}_i|+|\mathcal{S}_0|.}$ Then by union bound, at iteration t, we need to proof

$$\Pr\{\forall\theta : \left\|Q_t^l(\theta) - \nabla F(\theta)\right\| \leq 8\Delta_6\|\theta - \theta^*\| + 4\Delta_5\} \geq 1 - \frac{\delta_2}{|\mathcal{NC}_t|}.$$

The remaining proof is similar to the proof of Proposition 2.1 and hence is omitted for brevity. □

For the second term in (2.9), each Byzantine gradient information in $\mathcal{U}_t$ must follow the inequality

$$\|g(\theta) - \nabla F(\theta)\| \leq \max_{j \in \mathcal{H}} \left\| \nabla \overline{f}^{(j)}(\theta) - \nabla F(\theta) \right\|. \tag{2.15}$$

Using this fact, we have the following proposition.

Proposition 2.3. *Suppose Assumptions 2-4 hold, and* $\Theta \subset \{\theta :\| \theta - \theta^* \| \leq r\sqrt{d}\}$ *for some positive parameter* r*. For any* $\delta_2 \in (0,1)$

$$Pr\{\forall \theta : \max_{j \in \mathcal{H}} \left\| \nabla \overline{f}^{(j)}(\theta) - \nabla F(\theta) \right\| \leq 8\Delta_8 \|\theta - \theta^*\| + 4\Delta_7\} \geq 1 - \delta_2,$$

in which

$$\Delta_7 = \sqrt{2}\sigma_1 \sqrt{(d \log 6 + \log(3|\mathcal{H}|/\delta_2))(\min_{j \in \mathcal{H}} |\mathcal{S}_j|)}, \tag{2.16}$$

and

$$\Delta_8 = \sqrt{2}\sigma_2 \sqrt{(\tau_3 + \tau_4)/\min_{j \in \mathcal{H}} |\mathcal{S}_j|}, \tag{2.17}$$

with $\tau_3 = d \log 18 + d \log((M \vee M')/\sigma_2)$*, and* $\tau_4 = 0.5 d \log\left(\frac{\max_{j \in \mathcal{H}} |\mathcal{S}_j|}{d}\right) + \log(3/\delta_2) + \log(\frac{2r\sigma_2^2 \sqrt{\min_{j \in \mathcal{H}} |\mathcal{S}_j|}}{\alpha_2 \sigma_1})$.

Proof. The proof is similar to the proof of Proposition 2.2 and hence is omitted for brevity. □

Using these two propositions, we now further bound (2.9) from above by examining the worst-case scenario with regards to $\mathcal{U}_t$. At iteration t, the right-hand side of (2.9) has a high probability to be bounded by

$$\begin{aligned} &8\Delta_6\|\theta - \theta^*\| + 4\Delta_5 + 8(\Delta_8 - \Delta_6)\|\theta - \theta^*\| \frac{\sum_{j \in \mathcal{B} \cap \mathcal{U}_t} |\mathcal{S}_j|}{\sum_{j \in \mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|} + \\ &4(\Delta_7 - \Delta_5) \frac{\sum_{j \in \mathcal{B} \cap \mathcal{U}_t} |\mathcal{S}_j|}{\sum_{j \in \mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|} \end{aligned} \tag{2.18}$$

When $|\mathcal{H} \cap \mathcal{U}_t| \geq 1$, we can find $\Delta_8 \geq \Delta_6$ and $\Delta_7 \geq \Delta_5$, as Δ_8 and Δ_7 have a smaller denominator. Hence, the coefficient of $(\sum_{j \in \mathcal{B} \cap \mathcal{U}_t} |\mathcal{S}_j|)/(\sum_{j \in \mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|)$ in the second term and third term of (2.18) are non-negative. As the result, (2.18) is a nondecreasing function of $(\sum_{j\in\mathcal{B}\cap\mathcal{U}_t}|\mathcal{S}_j|)/(\sum_{j\in\mathcal{U}_t}|\mathcal{S}_j|+|\mathcal{S}_0|)$. We now consider two different cases with fixed denominator.

Case 1): $p < m/2$. In this case, $p < m-p$, hence $\max\{\sum_{j\in\mathcal{B}\cap\mathcal{U}_t}|\mathcal{S}_j|\}$ with setting $|\mathcal{B}\cap\mathcal{U}_t| = p$ will maximize (2.18) and also maximize the right-hand side of (2.9) in iteration t. This implies that, when $p < m/2$, the worst-case scenario is that there are p Byzantine workers and these gradients are all in $\mathcal{U}_t$ all the time.

Case 2): $p \geq m/2$. In this case, since $|\mathcal{B}\cap\mathcal{U}_t| \leq m-p$ and $m-p \leq p$, $\max\{\sum_{j\in\mathcal{B}\cap\mathcal{U}_t}|\mathcal{S}_j|\}$ with setting $|\mathcal{B}\cap\mathcal{U}_t| = m-p-1$ will maximize (2.18) as argued above. We need to consider additional value when $\max\{\sum_{j\in\mathcal{B}\cap\mathcal{U}_t}|\mathcal{S}_j|\}$ with $|\mathcal{B}\cap\mathcal{U}_t| = m-p$, which means $m-p$ gradient information in $\mathcal{U}_t$ are all from Byzantine workers. In this case $\Delta_8 < \Delta_6$ and $\Delta_7 < \Delta_5$, since we assume $\min_{j\in\mathcal{H}}|\mathcal{S}_j| \geq |\mathcal{S}_0|$. The obtained bound with $\max\{\sum_{j\in\mathcal{B}\cap\mathcal{U}_t}|\mathcal{S}_j|\}$ by setting $|\mathcal{B}\cap\mathcal{U}_t| = m-p$ is larger than the bound obtained by setting $|\mathcal{B}\cap\mathcal{U}_t| = m-p-1$ in (2.18). This implies that, when $p \geq m/2$, the worst-case occurs when all $m-p$ gradient information in $\mathcal{U}_t$ are from Byzantine workers.

From the discussion above, we know that with a high probability, the gradient information from all honest works will be accepted. Furthermore, regardless of the true number of attackers, the right-hand side of (2.9) is bounded by the scenario where $\min\{m-p, p\}$ number of gradient in $\mathcal{U}_t$ are from Byzantine workers all the time.

With these supporting lemmas and propositions, we are ready for our main convergence result under 2 cases.

Theorem 2. *If there are up to p attackers, Assumptions 1- 4 hold and* $\Theta \subset \{\theta : \parallel \theta - \theta^* \parallel \leq r\sqrt{d}\}$ *for some* $r > 0$*, choose* $0 < \eta < L/M^2$*, we have*

$$\|\theta_t - \theta^*\| \leq (1-\rho_2)^t\|\theta_0 - \theta^*\| + (\eta\gamma_1)/\rho_2, \tag{2.19}$$

hold simultaneously for all θ_t with probability at least $1 - 2\delta_2 - \delta_3$. Here

$$\rho_2 = 1 - \left(\sqrt{1 + \eta^2 M^2 - \eta h} + \eta\gamma_2\right), \tag{2.20}$$

$$\gamma_1 = 4(1 - w_{max})\Delta_5 + 4w_{max}\Delta_7, \tag{2.21}$$

and

$$\gamma_2 = 8(1 - w_{max})\Delta_6 + 8w_{max}\Delta_8, \tag{2.22}$$

with $w_{max} = \max\{(\sum_{j\in\mathcal{B}\cap\mathcal{U}_t} |\mathcal{S}_j|)/(\sum_{j\in\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|)\}$ and $|\mathcal{B} \cap \mathcal{U}_t| = \min\{m - p, p\}$ and $|\mathcal{U}_t| = m - p$.

Proof. Please see Appendix A.8. □

Theorem 2 shows that under the event which would happen with highly probability, the estimated θ can converge to the neighborhood of θ^* exponentially fast.

From the discussion above, since $\sum_{j\in\mathcal{H}\cap\mathcal{U}} |\mathcal{S}_j| + |\mathcal{S}_0|$ is greater or equal to $|\mathcal{S}_0|$, $\Delta_6 \leq \Delta_2$. Then, $\gamma_2 \leq (8\Delta_2 + 8\xi\Delta_2 + \xi M)$ and $\rho_2 \geq \rho_1$, Hence, the convergence performance benefits from knowing an upperbound on the number of Byzantine workers.

2.5 Numerical results

In this section, we provide numerical examples, with both synthesized data and real data, to illustrate the analytical results.

2.5.1 Synthesized data

We first use synthesized data. In this example, we focus on linear regression, in which

$$Y_i = X_i^T\theta^* + \epsilon_i, i = 1, 2, \cdots, N,$$

where $X_i \in \mathbb{R}^d$, θ^* is a $d \times 1$ vector and ϵ_i is the noise. We set $\mathbf{X} = [X_1, \cdots, X_N]$ as $d \times N$ data matrix.

In the simulation, we set the dimension $d = 20$, the total number of data $N = 100000$, the number of workers $m = 100$, and evenly distribute data among these machines. We set $\epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, 1)$. Here $\mathcal{N}(\mu, \sigma^2)$ denotes Gaussian variables with mean μ and variance σ^2. Furthermore, we set $|\mathcal{S}_0| = 1000$, $\xi = 1.5|\mathcal{S}_0|^{-\frac{1}{4}} = 0.2667$. We use $\mathcal{N}(0, 4)$ to independently generate each entry of θ^*. After θ^* is generated, we fix it. The data matrix $\mathbf{X}$ is generated randomly by Gaussian distribution with $\mu = 0$ and fixed known maximal and minimal eigenvalues of the correlation matrix $\mathbf{X}^T\mathbf{X}$. Let $\lambda_{max}(\cdot)$ and $\lambda_{min}(\cdot)$ denote the maximal and minimal eigenvalue of $\mathbf{X}^T\mathbf{X}$ respectively. In the following figures, we use $\lambda_{max}(\mathbf{X}^T\mathbf{X}) = 200$ and $\lambda_{min}(\mathbf{X}^T\mathbf{X}) = 2$ to generate the data matrix $\mathbf{X}$, then generate Y_i using the linear relationship mentioned above. We illustrate our results with two different attacks: 1) Inverse attack, in which each attacker first calculates the gradient information $\nabla\overline{f}^{(j)}(\theta_{t-1})$ based on the its local data but sends the inversed version $-\nabla\overline{f}^{(j)}(\theta_{t-1})$ to the server; and 2) Random attack, in which the attacker randomly generates gradient value. In our simulation, we compare three algorithms: 1) Gradient descent using only data from $\mathcal{S}_0$, i.e., the server ignores information from all workers; 2) Algorithm proposed in [13]; and 3) The proposed algorithm described in Table 2.1.

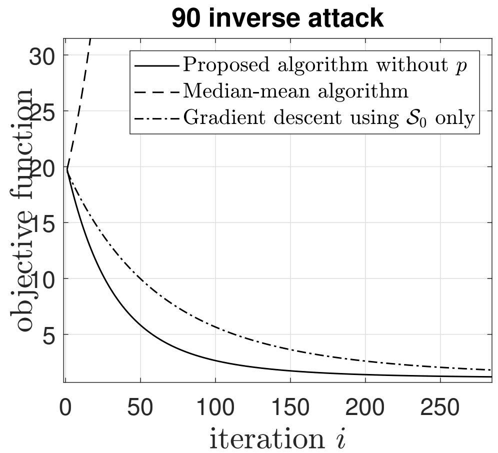

Figure 2.2: Synthesized data: 90 Inverse attack. Median-mean algorithm in [13]

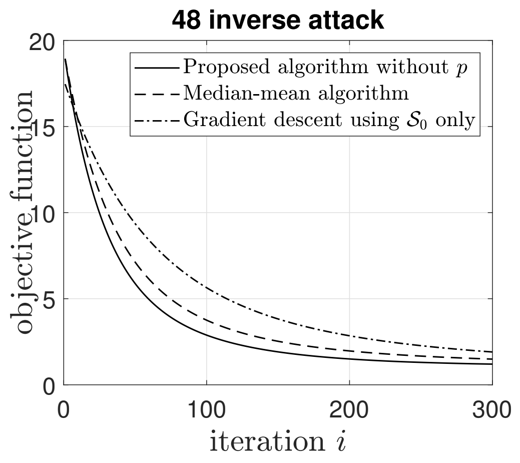

Figure 2.3: Synthesized data: 48 Inverse attack. Median-mean algorithm in [13]

Figures 2.2 and 2.3 plot the value of the loss function vs iteration with 90 and 48 inverse attacks respectively. When the attacker number is 48, which is less than half of the total number, all three algorithms can converge. However, from Figure 2.2, it is clear that the algorithm in [13] does not converge as the number of attackers is more than half of the total number of machines. The proposed algorithm, however, still converges in the presence of 90 attackers. Furthermore, even though there are only 10 honest workers and the server does not know the identities of these honest workers, the proposed algorithm can still benefit from these workers, as the proposed algorithm outperforms the algorithm that only relies on information from $\mathcal{S}_0$.

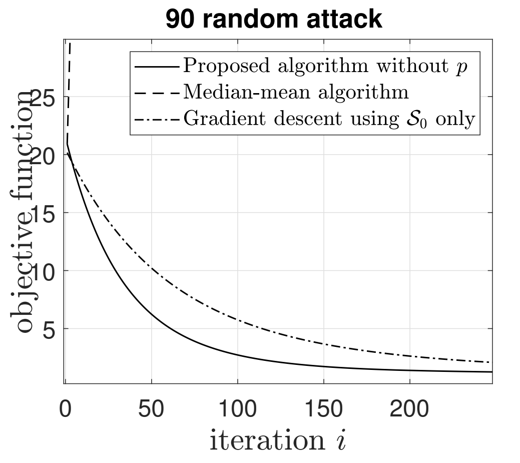

Figure 2.4: Synthesized data: 90 Random attack. Median-mean algorithm in [13]

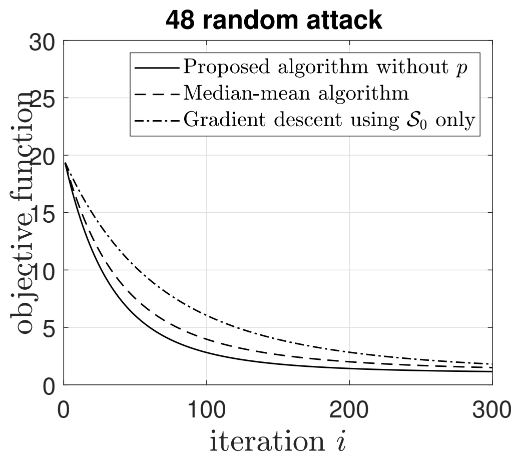

Figure 2.5: Synthesized data: 48 Random attack. Median-mean algorithm in [13]

Figures 2.4 and 2.5 plot the value of the loss function vs iteration with 90 and 48 random attacks respectively. Similar to the scenario with inverse attack, all three algorithms can converge when there are less than half of the total number attackers. However, when there are 90 attackers, our algorithm outperforms the algorithm that uses $\mathcal{S}_0$ only, while the algorithm in [13] diverges.

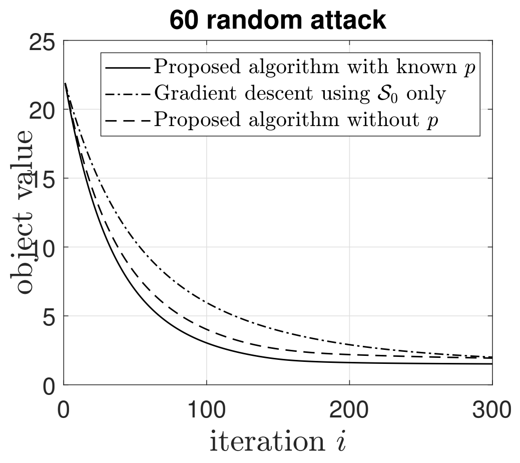

Figure 2.6: Synthesized data: 60 Random attack.

Figures 2.6 and 2.7 plot the value of the loss function vs iteration with 60 random and 60 inverse attacks respectively for the cases with and without knowledge of p. For the case with knowledge about p, we set $p = 75$. From Figures 2.6, we can see that the proposed algorithm without knowing p has a lower convergence accuracy and convergence rate when comparing with

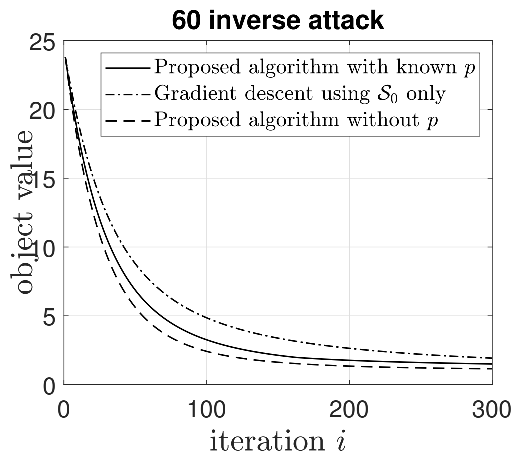

Figure 2.7: Synthesized data: 60 Inverse attack.

the proposed algorithm knowing p. The main reason is that, when facing random attack, some attack vectors can pass the comparison test. In Figure 2.7, since the attacks are inverse attack, the proposed algorithm can successfully reject all the information from attackers, then the proposed algorithm without knowing p has more data to update the parameter.

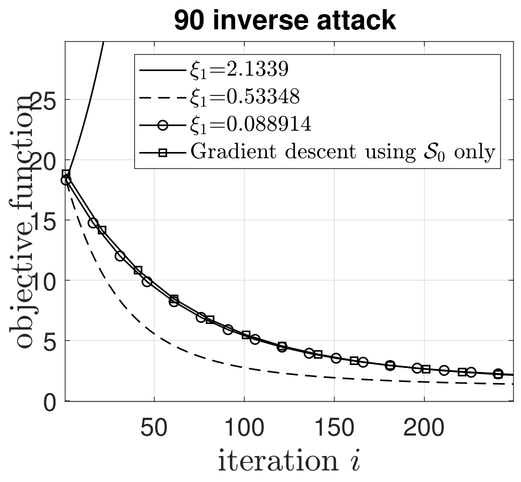

Figure 2.8: Synthesized data: Different ξ.

In Figure 2.8, we show the effect of different ξs on the convergence speed on testing with 90 inverse attack by using the proposed algorithm without knowledge of p. With $\xi = 0.55348$, the comparison step (2.3) can successfully reject attack data with large amplitude, and our proposed algorithm still benefit from the good workers (even though their identities are unknown). With $\xi = 2.1339$, the comparison step (2.3) can not reject inverse attack, then our proposed algorithm will diverge. With $\xi = 0.088914$, the comparison step (2.3) rejects all the accepted data since ξ is too small, then our proposed algorithm has the same performance as using $\mathcal{S}_0$ only. Hence, the convergence speed in our proposed algorithm can be improved by choosing an appropriate ξ.

Table 2.1 lists the running time for three algorithms under 60 inverse attacks, and we measure the number of iterations needed for the loss function to reach 1.9. In Table 2.1, the simulations are produced under the same testing environment. From Table 2.1, we can see that, compared with the case of using data from $\mathcal{S}_0$ only, the proposed algorithms have a higher complexity per iteration, but they reduce the number of iterations. Both proposed algorithms have a better performance than the gradient descent using $|\mathcal{S}_0|$ only, since they can benefit from the gradient information received from workers, even though the server does not know whether the workers are honest or not.

Table 2.1: Running time comparison

loss function 1.9	time/iter	iteration	time
algorithm without p	1.0904×10^{-4}	140	0.0153
algorithm with p	1.5575×10^{-4}	174	0.0271
GD using S_0 only	9.5889×10^{-5}	300	0.0288

2.5.2 Real data

Now we test our algorithms on real datasets MNIST [25] and CIFAR-10 [23], and compare our algorithms with various existing work [3,13,50]. MNIST is a widely used computer vision dataset that consists of 70,000 28×28 pixel images of handwritten digits 0 to 9. We use the handwritten images of 3 and 5, which are the most difficult to distinguish in this dataset, to build a logistic regression model. After picking all 3 and 5 images from the dataset, the total number of images is 13454. It is divided into a training subset of size 12000 and a testing subset of size 1454. The CIFAR-10 dataset consists of 60,000 32x32 images in 10 classes. For CIFAR-10 dataset, we pick the images of car and plane, and build a training subset of size 10000 and a testing subset of 2000.

For these two datasets, we set the number of workers to be 50 and we random pick 200 images from both subset to build $\mathcal{S}_0$, and set the step size to be 0.01 for MNIST and 0.005 for CIFAR-10. Similar to the synthesized data scenario, we illustrate our results with two different attacks, namely inverse attack and random attack, and compare the performance of five algorithms: Zeno [50], where we set the cutoff number (a design parameter in Zeno) to be 5, Krum [3], median-mean [13], proposed algorithm without known p and the algorithm that server using only data $\mathcal{S}_0$. The following figures show how the testing accuracy varies with training iteration.

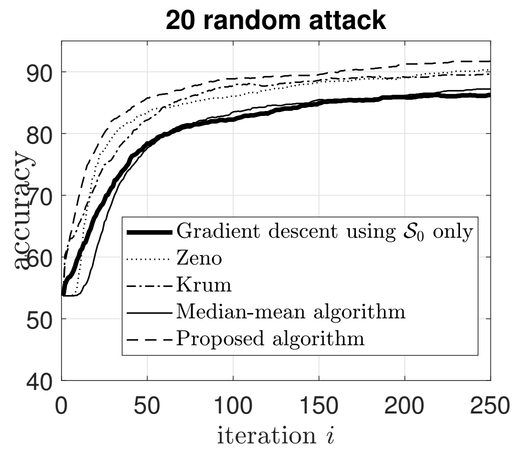

Figure 2.9: MNIST: 20 Random attack. Zeno [50], Krum [3], Median-mean algorithm [13]

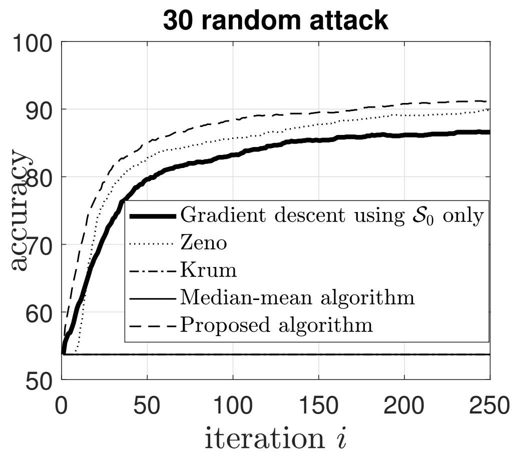

Figure 2.10: MNIST: 30 Random attack. Zeno [50], Krum [3], Median-mean algorithm [13]

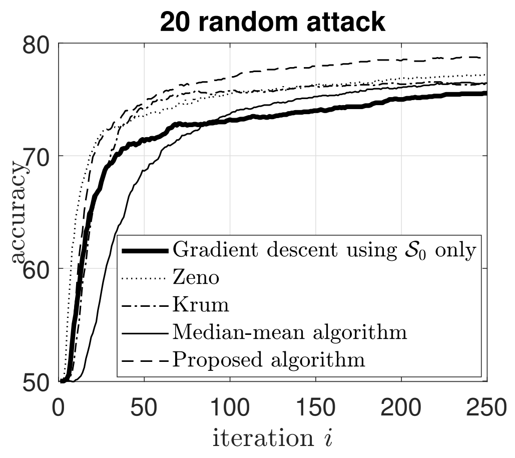

Figure 2.11: CIFAR-10: 20 Random attack. Zeno [50], Krum [3], Median-mean algorithm [13]

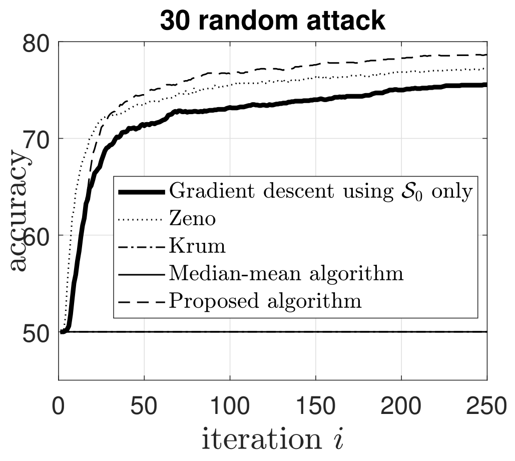

Figure 2.12: CIFAR-10: 30 Random attack. Zeno [50], Krum [3], Median-mean algorithm [13]

Figures 2.9, 2.10, 2.11, and 2.12 illustrate the impact of 20 and 30 random attacks on different algorithms respectively. Figures 2.9, 2.10 are generated using MNIST, while Figures 2.11, and 2.12 are generated using CIFAR-10. Figure 2.9 and 2.11 show that all algorithms have high accuracy when there are 20 attacks. Gradient descent using $\mathcal{S}_0$ only have the lowest accuracy since it uses a small size of data for training. The proposed algorithm has the best performance even though less than half of the workers are attackers. Figure 2.10 and 2.12 show the algorithm using median-mean and Krum fail to predict if there are 30 attackers. Our proposed algorithm and Zeno still show high accuracy, and outperform the algorithm that only relies on information from $\mathcal{S}_0$. Furthermore, the proposed algorithm has better performance than Zeno.

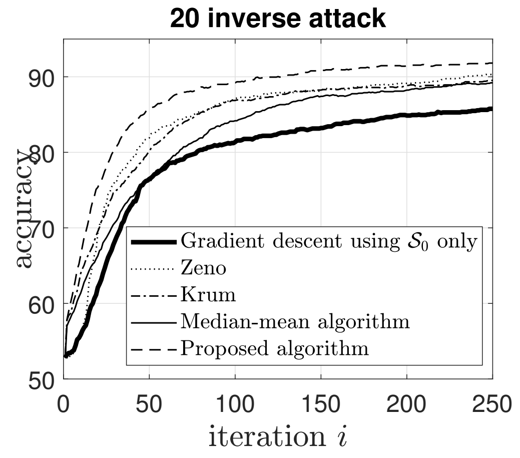

Figure 2.13: MNIST: 20 Inverse attack. Zeno [50], Krum [3], Median-mean algorithm [13]

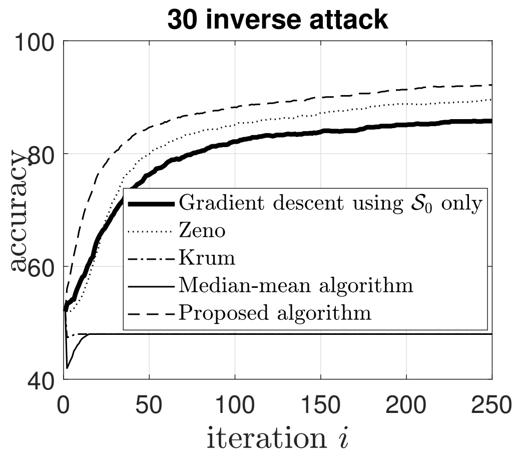

Figure 2.14: MNIST: 30 Inverse attack. Zeno [50], Krum [3], Median-mean algorithm [13]

We plot the impact of different number of inverse attacks on real data in Figures 2.13, 2.14, 2.15, and 2.16 using MNIST and CFAIR10 respectively. All algorithms can converge when there are 20 inverse attacks. However, as the number of attackers is very close to half of the total number, the algorithm in [13] converges very slowly. Again, the proposed algorithm has the best performance even though less than half of the workers are attackers. Furthermore, if there are 30 Byzantine workers, Krum and median-mean algorithm cannot properly work. The algorithm that only based on information from $\mathcal{S}_0$ still performs well, since it does not use the information from all workers.

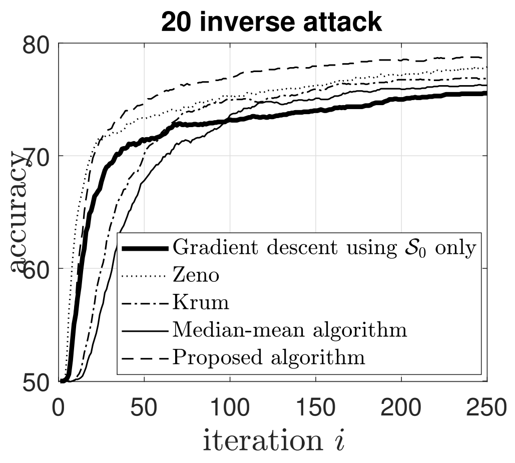

Figure 2.15: CIFAR-10: 20 Inverse attack. Zeno [50], Krum [3], Median-mean algorithm [13]

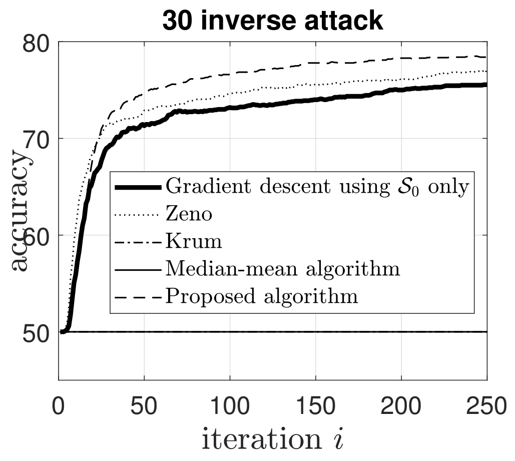

Figure 2.16: CIFAR-10: 30 Inverse attack. Zeno [50], Krum [3], Median-mean algorithm [13]

Our proposed algorithm and Zeno can still work well. They can benefit from the 20 good workers, and outperform the scheme with $\mathcal{S}_0$ only. Our proposed algorithm also outperforms Zeno.

In Figures 2.14 and 2.17, we plot the impact of choosing different cutoff values in Zeno. In Figure 2.17, the cutoff value is 20, Zeno and our proposed algorithm both use all good gradient information, so both algorithms have similar performance. In Figure 2.14, the cutoff value is 5. Although there are 20 good workers, Zeno can only benefit from 5 good workers, but our proposed algorithm can still benefit from all good workers and has a better performance.

Figures 2.18 and 2.19 illustrate the testing accuracy v.s. training time under 20 and 30 inverse

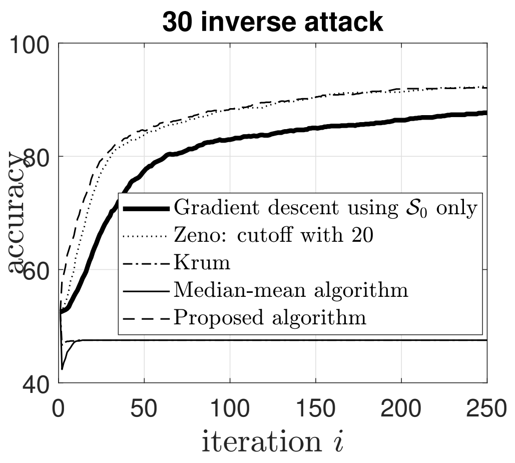

Figure 2.17: MNIST: 30 Inverse attack. Zeno [50], Krum [3], Median-mean algorithm [13]

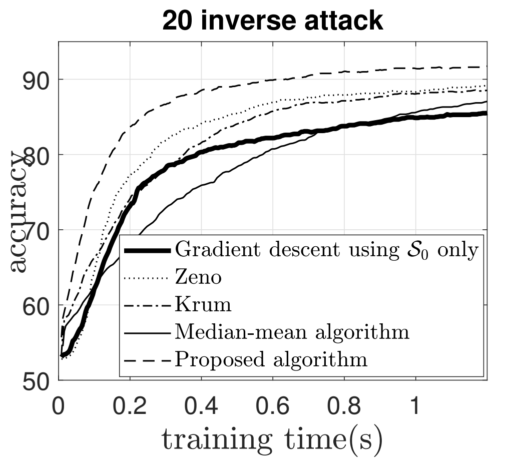

Figure 2.18: MNIST: 20 Inverse attack. Zeno [50], Krum [3], Median-mean algorithm [13]

attacks with different algorithms. All algorithms can converge when there are 20 inverse attacks. Since algorithms have higher complexity, some algorithms converges slower than the gradient descent using $|\mathcal{S}_0|$ only. But our proposed algorithm has a better performance in general.

Figure 2.20 plots the value of the accuracy for $t = 1, \cdots, 250$ with 3 different ξs on testing with 30 inverse attacks by using the proposed algorithm without p. Similar to the scenario with synthesized data, the choice of ξ has an impact on the accuracy of the algorithm. In particular, if ξ is too large, our algorithm will have a low accuracy since the the comparison step (2.3) will accept more data but it may fail to reject wrong information. However, when setting an appropriate ξ, our

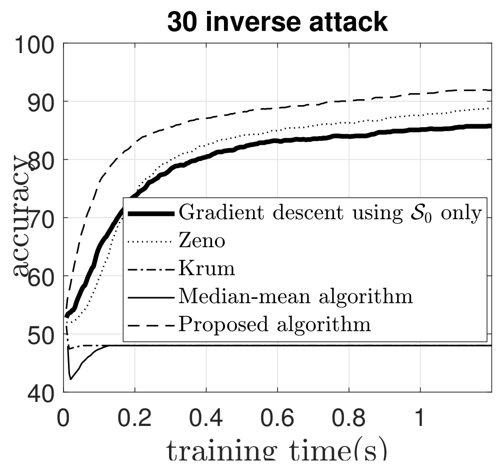

Figure 2.19: MNIST: 30 Inverse attack. Zeno [50], Krum [3], Median-mean algorithm [13]

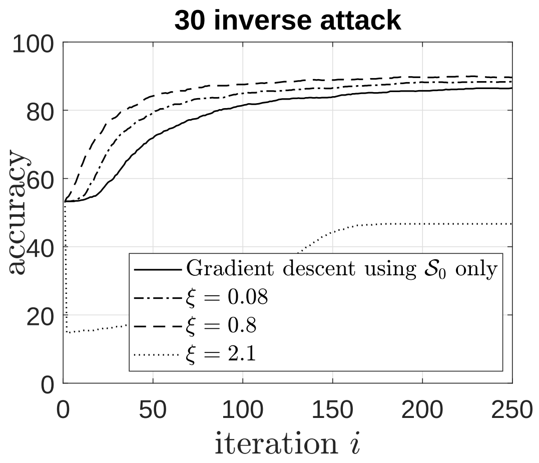

Figure 2.20: MNIST: Different ξ.

algorithm can have a better accuracy.

Figure 2.21 plots the accuracy of three algorithms: the proposed algorithm without knowing p, the proposed algorithm known p and the algorithm using $\mathcal{S}_0$ only. We consider the $p = 40$ while the actual number of Byzantine worker is 30. From Figure 2.21, we can see that the proposed algorithm without knowing p has a higher accuracy and convergence rate than the proposed algorithm knowing p, since the algorithm knowing p discards some correct gradient information and use less gradient information to update. Both proposed algorithms have a better performance than the gradient descent using $\mathcal{S}_0$ only, since they can benefit from the gradient information received from

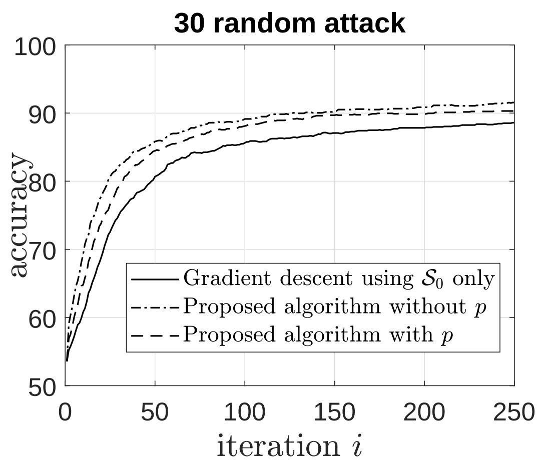

Figure 2.21: MNIST: 30 Random attack.

workers.

In Figure 2.22, we plot the accuracy of three algorithms: the proposed algorithm, the algorithm using $\mathcal{S}_0$ only and the gradient descent algorithm using all data. We consider the case when there is no Byzantine attacker in the system. From Figure 2.22, we can see that the proposed algorithm have similar performance when comparing with gradient descent using all data. Proposed algorithm and gradient descent using all data have a better performance than the gradient descent using $\mathcal{S}_0$ only, since they can benefit from the gradient information received from workers.

Figure 2.23 plots the accuracy of proposed algorithms and the algorithm using $\mathcal{S}_0$ only with three different size of S_0. We consider that there are 20 random attacks. From Figure 2.23, we can see that the algorithms using different size of S_0 only show different performance, but proposed algorithms showing similar performance since after comparison testing, proposed algorithms can benefit from the gradient information received from workers.

2.6 Conclusion

In this chapter, we have proposed a robust gradient descent algorithm that can tolerant an arbitrary number of Byzantine attackers. We have shown that the proposed algorithm converges regardless the number of Byzantine attackers and have provided numerical examples to illustrate

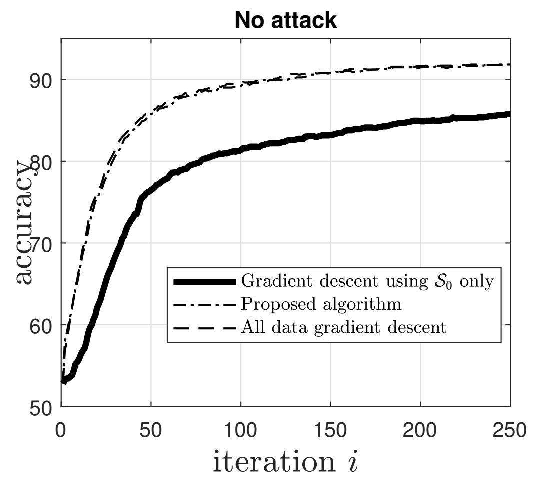

Figure 2.22: MNIST: No attack.

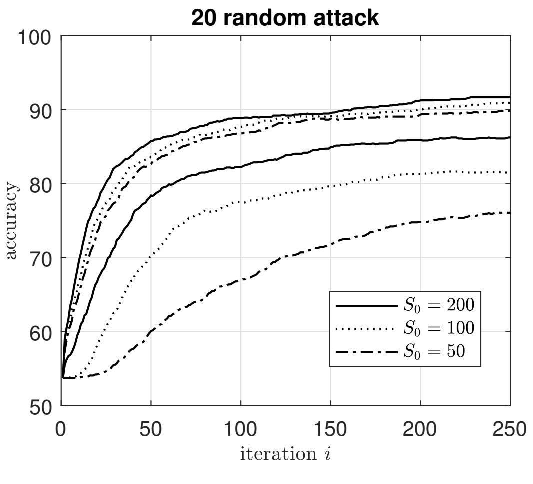

Figure 2.23: MNIST: 20 Random attack.

the performance of the proposed algorithm. In terms of future work, we hope to extend the analysis to scenarios with non-convex cost functions.

Chapter 3

Distributed Approximate Newton's Algorithm Robust to Byzantine Attackers

3.1 Introduction

First-order methods significantly reduce the amount of local computation. But first-order methods may require a far greater number of iterations for communication. Some algorithms also require synchronization in every iteration for parameter updating. In order to mitigate the negative impact of the large number of iterations for distributed optimization, we will propose two robust approximate Newton's algrothms in this chapter.

The first method, named median-based approximate Newton's method (MNM), can converge to the neighborhood of the population minimizer when less than half of the workers are Byzantine attackers. The main idea is to use geometric median, which enables the server to mitigate the impact of attackers when up to half of the workers are Byzantine attackers, to aggregate information from workers.

The second method, named comparison-based approximate Newton's Method (CNM), can converge to the neighborhood of the population minimizer server regardless whether q is larger or smaller than $m/2$. Compared with MNM, CNM requires additional computation at the server.

The main idea is to ask server to randomly collect a small clean dataset and compute noisy value as an approximation of the ground truth to filter out information from attackers.

In this chapter, we analyze the two proposed approximate Newton's method to tolerate Byzantine attackers, and show that that these methods can converge to the neighborhood of the population minimizer server. This chapter is organized as follows. In Section 3.2, we describe the model. In Section 3.3, we describe the proposed algorithms. In Section 3.4, we analyze the convergence property of the proposed algorithms. In Section 3.5, we provide numerical examples to validate the theoretic analysis. Finally, we offer several concluding remarks in Section 3.6. The proofs are collected in Appendix.

3.2 Model

We consider the same population optimizaiton problem in (1.1) and emprical optimization problem in (1.2). And our goal is the same goal as in Chapter 2, to infer the model parameter $\theta^* \in \mathbb{R}^d$ of the unknown distribution.

In this chapter, we assume the population risk function $F(\theta)$ satisfy Assumption 1.

We will again consider a system with Byzantine workers. Furthermore, the set of compromised workers might change over time. In each iteration, if a worker is compromised, it can send arbitrary information to the server when sending gradient information and Newton's direction. We will again let $\mathcal{B}_t$ denote the set of compromised workers at iteration t. When receiving gradient information, the server receives data $g_1^{(j)}(\theta_{t-1})$ from the j-th worker with

$$g_1^{(j)}(\theta_{t-1}) = \begin{cases} \nabla \overline{f}^{(j)}(\theta_{t-1}) & j \notin \mathcal{B}_t \\ \star & j \in \mathcal{B}_t \end{cases}, \tag{3.1}$$

in which $\star$ denotes an arbitrary vector chosen by the attacker. After receiving $g_1^{(j)}$ from workers, the server computes and broadcasts

$$g(\theta_{t-1}) = Aggre_1(g_1^{(1)}(\theta_{t-1}), ..., g_1^{(m)}(\theta_{t-1})), \tag{3.2}$$

in which $Aggre_1(\cdot)$ depends on how the server aggregates gradient information from workers. Each worker then computes Newton's direction based on $g(\theta_{t-1})$. After workers send Newton's direction, the server receives data $g_2^{(j)}(\theta_{t-1})$ from j-th worker

$$g_2^{(j)}(\theta_{t-1}) = \begin{cases} H_{j,t-1}^{-1} g(\theta_{t-1}) & j \notin \mathcal{B}_t \\ \star & j \in \mathcal{B}_t \end{cases}. \tag{3.3}$$

The server finally computes the final update direction using

$$G(\theta_{t-1}) = Aggre_2(g_2^{(1)}(\theta_{t-1}), ..., g_2^{(m)}(\theta_{t-1})), \tag{3.4}$$

in which in which $Aggre_2(\cdot)$ depends on how the server processes $g_2^{(j)}(\theta_{t-1})$ from workers.

Note that if both $Aggre_1(\cdot)$ and $Aggre_2(\cdot)$ are mean functions, the algorithm is the same as the GIANT algorithm [48]. But from Introduction 1.3, the GIANT algorithm is not robust to adversary attacks. The goal of this chapter is to design robust Newton's method algorithms, by designing proper $Aggre_1(\cdot)$ and $Aggre_2(\cdot)$, that can tolerate Byzantine attacks.

3.3 Algorithm

In this section, we describe two algorithms that can handle different number of Byzantine attackers. Let q be the number of attackers in the system. We will first describe our algorithm that can deal with $q < m/2$, i.e., up to half of the total number of workers are Byzantine attackers. We then describe our algorithm to deal with an arbitrary number of Byzantine attackers, i.e., there is no restrict on q. This algorithm requires additional computations at the server.

Algorithm 3.1 Median-based Approximate Newton's Method (MNM) Algorithm

```
Parameter server:
Initialize randomly selects $\theta_0 \in \Theta$.
repeat
    1: Broadcasts the current model parameter estimator $\theta_{t-1}$ to all workers;
    2: Waits to receive gradients from the $m$ workers; $g^{(j)}(\theta_{t-1})$ denote the value received from worker $j$;
    3: Computes $g(\theta_{t-1}) = med\{g_1^{(1)}(\theta_{t-1}), ..., g_1^{(m)}(\theta_{t-1})\}$;
    4: Broadcasts the current gradient estimator $g(\theta_{t-1})$ to all workers;
    5: Waits to receive estimators from the $m$ workers; $g_2^{(j)}(\theta_{t-1})$ denote the value received from worker $j$;
    6: Computes $G(\theta_{t-1}) = med\{g_2^{(1)}(\theta_{t-1}), ..., g_2^{(m)}(\theta_{t-1})\}$;
    7: Updates $\theta_t = \theta_{t-1} - \eta G(\theta_{t-1})$;
until $\|\theta_t - \theta^*\| \leq \epsilon$.
Worker $j$:
1: Receives model parameter estimator $\theta_{t-1}$, computes the gradient $\nabla \overline{f}^{(j)}(\theta_{t-1})$;
2: If worker $j$ is honest, it sends $\nabla \overline{f}^{(j)}(\theta_{t-1})$; If not, it sends the value determined by the attacker;
3: Receives gradient estimator $g(\theta_{t-1})$, computes the parameter $H_{j,t-1}^{-1} g(\theta_{t-1})$ ;
4: If worker $j$ is honest, it sends $H_{i,t-1}^{-1} g(\theta_{t-1})$ back to the server; If not, it sends the value determined by the attacker;
```

3.3.1 Case with $q < m/2$

In the first scenario, we consider the case where there are at most $q < m/2$ Byzantine attackers. We propose a median-based approximate Newton's method (MNM). Main steps of the algorithm are listed in Algorithm 3.1.

Instead of computing the average, the main idea of our algorithm is to use geometric median of the received information as the aggregation function $aggre_1(\cdot)$ and $aggre_2(\cdot)$. In particular, after receiving the gradient information from workers, the server computes

$$g(\theta_{t-1}) = med\{g_1^{(1)}(\theta_{t-1}), ..., g_1^{(m)}(\theta_{t-1})\}, \tag{3.5}$$

in which $med\{\cdot\}$ is the geometric median of the vectors. Geometric median is a generalization of median in one-dimension to multiple dimensions, and has been widely used in robust statistics. In particular, let $x_i \in \mathbb{R}^d, i = 1, \cdots, n$, then the geometric median of the set $\{x_1, x_2, ..., x_n\}$ is define as

$$med\{x_1, x_2, ..., x_n\} := \arg\min_x \sum_{i=1}^{n} \|x_i - x\|. \tag{3.6}$$

Then the server broadcasts value $g(\theta_{t-1})$ to all workers. After receiving the Newton's direction information, the server compute the final Newton's direction information by geometric median again,

$$G(\theta_{t-1}) = med\{g_2^{(1)}(\theta_{t-1}), ..., g_2^{(m)}(\theta_{t-1})\}. \tag{3.7}$$

Finally, the server uses $G(\theta_{t-1})$ to update parameter θ_{t-1},

$$\theta_t = \theta_{t-1} - \eta G(\theta_{t-1}). \tag{3.8}$$

3.3.2 Case with an arbitrary number of Byzantine attackers

The MNM algorithm described in Section 3.3.1 will converge if $q < m/2$, which will be shown in Section 3.3. However, it will fail to converge once $q > m/2$. In this subsection, we propose another algorithm, named comparison-based approximate Newton (CNM) method, that converges for an arbitrary value of q, regardless whether q is larger or smaller than $m/2$. Compared with the MNM algorithm, the CNM algorithm needs additional computation at the server side. In particular, we assume that the server keep a small set of clear data to compute a noisy gradient and a noisy Newton's direction. These information, which are noisy version of the ground truth, will help the server make decision to whether accept information from each worker or not. Main steps of the algorithm are listed in Algorithm 3.2.

More specifically, in our algorithm, the server will randomly select a small set of data points $\mathcal{S}_0$ at the very beginning, where $|\mathcal{S}_0| \leq |\mathcal{S}_j|$ and $j \in [1, m]$. Once $\mathcal{S}_0$ is selected, it is fixed throughout the algorithm. Then at each iteration t, the server calculates a noisy gradient using data points in $\mathcal{S}_0$:

$$\nabla \overline{f}^{(0)}(\theta_{t-1}) = \frac{1}{|\mathcal{S}_0|} \sum_{j \in \mathcal{S}_0} \nabla f(X_j, \theta_{t-1}). \tag{3.9}$$

Algorithm 3.2 Comparison-based approximate Newton's Method (CNM) Algorithm

```
Parameter server:
Initialize randomly selects $\theta_0 \in \Theta$.
repeat
    1: Broadcasts the current model parameter estimator $\theta_{t-1}$ to all workers;
    2: Waits to receive gradients from the $m$ workers; $g^{(j)}(\theta_{t-1})$ denote the value received from worker $j$;
    3: Accepts $g_1^{(j)}(\theta_{t-1})$ which pass test $\|g_1^{(j)}(\theta_{t-1}) - \nabla\overline{f}^{(0)}(\theta_{t-1})\| \leq \xi_1 \|\nabla\overline{f}^{(0)}(\theta_{t-1})\|$, consider them in set $\mathcal{A}^{(1)}$.
    4: Computes $g(\theta_{t-1}) = \frac{1}{1+|\mathcal{A}^{(1)}|}(\sum_{i\in\mathcal{A}^{(1)}} g_1^{(i)}(\theta_{t-1}) + \nabla\overline{f}^{(0)}(\theta_{t-1}))$;
    4: Broadcasts the current gradient estimator $g(\theta_{t-1})$ to all workers;
    5: Accepts $g_2^{(j)}(\theta_{t-1})$ which pass test $\|g_2^{(j)}(\theta_{t-1}) - \tilde{H}_0^{-1} g(\theta_{t-1})\| \leq \xi_2 \|\tilde{H}_0^{-1} g(\theta_{t-1})\|$, consider them in set $\mathcal{A}^{(2)}$.
    6: Computes $G(\theta_{t-1}) = \frac{1}{1+|\mathcal{A}^{(2)}|}(\sum_{i\in\mathcal{A}^{(2)}} g_2^{(i)}(\theta_{t-1}) + \tilde{H}_0^{-1} g(\theta_{t-1}))$;
    7: Update model parameter $\theta_t = \theta_{t-1} - G(\theta_{t-1})$ ;
until $\|\theta_t - \theta^*\| \leq \epsilon$.
Worker $j$:
1: Receives model parameter estimator $\theta_{t-1}$, computes the gradient $\nabla\overline{f}^{(j)}(\theta_{t-1})$;
2: If worker $j$ is honest, it sends $\nabla\overline{f}^{(j)}(\theta_{t-1})$; If not, it sends the value determined by the attacker;
3: Receives gradient estimator $g(\theta_{t-1})$, computes the parameter $\tilde{H}_{j,t-1}^{-1} g(\theta_{t-1})$;
4: If worker $j$ is honest, it sends $\tilde{H}_{i,t-1}^{-1} g(\theta_{t-1})$ back
to the server; If not, it sends the value determined by the attacker;
```

After computing $\nabla\overline{f}^{(0)}(\theta_{t-1})$, the server compares $g_1^{(j)}(\theta_{t-1})$ received from worker j with $\nabla\overline{f}^{(0)}(\theta_{t-1})$. The server will accept $g_1^{(j)}(\theta_{t-1})$ as authentic value and use it for further processing, if

$$\|g_1^{(j)}(\theta_{t-1}) - \nabla\overline{f}^{(0)}(\theta_{t-1})\| \leq \xi_1 \|\nabla\overline{f}^{(0)}(\theta_{t-1})\| \tag{3.10}$$

where ξ_1 is a constant. The server will collect all accepted $g_1^{(j)}(\theta_{t-1})$ in set $\mathcal{A}^{(1)}$. The main enabling observation is that, even though $\nabla\overline{f}^{(0)}(\theta_{t-1})$ is noisy, it is an approximation of the ground truth and hence can be used to filter out bad information from Byzantine workers as done in (3.10).

Then the server computes $g(\theta_{t-1})$ based on the accepted gradient information in set $\mathcal{A}^{(1)}$:

$$g(\theta_{t-1}) = \frac{1}{1+|\mathcal{A}^{(1)}|}\left(\sum_{i\in\mathcal{A}^{(1)}} g_1^{(i)}(\theta_{t-1}) + \nabla\overline{f}^{(0)}(\theta_{t-1})\right).$$

The server will broadcast $g(\theta_{t-1})$ to all workers, each of which will compute $\tilde{H}_{j,t-1}^{-1} g(\theta_{t-1})$, where $\tilde{H}_{j,t-1} = H_{j,t-1} + \mu\mathbf{I}$ with $\mu \geq 0$ and $\mathbf{I}$ being the identity matrix. Here $\mu\mathbf{I}$ is added to make sure the matrix is invertible. The server also computes a noisy Newton's direction $\tilde{H}_0^{-1} g(\theta_{t-1})$, in which $\tilde{H}_0$ is computed using data points in $\mathcal{S}_0$:

$$\tilde{H}_0 = \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \nabla^2 f(X_i, \theta_{t-1}) + \mu\mathbf{I}. \tag{3.11}$$

Then the server compares $g_2^{(j)}(\theta_{t-1})$ received from worker j with $\tilde{H}_0^{-1} g(\theta_{t-1})$. If the following condition is satisfied

$$\|g_2^{(j)}(\theta_{t-1}) - \tilde{H}_0^{-1} g(\theta_{t-1})\| \leq \xi_2 \|\tilde{H}_0^{-1} g(\theta_{t-1})\| \tag{3.12}$$

the server will collect $g_2^{(j)}(\theta_{t-1})$ in set $\mathcal{A}^{(2)}$. Here, ξ_2 is a constant. Then the server computes the final update direction:

$$G(\theta_{t-1}) = \frac{1}{1 + |\mathcal{A}^{(2)}|} \left(\sum_{i \in \mathcal{A}^{(2)}} g_2^{(i)}(\theta_{t-1}) + H_0^{-1} g(\theta_{t-1}) \right). \tag{3.13}$$

3.4 Convergence analysis

In this section, we analyze the convergence property of the proposed algorithms.

3.4.1 Convergence of MNM algorithm

In this section, we will prove results that hold simultaneously for all $\theta \in \Theta$ with a high probability. Hence, in the following, we will drop subscript $t-1$. Before presenting detailed analysis, here we describe the high level ideas. If $H^{-1}\nabla F(\theta)$ is available, where $H = \nabla^2 F(\theta)$, the Newton's method will converge to θ^*. The main idea of our proof is to show that the distance between $G(\theta)$ computed in (3.7) and $H^{-1}\nabla F(\theta)$ is universally bounded in Θ when the number of attackers is at most $m/2$. Hence, $G(\theta)$ is a good estimate of $H^{-1}\nabla F(\theta)$. As the result, we can then show that the proposed algorithm converge to the neighborhood of the population minimizer.

We first show that $\|G(\theta) - H^{-1}\nabla F(\theta)\|$ is universally bounded in Θ. To start with, we first write

$$\begin{aligned}
&\|H^{-1}\nabla F(\theta) - G(\theta)\| \\
= \ &\|H^{-1}\nabla F(\theta) - med\{g_2^{(1)}(\theta), ..., g_2^{(m)}(\theta)\}\| \\
= \ &\|Z(\theta) - H^{-1}g(\theta) + H^{-1}\nabla F(\theta)\| \\
\leq \ &\|Z(\theta)\| + \|H^{-1}(g(\theta) - \nabla F(\theta))\|, \\
\leq \ &\|Z(\theta)\| + \|H^{-1}J(\theta)\|, \qquad (3.14)
\end{aligned}$$

where

$$\begin{aligned}
Z(\theta) &= med\{H^{-1}g(\theta) - g_2^{(1)}(\theta), ..., H^{-1}g(\theta) - g_2^{(m)}(\theta)\} \\
&= med\{Z_1(\theta), ..., Z_m(\theta)\}, \qquad (3.15)
\end{aligned}$$

and

$$\begin{aligned}
J(\theta) &= med\{\nabla F(\theta) - g_1^{(1)}(\theta), ..., \nabla F(\theta) - g_1^{(m)}(\theta)\} \\
&= med\{J_1(\theta), ..., J_m(\theta)\}. \qquad (3.16)
\end{aligned}$$

To further bound the terms in (3.14), we need to use Assumption 2-4 in Chapter 2 and new assumption presented below.

We also assume data in each worker $j \in [1, m]$ has following assumption.

Assumption 5. *For any $\delta \in (0, 1/|\mathcal{S}_j|)$, there exists an $M' = M'(\delta)$ and $h' = h'(\delta)$ such that*

$$\mathbf{Pr}\left\{\forall \theta, \theta' \in \Theta, h' \leq \frac{\|\nabla f(X, \theta) - \nabla f(X, \theta')\|}{\|\theta - \theta'\|} \leq M'\right\} \geq 1 - \frac{\delta}{3}.$$

Assumption 5 ensures that $\nabla f(X, \theta)$ in each worker is M'-Lipschitz and $f(X, \theta)$ is h' strongly convex with high probability.

Now, we make another standard assumption in analyzing Newton's method for population risk.

Assumption 6. *The Hessian matrix $\nabla^2 F(\theta)$ is L-Lipschitz continuous,i.e, there exists an L such that for $\theta, \theta' \in \Theta$*

$$\|\nabla^2 F(\theta) - \nabla^2 F(\theta)\|_2 \leq L\|\theta - \theta'\|,$$

in which $\|\cdot\|_2$ is the matrix spectral norm.

From (3.15), we need to bound the geometric median $Z(\theta)$ of $Z_1(\theta), ..., Z_m(\theta)$. We will use the following property of the geometric median from [31].

Lemma 3.1. *[31] Let $x_1, x_2, ..., x_n$ be n points in a Hilbert space. Let x^* denote the geometric median of these points. For any $\alpha \in (0, 1/2)$, and given $r > 0$, if $\sum_{i=1}^{n} \mathbf{1}_{\{\|x_i\| \leq r\}} \geq (1 - \alpha)n$, then*

$$\|x^*\| \leq \mathcal{C}_\alpha r, \tag{3.17}$$

where

$$\mathcal{C}_\alpha = \frac{2(1-\alpha)}{1-2\alpha}. \tag{3.18}$$

From Lemma 3.1, we can see that, if majority of points are inside the Euclidean ball of radius r centered at origin, then the geometric median must be inside the Euclidean ball of radius $\mathcal{C}_\alpha r$. To use this lemma, we need to show more than half of information received by the server are bounded by some quantity.

We first have the following lemma regarding the spectral norm of $H_i - H$.

Lemma 3.2. *If Assumption 5 holds, for any $\delta \in (0, 1)$, with probability at least $1 - \frac{\delta}{3}$, $|\mathcal{S}_j|$ data satisfy*

$$h' \leq \|\nabla^2 f(X, \theta)\|_2 \leq M', \tag{3.19}$$

for any $\delta_3' \in (0,1)$, *let*

$$\Delta_3 = \sqrt{\frac{14(M \vee M')^2 \log(2d/\delta_3')}{3|\mathcal{S}_i|}}, \tag{3.20}$$

then

$$\mathbf{Pr}\left\{\|H_i - H\|_2 \le \Delta_3\right\} \ge 1 - \delta_2. \tag{3.21}$$

with $\delta_2 = \delta_3' + \frac{\delta}{3}$ *and* $\delta_2 \in (0,1)$.

Proof. Please see Appendix B.1 for detail. □

Now, with these lemmas and assumptions, when worker i is honest, we can show the bound for Z_i.

Proposition 3.1. *Suppose Assumptions 1-3, 5, 6 hold, and* $\Theta \subset \{\theta : \| \theta - \theta^* \| \le r\sqrt{d}\}$ *for some* $r > 0$. *For any* $\delta_3 \in (0,1), \alpha \in (q/m, 1/2)$ *and* $\delta_4 = \delta_2 + e^{-mD(\alpha - q/m\|\delta_3)}$,

$$\mathbf{Pr}\left\{\forall\theta : \|Z_i(\theta)\| \le \left(\frac{8\mathcal{C}_\alpha\Delta_3\Delta_2}{hh'} + \frac{\Delta_3 M}{hh'}\right)\|\theta - \theta^*\| + \frac{4\mathcal{C}_\alpha\Delta_3\Delta_1}{hh'}\right\} \ge 1 - \delta_4. \tag{3.22}$$

where $\Delta_1 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(6/\delta_3))/|\mathcal{S}_i|}$, $\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1+\tau_2)/|\mathcal{S}_i|}$, *with* $\tau_1 = d\log 18 + d\log(M \vee M'/\sigma_2)$, $\tau_2 = 0.5d\log(|\mathcal{S}_i|/d) + \log(6/\delta_3) + \log(\frac{2r\sigma_2^2\sqrt{|\mathcal{S}_i|}}{\alpha_2\sigma_1})$, $\mathcal{C}_\alpha = \frac{2(1-\alpha)}{1-2\alpha}$ *and* $D(\delta'\|\delta) = \delta'\log\frac{\delta'}{\delta} + (1-\delta')\log\frac{1-\delta'}{1-\delta}$.

Proof. Please see Appendix B.2 for details. □

Now we have already shown that for honest workers, the local Newton's direction received at the server is uniformly close to the true Newton's direction. Now using Lemma 3.1, we can show the median $Z(\theta)$ is bounded.

Proposition 3.2. *Suppose Assumptions 1-3, 5, 6 hold, and* $\Theta \subset \{\theta : \| \theta - \theta^* \| \le r\sqrt{d}\}$ *for some* $r > 0$. *For any* $\alpha \in (q/m, 1/2)$ *and* $0 < \delta_4 < \alpha - q/m$,

$$\begin{aligned}\mathbf{Pr}&\left\{\forall\theta : \|Z(\theta)\| \le \left(\frac{8\mathcal{C}_\alpha\Delta_3\Delta_2}{hh'} + \frac{\Delta_3 M}{hh'}\right)\mathcal{C}_\alpha\|\theta - \theta^*\| + \frac{4\mathcal{C}_\alpha^2\Delta_3\Delta_1}{hh'}\right\}\\ \ge\ & 1 - e^{-mD(\alpha - q/m\|\delta_4)}.\end{aligned} \tag{3.23}$$

Proof. Please see Appendix B.3. □

With Proposition 3.2, we are ready to show that $G(\theta)$ is a good approximation of $H^{-1}\nabla F(\theta)$ from (3.14), and show the convergence of the proposed MNM algorithm.

Theorem 3. *If Assumptions 1-3, 5, 6 hold, and* $\Theta \subset \{\theta :\| \theta - \theta^* \|\leq r\sqrt{d}\}$ *for some* $r > 0$*, then for any* $0 < \eta \leq 1$, $\alpha \in (q/m, 1/2)$, $0 < \delta_3 < \alpha - q/m$ *and* $0 < \delta_4 < \alpha - q/m$ *with probability at least* $1 - e^{-mD(\alpha - q/m\|\delta_3)} - e^{-mD(\alpha-q/m\|\delta_4)}$ *that*

$$\|\theta_t - \theta^*\| \;\leq\; \rho\|\theta_{t-1} - \theta^*\| + \frac{\eta L\|\theta_{t-1} - \theta^*\|^2}{2h} + \frac{4}{hh'}\mathcal{C}_\alpha^2\Delta_3\Delta_1 + \eta\frac{4}{h}\mathcal{C}_\alpha\Delta_1,$$

where

$$\rho = 1 - \eta + \eta\frac{8}{hh'}\mathcal{C}_\alpha^2\Delta_3\Delta_2 + \eta\frac{8}{h}\mathcal{C}_\alpha\Delta_2 + \eta\mathcal{C}_\alpha\frac{\Delta_3 M}{hh'} \tag{3.24}$$

Proof. Please see Appendix B.4. □

This theorem shows that under an event that happens with a high probability, the estimated θ can converge to the neighborhood of θ^* with a linear-quadratic rate. Since we consider $\theta^* \in \arg\min_{\theta\in\Theta} F(\theta)$, there is always a gap between estimator θ and θ^*. This gap is due to the approximation error introduced by solving (1.2), instead of (1.1).

3.4.2 Convergence of CNM algorithm

In this section, we prove the convergence of CNM algorithm regardless the number of Byzantine attackers. In other words, q could be larger than $m/2$. Towards this goal, we will show that the distance between $G(\theta)$ defined in (3.13) and $H^{-1}\nabla F(\theta)$ is universally bounded in Θ regardless the number of attackers. As the result, $G(\theta)$ is a good estimate of $H^{-1}\nabla F(\theta)$. Finally, we will show that the proposed algorithm converge to the neighborhood of minimizer of the population risk.

Lemma 3.3. *For an arbitrary number of attackers, the distance between* $G(\theta)$ *and* $H^{-1}\nabla F(\theta)$ *is bounded by*

$$\begin{aligned}
& \|H^{-1}\nabla F(\theta) - G(\theta)\| \\
< \;& (1+\xi_2)\|\tilde{H}_0^{-1}\|_2\|g(\theta) - \nabla F(\theta)\| \\
+ \;& \xi_2\|\tilde{H}_0^{-1}\|_2\|\nabla F(\theta)\| + \|H^{-1}\nabla F(\theta) - \tilde{H}_0^{-1}\nabla F(\theta)\|.
\end{aligned} \tag{3.25}$$

Proof. Please see Appendix B.5. □

Now, in order to bound the distance between $G(\theta)$ and $H^{-1}\nabla F(\theta)$, we need to bound the three terms in the right hand side of (3.25).

For the second term, from Assumption 1, we have $\|\nabla F(\theta)\| = \|\nabla F(\theta) - \nabla F(\theta^*)\| \leq M\|\theta - \theta^*\|$, since $\nabla F(\theta^*) = 0$.

For the third term, we have $\|(H^{-1} - \tilde{H}_0^{-1})\nabla F(\theta)\| = \|(\mathbf{I} - \tilde{H}_0^{-1}H)H^{-1}\nabla F(\theta)\| \leq \|\mathbf{I} - \tilde{H}_0^{-1}H\|_2\|H^{-1}\|_2\|\nabla F(\theta)\|$.

Then, we use the following lemma to bound $\|\mathbf{I} - \tilde{H}_0^{-1}H\|_2$.

Lemma 3.4. *If* $\|H_0 - H\|_2 \leq \beta$ *and* $\beta < \frac{h(h+\mu)}{3h+2\mu}$,

$$\|\mathbf{I} - \tilde{H}_0^{-1}H\|_2 \leq \frac{\mu}{h+\mu} + \frac{2\beta}{h+\mu-\beta} < 1. \tag{3.26}$$

Proof. Please see Appendix B.6. □

From this lemma, we have that $\|\mathbf{I} - \tilde{H}_0^{-1}H\|_2$ is bounded by a constant value smaller than 1, when $\|H_0 - H\|_2$ is bounded.

Proposition 3.3. *Suppose Assumptions 1-3, 5, 6 hold, and* $\Theta \subset \{\theta : \| \theta - \theta^* \| \leq r\sqrt{d}\}$ *for some*

$r > 0$, *and* $\Delta_3 < \frac{h(h+\mu)}{3h+2\mu}$. *For any* $\delta \in (0,1), \delta_3' \in (0,1), \delta_2 \in (0,1)$ *and* $\delta_2 = \frac{\delta}{3} + \delta_3'$

$$\mathbf{Pr}\left\{\forall\theta : \|(H^{-1} - \tilde{H}_0^{-1})\nabla F(\theta)\| \leq \frac{\Delta_4 M}{h}\|\theta - \theta^*\|\right\} \geq 1 - \delta_2, \tag{3.27}$$

with

$$\Delta_4 = \frac{\mu}{h+\mu} + \frac{2\Delta_3}{h+\mu-\Delta_3}, \tag{3.28}$$

and

$$\Delta_3 = \sqrt{\frac{14M^2\log(2d/\delta_3')}{3|\mathcal{S}_0|}}. \tag{3.29}$$

Proof. Please see Appendix B.7. □

Using these intermediate results, we have the following convergence result.

Theorem 4. *If Assumptions 1-3, 5, 6 hold, and* $\Theta \subset \{\theta :\| \theta - \theta^* \|\leq r\sqrt{d}\}$ *for some* $r > 0$, $\mu \geq 0$ *and* $|\mathcal{S}_0|$ *is sufficiently large, then for arbitrary number of attackers with probability at least* $1 - \delta_5 - \delta_2$ *that*

$$\|\theta_t - \theta^*\| \leq \frac{L}{2h}\|\theta_{t-1} - \theta^*\|^2 + \gamma_1\|\theta_{t-1} - \theta^*\| + \eta\gamma_2. \tag{3.30}$$

where $\Delta_4 = \frac{\mu}{h+\mu} + \frac{2\Delta_3}{h+\mu-\Delta_3}$, *and*

$$\gamma_1 = \left[(8\Delta_2 + \xi_1(8\Delta_2 + M))\frac{1+\xi_2}{h'+\mu} + \frac{\xi_2 M}{h'+\mu} + \frac{\Delta_4}{h}\right], \tag{3.31}$$

and

$$\gamma_2 = (4\Delta_1 + \xi_1 4\Delta_1)\frac{1+\xi_2}{h'+\mu}. \tag{3.32}$$

Proof. Please see Appendix B.8. □

This theorem shows that, with high probability, the estimated θ can converge to the neighborhood of θ^* with a linear-quadratic rate when there are arbitrary number of Byzantine attackers. Since we consider $\theta^* \in \arg\min_{\theta\in\Theta} F(\theta)$, we can only use empirical risk to approximate population risk, there is always a gap between estimator θ and θ^*.

3.5 Numerical results

In this section, we provide numerical examples, with both synthesized data and real data, to illustrate the performance of the proposed algorithms.

3.5.1 Synthesized data

We first use synthesized data. In this example, we focus on linear regression, in which

$$Y_i = X_i^T \theta^* + \epsilon_i, i = 1, 2, \cdots, N,$$

where $X_i \in \mathbb{R}^d$, θ^* is a $d \times 1$ vector and ϵ_i is the noise. We set $\mathbf{X} = [X_1, \cdots, X_N]$ as $d \times N$ data matrix.

In the simulation, we set the dimension $d = 20$, the total number of data $N = 50000$. We use $\mathcal{N}(0, 9)$ to independently generate each entry of θ^*. Here $\mathcal{N}(\nu, \sigma^2)$ denotes Gaussian variables with mean ν and variance σ^2. After θ^* is generated, we fix it. The data matrix $\mathbf{X}$ is generated randomly by Gaussian distribution with $\nu = 0$ and fixed known maximal and minimal eigenvalues of the correlation matrix $\mathbf{X}^T\mathbf{X}$. Let $\lambda_{max}(\cdot)$ and $\lambda_{min}(\cdot)$ denote the maximal and minimal eigenvalue of $\mathbf{X}^T\mathbf{X}$ respectively. In the following figures, we use $\lambda_{max}(\mathbf{X}^T\mathbf{X}) = 200$ and $\lambda_{min}(\mathbf{X}^T\mathbf{X}) = 2$ to generate the data matrix $\mathbf{X}$. We set ϵ_i as i.i.d. $\mathcal{N}(0, 1)$ random variable. Finally, we generate Y_i using the linear relationship mentioned above. In the simulation, we set the number of workers $m = 50$, and evenly distribute data among these machines. Furthermore, for robust gradient descent in [9] and proposed algorithm CNM, we set $|\mathcal{S}_0| = 1000$, $\xi = 1.5|\mathcal{S}_0|^{-\frac{1}{4}} = 0.2667, \xi_1 = 0.2667$ and $\xi_2 = 0.2667$. For the GIANT algorithm in [48] and proposed MNM, we set $\eta = 1$. For CNM, we set $\mu = 0.001$. We illustrate our results with 4 different cases: 1) 20 Inverse attack, in which each attacker first calculates the gradient and Newton's direction based on its local data but sends the inverse version of gradient information or vector information to the server; 2) 45 Inverse attack; 3) 20 Random attack, in which the attacker randomly generates gradient value; and 4) 45 Random attack. In our simulation, we compare four algorithms: 1) MNM in Table 3.1; 2) CNM described in Table 3.2; 3) Algorithm proposed in [9]; and 4) The GIANT algorithm proposed in [48]. The algorithm proposed in [9] is a first-order method which is robust to Byzantine attackers.

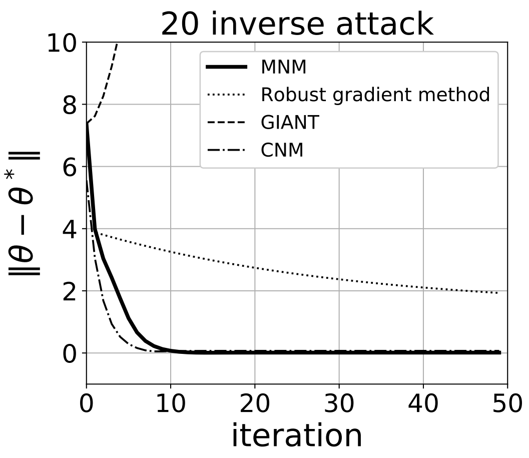

Figure 3.1: Synthesized data: 20 Inverse attack. Robust gradient method in [9], GIANT in [48]

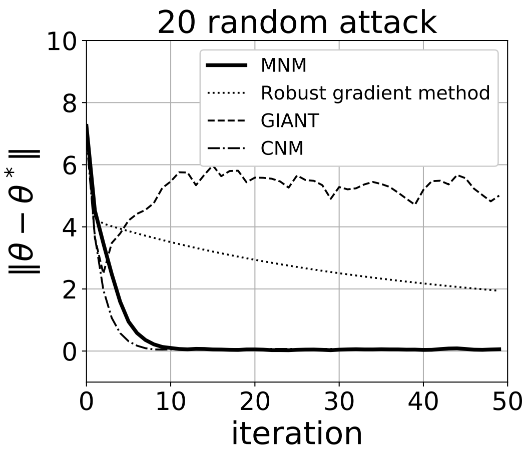

Figure 3.2: Synthesized data: 20 Random attack. Robust gradient method in [9], GIANT in [48]

Figures 3.1 and 3.2 plot the value of the norm of distance between estimated and the true parameter vs iteration with 20 inverse attacks and 20 random attacks respectively. From Figures 3.1 and 3.2, GIANT method does not converge, since computing average cannot defend Byzantine attacks, but the proposed MNM, CNM and robust gradient method can still converge. Furthermore, the proposed two algorithms still perform better than the robust gradient method in [9] in iteration, since our proposed algorithms compute Hessian matrix on each worker, which generate more information in each communication iteration.

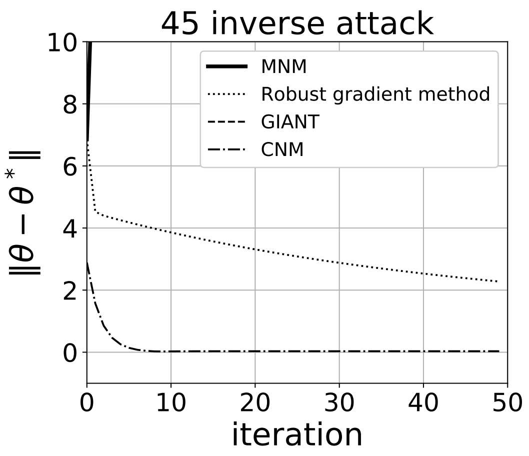

Figure 3.3: Synthesized data: 45 Inverse attack. Robust gradient method in [9], GIANT in [48]

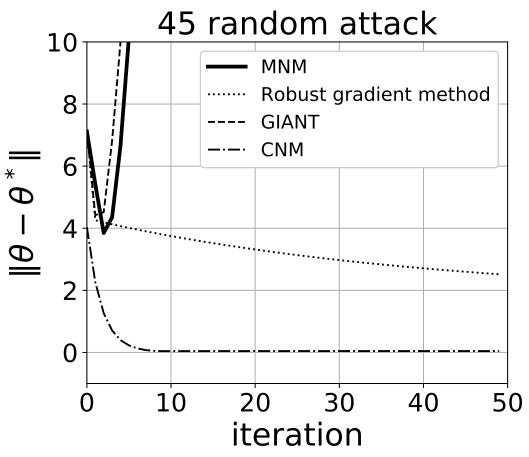

Figure 3.4: Synthesized data: 45 Random attack. Robust gradient method in [9], GIANT in [48]

Figures 3.3 and 3.4 plot the value of the norm of distance between the estimated and the true parameter vs iteration with 45 inverse attacks and 45 random attacks. From Figures 3.3 and 3.4, we can observe that GIANT and MNM do not converge, as more than half of the workers are compromised. However the proposed CNM and robust gradient method can still converge. Furthermore, the proposed CNM can benefit from Newton's direction information and outperforms the robust gradient method in [9] in iteration.

3.5.2 Real data

Now we test our algorithms on real datasets MNIST [25] and compare our algorithms with various existing gradient method work [9] and GIANT. MNIST is a widely used computer vision dataset that consists of 70,000 28×28 pixel images of handwritten digits 0 to 9. We use the handwritten images of 3 and 5, which are the most difficult to distinguish in this dataset, to build a logistic regression model. After picking all 3 and 5 images from the dataset, the total number of images is 13454. It is divided into a training subset of size 12000 and a testing subset of size 1454. For the dataset, we set the number of workers to be 50. For algorithm in [9] and algorithm CNM, we random pick 200 images from both subsets to build $\mathcal{S}_0$, For the proposed MNM and GIANT in [48], we set the learning rate $\eta = 1$. For CNM, we set $\mu = 0.0001$. Similar to the synthesized data scenario, we illustrate our results with four cases, namely 20 inverse attack, 20 random attack, 45 inverse attack and 45 random attack, and compare the performance of four algorithms. The following figures show how the testing accuracy varies with training iteration.

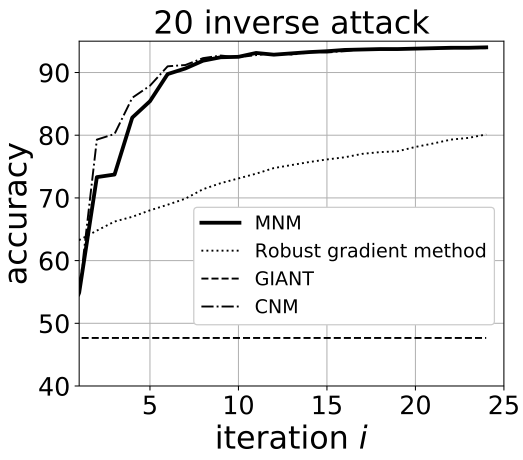

Figure 3.5: MNIST: 20 Inverse attack. Robust gradient method in [9], GIANT in [48]

Figures 3.5 and 3.6 illustrate the impact of two cases on different algorithms using MNIST respectively. Figures 3.5 and 3.6 show the GIANT fails to predict if there are 20 attackers. Our proposed algorithm and robust gradient descent still show high accuracy. Furthermore, the proposed MNM has a better performance than robust gradient descent in [9].

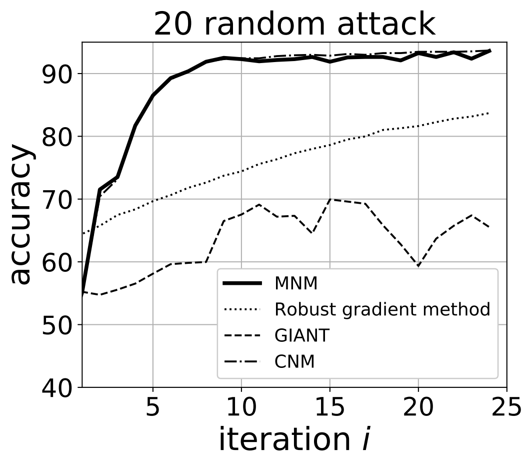

Figure 3.6: MNIST: 20 Randome attack. Robust gradient method in [9], GIANT in [48]

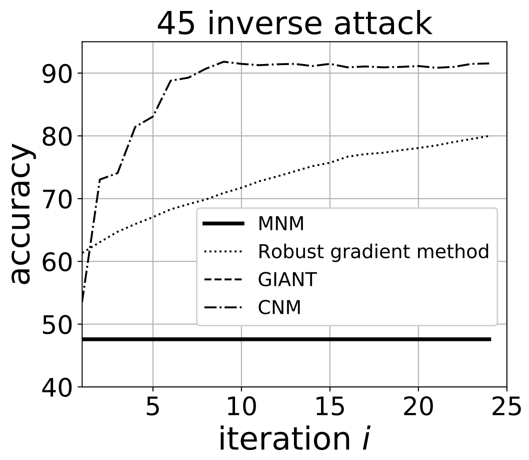

Figure 3.7: MNIST: 45 Inverse attack. Robust gradient method in [9], GIANT in [48]

We plot the impact of 45 attacker case on real data in Figures 3.7 and 3.8 using MNIST respectively. When there are 45 attackers, which is more than half of the total number of workers, MNM and GIANT can not properly work. CNM and robust gradient descent [9] still perform well, since these algorithm are generated to defend arbitrary number of attackers. Our proposed algorithms outperform the scheme using robust gradient descent in iteration.

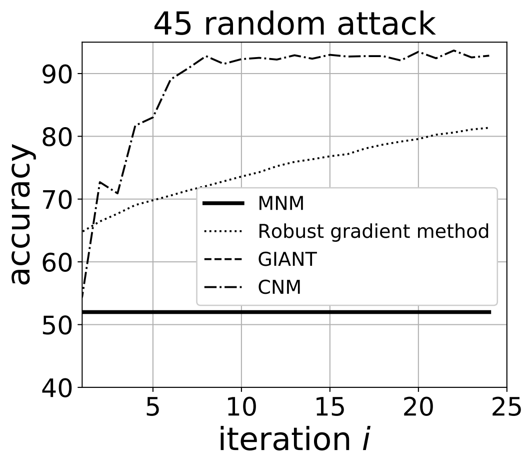

Figure 3.8: MNIST: 45 Random attack. Robust gradient method in [9], GIANT in [48]

3.6 Conclusion

In this chapter, we have proposed two robust distributed approximate Newton's method that can tolerant Byzantine attackers. We have shown that the proposed algorithms can converges to the neighborhood of true parameter and have provided numerical examples to illustrate the performance of the proposed algorithm.

Chapter 4

Distirbuted Zeroth-order ADMM Robust to Byzantine Attackers

4.1 Introduction

In this chapter, we focus on problems in which the first-order gradient information is difficult to obtain. In particular, we propose a new robust zeroth-order information based distributed optimization algorithm that is robust to Byzantine attacks. We name the method as zeroth-order adversarial robust alternating direction method of multipliers (ZOAR-ADMM). In the proposed method, at each iteration, each worker will first receive model parameter from its neighbors. Then each worker will test received parameter information by computing the distance from the received parameter to the model parameter computed using local data, and then sum all such distances obtained in history to build a deviation statistic for all neighbor workers. If the deviation statistic computed for its neighbor worker is smaller than a specially designed threshold, the worker will accept the model parameter from that neighbor. If the deviation statistic is larger than the threshold, the worker will reject the model parameter and decide that worker to be an attacker. After testing, each worker will first update dual variable by using accepted model parameter, then compute temporary model parameter based on accepting parameter and using deterministic gradient approximation from its own data and update new model parameter then broadcast it to its neighbors. By this method, we prove that the algorithm can solve the optimization problem and the objective function can converge to the minimum value. We show this result by first investigating how the distance between model parameter and optimal value is affected by the attack vector generated by the attackers, and then carefully analyzing how the proposed testing method can mitigate these effects and eventually proving that the value of objective function of the proposed algorithm will converge to the optimal value despite the presence of Byzantine attackers.

This chapter is organized as follows. In Section 4.2, we describe the model. In Section 4.3, we describe the proposed algorithm. In Section 4.4, we analyze the convergence property of the proposed algorithm. In Section 4.5, we provide numerical examples to validate the theoretic analysis. Finally, we offer several concluding remarks in Section 4.6. The proofs are collected in Appendix.

4.2 Model

In this section, we introduce our model. For an unknown distribution $\mathcal{D}$, our goal is to infer the model parameter $\theta^* \in \Theta$ of the unknown distribution. It is popular to formulate this inference problem as an optimization problem

$$\theta^* \in \arg\min_{\theta\in\Theta} F(\theta) = \mathbb{E}\{f(X,\theta)\}, \tag{4.1}$$

in which X is the data generated by the unknown distribution $\mathcal{D}$, $f : \mathcal{X} \times \Theta \to \mathbb{R}$ is the loss function, $\Theta \in \mathbb{R}^d$ is a closed convex set of all possible model parameters, and the expectation is over the distribution $\mathcal{D}$. $F(\theta)$ is called population risk function.

Since the expectation in (4.1) is over the unknown distribution $\mathcal{D}$, the population risk function $F(\theta)$ is unknown and hence we cannot solve (4.1) directly. Instead, one typically aims to minimize the empirical risk:

$$\min_{\theta\in\Theta} \frac{1}{N}\sum_{s=1}^{N} f(X_s,\theta), \tag{4.2}$$

which uses N data samples $X_s, s = 1, \dots, N$ generated by the unknown distribution $\mathcal{D}$. By solving (4.2), we obtain an estimate of the true model parameter θ^*. When the number of data points N is large, we can employ distributed optimization methods. In particular, we consider a network consisting of n workers bidirectionally connected with E edges. We can describe the network as a symmetric directed graph $\mathcal{G}_d = \{\mathcal{V}, \mathcal{A}\}$, where $\mathcal{V}$ is the set of workers with $|\mathcal{V}| = n$ and $\mathcal{A}$ is the set of directed edges with $|\mathcal{A}| = 2E$. In a distributed setup, a connected network of workers collaboratively minimize the sum of their local loss functions over a common optimization variable. Each worker generates local updates individually and communicates with its neighbors to reach a common minimizer in a consensus network. Then we can have a distirbuted optimization problem with population risk,

$$\min_{\theta_i,\phi_{ij}} \sum_{i=1}^{n} F^i(\theta_i), s.t.\theta_i = \phi_{ij}, \theta_j = \phi_{ij}, \forall(i,j) \in \mathcal{A}. \tag{4.3}$$

where $F^i(\theta_i) = \mathbb{E}\{f(X,\theta_i)\}$, where $f(X,\theta_i)$ repesents the loss function based on the data generated by the unknown distribution $\mathcal{D}$ and the model parameter θ_i. $\theta_i \in \mathbb{R}^d$ is the local optimization variable at worker i and $\phi_{ij} \in \mathbb{R}^d$ is an auxiliary variable imposing the consensus constraint on neighbor workers i and j. Again, since we do not know the distribution $\mathcal{D}$, we cannot solve (4.3) directly. Instead we can focus on the distributed optimization problem for empirical risk function formulated as follows

$$\min_{\theta_i,\phi_{ij}} \sum_{i=1}^{n} \overline{f}^{(i)}(\theta_i), s.t.\theta_i = \phi_{ij}, \theta_j = \phi_{ij}, \forall(i,j) \in \mathcal{A}. \tag{4.4}$$

where $\overline{f}^{(i)}(\theta_i) = \frac{1}{|\mathcal{S}_i|}\sum_{s\in\mathcal{S}_i} f(X_s,\theta_i)$ with $\mathcal{S}_i$ being the set of data samples at worker i.

Define $\theta \in \mathbb{R}^{nd}$ as a vector concatenating all θ_i, $\phi \in \mathbb{R}^{2Ed}$ as a vector concatenating all ϕ_{ij}, then (4.4) can be written in a matrix form as

$$\begin{aligned} \min_{\theta,\phi} \quad & f(\theta) + \Gamma(\phi), \\ s.t. \quad & A\theta + B\phi = 0, \end{aligned} \tag{4.5}$$

where $f(\theta) = \sum_{i=1}^{n} \overline{f}^{(i)}(\theta_i)$ and $\Gamma(\phi) = 0$. Here $A = [A_1; A_2]; A_1, A_2 \in \mathbb{R}^{2Ed\times nd}$ are both composed of $2E \times n$ blocks of $d \times d$ matrices. If $(i,j) \in \mathcal{A}$ and ϕ_{ij} is the qth block of ϕ, then the (q,i)th block of A_1 and the (q,j)th block of A_2 are $d \times d$ identity matrices I_d; otherwise the corresponding blocks are $d \times d$ zero matrices 0_d. Also, we have $B = [-I_{2Ed}; -I_{2Ed}]$ with I_{2Ed} being a $2Ed \times 2Ed$ identity matrix.

In this chapter, we assume that $F(\theta)$ and θ satisfy the following assumptions.

Assumption 7. *$F(\theta)$ is m_F-strongly convex and $F(\theta)$ has M_F-Lipschitz gradients on $\theta \in \Theta$ for any θ.*

Assumption 8. *The constrain set Θ is convex and compact, there exists some constant R such that $\|\theta - \theta'\| \leq R$ for any $\theta, \theta' \in \Theta$.*

These assumptions are common assumptions in existing works for optimization problem [5, 28].

The iterative updates of the distributed ADMM to solve problem (4.4) is given in [39]. In particular, consider the augmented Lagrangian of (4.5), we will have

$$L(\theta, \phi, \nu) = f(\theta) + \langle \nu, A\theta + B\phi \rangle + \frac{c}{2}\|A\theta + B\phi\|^2. \tag{4.6}$$

By using ADMM method, the updates are

$$\begin{aligned} \nabla f(\theta^{k+1}) + A^T\nu^{k+1} + cA^TB(\phi^k - \phi^{k+1}) &= 0, & (4.7) \\ B^T\nu^{k+1} &= 0, & (4.8) \\ \nu^{k+1} - \nu^k - c(A\theta^{k+1} + B\phi^{k+1}) &= 0. & (4.9) \end{aligned}$$

By letting $\nu = [\beta; \gamma]$ with $\beta, \gamma \in \mathbb{R}^{2Ed}$ and recalling $B = [-I_{2Ed}; -I_{2Ed}]$, we will have $\gamma = -\beta$. By choosing $\phi^0 = \frac{1}{2}M_+^T\theta^0$, the ADMM form will be reduced to the following form:

$$\theta - update : \nabla f(\theta^{k+1}) + M_-\beta^{k+1} - \frac{c}{2}M_+M_+^T\theta^k \tag{4.10}$$

$$+\frac{c}{2}M_+M_+^T\theta^{k+1} = 0, \tag{4.11}$$

$$\beta - update : \beta^{k+1} - \beta^k - \frac{c}{2}M_-^T\theta^{k+1} = 0, \tag{4.12}$$

where $\beta \in \mathbb{R}^{2Ed}$, the matrices $M_+ = A_1^T + A_2^T$ and $M_- = A_1^T - A_2^T$. Let $W \in \mathbb{R}^{nd\times nd}$ be a block diagonal matrix with its (i, i)th block being the degree of agent i multiplying I_d and other blocks being 0_d, $L_+ = \frac{1}{2}M_+M_+^T$, $L_- = \frac{1}{2}M_-M_-^T$, and $W = \frac{1}{2}(L_+ + L_-)$. By defining a new multiplier $\alpha = M_-\beta \in \mathbb{R}^{nd}$, the algorithm reduces to the following form:

$$\theta - update : \nabla f(\theta^{k+1}) + \alpha^k + 2cW\theta^{k+1} = cL_+^{k+1}\theta^k, \tag{4.13}$$

$$\alpha - update : \alpha^{k+1} - \alpha^k - cL_-^{k+1}\theta^{k+1} = 0. \tag{4.14}$$

Note $\theta = [\theta_1, ...\theta_n]$, $\alpha = [\alpha_1, ..., \alpha_n] \in \mathbb{R}^{nd}$, and there is an optimal solution $\theta^* \in \Theta$. These matrices are related to the underlying network topology. From above, we can find that W is a block diagonal matrix with its (i, i)th being the number of neighbor of worker i. L_- is the Laplacian matrix, and L_+ is the nonnegative Laplacian matrix.

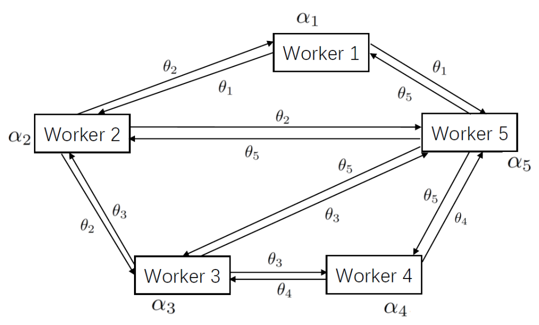

Figure 4.1: Information flow of ADMM algorithm in [39].

Using the matrices defined above, the matrices form iterative updates in (4.14) can be distributed to each worker. For example, Figure 4.1 illustrates information flow of 5 workers in the network by using this algorithm. In iteration k, worker i will receive all model parameter $\theta_j^k, j \in \mathcal{N}_i$ from its neighbors, then it will first calculate α_i^k based on received information:

$$\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i|\theta_i^k - c\sum_{j\in\mathcal{N}_i}\theta_j^k. \tag{4.15}$$

Then it will update θ_i^{k+1} by solving

$$\nabla\overline{f}^{(i)}(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i|\theta_i^{k+1} = c|\mathcal{N}_i|\theta_i^k + c\sum_{j\in\mathcal{N}_i}\theta_j^k, \tag{4.16}$$

based on received model information θ_j^k and local data. After updating θ_i^{k+1}, worker i will broadcast it to its all neighbors. Algorithm 4.1 (from [39]) summarizes these steps.

In this chapter, we consider two problems based on Algorithm 4.1. First, we consider a system with Byzantine attackers, in which an unknown subset of workers might be compromised. In each iteration, compromised worker i can send arbitrary information to its neighbors which can be defined as: $z_i = \theta_i + e_i$. In particular, let $\mathcal{B}$ denote the set of compromised workers, then e_i has the following form

$$e_i = \begin{cases} 0 & i \notin \mathcal{B} \\ \star & i \in \mathcal{B} \end{cases} \tag{4.17}$$

in which $\star$ denotes an arbitrary vector chosen by the attacker. Secondly, We also consider the system where gradient or subgradient information is hard to be explicitly evaluated. Instead, we will use a deterministic estimator $g_i(\theta_i)$ to estimate $\nabla\overline{f}^{(i)}(\theta_i)$, which approximates each coordinate of the gradient and then sums them up [1]:

$$g_i(\theta_i) = \frac{1}{m}\sum_{s\in\mathcal{S}_i}\sum_{l=1}^{d}\frac{f(X_s,\theta_i+uv_l)-f(X_s,\theta_i-uv_l)}{2u}v_l.$$

Here u is a scalar, whose value will be specified in the algorithm analysis, and v_l is a standard basis vector with 1 at its lth coordinate.

Then the corresponding algorithm becomes

$$g_i(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i|\theta_i^{k+1} = c|\mathcal{N}_i|z_i^k + c\sum_{j\in\mathcal{N}_i} z_j^k, \tag{4.18}$$

$$\alpha_i^{k+1} = \alpha_i^k + c|\mathcal{N}_i|z_i^{k+1} - c\sum_{j\in\mathcal{N}_i} z_j^{k+1}. \tag{4.19}$$

For a clearer presentation, we will use following equivalent form of the updates in analysis when there are Byzantine attackers:

$$\theta - update : g(\theta^{k+1}) + \alpha^k + 2cW^{k+1}\theta^{k+1} = cL_+^{k+1}z^k, \tag{4.20}$$

$$\alpha - update : \alpha^{k+1} - \alpha^k - cL_-^{k+1}z^{k+1} = 0, \tag{4.21}$$

where $g(\theta) = \sum_{i=1}^n g_i(\theta_i)$. Compared with (4.14), θ^k is replaced by z^k and θ^{k+1} is replaced by z^{k+1}. The goal of this chapter is to design robust zeroth-order algorithms, by designing proper tests for each worker that can tolerate Byzantine attacks. For $g(\theta)$ generated by deterministic estrimator, we will use $g(\theta)$ to estimate $\nabla f(\theta)$. For $\nabla f(\theta)$, we have following assumption, which are similar to those used in [13], [41,51],

We also assume data in each worker has following assumption.

Assumption 9. *For any $\delta \in (0, 1/m)$, there exists an $M_f = M_f(\delta)$ and $m_f = m_f(\delta)$ such that*

$$\mathbf{Pr}\left\{\forall\theta,\theta' \in \Theta, m_f \leq \frac{\|\nabla f(X,\theta) - \nabla f(X,\theta')\|}{\|\theta-\theta'\|} \leq M_f\right\} \geq 1 - \frac{\delta}{3}. \tag{4.22}$$

Assumption 9 ensures that $\nabla f(X,\theta)$ in each worker is M_f-Lipschitz and $f(X,\theta)$ is m_f strongly convex with high probability.

Algorithm 4.1 ADMM [39]

```
Initialize $\theta^1 = 0, c, \alpha^0 = 0, T$.
for $k = 1$ to $T$ do
    For the worker $i$:
    1: Recieves the model parameter $\theta_j^k$ from its neighbor;
    2: Computes $\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i|\theta_i^k - c\sum_{j\in\mathcal{N}_i}\theta_j^k$
    3: Solves $\nabla f_i(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i|\theta_i^{k+1} = c|\mathcal{N}_i|\theta_i^k + c\sum_{j\in\mathcal{N}_i}\theta_j^k$
    to gets updated $\theta_i^{k+1}$ and communicates it with its neighbors;
end for
```

Algorithm 4.2 ZOAR-ADMM

```
Initialize $\theta^1 = 0, c, \alpha^0 = 0, T, U$.
for $k = 1$ to $T$ do
    For the worker $i$:
    1: Recieves the model parameter $\theta_j^k$ from its neighbor;
    if $\sum_{t=1}^k \|\theta_i^t - \theta_j^t\| > U$ then
        2: worker $i$ detects that worker $j$ is an attacker, rejects $\theta_j^k$ and removes worker $j$ from $\mathcal{N}_i^k$;
    else
        2: worker $i$ accepts $\theta_j^k$;
    end if
    3: Computes $\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i^k|\theta_i^k - c\sum_{j\in\mathcal{N}_i^k}\theta_j^k$
    4: Solves $g_i(\theta_i^{k+1}) + \alpha_i^k + 2c|\mathcal{N}_i^k|\theta_i^{k+1} = c|\mathcal{N}_i^k|\theta_i^k + c\sum_{j\in\mathcal{N}_i^k}\theta_j^k$
    to gets updated $\theta_i^{k+1}$ and communicates it with its neighbors;
end for
```

4.3 Algorithm

In this section, we describe our algorithm in distributed network that can tolerate Byzantine attacks in ADMM updates.

If there is no network, each worker will compute model parameter by itself, then in each iteration, different workers will have different model parameter. But in a network, workers will communicate with its neighbor, then each worker can know the model parameter deviation between itself and its neighbor. The main idea of our algorithm is to use this model parameter deviation to detect Byzantine attackers. As we will show in Lemma 4.4, for the case where all the workers are honest, the deviation statistic $\sum_{t=1}^k \sum_{(i,j)\in\mathcal{A}} \|\theta_i^t - \theta_j^t\|$ will be bounded by a quantity value U no matter what the value k is. From this property, this bound can serve as the standard threshold for each worker to decide whether its neighbor is honest or not. We will discuss how to choose U in Lemma 4.4 in the analysis. Inspired by this bound, in our algorithm, each worker maintains the local deviation statistic $\sum_{t=1}^{k} \|\theta_i^t - \theta_j^t\|$ for every neighboring worker j, and compares it with U to test if neighboring worker j is providing a reasonable value or not. The local deviation statistic from an honest worker will always smaller than U, no matter how many iterations have passed.

In particular, in iteration k, worker i tests all the model information θ_j^k from its neighbor $j, j \in \mathcal{N}_i$. If the local deviation statistic $\sum_{t=1}^{k} \|\theta_i^t - \theta_j^t\|$ from neighbor j is larger than U, neighbor j will be considered as a Byzantine attacker. The model parameter sent by a Byzantine attacker will be rejected forever and worker i will not send information to worker j. Worker j will be removed from set $\mathcal{N}_i$ and worker i will be removed from set $\mathcal{N}_j$. Then worker i and worker j will have new neighbor set $\mathcal{N}_i^k$ and $\mathcal{N}_j^k$. After testing all neighbors, worker i updates α_i^k first:

$$\alpha_i^k = \alpha_i^{k-1} + c|\mathcal{N}_i^k|\theta_i^k - c \sum_{j \in \mathcal{N}_i^k} \theta_j^k. \tag{4.23}$$

Then worker i will update θ_i by solving

$$g_i(\theta_i) + \alpha_i^k + 2c|\mathcal{N}_i^k|\theta_i = c|\mathcal{N}_i^k|\theta_i^k + c \sum_{j \in \mathcal{N}_i^k} \theta_j^k, \tag{4.24}$$

where we use deterministic gradient estimator $g_i(\theta_i)$ using its own local m data:

$$g_i(\theta_i) = \frac{1}{m} \sum_{s \in \mathcal{S}_i} \sum_{l=1}^{d} \frac{f(X_s, \theta_i + u_k v_l) - f(X_s, \theta_i - u_k v_l)}{2u_k} v_l, \tag{4.25}$$

here $u_k = \frac{1}{dk^2}$ is a scalar and v_l is a standard basis vector with 1 at it lth coordinate. After worker i update θ_i, it will communicate its value with its neighbors.

Main steps of the algorithm are list in Algorithm 4.2.

4.4 Convergence Analysis

In this section, we analyze the convergence property of ZOAR-ADMM in the consensus network with Byzantine attackers.

Before presenting detailed analysis, here we introduce some notations for the network and describe the high level ideas. On iteration k, when we describe the network, we let $Q^k = LD^{\frac{1}{2}}L^T$, where $LDL^T = \frac{L_-^k}{2}$ is the singular value decomposition of the positive semidefinite matrix $\frac{L_-^k}{2}$, and L_-^k represents the Laplacian matrix of the network at iteration k. We will define a new auxiliary sequence $r^k = \sum_{s=0}^{k} Q^s(\theta^s + e^s)$ to represent the accumulation of the network constraint in optimization problem over iterations. In addition, we define matrix p and matrix G as

$$p^k = \begin{bmatrix} r^k \\ \theta^k \end{bmatrix}, G^{k+1} = \begin{bmatrix} cI & 0 \\ 0 & cL_+^{k+1}/2 \end{bmatrix}. \tag{4.26}$$

We also define two constants that will be used in the analysis:

$$\Delta_1 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(3/\delta))/m}, \tag{4.27}$$

$$\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1 + \tau_2)/m} \tag{4.28}$$

with $\tau_1 = d\log 18 + d\log(M_F \vee M_f/\sigma_2)$, $\tau_2 = 0.5d\log(m/d) + \log(6/\delta) + \log(\frac{2r\sigma_2^2\sqrt{m}}{\alpha_2\sigma_1})$.

In our analysis, we will first study the properties of the zeroth-order gradient estimation at an honest worker. We will then analyze the impacts of attacks on each iteration of ADMM. Finally, we will show that our proposed algorithm can reduce the error caused by Byzantine attackers and the function value will converge to the function value based on the optimal parameter.

4.4.1 Bound of zeroth-order gradient estimation

In this section, we will derive an upper bound on the gradient estimate at an honest worker. This bound will be used in the subsequent analysis.

Recall that we have $f(\theta) = \sum_{i=1}^{n} \overline{f}^{(i)}(\theta_i)$. To consider the difference between zeroth-order gradient estimation and the true unknown gradient of $f(\theta)$, we denote $h(\theta) = \nabla f(\theta) - g(\theta)$. For $h(\theta)$, we have

Lemma 4.1. *([1]) Under Assumptions 7-9, in iteration k, for any $\delta \in (0, 1)$, with probability at least $1 - \delta/3$, the deterministic estimator $g(\theta^k)$ satisfies*

$$\|g(\theta^k) - \nabla f(\theta^k)\|^2 \leq \frac{nM_f^2 d^2 u_k^2}{4m}. \tag{4.29}$$

Lemma 4.1 illustrates that there is a bound for the distance between zeroth-order estimate and the true gradient. From this lemma and assumptions mentioned above, we have the following upper bound on $\|g_i(\theta)\|$.

Lemma 4.2. *Under Assumptions 2-3, 7-9, in iteration k, for any $\delta \in (0, 1)$, with probability at least $(1 - \delta)$, the deterministic estimator $g_i(\theta_i^k)$ satisfies*

$$\|g_i(\theta_i^k)\| \leq V_k + M_f \|\theta_i^k - \theta^*\|, \tag{4.30}$$

where $V_k = \frac{M_f^2 d^2 u_k^2}{m} + \Delta_1$.

Proof. Please see Appendix C.1 for details. $\square$

4.4.2 Impact of Byzantine attackers in ADMM

In this section, we analyze the impact of Byzantine attacks on the iterations of ADMM. To facilitate the analysis of the algorithm, we show that the algorithm has the following equivalent form.

Lemma 4.3. *The algorithm satisfies*

$$g(\theta^{k+1}) = 2cW^{k+1}e^{k+1} - cL_+^{k+1}(z^{k+1} - z^k) - 2cQr^{k+1},$$

where $W^{k+1} = \frac{L_+^{k+1} + L_-^{k+1}}{2}$ and Q is a matrix that makes $2Qr^{k+1} = \sum_{s=0}^{k+1} L_-^s(\theta^s + e^s)$

Proof. Please see Appendix C.2 for details. ☐

Using this lemma, we are ready to show that, if each node blindly accepts information from neighboring workers, Byzantine attackers can change the distance between θ^k and θ^* by changing the model parameter during information transmission.

Theorem 5. *If Assumptions 2-3, 7-9 hold, by choosing $u_k = \frac{1}{dk^2}$ for k iteration, for any $\delta \in (0,1)$, with optimal value*

$$p = \begin{bmatrix} 0 \\ \theta^* \end{bmatrix}, \tag{4.31}$$

then with probability at least $(1-\delta)^n$, we have

$$\|p^{k+1} - p\|^2_{G^{k+1}} \leq \frac{1}{1+\rho}\left(\|p^k - p\|^2_{G^{k+1}} + \Delta(k+1)\right), \tag{4.32}$$

where

$$\begin{aligned}\Delta(k+1) &= c\frac{\sigma^2_{max}(L^{k+1}_+)}{2\sigma_{min}(L^{k+1}_-)}\|e^k\|^2 + \frac{\sqrt{n}M_f R}{\sqrt{m}k^2} + \Delta_1 R \\ &\quad + c^2\sigma^2_{max}(L^{k+1}_+)\|e^k\|^2 + c^2\sigma^2_{max}(L^{k+1}_-)\|e^{k+1}\|^2 \\ &\quad + c\langle e^{k+1}, 2Qr^{k+1}\rangle + 2(\mu-1)nV^2_{k+1} + 8\Delta_2 R^2,\end{aligned} \tag{4.33}$$

and

$$\rho = \min\left\{\frac{(\mu-1)\sigma^2_{min}(L^{k+1}_-)}{2\mu\sigma^2_{max}(L^{k+1}_+)\sigma_{max}(L^0_-)}, \frac{m_f}{\frac{c\sigma^2_{max}(L_+)}{2} + \frac{\mu}{c}2M_f^2\sigma^{-2}_{min}(L^{k+1}_-)\sigma_{max}(L^0_-)}\right\} > 0. \tag{4.34}$$

Proof. Please see Appendix C.3 for details. ☐

From this theorem, we can see that when there is no attacker, i.e., $\|e^k\| = \|e^{k+1}\| = 0$, then $\Delta(k+1)$ decreases and goes to $2(\mu-1)\Delta_1^2 + \Delta_1 R + 8\Delta_2 R^2$ as $k \to \infty$, which is generated from the approximation of population risk function by using empirical risk function. We can find the sequence $\|p^k - p\|^2_{G^k}$ converges linearly to the neighbor of optimal p with a rate of $\frac{1}{1+\rho}$ when there is no attacker in the network. However, when there are attackers, this theorem shows how the error values $\|e^k\|$ introduced by the attackers affect the term $\Delta(k+1)$, and these errors will accumulate after each iteration. These error values can be any value decided by the Byzantine attackers. The bound will become larger and larger, the ADMM algorithm will not converge.

To provide further insights on how attackers can impact the algorithm, we analyze how the convergence rate is related to the value of ρ. In the no attacker case, by maximizing ρ, we can have a better convergence result. Then we will show how to maximal ρ.

Proposition 4.1. *If the algorithm parameter c is chosen as*

$$c = \frac{2M_f\sqrt{\sigma_{max}(L_-^0)\sqrt{\mu}}}{\sigma_{max}(L_+^{k+1})\sigma_{min}(L_-^{k+1})}, \tag{4.35}$$

and

$$\mu = 1 + \frac{K_L^2\sigma_{max}(L_-^0)}{K_f^2} - \frac{K_L\sigma_{max}(L_-^0)}{2K_f}\sqrt{\frac{8}{\sigma_{max}(L_-^0)} + 4\frac{K_L^2}{K_f^2}},$$

then we have

$$\rho = \frac{1}{2K_f}\sqrt{\frac{8}{\sigma_{max}(L_-^0)K_{L^{k+1}}^2} + \frac{4}{K_f^2}} - \frac{1}{2K_f^2} \tag{4.36}$$

maximizes the value of ρ in iteration $k+1$, where $K_{L^{k+1}} = \frac{\sigma_{max}(L_+^{k+1})}{\sigma_{min}(L_-^{k+1})}$ and $K_f = \frac{M_f}{m_f}$.

Proof. Please see Appendix C.4 for details. □

The minimum non-zero singular value of the signed Laplacian matrix L_- and the maximum singular value of signless Laplacian matrix L_+ are related to network connectedness but former is less. Roughly speaking, larger L_+ and L_- mean stronger connectedness, and a larger K_L means weaker connectedness. From this proposition, we can observe that the value of ρ is related to K_L. The value of ρ decreases as K_L increases. This proposition suggests that another way that the Byzantine attacker can influence the result is to reduce the network connectedness, which makes the convergence arbitrarily slow.

In summary, Theorem 5 and Proposition 4.1 provide useful insights the impact the adversary attacks. In particular, when we consider the defending method as in the proposed ZOAR-ADMM, we are going to identify the Byzantine attackers and remove them from the network. Then in the network, the attackers may have two difference methods for attacking: 1) From insights in Theorem 5, the attacker may choose to change the model parameter but only make small changes so that changed model parameter pass the test to accumulate the wrong information; 2) From insights in Proposition 4.1, the attacker may choose to make large changes to the value so it does not pass the test, which will break the network and change the value of ρ and impact the convergence.

4.4.3 Convergence analysis of ZOAR-ADMM

Using the insights obtained in Section 4.4.2, in this section, we will prove the convergence of ZOAR-ADMM when there are Byzantine attackers in the network.

In Section 4.3, we mention that, when there is no Byzantine attackers, the deviation statistic $\sum_{t=1}^{k}\|Q\theta^t\|$ will be bounded by some value no matter what the value k. The following lemma shows how to find such a bound.

Lemma 4.4. *Consider a network without attacker, starting from $\theta^0 = 0$ and $u_t = \frac{1}{dt^2}$, for any $\delta \in (0,1)$, with probability at least $(1-\frac{\delta}{3})^n$, we have*

$$\frac{1}{T}\sum_{t=1}^{T}\|Q\theta^t\| \le \frac{1}{4T}\left(\sigma_{max}(L_+^0)R^2 + \frac{4C}{\sigma_{min}(L_-^0)c^2} + 4\right) + \frac{R}{2cT}\frac{\sqrt{n}M_f\pi^2}{12\sqrt{m}}, \tag{4.37}$$

where $C = nV_1^2 + M_f^2R^2$.

Proof. Please see Appendix C.5 for details. □

Using this lemma, we can set the bound for testing as $U = \frac{1}{2\sqrt{2}}\left(\sigma_{max}(L_+^0)R^2 + \frac{4C}{\sigma_{min}(L_-^0)c^2} + 4\right) + \frac{R}{c}\frac{\sqrt{n}M_f\pi^2}{12\sqrt{2m}}$. When there is no attacker, from Lemma 4.4, $\sum_{t=1}^{T}\|Q\theta^t\| \le U/\sqrt{2}$. Note that $\sum_{t=1}^{T}\|Q\theta^t\| = \frac{1}{\sqrt{2}}\sum_{t=1}^{T}\sum_{(i,j)\in\mathcal{A}}\|\theta_i^t - \theta_j^t\|$, thus, we will have $\frac{1}{\sqrt{2}}\sum_{t=1}^{T}\|\theta_i^t - \theta_j^t\| \le U/\sqrt{2}, \forall(i,j) \in \mathcal{A}$. Then we can design our attacker testing method in the following way: in each iteration k, each worker i maintains the local deviation statsitics $\sum_{t=1}^{k}\|\theta_i^t - \theta_j^t\|$ for every neighbor worker $j \in \mathcal{N}_i$. For an honest worker, this deviation statsitics will not exceed U. If this value is greater than U, then worker j will be regarded as a Byzantine attacker by worker i, since if in one iteration, this value is greater than U, then after this iteration, the value will still be greater, so worker i will reject the information from worker j forever.

Next, we show that the proposed ZOAR-ADMM algorithm can converge to the optimal value in a consensus network. Considering after T iteration, the whole consensus network has been attacked to several small consensus networks. Assume first $\hat{n} \leq n$ workers are in one consensus network. Then consider the initial network between these workers, we will have $\hat{L}_+$, $\hat{L}_-$ for such network and $\hat{f}(\theta) = \sum_{i=1}^{\hat{n}} \overline{f}^{(i)}(\theta_i)$. Then we have the following theorem showing the proposed algorithm can work in a consensus network.

Theorem 6. *If Assumptions 2-3, 7-9 hold, there exists optimal* $p = \begin{bmatrix} r \\ \theta^* \end{bmatrix}$*, with* $r = 0$ *and* $\hat{\theta}_T = \frac{\sum_{k=1}^{T} \theta^k}{T}$*, with* $u_k = \frac{1}{dk^2}$ *and for any* $\delta \in (0,1)$*, with probability* $(1-\delta)^{\hat{n}}$*, it holds*

$$\hat{f}(\hat{\theta}_T) - \hat{f}(\theta^*) \ \leq \ \frac{1}{T}\left(\|\hat{p}^0 - p\|^2_{\hat{G}^1} + c\frac{\sigma^2_{max}(\hat{L}^T_+)}{\sigma^2_{min}(\hat{L}^T_-)} 8E^2U^2 + \frac{\pi^2}{6}\frac{\hat{n}\sqrt{\hat{n}}M_f R}{2n\sqrt{m}} \right). \tag{4.38}$$

Proof. Please see Appendix C.6 for detail. □

This theorem shows that, when the whole network is separated by Byzantine attackers into several smaller network, ZOAR-ADMM can work in each small consensus network. Now we consider the convergence of ZOAR-ADMM in the whole network. Consider different network in whole algorithm, for signless Laplacian matrix, we have $\|x^k - x^*\|^2_{\frac{L^k_+}{2}} = \frac{1}{4}\sum_{i=1}^{m}\sum_{j \in \mathcal{N}_i} \|x_i - x^* + x_j - x^*\|^2$. Now consider the whole network, define $f_{all}(x) = \sum f(x) = \sum_{i=1}^{n} f_i(x_i)$, which consider the whole network, then we get the following theorem for whole network.

Theorem 7. *If Assumptions 2-3, 7-9 holds, there exists optimal* $p = \begin{bmatrix} r \\ \theta^* \end{bmatrix}$*, with* $r = 0$ *and*

$\hat{\theta}_T = \frac{\sum_{k=1}^{T} \theta^k}{T}$, *with* $u_k = \frac{1}{dk^2}$ *and for any* $\delta \in (0,1)$, *with probability* $(1-\delta)^n$, *it holds*

$$f(\hat{\theta}_T) - f(\theta^*) = \sum \hat{f}(\hat{\theta}_T) - \hat{f}(\theta^*) \tag{4.39}$$
$$\leq \frac{1}{T}\left(\|p^0 - p\|_{G^1}^2 + c\frac{\sigma_{max}^2(L_+^T)}{\sigma_{min}^2(L_-^T)} 8E^2U^2 + \frac{\pi^2}{6}\frac{\sqrt{n}M_f R}{2\sqrt{m}}\right). \tag{4.40}$$

Proof. Please see Appendix C.7 for details. □

This theorem shows that the algorithm achieves a sub-linear convergence rate of $\mathcal{O}(\frac{1}{T})$. The upper bound in (4.40) introduces two additional terms. The first term comes from the method for defending against Byzantine attackers and the second term comes from the estimate gradient by using zeroth-order approximation.

4.5 Numerical results

In this section, we provide numerical examples, with both synthesized data and real data, to illustrate the performance of the proposed algorithm.

4.5.1 Synthesized data

We first use synthesized data. In this example, we focus on linear regression, in which

$$Y_i = H_i^T x^* + \epsilon_i, i = 1, 2, \cdots, N,$$

where $H_i \in \mathbb{R}^d$, x^* is a $d \times 1$ vector and ϵ_i is the noise. We set $\mathbf{H} = [H_1, \cdots, H_N]$ as $d \times N$ data matrix.

In the simulation, we set the dimension $d = 10$, the total number of data $N = 50000$. We use $\mathcal{N}(0, 9)$ to independently generate true model parameter x^*, where $\mathcal{N}(\nu, \sigma^2)$ denotes Gaussian variables with mean ν and variance σ^2. After x^* is generated, we fix it. The data matrix $\mathbf{H}$ is generated randomly by Gaussian distribution with $\nu = 0$ and fixed known maximal and minimal eigenvalues of the correlation matrix $\mathbf{H}^T\mathbf{H}$. Let $\lambda_{max}(\cdot)$ and $\lambda_{min}(\cdot)$ denote the maximal and minimal eigenvalue of $\mathbf{H}^T\mathbf{H}$ respectively. In the following figures, we use $\lambda_{max}(\mathbf{H}^T\mathbf{H}) = 100$ and $\lambda_{min}(\mathbf{H}^T\mathbf{H}) = 1$ to generate the data matrix $\mathbf{H}$. We set the white noise ϵ_i as i.i.d. $\mathcal{N}(0, 1)$ random variable. Finally, we generate Y_i using the linear relationship mentioned above. In the synthesized data simulation, we set the number of workers $n = 100$, and data are evenly distributed in each worker. The original network is generated by a connected Erdos-Renyi graph $ER(100, 0.2)$, meaning that 100 workers connect with each other with probability 0.2. We first randomly select 20 workers to be attackers. We illustrate our results with 2 different cases: 1) 20 Inverse attack, in which each attacker first calculates the gradient based on its local data but sends the inverse version of gradient information or vector information to the server; 2) 20 Random attack, in which the attacker randomly generates gradient value. In our simulation, we compare 2 algorithms: 1) The proposed ZOAR-ADMM as presented in Algorithm 4.2; 2) The DS-ADMM in [28] which considers zeroth-order ADMM with two times communication in each iteration.

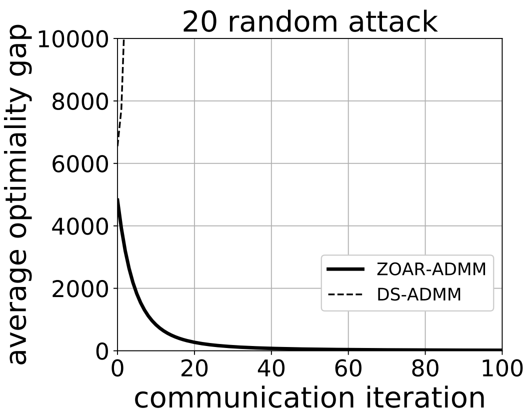

Figure 4.2: Optimality gap comparison using synthesized data: 20 Random attack.

Figures 4.2 and 4.3 plot the value of the average optimality gap vs iteration with 20 inverse attacks and 20 random attacks respectively, where the average optimality gap is defined as: $\frac{1}{n}\sum_{j=1}^{n}[\sum_{i=1}^{n} f_i(x_j^k) - \sum_{i=1}^{n} f_i(x^*)]$. From Figures 4.2 and 4.3, we can see that DS-ADMM method does not converge, since computing average cannot defend Byzantine attacks. On the other hand, the proposed ZOAR-ADMM can still converge, since it helps workers to detect the Byzantine

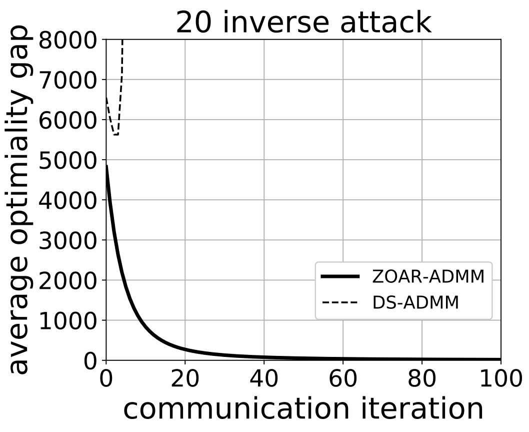

Figure 4.3: Optimality gap comparison using synthesized data: 20 Inverse attack.

attackers and converge under the trusted sub network.

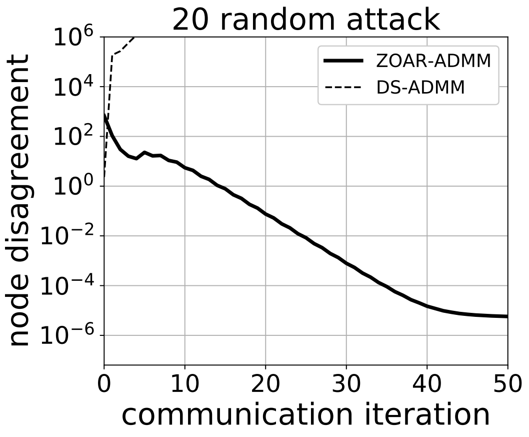

Figure 4.4: Node disagreement comparison using synthesized data: 20 Random attack.

Figures 4.4 and 4.5 plot the value of $\|Q^0 x^k\|^2$ vs iteration with 20 random attacks and 20 inverse attacks. As we discussed above, $\|Q^0 x^k\|^2$ can be used to show the node disagreement. From Figures 4.4 and 4.5, we can observe that DS-ADMM has a large disagreement, since the attackers successfully make the algorithm fail. However the proposed ZOAR-ADMM has a small disagreement.

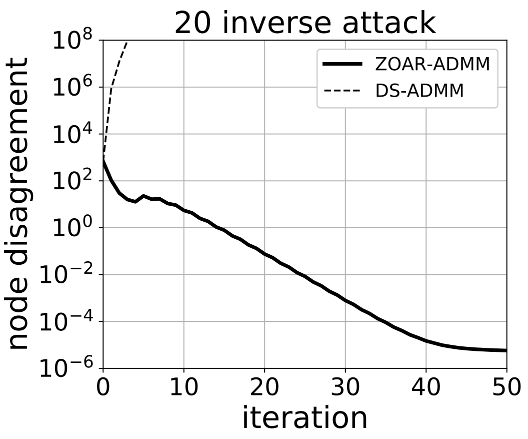

Figure 4.5: Node disagreement comparison using synthesized data: 20 Inverse attack.

4.5.2 Real data

Now we test our algorithms on real datasets MNIST [25] and compare our algorithms with the existing zeroth order method in [28]. MNIST is a widely used computer vision dataset that consists of 70,000 28×28 pixel images of handwritten digits 0 to 9. We use the handwritten images of 3 and 5, which are the most difficult to distinguish in this dataset, to build a logistic regression model. After picking all 3 and 5 images from the dataset, the total number of images is 13454. It is divided into a training subset of size 12000 and a testing subset of size 1454. For the dataset, we set the number of workers to be 50, and generate network by a connected Erdos-Renyi graph $ER(50, 0.2)$. We then randomly select 20 workers from these 50 workers to be attackers. Similar to the synthesized data scenario, we illustrate our results with two cases, namely 20 inverse attack, 20 random attack, and compare the performance of two algorithms by comparing the testing accuracy and node disagreement. When testing accuracy, we consider $\overline{x} = \frac{1}{50}\sum_{i=1}^{50} x_i$ to be the output testing model parameter and testing with testing data. The following figures show the result.

Figures 4.6 and 4.7 illustrate the impact of two cases on different algorithms using MNIST respectively. Figures 4.6 and 4.7 show that the DS-ADMM fails to predict if there are 20 attackers. On the other hand, the proposed ZOAR-ADMM algorithm still show high accuracy.

We then plot the impact of 20 attackers case on real data with value of $\|Q^0 x^k\|^2$ to show

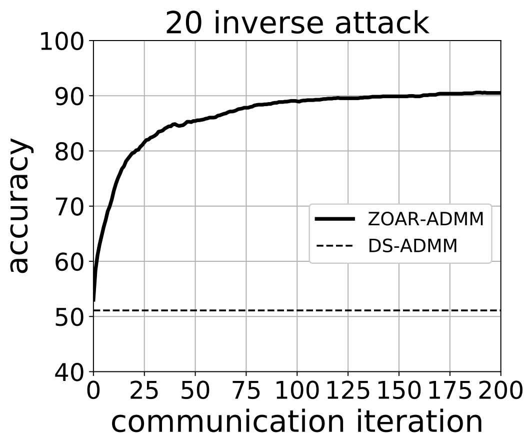

Figure 4.6: Accuracy comparison using MNIST: 20 Inverse attack.

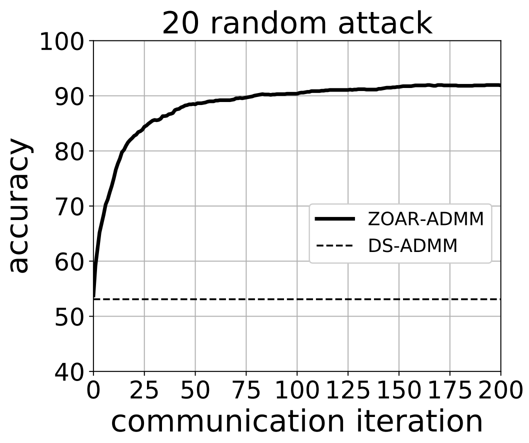

Figure 4.7: Accuracy comparison using MNIST: 20 Random attack.

node disagreement in Figures 4.8 and 4.9 using MNIST respectively. When there are 20 attackers, DS-ADMM has large disagreement, it cannot properly work. Our proposed ZOAR-ADMM has a low disagreement. As the iterations increase, the simulation result shows that our proposed ZOAR-ADMM has better accuracy and lower disagreement.

Figure 4.10 plots the average test error vs iterations when there are 20 inverse attackers. Figure 4.10 shows the number of honest worker that our proposed algorithm may misjudge when there are exist 30 honest workers and 20 attackers. Our algorithm can successfully defend all Byzantine attackers, but it may misjudge 1 or 2 honest workers in 30 honest workers.

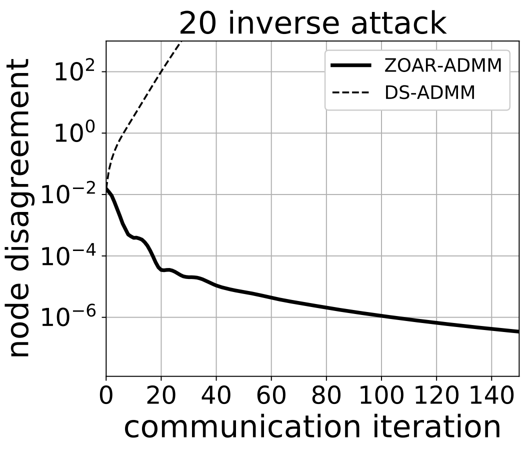

Figure 4.8: Node disagreement comparison using MNIST: 20 Inverse attack.

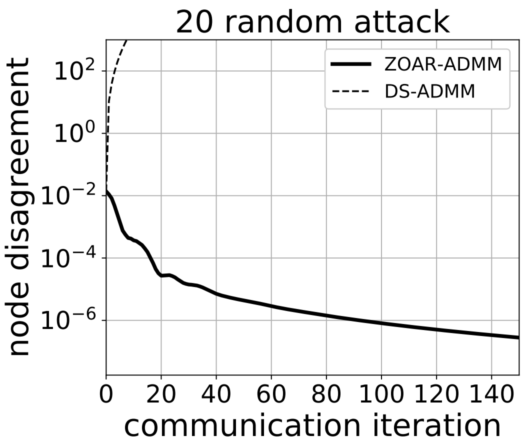

Figure 4.9: Node disagreement comparison using MNIST: 20 Random attack.

4.6 Conclusion

In this chapter, we have proposed a robust zeroth-order ADMM named ZOAR-ADMM algorithm that can tolerant Byzantine attackers in a distributed network. We have analyzed the effect of Byzantine attacks, and have proved that the proposed algorithm can converge to optimal value. We also have provided numerical examples to illustrate the performance of the proposed algorithm.

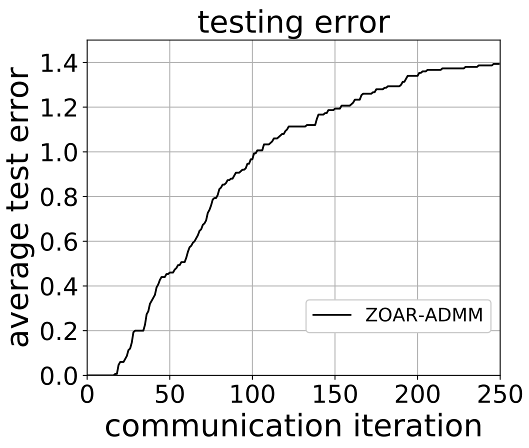

Figure 4.10: Testing error using MNIST: 20 Inverse attack.

Chapter 5

Conclusion

This dissertation discusses three different distributed network scenarios where might be attacked by Byzantine attackers and alogrithms to defend against it.

Firstly, we have proposed robust distributed gradient descent algorthm to defend againts Byzantine attackers in distributed network. We prove that the algorithm can converge to the neighborhood of the population minimizer regardless of the number of compromised workers by proving that the distance between the estimated gradient and the true gradient can be universally bounded.

Secondly, we have analyzed ditributed network which uses second order information. For robust distributed second order algorithm, we have proposed two algorithms to defend against Byzantine attackers. The two algorithms are Median-based approximate Newton's method (MNM) and comparison-based approximate Newton's Method (CNM).For MNM, we have proved that the algorithm can converge to the neighborhood of the population minimizer when q, the number of Byzantine attackers, is less than $m/2$ with m being the total number of workers. For CNM, we have proved that the algorithm can converge to the neighborhood of population minimizer regardless number of Byzantine attackers.

Finally, we have designed and analyzed a new robust zeroth-order information based distributed optimization algorithm that is robust to Byzantine attacks in decentralized distirbuted network. We name the method as zeroth-order adversarially robust alternating direction method of multipliers (ZOAR-ADMM). We have proved that the algorithm can solve the optimization problem and the objective function can converge to the minimum value. We show this result by first investigating how the distance between model parameter and optimal value is affected by the attack vector generated by the attackers, and then carefully analyzing how the algorithm can mitigate these effects and eventually proving that the value of objective function of the proposed algorithm will converge to the optimal value despite the presence of Byzantine attackers.

Appendix A

Appendix of Chapter 2

A.1 Proof of Lemma 2.1

$$
\begin{aligned}
&\|G(\theta) - \nabla F(\theta)\| \\
&= \left\| \sum_{l \in \mathcal{V}_t} w_l q_t^{(l)}(\theta_{t-1}) + w_0 \nabla \overline{f}^{(0)}(\theta_{t-1}) - \nabla F(\theta) \right\| \\
&= \left\| \sum_{l \in \mathcal{V}_t} w_l (q_t^{(l)}(\theta_{t-1}) - \nabla \overline{f}^{(0)}(\theta_{t-1})) + \nabla \overline{f}^{(0)}(\theta) - \nabla F(\theta) \right\| \\
&\leq \sum_{l \in \mathcal{V}_t} w_l \|q_t^{(l)}(\theta) - \nabla \overline{f}^{(0)}(\theta)\| + \|\nabla \overline{f}^{(0)}(\theta) - \nabla F(\theta)\| \\
&\leq \sum_{l \in \mathcal{V}_t} w_l \xi \|\nabla \overline{f}^{(0)}(\theta)\| + \|\nabla \overline{f}^{(0)}(\theta) - \nabla F(\theta)\| \\
&\leq \sum_{l \in \mathcal{V}_t} w_l \xi \|\nabla \overline{f}^{(0)}(\theta) - \nabla F(\theta)\| + \sum_{l \in \mathcal{V}_t} w_l \xi \|\nabla F(\theta)\| + \|\nabla \overline{f}^{(0)}(\theta) - \nabla F(\theta)\| \\
&\leq (1 + \xi) \|\nabla \overline{f}^{(0)}(\theta) - \nabla F(\theta)\| + \xi \|\nabla F(\theta)\|.
\end{aligned} \tag{A.1}
$$

A.2 Proof of Lemma 2.2

Let $V = \{v_1, v_2, ..., v_{N_{1/2}}\}$ denote an $\frac{1}{2}$-cover of unit sphere B, i.e., for fix any $v \in B$, there exists a $v_j \in V$ such that $\| v - v_j \| \leq \frac{1}{2}$. From [46], we have $\log N_{1/2} \leq d \log 6$, and

$$\left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \nabla f(X_i, \theta^*) - \nabla F(\theta^*) \right\| \leq 2 \sup_{v \in V} \left\{ \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \langle \nabla f(X_i, \theta^*) - \nabla F(\theta^*), v \rangle \right\}. \quad \text{(A.2)}$$

By assumption 2 and the condition $\Delta_1 \leq \sigma_1^2/\alpha_1$, it follows from concentration inequalities for sub-exponential random variables [47] that

$$\Pr \left\{ \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \langle \nabla f(X_i, \theta^*) - \nabla F(\theta^*), v \rangle \geq \Delta_1 \right\} \leq \exp(-|\mathcal{S}_0| \Delta_1^2 / (2\sigma_1^2)). \quad \text{(A.3)}$$

By union bound and (A.2), we have

$$\Pr \left\{ \left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \nabla f(X_i, \theta^*) - \nabla F(\theta^*) \right\| \geq 2\Delta_1 \right\} \leq \exp(-|\mathcal{S}_0| \Delta_1^2 / (2\sigma_1^2) + d \log 6). \quad \text{(A.4)}$$

Setting $\Delta_1 = \sqrt{2}\sigma_1 \sqrt{(d \log 6 + \log(3/\delta))(|\mathcal{S}_0|)}$ in (A.4), we obtain the desired result.

A.3 Proof of Lemma 2.3

Define a set V using the same way in Appendix B. We have

$$\frac{\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} h(X_i, \theta) - \mathbb{E}[h(X, \theta)] \|}{\|\theta - \theta^*\|} \leq 2 \sup_{v \in V} \left\{ \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \frac{\langle h(X_i, \theta) - \mathbb{E}[h(X, \theta)], v \rangle}{\| \theta - \theta^* \|} \right\}. \quad \text{(A.5)}$$

By assumption 3 and the condition $\Delta_1' \leq \sigma_2^2/\alpha_2$, it follows from concentration inequalities for sub-exponential random variables [47] that

$$\Pr \left\{ \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \frac{\langle h(X_i, \theta) - E[h(X, \theta)], v \rangle}{\| \theta - \theta^* \|} \geq \Delta_1' \right\} \leq \exp(-|\mathcal{S}_0| (\Delta_1')^2 / (2\sigma_2^2)). \quad \text{(A.6)}$$

By union bound and (A.5),

$$\Pr\left\{\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0} h(X_i,\theta) - \mathbb{E}[h(X,\theta)]\right\| \geq 2\Delta_1' \left\|\theta-\theta^*\right\|\right\} \leq \exp(-|\mathcal{S}_0|(\Delta_1')^2/(2\sigma_2^2) + d\log 6).$$

By setting $\Delta_1' = \sqrt{2}\sigma_2\sqrt{(d\log 6 + \log(3/\delta))(|\mathcal{S}_0|)}$, the proof is complete.

A.4 Proof of Proposition 2.1

Suppose assumption 2, assumption 3 and assumption 4 hold, $\delta_1 \in (0,1)$ and $\Theta \subset \{\theta :\| \theta-\theta^* \|\leq r\sqrt{d}\}$ for some positive parameter r.let

$$\tau = \frac{\alpha_2\sigma_1}{2\sigma_2^2}\sqrt{\frac{d}{|\mathcal{S}_0|}}, u^* = \left\lceil \frac{r\sqrt{d}}{\tau}\right\rceil, \tag{A.7}$$

We define Θ_u for any positive integer $1 \leq u \leq u^*$. $\Theta_u \triangleq \{\theta :\| \theta-\theta^* \|\leq \tau u\}$. Suppose that $\theta_1, ..., \theta_{N_\epsilon}$ is an ϵ-cover of Θ_τ, where ϵ is given by

$$\epsilon = \frac{\sigma_2\tau u}{M \vee M'}\sqrt{\frac{d}{|\mathcal{S}_0|}}. \tag{A.8}$$

Then $\log N_\epsilon \leq d\log(3\tau u/\epsilon)$. Fix any $\theta \in \Theta_u$, there exists a $1 \leq j \leq N_\epsilon$ that $\| \theta-\theta_j \|\leq \epsilon$. By triangle's inequality,

$$\begin{aligned}\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta) - \nabla F(\theta)\right\| &\leq \|\nabla F(\theta) - \nabla F(\theta_j)\| \\ &+ \left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}(\nabla f(X_i,\theta) - \nabla f(X_i,\theta_j))\right\| \\ &+ \left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta_j) - \nabla F(\theta_j)\right\|. \end{aligned} \tag{A.9}$$

By assumption 1,

$$\|\nabla F(\theta) - \nabla F(\theta_j)\| \leq M\|\theta-\theta_j\| \leq M\epsilon. \tag{A.10}$$

Define event

$$\varepsilon_1 = \left\{ \sup_{\theta,\theta' \in \Theta : \theta \neq \theta'} \frac{\|\nabla f(X,\theta) - \nabla f(X,\theta')\|}{\|\theta - \theta'\|} \leq M' \right\}. \tag{A.11}$$

By assumption 4, $Pr\{\varepsilon_1\} \geq 1 - \frac{\delta_1}{3}$, and on event ε_1,

$$\sup_{\theta \in \Theta_\tau} \left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} (\nabla f(X_i,\theta) - \nabla f(X_i,\theta_j)) \right\| \leq M' \parallel \theta - \theta_j \parallel \leq M'\epsilon.$$

By triangle's inequality,

$$\begin{aligned} & \left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \nabla f(X_i,\theta_j) - \nabla F(\theta_j) \right\| \\ \leq & \left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \nabla f(X_i,\theta^*) - \nabla F(\theta^*) \right\| + \left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} h(X_i,\theta_j) - \mathbb{E}[h(X,\theta_j)] \right\|. \end{aligned}$$

Define event

$$\varepsilon_2 = \left\{ \left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} \nabla f(X_i,\theta^*) - \nabla F(\theta^*) \right\| \geq 2\Delta_1 \right\}, \tag{A.12}$$

and event

$$\mathcal{F}_u = \left\{ \left\| \frac{1}{|\mathcal{S}_0|} \sum_{i \in \mathcal{S}_0} h(X_i,\theta_j) - E[h(X,\theta_j)] \right\| \geq 2\tau u \Delta_2 \right\}, \tag{A.13}$$

where

$$\Delta_1 = \sqrt{2}\sigma_1 \sqrt{\frac{d\log 6 + \log(3/\delta_1)}{|\mathcal{S}_0|}}, \tag{A.14}$$

$\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1+\tau_2)(|\mathcal{S}_0|)}$, with

$$\begin{aligned} \tau_1 &= d\log 18 + d\log((M \vee M')/\sigma_2), \\ \tau_2 &= \frac{1}{2} d\log(|\mathcal{S}_0|/d) + \log(3/\delta_1) + \log\left(\frac{2r\sigma_2^2\sqrt{|\mathcal{S}_0|}}{\alpha_2\sigma_1} \right). \end{aligned} \tag{A.15}$$

Since $\Delta_1 \leq \sigma_1^2/\alpha_1$, by Lemma 2.2, Pr$\{\varepsilon_2\} \leq \delta_1/3$. Similarly, by Lemma 2.3, Pr$\{\mathcal{F}_u\} \leq \delta_1/(3u^*)$. In conclusion, it follows that on event $\varepsilon_1 \cap \varepsilon_2^c \cap \mathcal{F}_u^c$,

$$\begin{aligned}&\sup_{\theta\in\Theta_\tau}\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta)-\nabla F(\theta)\right\| \\ &\leq (M+M')\epsilon + 2\Delta_1 + 2\Delta_2\tau \leq 4\Delta_2\tau u + 2\Delta_1,\end{aligned} \tag{A.16}$$

where the last inequality holds due to $(M \vee M')\epsilon \leq \Delta_2\tau u$. Let

$$\varepsilon = \varepsilon_1 \cap \varepsilon_2^c \cap (\cap_{\tau=1}^{u*}\mathcal{F}_u^c). \tag{A.17}$$

It follows from the union bound, $Pr\{\varepsilon\} \geq 1-\delta_1$. On event ε, for all $\theta \in \Theta_{u^*}$, there exist a u such that $(u-1)\tau \leq \|\theta-\theta^*\| \leq u\tau$. For $u \geq 2$, $u \leq 2(u-1)$, then

$$\sup_{\theta\in\Theta_r}\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta)-\nabla F(\theta)\right\| \leq 8\Delta_2\|\theta-\theta^*\| + 2\Delta_1.$$

For $u=1$, since $\Delta_1 \geq \sigma_1\sqrt{d/|\mathcal{S}_0|}$ and $\Delta_2 \leq \sigma_2^2/\alpha_2$, by using τ in (A.7), we get

$$\sup_{\theta\in\Theta_r}\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta)-\nabla F(\theta)\right\| \leq 4\Delta_1.$$

Then on event ε, we have

$$\sup_{\theta\in\Theta_r}\left\|\frac{1}{|\mathcal{S}_0|}\sum_{i\in\mathcal{S}_0}\nabla f(X_i,\theta)-\nabla F(\theta)\right\| \leq 8\Delta_2\|\theta-\theta^*\| + 4\Delta_1.$$

As $\Delta_1 \leq \sigma_1^2/\alpha_1$ and $\Delta_2 \leq \sigma_2^2/\alpha_2$, then

$$\Pr\{\forall\theta : \|\nabla F(\theta) - \nabla\overline{f}^{(0)}(\theta)\| \leq 8\Delta_2\|\theta-\theta^*\| + 4\Delta_1\} \geq 1-\delta_1, \tag{A.18}$$

is proved by the assumption $\Theta \subset \Theta_r$.

A.5 Proof of Theorem 1

Under proposition 2.1, fix any $t \geq 1$,

$$
\begin{aligned}
&\|\theta_t - \theta^*\| \\
&= \left\|\theta_{t-1} - \eta \left[\sum_{l \in \mathcal{V}_t} w_l q_t^{(l)}(\theta_{t-1}) + w_0 \nabla \overline{f}^{(0)}(\theta_{t-1})\right] - \theta^*\right\| \\
&= \left\|\theta_{t-1} - \eta \nabla F(\theta_{t-1}) - \theta^* + \eta(\nabla F(\theta_{t-1}) - \nabla \overline{f}^{(0)}(\theta_{t-1})) + \eta \left[\sum_{l \in \mathcal{V}_t} w_l (\overline{f}^{(0)}(\theta_{t-1}) - q_t^{(l)}(\theta_{t-1}))\right]\right\| \\
&\leq \|\theta_{t-1} - \eta \nabla F(\theta_{t-1}) - \theta^*\| + \eta \|\nabla F(\theta_{t-1}) - \nabla \overline{f}^{(0)}(\theta_{t-1})\| + \eta \sum_{l \in \mathcal{V}_t} w_l \left\|\overline{f}^{(0)}(\theta_{t-1}) - q_t^{(l)}(\theta_{t-1})\right\| \\
&\leq \ \|\theta_{t-1} - \eta \nabla F(\theta_{t-1}) - \theta^*\| + \eta \|\nabla F(\theta_{t-1}) - \nabla \overline{f}^{(0)}(\theta_{t-1})\| + \eta \xi \sum_{l \in \mathcal{V}_t} w_l \|\overline{f}_t^{(0)}(\theta_{t-1})\| \\
&\leq \ \|\theta_{t-1} - \eta \nabla F(\theta_{t-1}) - \theta^*\| + \eta \|\nabla F(\theta_{t-1}) - \nabla \overline{f}_t^{(0)}(\theta_{t-1})\| \\
&\quad + \eta \xi \sum_{l \in \mathcal{V}_t} w_l (\|\overline{f}_t^{(0)}(\theta_{t-1}) - \nabla F(\theta_{t-1})\| + \|\nabla F(\theta_{t-1}) - \nabla F(\theta^*)\|) \\
&\leq \ \left(\sqrt{1 + \eta^2 M^2 - \eta L} + 8\Delta_2 \eta + \eta \xi (8\Delta_2 + M)\right) \|\theta_{t-1} - \theta^*\| + (\eta 4 \Delta_1 + \eta \xi 4 \Delta_1). \qquad \text{(A.19)}
\end{aligned}
$$

Then

$$
\|\theta_t - \theta^*\| \leq (1 - \rho_1)^t \|\theta_0 - \theta^*\| + (4\eta\Delta_1 + 4\eta\xi\Delta_1)/\rho_1, \qquad \text{(A.20)}
$$

where

$$
\rho_1 = 1 - \left(\sqrt{1 + \eta^2 M^2 - \eta L} + 8\Delta_2 \eta + \eta \xi (8\Delta_2 + M)\right).
$$

A.6 Proof of Lemma 2.4

$$\begin{aligned}
&\|G(\theta)-\nabla F(\theta)\| \\
&= \left\| \left(\sum_{j\in\mathcal{H}\cap\mathcal{U}_t} w_j \nabla \overline{f}^{(j)}(\theta) + w_0 \nabla \overline{f}^{(0)}(\theta) + \sum_{j\in\mathcal{A}\cap\mathcal{U}_t} w_j g^{(j)}(\theta) \right) - \nabla F(\theta) \right\| \\
&\le \frac{\sum_{j\in\mathcal{H}\cap\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|}{\sum_{j\in\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|} \|\mathcal{C}_t(\theta) - \nabla F(\theta)\| + \left\| \left(\sum_{j\in\mathcal{A}_t\cap\mathcal{U}_t} w_j g^{(j)}(\theta) - \frac{\sum_{j\in\mathcal{A}_t\cap\mathcal{U}_t} |\mathcal{S}_j|}{\sum_{j\in\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|} \nabla F(\theta) \right) \right\| \\
&\le \frac{\sum_{j\in\mathcal{H}\cap\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|}{\sum_{j\in\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|} \|\mathcal{C}_t(\theta) - \nabla F(\theta)\| + \sum_{j\in\mathcal{A}_t\cap\mathcal{U}_t} w_j \|g^{(j)}(\theta) - \nabla F(\theta)\|.
\end{aligned}$$

A.7 Proof of Lemma 2.5

By triangle inequality,

$$\|\nabla \overline{f}^{(j)}(\theta) - \nabla \overline{f}^{(0)}(\theta)\| \le \|\nabla \overline{f}^{(j)}(\theta) - \nabla F(\theta)\| + \|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\|. \tag{A.21}$$

From Proposition 2.1, we know that $\|\nabla F(\theta) - \nabla \overline{f}^{(0)}(\theta)\|$ can be universally bounded. Using the same arguments, we have that $\|\nabla F(\theta) - \nabla \overline{f}^{(j)}(\theta)\|$ is universally bounded. In particular, under the same assumption as that of Proposition 2.1, for any $\delta_1 \in (0,1)$

$$\Pr\{\forall\theta : \|\nabla F(\theta) - \nabla \overline{f}^{(j)}(\theta)\| \le 8\Delta_4\|\theta-\theta^*\| + 4\Delta_3\} \ge 1-\delta_1, \tag{A.22}$$

in which

$$\Delta_3 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(3/\delta_1))/|\mathcal{S}_j|}, \tag{A.23}$$

and $\Delta_4 = \sqrt{2}\sigma_2\sqrt{(\tau_1+\tau_2)/|\mathcal{S}_j|}$, with $\tau_1 = d\log 18 + d\log((M \vee M')/\sigma_2)$, and $\tau_2 = 0.5d\log(|\mathcal{S}_j|/d) + \log(3/\delta_1) + \log(\frac{2r\sigma_2^2\sqrt{|\mathcal{S}_j|}}{\alpha_2\sigma_1})$.

Combining Proposition 2.1 and equation (A.22), we know that for each good worker,

$$\|\nabla \overline{f}^{(j)}(\theta) - \nabla \overline{f}^{(0)}(\theta)\| \le 8(\Delta_2+\Delta_4)\|\theta-\theta^*\| + 4(\Delta_1+\Delta_3), \forall\theta\in\Theta \tag{A.24}$$

with a probability larger than $(1-\delta_1)^2$.

In the following, we provide a lower bound on $\xi\|\nabla\overline{f}^{(0)}(\theta)\|$. By triangle inequality,

$$\xi\|\nabla\overline{f}^{(0)}(\theta)\| \geq \xi\|\nabla F(\theta)\| - \xi\|\nabla F(\theta) - \nabla\overline{f}^{(0)}(\theta)\|. \tag{A.25}$$

The second term of (A.25) can be bounded using Proposition 2.1. Next we bound the first term of (A.25). Using Assumption 1, we have

$$\begin{aligned} F(\theta^*) &\geq F(\theta) + < \nabla F(\theta), \theta^* - \theta > + \frac{L}{2} \parallel \theta^* - \theta \parallel^2 \\ &\geq F(\theta) - \|\nabla F(\theta)\|\|\theta^* - \theta\| + \frac{L}{2} \parallel \theta^* - \theta \parallel^2 . \end{aligned} \tag{A.26}$$

Since $F(\theta^*) \leq F(\theta)$,

$$-\|\nabla F(\theta)\|\|\theta^* - \theta\| + \frac{L}{2} \parallel \theta^* - \theta \parallel^2 \leq 0, \tag{A.27}$$

hence,

$$\|\nabla F(\theta)\| \geq \frac{L}{2}\|\theta - \theta^*\|. \tag{A.28}$$

Plugging (A.28) and Proposition 2.1 to (A.25), we have $\forall\theta \in \Theta$

$$\xi\|\nabla\overline{f}^{(0)}(\theta)\| \geq \frac{L\xi}{2}\|\theta - \theta^*\| - 8\xi\Delta_2\|\theta - \theta^*\| - 4\xi\Delta_1. \tag{A.29}$$

with probability larger than $1 - \delta_1$. Then we need to choose value of ξ to guarantee that the right-hand side of (A.29) will be larger than the right-hand side of (A.24).

$$8(\Delta_2 + \Delta_4)\|\theta - \theta^*\| + 4(\Delta_1 + \Delta_3) \leq 16\Delta_2\|\theta - \theta^*\| + 8\Delta_1. \tag{A.30}$$

Since $\xi \leq 1$,

$$\begin{aligned}
&(16+8\xi)\Delta_2\|\theta-\theta^*\| + (8+4\xi)\Delta_1 \\
&\leq 24\Delta_2\|\theta-\theta^*\| + 12\Delta_1 \leq \frac{L\xi}{2}\|\theta-\theta^*\|. \qquad \text{(A.31)}
\end{aligned}$$

Since $|\mathcal{S}_0|^{-1/4}$ converges more slowly than $\sqrt{\frac{\log(|\mathcal{S}_0|)}{|\mathcal{S}_0|}}$, we set $\xi = c|\mathcal{S}_0|^{-1/4}$, then we can choose $c = (48\Delta_2\|\theta-\theta^*\| + 24\Delta_1)|\mathcal{S}_0|^{1/4}/(L\|\theta-\theta^*\|)$, when $\|\theta-\theta^*\| \neq 0$. As the result,

$$\|\nabla\overline{f}^{(j)}(\theta) - \nabla\overline{f}^{(0)}(\theta)\| \leq \xi\|\nabla\overline{f}^{(0)}(\theta)\|, \forall\theta \in \Theta \qquad \text{(A.32)}$$

holds with probability $(1-\delta_1)^2 - \delta_1$.

A.8 Proof of Theorem 2

From Assumption 1, Proposition 2.1, Proposition 2.2 and Lemma 2.4 , fix any $t \geq 1$, the norm of difference between $G_t(\theta)$ and $\nabla F(\theta)$ is

$$\begin{aligned}
&\|G(\theta) - \nabla F(\theta)\| \\
&\leq \frac{\sum_{j\in\mathcal{H}\cap\mathcal{U}_t}|\mathcal{S}_j| + |\mathcal{S}_0|}{\sum_{j\in\mathcal{U}_t}|\mathcal{S}_j| + |\mathcal{S}_0|}\|C_t(\theta) - \nabla F(\theta)\| + \sum_{j\in\mathcal{A}_t\cap\mathcal{U}_t} w_j\|\nabla g^{(j)}(\theta) - \nabla F(\theta)\| \\
&\leq \gamma_2\|\theta-\theta^*\| + \gamma_1, \qquad \text{(A.33)}
\end{aligned}$$

where

$$\gamma_1 = 4(1-w_{max})\Delta_5 + 4w_{max}\Delta_7, \qquad \text{(A.34)}$$

and

$$\gamma_2 = 8(1-w_{max})\Delta_6 + 8w_{max}\Delta_8. \qquad \text{(A.35)}$$

with $w_{max} = \max\{(\sum_{j\in\mathcal{B}\cap\mathcal{U}_t} |\mathcal{S}_j|)/(\sum_{j\in\mathcal{U}_t} |\mathcal{S}_j| + |\mathcal{S}_0|)\}$ and $|\mathcal{B}\cap\mathcal{U}_t| = \min\{m-p, p\}$ and $|\mathcal{U}_t| = m - p$. Fix any $t \geq 1$,

$$\begin{aligned} \|\theta_t - \theta^*\| &= \|\theta_{t-1} - \eta G(\theta_{t-1}) - \theta^*\| \\ &\leq \|\theta_{t-1} - \eta\nabla F(\theta_{t-1}) - \theta^*\| + \eta\|G(\theta_{t-1}) - \nabla F(\theta_{t-1})\| \\ &\leq \left(\sqrt{1+\eta^2M^2 - \eta L} + \eta\gamma_2\right)\|\theta_{t-1} - \theta^*\| + \eta\gamma_1. \end{aligned} \tag{A.36}$$

Then,

$$\|\theta_t - \theta^*\| \leq (1-\rho_2)^t\|\theta_0 - \theta^*\| + (\eta\gamma_1)/\rho_2, \tag{A.37}$$

where $\rho_2 = 1 - \left(\sqrt{1+\eta^2M^2 - \eta L} + \eta\gamma_2\right)$.

Appendix B

Appendix of Chapter 3

B.1 Proof of Lemma 3.2

When Assumption 5 holds, from union bound theorem, for any $\delta \in (0,1)$, with probability at least $1-\frac{\delta}{3}$, $|\mathcal{S}_j|$ data satisfy

$$h' \leq \|\nabla^2 f(X,\theta)\|_2 \leq M', \tag{B.1}$$

When $\|\nabla^2 f(X,\theta)\|_2 \leq M'$, we have

$$H_i - H = \sum_{j\in\mathcal{S}_i} \frac{1}{|\mathcal{S}_i|}(\nabla^2 f(X_j,\theta) - H), \tag{B.2}$$

and

$$\begin{aligned} \left\| \frac{1}{|\mathcal{S}_i|}(\nabla^2 f(X_j,\theta) - H) \right\|_2 &\leq \frac{1}{|\mathcal{S}_i|}(\|\nabla^2 f(X_j,\theta)\|_2 + \|H\|_2) \\ &\leq \frac{2(M \vee M')}{|\mathcal{S}_i|}. \end{aligned} \tag{B.3}$$

Before proceed further, we define matrix variance statistic $v(Y)$ of a random Hermitian matrix with zero mean Y as

$$v(Y) = \|Var(Y)\|_2 = \|\mathbf{E}[(Y - E[Y])^2]\|_2 = \|\mathbf{E}[Y^2]\|_2.$$

Using this definition, we have

$$\begin{aligned} v(H_i - H) &= \left\| \sum_{j \in \mathcal{S}_i} \mathbf{E}\left[\frac{1}{|\mathcal{S}_i|} (\nabla^2 f(X_j, \theta) - H)^2 \right] \right\|_2 \\ &= \left\| \sum_{j \in \mathcal{S}_i} \frac{1}{|\mathcal{S}_i|^2} \mathbf{E}\left[(\nabla^2 f(X_j, \theta) - H)^2 \right] \right\|_2 \\ &\leq \left\| \sum_{j \in \mathcal{S}_i} \frac{1}{|\mathcal{S}_i|^2} \mathbf{E}[\nabla^2 f(X_j, \theta)^2] \right\|_2 \\ &\leq \frac{1}{|\mathcal{S}_i|} \mathbf{E} \left\| \nabla^2 f(X_j, \theta)^2 \right\|_2 \\ &= \frac{1}{|\mathcal{S}_i|} \mathbf{E} \left\| \nabla^2 f(X_j, \theta) \right\|_2^2 \\ &\leq \frac{(M \vee M')^2}{|\mathcal{S}_i|}. \end{aligned} \tag{B.4}$$

Since $H = \mathbf{E}[H_i]$, for $0 \leq \gamma \leq 2(M \vee M')$, we can use Matrix Bernstein inequality from [45] to get

$$\begin{aligned} &\mathbf{Pr}\left\{ \|H_i - H\|_2 \geq \gamma \right\} \\ \leq\ & 2d \exp\left(\frac{-\gamma^2/2}{v(H_i - H) + 2(M \vee M')\gamma/3|\mathcal{S}_i|} \right) \\ \leq\ & 2d \exp\left(\frac{-\gamma^2/2}{(M \vee M')^2/|\mathcal{S}_i| + 2(M \vee M')\gamma/3|\mathcal{S}_i|} \right) \\ \leq\ & 2d \exp\left(\frac{-3\gamma^2|\mathcal{S}_i|}{14(M \vee M')^2} \right). \end{aligned} \tag{B.5}$$

By picking $\Delta_3 = \gamma = \sqrt{\frac{14(M \vee M')^2 \log(2d/\delta_3')}{3|\mathcal{S}_i|}}$, we achieve

$$\mathbf{Pr}\left\{ \|H_i - H\|_2 \geq \Delta_3 \right\} \leq \delta_3'. \tag{B.6}$$

when $\|\nabla^2 f(X,\theta)\|_2 \leq M'$.

From union bound theorem, suppose Assumption 4 holds, let $\delta_2 = \frac{\delta}{3} + \delta_3'$ and $\delta_2 \in (0,1)$, we have

$$\mathbf{Pr}\left\{\|H_i - H\|_2 \leq \Delta_3\right\} \geq 1 - \delta_2. \tag{B.7}$$

B.2 Proof of Proposition 3.1

Suppose Assumptions 1-3 hold, and $\Theta \subset \{\theta : \| \theta - \theta^* \| \leq r\sqrt{d}\}$ for some $r > 0$. From (3.15), for an honest worker i, we have

$$\begin{aligned} Z_i(\theta) &= (H^{-1} - H_i^{-1})g(\theta) \\ &= H^{-1}(H_i - H)H_i^{-1}g(\theta) \\ &= H^{-1}(H_i - H)H_i^{-1}(J(\theta) + \nabla F(\theta)). \end{aligned} \tag{B.8}$$

Using the properties of the spectral norm, we have

$$\|Z_i(\theta)\| \leq \|H^{-1}\|_2\|H_i - H\|_2\|H_i^{-1}\|_2(\|J(\theta)\| + \|\nabla F(\theta)\|).$$

Following similar steps in [9, 13], we can show that, for any $\alpha \in (q/m, 1/2)$ and $0 < \delta_3 < \alpha - q/m$, we have

$$\mathbf{Pr}\{\|J(\theta)\| \leq 8\mathcal{C}_\alpha\Delta_2\|\theta - \theta^*\| + 4\mathcal{C}_\alpha\Delta_1\} \geq 1 - e^{-mD(\alpha - q/m\|\delta_3)} \tag{B.9}$$

where $\Delta_1 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(6/\delta_3))/|\mathcal{S}_i|}$, $\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1+\tau_2)/|\mathcal{S}_i|}$, with $\tau_1 = d\log 18 + d\log(M/\sigma_2)$, $\tau_2 = 0.5d\log(|\mathcal{S}_i|/d) + \log(6/\delta_3) + \log(\frac{2r\sigma_2^2\sqrt{|\mathcal{S}_i|}}{\alpha_2\sigma_1})$, $\mathcal{C}_\alpha = \frac{2(1-\alpha)}{1-2\alpha}$ and $D(\delta'\|\delta) = \delta'\log\frac{\delta'}{\delta} + (1-\delta')\log\frac{1-\delta'}{1-\delta}$.

Combining it with Assumption 1, Assumption 4, Lemma 3.2, we have the following bound

$$\mathbf{Pr}\left\{\forall\theta : \|Z_i(\theta)\| \leq \left(\frac{8\mathcal{C}_\alpha\Delta_3\Delta_2}{hh'} + \frac{\Delta_3 M}{hh'}\right)\|\theta-\theta^*\| + \frac{4\mathcal{C}_\alpha\Delta_3\Delta_1}{hh'}\right\} \geq 1-\delta_4, \quad \text{(B.10)}$$

with $1-\delta_4 = 1-\delta_2 - e^{-mD(\alpha-q/m\|\delta_3)}$.

B.3 Proof of Proposition 3.2

Suppose Assumptions 1-3 hold, and $\Theta \subset \{\theta :\| \theta-\theta^* \|\leq r\sqrt{d}\}$ for some $r > 0$. From Proposition 3.1, we have the bound $\|Z_i(\theta)\|$ for honest worker i.

From Lemma 3.1, in order to bound the geometric median $Z(\theta)$ of $Z_1(\theta), ..., Z_m(\theta)$, we need to have more than half of the workers to be honest.

Then we can define a good event $\mathcal{E}_{2,\alpha,\xi_1,\xi_2} = \left\{\sum_{i=1}^m \mathbf{1}_{\{\mathcal{C}_\alpha\|Z_i(\theta)\|_2 \leq \xi_3\|\theta-\theta^*\|+\xi_4\}} \geq m(1-\alpha)+q\right\}$, where

$$\xi_3 = \left(\frac{8\mathcal{C}_\alpha\Delta_3\Delta_2}{hh'} + \frac{\Delta_3 M}{hh'}\right)\mathcal{C}_\alpha,$$

and

$$\xi_4 = \frac{4\mathcal{C}_\alpha^2\Delta_3\Delta_1}{hh'}.$$

From proposition 3.1, for all $1 \leq i \leq m$, correct Z_i satisfied

$$\mathbf{Pr}\left\{\mathcal{C}_\alpha\|Z_i(\theta)\| \leq \xi_3\|\theta-\theta^*\| + \xi_4\right\} \geq 1-\delta_4, \quad \text{(B.11)}$$

for any $\alpha \in (q/m, 1/2)$ and $0 < \delta_4 < \alpha - q/m$. Then following similar steps as in [13], we have

$$\mathbf{Pr}\{\mathcal{E}_{2,\alpha,\xi_1,\xi_2}\} \geq 1 - e^{-mD(\alpha-q/m\|\delta_4)}. \quad \text{(B.12)}$$

Then using Lemma 3.1, we obtain an bound for norm of geometric median $Z(\theta)$,

$$\begin{aligned}
&\mathbf{Pr}\left\{\forall\theta : \|Z(\theta)\| \le \left(\frac{8\mathcal{C}_\alpha\Delta_3\Delta_2}{hh'} + \frac{\Delta_3 M}{hh'}\right)\mathcal{C}_\alpha\|\theta-\theta^*\| + \frac{4\mathcal{C}_\alpha^2\Delta_3\Delta_1}{hh'}\right\} \\
\ge\ & 1 - e^{-mD(\alpha - q/m\|\delta_4)}. \qquad \text{(B.13)}
\end{aligned}$$

B.4 Proof of Theorem 3

Suppose Assumptions 1-3,5,6 hold, and $\Theta \subset \{\theta :\| \theta - \theta^* \|\le r\sqrt{d}\}$ for some $r > 0$. Following similar steps in [9,13], we have the following bound for any $\alpha \in (q/m, 1/2)$ and $0 < \delta_3 < \alpha - q/m$,

$$\mathbf{Pr}\{\|J(\theta)\| \le 8\mathcal{C}_\alpha\Delta_2\|\theta-\theta^*\| + 4\mathcal{C}_\alpha\Delta_1\} \ge 1 - e^{-mD(\alpha-q/m\|\delta_3)} \qquad \text{(B.14)}$$

From (3.14), combined with Proposition 3.2, we have

$$\begin{aligned}
&\|H^{-1}\nabla F(\theta) - G(\theta)\| \\
\le\ & \|Z(\theta)\| + \|H^{-1}J(\theta)\| \\
\le\ & \left(\frac{8}{hh'}\mathcal{C}_\alpha^2\Delta_3\Delta_2 + \frac{8}{h}\mathcal{C}_\alpha\Delta_2 + \mathcal{C}_\alpha\frac{\Delta_3 M}{hh'}\right)\|\theta-\theta^*\| + \frac{4}{hh'}\mathcal{C}_\alpha^2\Delta_3\Delta_1 + \frac{4}{h}\mathcal{C}_\alpha\Delta_1. \qquad \text{(B.15)}
\end{aligned}$$

Then for any $0 < \eta \leq 1$,$\alpha \in (q/m, 1/2)$, $0 < \delta_3 < \alpha - q/m$ and $0 < \delta_4 < \alpha - q/m$ with probability at least $1 - e^{-mD(\alpha - q/m\|\delta_3)} - e^{-mD(\alpha-q/m\|\delta_4)}$, for any $t \geq 1$,

$$\begin{aligned}
&\|\theta_t - \theta^*\| \\
=&\|\theta_{t-1} - \eta G(\theta_{t-1}) - \theta^*\| \\
=&\|\theta_{t-1} - \eta H_{t-1}^{-1}\nabla F(\theta_{t-1}) - \theta^* + \eta H_{t-1}^{-1}\nabla F(\theta_{t-1}) - \eta G(\theta_{t-1})\| \\
=&\|\theta_{t-1} - \eta H_{t-1}^{-1}\nabla F(\theta_{t-1}) - \theta^* + \eta Z(\theta_{t-1}) - \eta H_{t-1}^{-1} g(\theta_{t-1}) + \eta H_{t-1}^{-1}\nabla F(\theta_{t-1})\| \\
\leq&\|\theta_{t-1} - \eta H_{t-1}^{-1}\nabla F(\theta_{t-1}) - \theta^*\| + \eta\|Z(\theta_{t-1})\| + \|\eta H_{t-1}^{-1} J(\theta_{t-1})\| \\
\leq&\left(1 - \eta + \eta\frac{8}{hh'}\mathcal{C}_\alpha^2\Delta_3\Delta_2 + \eta\frac{8}{h}\mathcal{C}_\alpha\Delta_2 + \eta\mathcal{C}_\alpha\frac{\Delta_3 M}{hh'}\right)\|\theta_{t-1} - \theta^*\| \\
&\quad + \frac{\eta L\|\theta_{t-1} - \theta^*\|^2}{2h} + \eta\frac{4}{hh'}\mathcal{C}_\alpha^2\Delta_3\Delta_1 + \eta\frac{4}{h}\mathcal{C}_\alpha\Delta_1. \qquad \text{(B.16)}
\end{aligned}$$

B.5 Proof of Lemma 3.3

$$\begin{aligned}
&\|H^{-1}\nabla F(\theta) - G(\theta)\| \\
=&\left\| H^{-1}\nabla F(\theta) - \frac{1}{1+|\mathcal{A}^{(2)}|}(\sum_{i\in\mathcal{A}^{(2)}} g_2^{(i)}(\theta) + H_0^{-1}g(\theta)) \right\| \\
\leq&\frac{1}{1+|\mathcal{A}^{(2)}|}\left\|(\sum_{i\in\mathcal{A}^{(2)}}(g_2^{(i)}(\theta) - H_0^{-1}g(\theta)))\right\| + \|H^{-1}\nabla F(\theta) - H_0^{-1}g(\theta)\| \\
\leq&\xi_2\frac{|\mathcal{A}^{(2)}|}{1+|\mathcal{A}^{(2)}|}\|H_0^{-1}g(\theta)\| + \|H_0^{-1}g(\theta) - H_0^{-1}\nabla F(\theta)\| + \|H^{-1}\nabla F(\theta) - H_0^{-1}\nabla F(\theta)\| \\
<&(1+\xi_2)\|H_0^{-1}\|_2\|g(\theta) - \nabla F(\theta)\| + \xi_2\|H_0^{-1}\|_2\|\nabla F(\theta)\| + \|H^{-1}\nabla F(\theta) - H_0^{-1}\nabla F(\theta)\|.
\end{aligned}$$

B.6 Proof of Lemma 3.4

If $\|H_0 - H\|_2 \leq \beta$ and $\beta < \frac{h(h+\mu)}{3h+2\mu}$, we have

$$\begin{aligned} & \|\mathbf{I} - \tilde{H}_0^{-1} H\|_2 \\ = & \|\mathbf{I} - (H + \mu\mathbf{I})^{-1}H + (H + \mu\mathbf{I})^{-1}H - \tilde{H}_0^{-1}H\|_2 \\ \leq & \frac{\mu}{h+\mu} + \|(\tilde{H}_0^{-1} - (H+\mu)^{-1})H\|_2. \end{aligned} \tag{B.17}$$

Consider $\tilde{H}_0^{-1}$, let $A = H + \mu\mathbf{I}$ and $\Delta_0 = H_0 - H$,noting that $\|A^{-1}\Delta_0\|_2 \leq \|A^{-1}\|_2 \|\Delta_0\|_2 \leq \frac{1}{h+\mu}\frac{h(h+\mu)}{3h+2\mu} < 1$, we have

$$\begin{aligned} \tilde{H}_0^{-1} &= (H + \mu\mathbf{I} + H_0 - H)^{-1} \\ &= (A + \Delta_0)^{-1} \\ &= (A(\mathbf{I} + A^{-1}\Delta_0))^{-1} \\ &= (\mathbf{I} + A^{-1}\Delta_0)^{-1}A^{-1} \\ &= A^{-1} + \sum_{r=1}^{\infty}(-1)^r (A^{-1}\Delta_0)^r A^{-1}. \end{aligned} \tag{B.18}$$

Then, we can have

$$\begin{aligned}
& \|(\tilde{H}_0^{-1} - (H+\mu\mathbf{I})^{-1})H\|_2 \\
= \;& \|\sum_{r=1}^{\infty}(-1)^r(A^{-1}\Delta_0)^r A^{-1}(A-\mu\mathbf{I})\|_2 \\
= \;& \|\sum_{r=1}^{\infty}(-1)^r(A^{-1}\Delta_0)^r(\mathbf{I}-\mu A^{-1})\|_2 \\
\leq \;& \sum_{r=1}^{\infty}\|A^{-1}\|_2^r\|\Delta_0\|_2^r\|\mathbf{I}-\mu A^{-1}\|_2 \\
\leq \;& \sum_{r=1}^{\infty}\frac{\beta^r}{(h+\mu)^r}(1+\frac{\mu}{h+\mu}) \\
\leq \;& \frac{2\beta}{h+\mu}\sum_{r=0}^{\infty}\frac{\beta^r}{(h+\mu)^r} \\
= \;& \frac{2\beta}{h+\mu-\beta}.
\end{aligned} \tag{B.19}$$

Then, we have

$$\|\mathbf{I} - \tilde{H}_0^{-1}H\|_2 \leq \frac{\mu}{h+\mu} + \frac{2\beta}{h+\mu-\beta} < 1, \tag{B.20}$$

when $\beta < \frac{h(h+\mu)}{3h+2\mu}$.

B.7 Proof of Proposition 3.3

From Lemma 3.4, we have the bound for $\|\mathbf{I} - \tilde{H}_0^{-1}H\|_2$, if $\|H_0 - H\|_2 \leq \beta$ and $\beta < \frac{h(h+\mu)}{3h+2\mu}$. From Lemma 3.2, we have showned when Assumption 4 holds, that for any $\delta \in (0,1), \delta_2 \in (0,1), \delta_3' \in (0,1)$ and $\delta_2 = \delta_3' + \delta/3$ with probability at least $1-\delta_2$, $\|H_0 - H\|_2 \leq \sqrt{\frac{14M^2\log(2d/\delta_3')}{3|\mathcal{S}_0|}}$. Then if $\|\mathcal{S}_0\|$ is sufficiently large, we have $\Delta_3 = \sqrt{\frac{14M^2\log(2d/\delta_3')}{3|\mathcal{S}_0|}} < \frac{h(h+\mu)}{3h+2\mu}$, and we have the following bound for any $\delta_2 \in (0,1)$ with probability at least $1-\delta_2$

$$\begin{aligned}
& \|(H^{-1} - \tilde{H}_0^{-1})\nabla F(\theta)\| \\
= \ & \|(\mathbf{I} - \tilde{H}_0^{-1}H)H^{-1}\nabla F(\theta)\| \\
\leq \ & \|\mathbf{I} - \tilde{H}_0^{-1}H\|_2 \|H^{-1}\|_2 \|\nabla F(\theta) - \nabla F(\theta^*)\| \\
\leq \ & \left(\frac{\mu}{h+\mu} + \frac{2\Delta_3}{h+\mu-\Delta_3}\right)\frac{M}{h}\|\theta - \theta^*\|. \qquad \text{(B.21)}
\end{aligned}$$

B.8 Proof of Theorem 4

If Assumptions 1-3, 5, 6 hold, and $\Theta \subset \{\theta : \| \theta - \theta^* \| \leq r\sqrt{d}\}$ for some $r > 0$ and $\Delta_3 < \frac{h(h+\mu)}{3h+2\mu}$ for the first term in Lemma 3.3, we already have the following bound from [9], regardless of the number of attackers, that with probability at least $1-\delta_5$

$$\|g(\theta) - \nabla F(\theta)\| \leq (8\Delta_2 + \xi_1(8\Delta_2 + M))\|\theta_{t-1} - \theta^*\| + (4\Delta_1 + \xi_1 4\Delta_1),$$

in which $\Delta_1 = \sqrt{2}\sigma_1\sqrt{(d\log 6 + \log(3/\delta_5))/|\mathcal{S}_0|}$ and $\Delta_2 = \sqrt{2}\sigma_2\sqrt{(\tau_1+\tau_2)/|\mathcal{S}_0|}$, with $\tau_1 = d\log 18 + d\log((M \vee M')/\sigma_2)$, and $\tau_2 = 0.5d\log(n/d) + \log(3/\delta_5) + \log(\frac{2r\sigma_2^2\sqrt{|\mathcal{S}_0|}}{\alpha_2\sigma_1})$.

Combine it with Assumption 1 and Proposition 3.3 and Lemma 3.3, fix any $t \geq 1$, for any $\delta_2 \in (0,1)$, and $\delta_5 \in (0,1)$, with probability at least $1-\delta_2-\delta_5$, the norm of difference between $G_t(\theta)$ and $\nabla F(\theta)$ is

$$\begin{aligned}
& \|H^{-1}\nabla F(\theta) - G(\theta)\| \\
< \ & (1+\xi_2)\|\tilde{H}_0^{-1}\|_2\|g(\theta) - \nabla F(\theta)\| + \xi_2\|\tilde{H}_0^{-1}\|_2\|\nabla F(\theta)\| + \|H^{-1}\nabla F(\theta) - \tilde{H}_0^{-1}\nabla F(\theta)\| \\
\leq \ & \gamma_1\|\theta - \theta^*\| + \gamma_2, \qquad \text{(B.22)}
\end{aligned}$$

where $\Delta_4 = \frac{\mu}{h+\mu} + \frac{2\Delta_3}{h+\mu-\Delta_3}$,

$$\gamma_1 = \left[(8\Delta_2 + \xi_1(8\Delta_2 + M)) \frac{1+\xi_2}{h'+\mu} + \frac{\xi_2 M}{h'+\mu} + \frac{\Delta_4 M}{h} \right], \tag{B.23}$$

and

$$\gamma_2 = (4\Delta_1 + \xi_1 4\Delta_1) \frac{1+\xi_2}{h'+\mu}. \tag{B.24}$$

Fix any $t \geq 1$,

$$\begin{aligned}
&\|\theta_t - \theta^*\| \\
&= \|\theta_{t-1} - G(\theta_{t-1}) - \theta^*\| \\
&\leq \|\theta_{t-1} - H^{-1}\nabla F(\theta_{t-1}) - \theta^*\| + \|G(\theta_{t-1}) - H^{-1}\nabla F(\theta_{t-1})\| \\
&\leq \frac{L}{2h}\|\theta_{t-1} - \theta^*\|^2 + \gamma_1\|\theta_{t-1} - \theta^*\| + \eta\gamma_2.
\end{aligned} \tag{B.25}$$

Appendix C

Appendix of Chapter 4

C.1 Proof of Lemma 4.2

To bound the zeroth-order estimate, we first need to bound the distance between the empirical gradient and the population gradient.

From [13], under Assumption 2, for any $\delta \in (0, 1)$,with high probability we have the following bound for the optimal parameter θ^*.

$$\Pr\left\{\left\|\nabla \overline{f}^{(i)}(\theta^*) - \nabla F(\theta^*)\right\| \geq 2\Delta_1\right\} \leq \frac{\delta}{3}.$$

Then for any θ, when Assumptions 2-3, 7-9 hold, for any $\delta \in (0, 1)$, from [13], we have

$$\Pr\{\forall \theta : \|\nabla F(\theta) - \nabla \overline{f}^{(i)}(\theta)\| \leq 8\Delta_2 \|\theta - \theta^*\| + 4\Delta_1\} \geq 1 - \delta.$$

From this inequality, we know that by increasing the number of data samples in each worker, Δ_1 and Δ_2 will decrease to zero, then we know the gradient of emprical risk is a good apprixmation of gradient of the population risk.

Then for the zeroth-order gradient estimation, we have

$$\begin{aligned}\|g_i(\theta_i^k)\| &\leq \|g_i(\theta_i^k) - \nabla \overline{f}^{(i)}(\theta_i^k)\| + \|\nabla \overline{f}^{(i)}(\theta_i^k) - \nabla \overline{f}^{(i)}(\theta^*)\| + \|\nabla \overline{f}^{(i)}(\theta^*) - \nabla F(\theta^*)\| \\ &\leq \frac{M_f^2 d^2 u_k^2}{m} + M_f\|\theta_i - \theta^*\| + \Delta_1. \qquad \text{(C.1)}\end{aligned}$$

C.2 Proof of Lemma 4.3

Using the second step of the algorithm, we have

$$\alpha^{k+1} = \alpha^k + cL_-(\theta^{k+1} + e^{k+1}). \qquad \text{(C.2)}$$

Then sum and telescope from iteration 0 to k, and assume $\alpha^0 = 0$, we have

$$\alpha^k = c\sum_{s=0}^{k} L_-^s(\theta^s + e^s). \qquad \text{(C.3)}$$

Then consider the first step of the algorthm, we have

$$g(\theta^{k+1}) = -2cW^{k+1}\theta^{k+1} + cL_+^{k+1}z^k - c\sum_{s=0}^{k} L_-^s(\theta^s + e^s). \qquad \text{(C.4)}$$

Then we have

$$g(\theta^{k+1}) + c\sum_{s=0}^{k} L_-^s(\theta^s + e^s) = -2cW^{k+1}\theta^{k+1} + cL_+^{k+1}z^k. \qquad \text{(C.5)}$$

By adding $cL_-^{k+1}(\theta^{k+1} + e^{k+1})$ on both size and rearrange the equation, we obtain

$$g(\theta^{k+1}) = 2cW^{k+1}e^{k+1} - cL_+^{k+1}(z^{k+1} - z^k) - 2cQr^{k+1}. \qquad \text{(C.6)}$$

C.3 Proof of Theorem 5

We have

$$
\begin{aligned}
& m_F\|\theta^{k+1}-\theta^*\|^2 \le \langle \theta^{k+1}-\theta^*, \sum_{i=1}^{n} \nabla F^i(\theta_i^{k+1})\rangle \\
= \quad & \langle \theta^{k+1}-\theta^*, g(\theta^{k+1})\rangle + \langle \theta^{k+1}-\theta^*, h(\theta^{k+1})\rangle \\
+ \quad & \langle \theta^{k+1}-\theta^*, \sum_{i=1}^{n} \nabla F^i(\theta_i^{k+1}) - \nabla f(\theta^{k+1})\rangle. \qquad \text{(C.7)}
\end{aligned}
$$

For the first part, we have

$$\begin{aligned}
& \langle\theta^{k+1}-\theta^*, g(\theta^{k+1})\rangle \\
\leq\ & -c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(z^{k+1}-z^k)\rangle + c\langle\theta^{k+1}-\theta^*, L_+^{k+1}e^{k+1}\rangle \\
& +c\langle\theta^{k+1}-\theta^*, L_-^{k+1}e^{k+1}\rangle - c\langle\theta^{k+1}-\theta^*, 2Q(r^{k+1})\rangle \\
=\ & c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(\theta^k-\theta^{k+1})\rangle + c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(z^k-\theta^k)\rangle \\
& +c\langle\theta^{k+1}-\theta^*, L_-^{k+1}(z^{k+1}-\theta^{k+1})\rangle + c\langle\theta^{k+1}-\theta^*, -2Qr^{k+1}\rangle \\
=\ & c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(\theta^k-\theta^{k+1})\rangle + c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(z^k-\theta^k)\rangle \\
& +c\langle z^{k+1}-\theta^*, 2Q(0-r^{k+1})\rangle + c\langle e^{k+1}, 2Qr^{k+1}\rangle \\
& +c\langle\theta^{k+1}-\theta^*, L_-^{k+1}(z^{k+1}-\theta^{k+1})\rangle \\
=\ & c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(\theta^k-\theta^{k+1})\rangle + c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(z^k-\theta^k)\rangle \\
& +c\langle r^{k+1}-r^k, 2(0-r^{k+1})\rangle + c\langle e^{k+1}, 2Qr^{k+1}\rangle \\
& +c\langle\theta^{k+1}-\theta^*, L_-^{k+1}(z^{k+1}-\theta^{k+1})\rangle \\
=\ & \|p^k-p\|^2_{G^{k+1}} - \|p^{k+1}-p\|^2_{G^{k+1}} - \|p^{k+1}-p^k\|^2_{G^{k+1}} \\
& +c\langle\theta^{k+1}-\theta^*, L_+^{k+1}(z^k-\theta^k)\rangle + c\langle e^{k+1}, 2Qr^{k+1}\rangle \\
& +c\langle\theta^{k+1}-\theta^*, L_-^{k+1}(z^{k+1}-\theta^{k+1})\rangle \\
\leq\ & \|p^k-p\|^2_{G^{k+1}} - \|p^{k+1}-p\|^2_{G^{k+1}} - \|p^{k+1}-p^k\|^2_{G^{k+1}} - c\|Q^{k+1}\theta^{k+1}\|^2 \\
& -c\|Q^{k+1}e^{k+1}\|^2 + 2c\langle\theta^{k+1}-\theta^*, \frac{L_+}{2}(z^k-\theta^k)\rangle + c\langle e^{k+1}, 2Qr^{k+1}\rangle \\
\leq\ & \|p^k-p\|^2_{G^{k+1}} - \|p^{k+1}-p\|^2_{G^{k+1}} - \|p^{k+1}-p^k\|^2_{G^{k+1}} \\
& -c\frac{\sigma_{min}(L_-^{k+1})}{2}\|\theta^{k+1}-\theta^*\|^2 - c\|Q^{k+1}e^{k+1}\|^2 + c\beta\|z^k-\theta^k\|^2 \\
& +\frac{c}{\beta}\|\frac{L_+^{k+1}}{2}(\theta^{k+1}-\theta^*)\|^2 + c\langle e^{k+1}, 2Qr^{k+1}\rangle \\
=\ & \|p^k-p\|^2_{G^{k+1}} - \|p^{k+1}-p\|^2_{G^{k+1}} - \|p^{k+1}-p^k\|^2_{G^{k+1}} \\
& +c\frac{\sigma^2_{max}(L_+^{k+1})}{2\sigma_{min}(L_-^{k+1})}\|e^k\|^2 + c\langle e^{k+1}, 2Qr^{k+1}\rangle. \qquad \text{(C.8)}
\end{aligned}$$

The last equality comes from setting $\beta = \frac{\sigma_{max}^2(L_+^{k+1})}{2\sigma_{min}(L_-^{k+1})}$. Now we need to show

$$\begin{aligned}&\|p^{k+1} - p^k\|_{G^{k+1}}^2 + m_f\|\theta^{k+1} - \theta^*\|^2 + c^2\sigma_{max}^2(L_+^{k+1})\|e^k\|^2 \\ &+c^2\sigma_{max}^2(L_-^{k+1})\|e^{k+1}\|^2 \geq \delta\|p^{k+1} - p\|_{G^{k+1}}^2 \end{aligned} \tag{C.9}$$

which is equivalent to

$$\begin{aligned}& c\|r^{k+1} - r^k\| + c\|\theta^{k+1} - \theta^k\|_{\frac{L_+^{k+1}}{2}}^2 + m_f\|\theta^{k+1} - \theta^*\|^2 \\ \geq\ & \delta c\|r^{k+1} - r^*\|^2 + c\|\theta^{k+1} - \theta^*\|_{\frac{L_+^{k+1}}{2}}^2. \end{aligned} \tag{C.10}$$

First, we have

$$c\|\theta^{k+1} - \theta^*\|_{\frac{L_+^{k+1}}{2}}^2 \leq \frac{c\sigma_{max}^2(L_+^{k+1})}{2}\|\theta^{k+1} - \theta^*\|^2. \tag{C.11}$$

For the other part, we have

$$\|r^{k+1} - r^*\|^2 \leq \frac{\sigma_{max}(L_-)}{2}\|\sum_{s=0}^{k+1}\theta^s - \theta^*\|^2. \tag{C.12}$$

We have $2M_f^2\|\theta^{k+1} - \theta^*\|^2 + 2nV_{k+1}^2 \geq \|g(\theta^{k+1})\|^2$. By using inequality $\|a+b\|^2 + (\mu-1)\|a\|^2 \geq (1 - \frac{1}{\mu})\|b\|^2$, which holds for any $\mu > 1$, we have

$$\begin{aligned}& 2c^2\sigma_{max}^2(L_+^{k+1})\|\theta^{k+1} - \theta^k\|_{\frac{L_+^{k+1}}{2}}^2 + c^2\sigma_{max}^2(L_+^{k+1})\|e^k\|^2 \\ & \quad +c^2\sigma_{max}^2(L_-^{k+1})\|e^{k+1}\|^2 + (\mu-1)2M_f^2\|\theta^{k+1} - \theta^*\|^2 + 2(\mu-1)nV_{k+1}^2 \\ \geq\ & \|cL_+^{k+1}(z^{k+1} - z^k) - 2cW^{k+1}e^{k+1}\|^2 + (\mu-1)\|g(\theta^{k+1}) - g(\theta^*)\|^2 \\ \geq\ & (1 - \frac{1}{\mu})\|cL_-^{k+1}(\sum_{s=0}^{k+1}\theta^s - \theta^*)\|^2 \\ \geq\ & (1 - \frac{1}{\mu})c^2\sigma_{min}^2(L_-^{k+1})\|\sum_{s=0}^{k+1}\theta^s - \theta^*\|^2. \end{aligned} \tag{C.13}$$

Combining these two parts, we can have ρ as following to achieve the inequality:

$$\begin{aligned}\rho &= \min\left\{\frac{(\mu-1)\sigma_{min}^2(L_-^{k+1})}{2\mu\sigma_{max}^2(L_+^{k+1})\sigma_{max}(L_-^0)},\right.\\ &\left.\frac{m_f}{\frac{c\sigma_{max}^2(L_+)}{2}+\frac{\mu}{c}2M_f^2\sigma_{min}^{-2}(L_-^{k+1})\sigma_{max}(L_-^0)}\right\}. \end{aligned} \tag{C.14}$$

Then we have

$$\begin{aligned}\|p^{k+1}-p\|_{G^{k+1}}^2 &\leq \frac{1}{1+\rho}\left(\|p^k-p\|_{G^{k+1}}^2 + c\frac{\sigma_{max}^2(L_+^{k+1})}{2\sigma_{min}(L_-^{k+1})}\|e^k\|^2 + c\langle e^{k+1}, 2Qr^{k+1}\rangle\right.\\ &+\langle\theta^{k+1}-\theta^*, h(\theta^{k+1})\rangle + c^2\sigma_{max}^2(L_+^{k+1})\|e^k\|^2\\ &+c^2\sigma_{max}^2(L_-^{k+1})\|e^{k+1}\|^2 + 2(\mu-1)nV_{k+1}^2\\ &\left.+\langle\theta^{k+1}-\theta^*, \sum_{i=1}^n \nabla F^i(\theta_i^{k+1}) - \nabla f(\theta^{k+1})\rangle\right). \end{aligned} \tag{C.15}$$

For $h(\theta)$, we have

$$\|g(\theta^k)-\nabla f(\theta^k)\|^2 \leq \frac{nM_f^2d^2u_k^2}{4m}. \tag{C.16}$$

For last term, we have

$$\Pr\{\forall\theta : \|\nabla F(\theta) - \nabla\overline{f}^{(i)}(\theta)\| \leq 8\Delta_2\|\theta-\theta^*\| + 4\Delta_1\} \geq 1-\delta.$$

Then combining them, we get obtain the result in theorem.

C.4 Proof of Proposition 4.1

Since we have

$$\rho = \min\left\{\frac{(\mu-1)\sigma_{min}^2(L_-^{k+1})}{2\mu\sigma_{max}^2(L_+^{k+1})\sigma_{max}(L_-^0)}, \frac{m_f}{\frac{c\sigma_{max}^2(L_+)}{2}+\frac{\mu}{c}2M_f^2\sigma_{min}^{-2}(L_-^{k+1})\sigma_{max}(L_-^0)}\right\},$$

only the second term is related to parameter c. In order to maximize δ, the parameter c is chosen as

$$c = \frac{2M_f\sqrt{\sigma_{max}(L_-^0)\sqrt{\mu}}}{\sigma_{max}(L_+^{k+1})\sigma_{min}(L_-^{k+1})}. \tag{C.17}$$

Then the first term and second term are monotonically increasing and decreasing with parameter $\mu > 1$ respectively. So we choose the value of μ to make the first term and second term equal:

$$\mu = 1 + \frac{K_L^2\sigma_{max}(L_-^0)}{K_f^2} - \frac{K_L\sigma_{max}(L_-^0)}{2K_f}\sqrt{\frac{8}{\sigma_{max}(L_-^0)} + 4\frac{K_L^2}{K_f^2}}. \tag{C.18}$$

Then we have

$$\rho = \frac{1}{2K_f}\sqrt{\frac{8}{\sigma_{max}(L_-^0)K_{L^{k+1}}^2} + \frac{4}{K_f^2}} - \frac{1}{2K_f^2} \tag{C.19}$$

maximizes the value of δ in iteration $k+1$, where $K_{L^{k+1}} = \frac{\sigma_{max}(L_+^{k+1})}{\sigma_{min}(L_-^{k+1})}$ and $K_f = \frac{M_f}{m_f}$.

C.5 Proof of Lemma 4.4

When proving this lemma, we consider the optimal model parameter $\hat{\theta}^*$ for the empirical risk distributed problem,

$$\min_{\theta_i,\phi_{ij}} \sum_{i=1}^{n} \overline{f}^{(i)}(\theta_i), s.t.\theta_i = \phi_{ij}, \theta_j = \phi_{ij}, \forall(i,j) \in \mathcal{A}. \tag{C.20}$$

Consider a network without attacks, we have

$$\begin{aligned}
& \frac{f(\theta^{k+1}) - f(\hat{\theta}^*)}{c} + \langle 2Qr, \theta^{k+1}\rangle \\
\leq\ & \frac{1}{c}\langle \theta^{k+1} - \hat{\theta}^*, \nabla f(\theta^{k+1})\rangle + \langle 2Qr, \theta^{k+1}\rangle \\
=\ & \frac{1}{c}\langle \theta^{k+1} - \hat{\theta}^*, g(\theta^{k+1})\rangle + \langle 2Qr, \theta^{k+1}\rangle + \frac{1}{c}\langle \theta^{k+1} - \hat{\theta}^*, \nabla f(\theta^{k+1}) - g(\theta^{k+1})\rangle \\
\leq\ & \langle \theta^{k+1} - \hat{\theta}^*, -L_+(\theta^{k+1} - \theta^k)\rangle + \langle r^{k+1} - r^k, -2(r^{k+1} - r)\rangle \\
& + \frac{1}{c}\langle \theta^{k+1} - \hat{\theta}^*, \nabla f(\theta^{k+1}) - g(\theta^{k+1})\rangle.
\end{aligned} \tag{C.21}$$

Telescope and sum for $k = 0$ to T, we have

$$
\begin{aligned}
& \frac{1}{c}\sum_{k=1}^{T} f(\theta^k) - f(\hat{\theta}^*) + \langle 2Qr, \theta^k\rangle \\
\leq\ & \|\theta^0 - \hat{\theta}^*\|^2_{\frac{L_+}{2}} + \|r^0 - r\|^2 + \frac{R}{c}\sum_{k=1}^{T}\langle \theta^{k+1} - \hat{\theta}^*, \nabla f(\theta^{k+1}) - g(\theta^{k+1})\rangle \\
\leq\ & \|\theta^0 - \hat{\theta}^*\|^2_{\frac{L_+}{2}} + \|r^0 - r\|^2 + \frac{R}{c}\sum_{k=1}^{T}\|\nabla f(\theta^{k+1}) - g(\theta^{k+1})\|. \qquad \text{(C.22)}
\end{aligned}
$$

Define $\hat{\theta_T} = \frac{\sum_{k=1}^{T}\theta^k}{T}$, by Jensen's inequality, we have

$$
f(\hat{\theta_T}) - f(\hat{\theta}^*) + 2cr'Q\hat{\theta_T} \leq \frac{c}{T}\|p^0 - p\|^2_G + \frac{R}{T}\sum_{k=1}^{T}\frac{\sqrt{n}M_f du_k}{2\sqrt{m}}. \qquad \text{(C.23)}
$$

Since for vector $y \in \mathbb{R}^{nd}$ and $\sigma_{min}(yy') = 1$, we have this property such that $\forall\theta \in \mathbb{R}^{nd}, y^T\theta \geq \|\theta\|$. Then let $r = \hat{r}^* + y$ and y has property that $\sigma_{min}(yy') = 1$, then

$$
\begin{aligned}
& f(\hat{\theta_T}) - f(\theta^*) + 2c\hat{r^*}'Q\hat{\theta_T} + 2cy'Q\hat{\theta_T} \\
\leq\ & \frac{c}{T}(\|\theta^0 - \theta^*\|^2_{\frac{L_+}{2}} + \|r^0 - \hat{r}^* - y\|^2) + \frac{R}{T}\sum_{k=1}^{T}\frac{\sqrt{n}M_f du_k}{2\sqrt{m}}. \qquad \text{(C.24)}
\end{aligned}
$$

Since $(\hat{\theta}^*, \hat{r}^*)$ is a primal dual optimal solution, by the saddle point inequality, we have

$$
f(\hat{\theta_T}) - f(\hat{\theta}^*) + 2c\hat{r^*}'Q\hat{\theta_T} \geq 0. \qquad \text{(C.25)}
$$

Then we have

$$
\frac{2c}{T}\sum_{k=1}^{T}\|Q\theta^k\| \leq \frac{c}{T}(\|\theta^0 - \hat{\theta}^*\|^2_{\frac{L_+}{2}} + \|r^0 - \hat{r}^* - y\|^2) + \frac{R}{T}\sum_{k=1}^{T}\frac{M_f du_k}{2\sqrt{m}}, \qquad \text{(C.26)}
$$

which leads to

$$\frac{1}{T}\sum_{k=1}^{T}\|Q\theta^k\| \le \frac{1}{2T}(\|\theta^0-\hat{\theta}^*\|^2_{\frac{L_+}{2}}+2\|r^0-\hat{r}^*\|^2+2)+\frac{R}{2cT}\sum_{k=1}^{T}\frac{M_f d u_k}{2\sqrt{m}}. \tag{C.27}$$

Choosing $u_k=\frac{1}{dk^2}$, starting point $\theta^0=0$ and thus $r^0=0$, since $Q\hat{r}^*+\frac{1}{c}g(\hat{\theta}^*)=0$, we have

$$\begin{aligned}\frac{1}{T}\sum_{k=1}^{T}\|Q\theta^k\| \;&\le\; \frac{1}{4T}\left(\sigma_{max}(L_+)R^2+\frac{2\|g(\hat{\theta}^*)\|^2}{\sigma_{min}(L_-)c^2}+4\right)+\frac{R}{2cT}\frac{\sqrt{n}M_f\pi^2}{12\sqrt{m}}\\ &\le \frac{1}{4T}\left(\sigma_{max}(L_+)R^2+\frac{4(nV_T^2+M_f^2R^2)}{\sigma_{min}(L_-)c^2}+4\right)+\frac{R}{2cT}\frac{\sqrt{n}M_f\pi^2}{12\sqrt{m}}.\end{aligned} \tag{C.28}$$

C.6 Proof of Theorem 6

Consider the network with attacks, in iteration k, we will have

$$\begin{aligned}&\frac{\hat{f}(\theta^{k+1})-\hat{f}(\theta^*)}{c}+2r'\hat{Q}\theta^{k+1}\\ \le\;& \frac{1}{c}(\|\hat{p}^k-p\|^2_{G^{k+1}}-\|\hat{p}^{k+1}-p\|^2_{G^{k+1}})-\|\hat{Q}^{k+1}e^{k+1}\|^2+\frac{\sigma^2_{max}(\hat{L_+}^{k+1})}{2\sigma_{min}(\hat{L_-}^{k+1})}\|e^k\|^2\\ &+\langle e^{k+1},2\hat{Q}(\hat{r}^{k+1}-r)\rangle+\frac{1}{c}\langle\theta^{k+1}-\theta^*,\nabla\hat{f}(\theta^{k+1})-\hat{g}(\theta^{k+1})\rangle\\ \le\;& \frac{1}{c}(\|\hat{p}^k-p\|^2_{\hat{G}^{k+1}}-\|\hat{p}^{k+1}-p\|^2_{G^{k+1}})-\|\hat{Q}^{k+1}e^{k+1}\|^2+\frac{\sigma^2_{max}(\hat{L_+}^{k+1})}{2\sigma_{min}(\hat{L_-}^{k+1})}\|e^k\|^2\\ &+\|2\hat{Q}e^{k+1}\|(\sqrt{2}EU+\|r\|)+\frac{1}{c}\langle\theta^{k+1}-\theta^*,\nabla\hat{f}(\theta^{k+1})-\hat{g}(\theta^{k+1})\rangle.\end{aligned} \tag{C.29}$$

Telescope and sum from $k = 0$ to $T - 1$, we will get

$$\begin{aligned}
& \sum_{k=1}^{T} \hat{f}(\theta^k) - \hat{f}(\theta^*) + 2cr'\hat{Q}\theta^k \\
\leq \; & \|\hat{p}^0 - p\|^2_{G^1} - \|\hat{p}^T - p\|^2_{\hat{G}^{T+1}} + c\frac{\sigma^2_{max}(\hat{L_+}^T) - \sigma^2_{min}(\hat{L_-}^T)}{\sigma^2_{min}(\hat{L_-}^T)} \sum_{k=1}^{T} \|\hat{Q}^k e^k\|^2 \\
& +2c\sum_{k=0}^{T} \|\hat{Q}e^k\|(\sqrt{2}EU + \|r\|) + \sum_{k=1}^{T} \langle \theta^k - \theta^*, \nabla\hat{f}(\theta^k) - \hat{g}(\theta^k)\rangle \\
\leq \; & \|\hat{p}^0 - p\|^2_{G^1} - \|\hat{p}^T - p\|^2_{\hat{G}^{T+1}} + c\frac{\sigma^2_{max}(\hat{L_+}^T) - \sigma^2_{min}(\hat{L_-}^T)}{\sigma^2_{min}(\hat{L_-}^T)} 8E^2U^2 \\
& +c4\sqrt{2}EU(\sqrt{2}EU + \|r\|) + \frac{\pi^2}{6}\frac{\hat{n}\sqrt{\hat{n}}M_f R}{2n\sqrt{m}} \\
\leq \; & \|\hat{p}^0 - p\|^2_{G^1} - \|\hat{p}^T - p\|^2_{\hat{G}^{T+1}} + c\frac{\sigma^2_{max}(\hat{L_+}^T)}{\sigma^2_{min}(\hat{L_-}^T)} 8E^2U^2 \\
& +c4\sqrt{2}EU\|r\| + \frac{\pi^2}{6}\frac{\hat{n}\sqrt{\hat{n}}M_f R}{2n\sqrt{m}}. \qquad \text{(C.30)}
\end{aligned}$$

Choosing $r = 0$, and $\hat{\theta}_T = \frac{\sum_{k=1}^{T} \theta^k}{T}$, by Jensen's inequality we will obtain

$$\hat{f}(\hat{\theta_T}) - \hat{f}(\theta^*) \;\; \leq \;\; \frac{1}{T}\left(\|\hat{p}^0 - p\|^2_{\hat{G}^1} + c\frac{\sigma^2_{max}(\hat{L_+}^T)}{\sigma^2_{min}(\hat{L_-}^T)} 8E^2U^2 + \frac{\pi^2}{6}\frac{\hat{n}\sqrt{\hat{n}}M_f R}{2n\sqrt{m}} \right). \qquad \text{(C.31)}$$

C.7 Proof of Theorem 7

Since the eigenvalue of a block diagonal matrix is equal to the eigenvalue of each matrix block, we have

$$\begin{aligned}
& f(\hat{\theta_T}) - f(\theta^*) \\
= \; & \sum \hat{f}(\hat{\theta_T}) - f(\theta^*) \\
\leq \; & \frac{1}{T}\left(\sum \|\hat{p}^0 - p\|^2_{\hat{G}^1} + c\sum \frac{\sigma^2_{max}(\hat{L_+}^T)}{\sigma^2_{min}(\hat{L_-}^T)} 8E^2U^2 + \sum \frac{\pi^2}{6}\frac{\hat{n}\sqrt{\hat{n}}M_f R}{2n\sqrt{m}}\right) \\
\leq \; & \frac{1}{T}\left(\|p^0 - p\|^2_{G^1} + c\frac{\sigma^2_{max}(L_+^T)}{\sigma^2_{min}(L_-^T)} 8E^2U^2 + \frac{\pi^2}{6}\frac{\sqrt{n}M_f R}{2\sqrt{m}}\right).
\end{aligned} \tag{C.32}$$

Bibliography

[1] A. Agarwal, O. Dekel, and L. Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, pages 28–40. Citeseer, 2010.

[2] D. Alistarh, Z. Allen-Zhu, and J. Li. Byzantine stochastic gradient descent. In *Proc. Advances in Neural Information Processing Systems*, pages 4613–4623, Dec. 2018.

[3] P. Blanchard, E. Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. pages 119–129, Dec. 2017.

[4] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. Intl. Conf. on Computational Statistics*, pages 177–186. Springer, Paris, France, Aug. 2010.

[5] L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, Jan. 2011.

[7] X. Cao and L. Lai. ZOAR-ADMM: Zeroth-order ADMM robust to Byzantine attackers,submitted. *IEEE Trans. Signal Processing*.

[8] X. Cao and L. Lai. Robust distributed gradient descent with arbitrary number of Byzantine attackers. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 6373–6377, Calgary, Canada, Apr. 2018.

[9] X. Cao and L. Lai. Distributed gradient descent algorithm robust to an arbitrary number of Byzantine attackers. *IEEE Trans. Signal Processing*, 67(22):5850–5864, Nov. 2019.

[10] X. Cao and L. Lai. Distributed approximate Newton's method robust to Byzantine attackers. *IEEE Trans. Signal Processing*, 68:6011–6025, Oct. 2020.

[11] Shicong Cen, Huishuai Zhang, Yuejie Chi, Wei Chen, and Tie-Yan Liu. Convergence of distributed stochastic variance reduced methods without sampling extra data. *IEEE Transactions on Signal Processing*, 68:3976–3989, 2020.

[12] L. Chen, Z. Charles, D. Papailiopoulos, et al. Draco: Robust distributed training via redundant gradients. *arXiv preprint arXiv:1803.09877*, Jun. 2018.

[13] Y. Chen, L. Su, and J. Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, Dec. 2017.

[14] S. Chouvardas, G. Mileounis, N. Kalouptsidis, and S. Theodoridis. Greedy sparsity-promoting algorithms for distributed learning. *IEEE Trans. Signal Processing*, 63(6):1419–1432, Mar. 2015.

[15] A. Crotty, A. Galakatos, and T. Kraska. Tupleware: Distributed machine learning on small clusters. *IEEE Data Eng. Bull.*, 37(3):63–76, Sept. 2014.

[16] G. Damaskinos, R. Guerraoui, R. Patra, M. Taziki, et al. Asynchronous Byzantine machine learning (the case of SGD). In *International Conference on Machine Learning*, pages 1145–1154. PMLR, 2018.

[17] C. Dünner, A. Lucchi, M. Gargiani, A. Bian, T. Hofmann, and M. Jaggi. A distributed second-order algorithm you can trust. *arXiv preprint arXiv:1806.07569*, Jun. 2018.

[18] D. Friend, R. Thomas, A. MacKenzie, and L. Silva. Distributed learning and reasoning in cognitive networks: Methods and design decisions. *Cognitive networks: Towards self-aware networks*, pages 223–246, Jul. 2007.

[19] L. Gordon, M. Loeb, W. Lucyshyn, and R. Richardson. 2006 CSI/FBI computer crime and security survey. *Computer Security Journal*, 22(3):1, 2006.

[20] L. Huang, A. Joseph, B. Nelson, B. Rubinstein, and J. Tygar. Adversarial machine learning. In *Proc. ACM Workshop on Security and Artificial Intelligence*, pages 43–58. ACM, Oct. 2011.

[21] A. Jochems, T. Deist, J. Van Soest, M. Eble, P. Bulens, P. Coucke, W. Dries, P. Lambin, and A. Dekker. Distributed learning: Developing a predictive model based on data from multiple hospitals without data leaving the hospital–a real life proof of concept. *Radiotherapy and Oncology*, 121(3):459–467, Dec. 2016.

[22] M. Jordan, J. Lee, and Y. Yang. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 114(526):668–681, Apr. 2019.

[23] A. Krizhevsky, V. Nair, and G. Hinton. The CIFAR-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[24] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.

[25] Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[26] J. Lee, Q. Lin, T. Ma, and T. Yang. Distributed stochastic variance reduced gradient methods by sampling extra data with replacement. *Journal of Machine Learning Research*, 18(1):4404–4446, Feb. 2017.

[27] S. Liu, J. Chen, P. Chen, and A. Hero. Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. In *International Conference on Artificial Intelligence and Statistics*, pages 288–297. PMLR, Apr. 2018.

[28] A. Makhdoumi and A. Ozdaglar. Convergence rate of distributed ADMM over networks. *IEEE Trans. Automatic Control*, 62(10):5082–5095, Mar. 2017.

[29] S. Marano, V. Matta, and P. Willett. Nearest-neighbor distributed learning by ordered transmissions. *IEEE Trans. Signal Processing*, 61(21):5217–5230, Nov. 2013.

[30] P. Mertikopoulos, E. Belmega, R. Negrel, and L. Sanguinetti. Distributed stochastic optimization via matrix exponential learning. *IEEE Trans. Signal Processing*, 65(9):2277–2290, May 2017.

[31] S. Minsker et al. Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335, Mar. 2015.

[32] P. Moritz, R. Nishihara, I. Stoica, and M. Jordan. Sparknet: Training deep networks in spark. *arXiv preprint arXiv:1511.06051*, 2015.

[33] F. Provost and D. Hennessy. Scaling up: Distributed machine learning with cooperation. In *Proc. National Conf. on Artificial Intelligence*, volume 1, pages 74–79, Portland, Oregon, Aug. 1996.

[34] S. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. On variance reduction in stochastic gradient descent and its asynchronous variants. In *Proc. Advances in Neural Information Processing Systems*, pages 2647–2655, Dec. 2015.

[35] S. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, and A. Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, Aug. 2016.

[36] P. Richtárik and M. Takáč. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 17(1):2657–2681, Jan. 2016.

[37] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, Mar. 2016.

[38] O. Shamir, N. Srebro, and T. Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *Proc.Intl. Conf. on Machine Learning*, pages 1000–1008, Beijing, China, Jun. 2014.

[39] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Trans. Signal Processing*, 62(7):1750–1761, Feb. 2014.

[40] V. Smith, S. Forte, C. Ma, M. Takac, M. Jordan, and M. Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18(1):8590–8638, Jan. 2017.

[41] L. Su and J. Xu. Securing distributed gradient descent in high dimensional statistical learning. *Proc. ACM on Measurement and Analysis of Computing Systems*, 3(1):12, Mar. 2019.

[42] B. Swenson, S. Kar, and J. Xavier. Empirical centroid fictitious play: An approach for distributed learning in multi-agent games. *IEEE Trans. Signal Processing*, 63(15):3888–3901, Aug. 2015.

[43] Y. Tang, J. Zhang, and N. Li. Distributed zero-order algorithms for nonconvex multi-agent optimization. *IEEE Transactions on Control of Network Systems*, pages 269–281, Sep. 2020.

[44] C. Tekin and M. van der Schaar. Distributed online learning via cooperative contextual bandits. *IEEE Trans. Signal Processing*, 63(14):3700–3714, Jul. 2015.

[45] J. Tropp et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, May 2015.

[46] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, pages 7–8, Nov. 2010.

[47] J. Wainwright. High-dimensional statistics: A non-asymptotic viewpoint. *preparation. University of California, Berkeley*, 2015.

[48] S. Wang, F. Roosta-Khorasani, P. Xu, and M. Mahoney. Giant: Globally improved approximate newton method for distributed optimization. In *Proc. Advances in Neural Information Processing Systems*, pages 2332–2342, Dec. 2018.

[49] C. Xie, O. Koyejo, and I. Gupta. Phocas: Dimensional Byzantine-resilient stochastic gradient descent. *arXiv preprint arXiv:1805.09682*, May 2018.

[50] C. Xie, O. Koyejo, and I. Gupta. Zeno: Byzantine-suspicious stochastic gradient descent. *arXiv preprint arXiv:1805.10032*, Sep. 2018.

[51] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, Mar. 2018.

[52] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Defending against saddle point attack in Byzantine-robust distributed learning. *arXiv preprint arXiv:1806.05358*, Sep. 2018.

[53] C. Yu, M. van der Schaar, and A. Sayed. Distributed learning for stochastic generalized Nash equilibrium problems. *IEEE Trans. Signal Processing*, 65(15):3893–3908, Apr. 2017.

[54] Z. Yu, D. Ho, and D. Yuan. Distributed randomized gradient-free mirror descent algorithm for constrained optimization. *IEEE Trans. Automatic Control*, pages 1–1, Apr. 2021.

[55] Y. Zhang and X. Lin. Disco: Distributed optimization for self-concordant empirical loss. In *Proc.Intl. Conf. on Machine Learning*, pages 362–370, Lille, France, Jul. 2015.