ABSTRACT

In today's highly connected world, the number of smart devices worldwide has increased exponentially. These devices generate huge amounts of real-time data, perform complicated computational tasks, and provide actionable information. Over the past decade, numerous machine learning approaches have been widely adopted to infer hidden information from this massive and complex data. Accuracy is not enough when developing machine learning systems for some crucial application domains. The safety and reliability guarantees on the underlying learning models are critical requirements as well. This in turn necessitates that the learned models be robust towards processing corrupted data. Data can be corrupted by adversarial attacks where the attack may consist of data taking arbitrary values adversely affecting the efficiency of the algorithm. An adversary can replace samples with erroneous or malicious samples such as false labels or arbitrary inputs. In this dissertation, we refer to this type of attack as *attack on data*.

Moreover, with the rapid increase in the volume of the data, storing and processing all this data at a central location becomes computationally expensive. Therefore, utilizing a distributed system is warranted to distribute tasks across multiple machines (known as *distributed learning*). Improvement of the efficiency of the optimization algorithms with respect to computational and communication costs along with maintaining a high level of accuracy is critical in distributed learning. However, an attack can occur by replacing the transmitted data of the machines in the system with arbitrary values that may negatively impact the performance of the learning task. We refer to this attack as *attack on devices*. The aforementioned attack scenarios can significantly impact the accuracy of the results, thereby, negatively impacting the expected model outcome. Hence, the development of a new generation of systems that are robust to such adversarial attacks and provide provable performance guarantees is warranted. The goal of this dissertation is to develop learning algorithms that are robust to such adversarial attacks.

In this dissertation, we propose learning algorithms that are robust to adversarial attacks under two frameworks: 1) *supervised learning*, where the true label of the samples are known; and 2) *unsupervised learning*, where the labels are not known.

Although neural networks have gained widespread success, theoretical understanding of their performance is lacking. Therefore, in the first part of the dissertation (Chapter 2), we try to understand the inner workings of a neural network. We achieve this by learning the parameters of the network. In fact, we generalize the estimation procedure by considering the robustness aspect along with the parameter estimation in the presence of adversarial attacks (*attack on data*). We devise a learning algorithm to estimate the parameters (weight matrix and bias vector) of a single-layer neural network with rectified linear unit activation in the unsupervised learning framework where each output sample can potentially be an arbitrary outlier with a fixed probability. Our estimation algorithm uses gradient descent algorithms along with the median-based filter to mitigate the effect of the outliers. We further determine the number of samples required to estimate the parameters of the network in the presence of the outliers.

Combining the use of distributed systems to solve large-scale problems with the recent success of deep learning, there has been a surge of development in the field of distributed learning. In fact, the research in this direction has been further catalyzed by the development of federated learning. Despite extensive research in this area, distributed learning faces the challenge of training a high-dimensional model in a distributed manner while maintaining robustness against adversarial attacks. Hence, in the second part of the dissertation (Chapters 3 and 4), we study the problem of distributed learning in the presence of adversarial nodes (*attack on nodes*). Specifically, we consider the worker-server architecture to minimize a global loss function under both the learning frameworks in the presence of adversarial nodes (Byzantines). Each honest node performs some computation based only on its own local data, then communicates with the central server that performs aggregation. However, an adversarial node may send arbitrary information to the central server. In Chapter 3, we consider robust distributed learning under the supervised learning framework. We propose a novel algorithm that combines the idea of variance-reduction with a filtering technique based on *vector median* to mitigate the effect of the Byzantines. We prove the convergence of the approach to a first-order stationary point. Further, in Chapter 4, we consider robust distributed learning under the unsupervised learning framework (*robust clustering*). We propose a novel algorithm that combines the idea of redundant data assignment with the paradigm of distributed clustering. We show that our proposed approaches obtain constant factor approximate solutions in the presence of adversarial nodes.

ON ROBUST MACHINE LEARNING IN THE PRESENCE OF ADVERSARIES

By

Saikiran Bulusu
B.Tech., Jawaharlal Nehru Technological University, India, 2009
M.Tech., Indian Institute of Technology Madras, India, 2012

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering

Syracuse University
August 2023

To my parents

ACKNOWLEDGMENTS

" If I have seen further it is by standing on the shoulders of Giants."

attributed to Isaac Newton

" Take up one idea. Make that one idea your life - think of it, dream of it, live on that idea. Let the brain, muscles, nerves, every part of your body, be full of that idea, and just leave every other idea alone."

attributed to Swami Vivekananda

Although words can not do justice, this is my humble attempt at expressing my gratitude towards all the *Giants* in my life. To begin with, I am extremely grateful to my advisor Dr. Pramod K. Varshney for his invaluable guidance along with the continued intellectual and motivational support throughout my doctoral studies. Without his patience and consistent encouragement, this dissertation would not have been possible. I would also like to express my deepest appreciation to my co-advisor Dr. Venkata Gandikota for giving me the opportunity to learn from his deep insights and in-depth knowledge. I have thoroughly enjoyed discussing with him about research, philosophy, and future. I am honored to be his first doctoral advisee. I would like to thank them for their continuous support without which my doctoral studies would not be so enjoyable. I am deeply indebted to Dr. M. Cenk Gursoy, Dr. Sidharth Jaggi, Dr. Geethu Joseph, Dr. Bhavya Kailkhura, and Dr. Arya Mazumdar, for mentoring me during the course of my graduate studies. I would also like to express my deepest gratitude to my defense committee members Dr. Lixin Shen and Dr. Biao Chen for their valuable suggestions.

I would like to extend my sincere thanks to my current and past lab members and friends, including Adarsh, Alex, Anthony, Arick, Bao, Chen, Hanne, Haodong, Nandan, Pranay, Prashant, Qunwei, Shan, and Swatantra, for all the helpful technical and philosophical discussions we had during the course of this dissertation. I am thankful to them for being a constant source of inspiration to carry out my research. Times spent with Arick, Hanne, Haodong, Manish, Nandan, Pranay, Prashant, Romesh, and Swatantra, will forever be cherished. I would also like to take this opportunity to thank Mrs. Anju Varshney who made Syracuse home away from home with her love and warmth. I am also thankful to the administrative staff at the Department of EECS, Syracuse University. I also extend my gratitude to Skype and Zoom for my research online.

I am forever grateful to Dr. A. Paulraj, Dr. A. Chockalingam, Dr. Neelesh Mehta, Dr. Srikrishna Bhashyam, Dr. Arun Pachai Kannu, and Dr. R. Aravind for inspiring me and teaching me to take baby steps in research. I am also thankful to all my friends at IISc, who inspired and helped me in one way or another before coming to Syracuse. I am also grateful to all the professors who taught me during my undergraduate and masters days. I owe a greater debt to my school teacher, Vinod Kumar, then any other teacher in my life. A heartfelt thanks to all my undergraduate and masters friends for their support throughout this journey. I have missed many names here, but they are in my heart.

Lastly and most importantly, I am grateful to my parents for their unconditional love and unwavering support at the toughest of times; to Murthy uncle and Ramani aunt for being there when it was most needed; to Sriram uncle and Sashi aunt for their backing; to my cousins Divya, Shekhar, my sister-in-law Anusha, and my brother Arun for their constant support throughout this journey. It is needless to say that this dissertation would not have been possible without their support and blessings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Traditionally, data fusion techniques were applied to infer about a phenomenon from the information acquired from various sources. The assumption was that the statistical model over the underlying processes is known. However, in today's highly connected world, the number of smart devices, equipped with a multitude of sensors, worldwide has increased exponentially. In fact, it is predicted that the total number of smart devices around the world would rise up to more than 75 billion at the end of 2025 [58, 80]. Moreover, it is anticipated that there will be more than 9 smart devices per person at the end of 2025 [89]. These devices generate huge amounts of real-time data. Hence, the assumption of the statistical model being known is no longer feasible in many scenarios which makes the traditional data fusion techniques restrictive.

This has led to the development of the machine learning approaches to understand the structure of data and fit that data into models that can be understood and utilized by people. These approaches provide a principled set of mathematical methods for extracting meaningful features from the data. Machine learning can be applied for tasks such as classification, regression, data mining, binning data into distinct and meaningful patterns that can be exploited for decision making, state estimation, and forecasting. The goal of machine learning is to build algorithms that accept data as input and use statistical techniques to predict an output while updating the outputs when new data is available.

Typically, machine learning approaches can be broadly classified into two classes, namely supervised learning (SL) and unsupervised learning (UL).

Definition 1.1 (Supervised Learning)**.** *SL is a learning paradigm that uses labeled data to train algorithms to predict outcomes accurately. The association between the input examples and the labels is known.*

The labeled data available for the problem under consideration is used to train an appropriate model. Once it is trained, we can test the model during the testing phase to check if it is able to predict the right output with new examples. Classification and regression problems fall under the SL framework. In applications where training data is not available, unsupervised learning approaches can be employed where they look at complex data to organize it in potentially meaningful ways.

Definition 1.2 (Unsupervised Learning)**.** *UL is a learning paradigm that uses unlabeled data to train algorithms for data exploration or to analyze or generate new data. The association between the input examples and the labels is not known.*

Clustering and association problems fall under the UL framework. UL approaches are particularly valuable as unlabeled data are more abundant in practice than labeled data.

When developing machine learning systems for some crucial application domains, accuracy is not sufficient to characterize performance. The safety and reliability guarantees on the underlying learning models are critical requirements as well. This in turn necessitates that the learned models be robust in the presence of potentially corrupted data. Data can be corrupted by adversarial attacks where the attack may consist of arbitrary values adversely affecting the efficiency of the algorithm. An adversary can replace samples with erroneous or malicious samples such as false labels or arbitrary inputs. We refer to this type of attack as *attack on data*.

Moreover, with the rapid increase in the volume of the data, storing and processing all this data at a central location becomes computationally expensive [107]. Therefore, utilizing a distributed system is warranted to distribute tasks across multiple worker machines monitored by a central server (known as *distributed learning*). Thus, improving the efficiency of the learning algorithms with respect to computational and communication costs along with maintaining a high level of accuracy is critical in distributed learning. However, an attack can occur when the data transmitted from the machines in the system is replaced by arbitrary values that may negatively impact the performance of distributed learning. We refer to this type of attack as *attack on devices*. The aforementioned attack scenarios can significantly impact the accuracy of the results and thus, negatively impacting the expected model outcome. Hence, the development of a new generation of systems that are robust to such adversarial attacks is warranted which provide provable performance guarantees. The goal of this dissertation is to develop learning algorithms that are robust to such adversarial attacks under both the supervised and unsupervised learning frameworks that provide sufficient performance guarantees for the task at hand and are also simple to implement.

Next, we list the major contributions of the dissertation and discuss the organization of the dissertation.

1.1 Major Contributions

The goal of this dissertation is to develop machine learning algorithms that are robust to adversarial attacks under the two learning frameworks. Specifically, we want the proposed algorithms to be robust to adversarial attacks along with having provable guarantees for convergence. We list the major contributions of the dissertation as the following.

- Although neural networks (NNs) have seen considerable success, theoretical understanding of their performance is lacking. One of the approaches to obtain insights into the inner workings of a NN is to learn the parameters of the NN under supervised as well as unsupervised learning frameworks. An additional challenge is posed when some of the available data is replaced with arbitrary data by an adversary (*attack on data*). Under the supervised learning framework, in [44], the network parameters for a single-layer NN with non-linear activation where the labels are corrupted by noise were learned. Despite extensive research in unsupervised learning, the state-of-the-art parameter estimation of a neural network faces the challenge of estimation of the parameters of the NN in the presence of arbitrary data. To understand the workings of a neural network (*opening the black box*) and to address the above problem, we develop an algorithmic framework to estimate the weight matrix and the bias vector of a single-layer rectified linear unit (ReLU) neural network in the unsupervised learning framework in the presence of arbitrary outliers. The setting assumes a probabilistic model where each data sample at the output of the network can be an outlier with fixed probability and there is no deterministic upper bound on the number of outliers. The key idea to achieve optimal performance is the combination of techniques from robust statistics [32, 33] and the parameter estimation from truncated Gaussian samples [27]. To our knowledge, this is the first work that solves the problem of parameter estimation of a NN in the presence of arbitrary outliers in the unsupervised learning framework.

- In the distributed learning framework, the worker-server architecture is considered to minimize a global loss function or to learn a joint model under supervised as well as unsupervised learning frameworks. Despite extensive research, the state-of-the-art distributed learning problems face the challenge of training a high-dimensional model in a distributed manner while maintaining robustness against adversarial attacks.

  - *Supervised Learning:* Under this framework, we propose a robust variant of the stochastic gradient descent (SGD) algorithm to minimize a non-convex global objective function where a fraction ($< 1/2$) of the devices send arbitrary information. We provide the analysis for the convergence rate of the proposed algorithm that closely matches the convergence rate results in the literature for the non-convex objective function. Although the computational complexity is less in SGD compared to the gradient descent algorithm, it may introduce large variance in each step due to stochasticity. A solution to this problem is variance reduction which is employed to propose the stochastic variance reduced gradient descent (SVRG) algorithm. However, adversarial attacks under this framework have received less attention. Therefore, we propose robust variants of the SVRG algorithm to minimize convex and nonconvex objective functions for the above setting. In particular, we develop a framework that significantly departs from most of the approaches in the literature, in that the convergence rate of the algorithms does not depend on the dimension of the problem. This is due to a key design element of the proposed learning framework which is a filtering technique called the *vector median* to identify and prune the devices that exhibit adversarial behavior (also known as Byzantine devices) so that the convergence rate of the algorithms does not depend on the dimension of the problem.

– *Unsupervised Learning:* Unlike the supervised learning framework, a popular method to learn the structure of the data is clustering under the unsupervised learning framework. We propose robust algorithms for k means and k median clustering. Specifically, we develop a distributed clustering algorithm for large datasets using *coded computations* which is robust to adversarial attacks. A key feature of the framework is the *redundant data assignment scheme* such that the information obtained from a subset of devices is sufficient to compute the desired objective function on the entire dataset which is especially required when some of the devices are adversarial in nature. To our knowledge, this is the first work that proposes robust algorithms to solve the distributed clustering problem in the presence of adversaries.

Below, we discuss the organization of the dissertation.

1.2 Organization of the Dissertation

The dissertation is organized into five chapters. In Chapter 2, we propose a robust algorithm based on the gradient descent algorithm combined with the median-based filter to mitigate the effect of the arbitrary outliers for estimating the parameters (weight matrix and bias vector) of the neural network, assuming the bias vector to be non-negative. We then obtain bounds on the number of samples and the run time that is sufficient for our algorithm to estimate the neural network parameters. Our analysis provides insights into the training complexity of ReLU NNs in terms of the probability of outliers and problem dimension.

In Chapter 3, we propose the robust variant of SGD and SVRG algorithms to solve the distributed stochastic optimization problem in the presence of adversarial nodes. We then provide the convergence rates of the proposed algorithms that employ a novel filtering rule as a result of which they are independent of the problem dimension.

In Chapter 4, we propose redundant data assignment schemes that enable us to obtain global information about the entire dataset to solve the distributed clustering problem, even when some machines can either be straggling machines that fail to respond within a stipulated time or Byzantines that send arbitrary responses. We then propose robust algorithms to solve the distributed clustering problem in the presence of stragglers or Byzantines.

Finally, in Chapter 5 we conclude the dissertation with a summary and some possible future directions we intend to pursue.

Before proceeding further we first discuss the notations we will use in the rest of the dissertation.

1.3 Notations

The probability of an event A and the conditional probability of A given B are denoted by $\mathbb{P}[A]$ and $\mathbb{P}[A|B]$, respectively. The expectation of a RV is given by $\mathbb{E}[\cdot]$. The ℓ_2-norm is denoted by $\|\cdot\|$. We use the notation $[K] = \{1, \ldots, K\}$. $\mathbf{1}_n$ denotes a vector of all 1's of length n, and $d(\mathbf{x}, \mathbf{y})$ denotes the Euclidean distance between two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$.

1.4 Bibliographic Note

Most of the research work appearing in this dissertation has either appeared in the publications listed below or is under review.

Work Included in the Dissertation

Journal Papers:

- S. Bulusu, V. Gandikota, A. Mazumdar, A. S. Rawat, and P. K. Varshney, “Robust Distributed Clustering with Redundant Data Assignment", *submitted to IEEE Trans. on Information Theory*, 2023. (Chapter 4)

- S. Bulusu, P. Khanduri, S. Kafle, P. Sharma, and P. K. Varshney, “Byzantine Resilient Non-Convex SCSG With Distributed Batch Gradient Computations", *IEEE Trans. Signal and Information Process. over Networks*, vol. 7, pp. 754-766, 2021. (Chapter 3)

Conference Papers:

- S. Bulusu, G. Joseph, M. C. Gursoy, and P. K. Varshney, “Learning Distributions Generated by Single-Layer ReLU Networks in the Presence of Arbitrary Outliers", *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)*, 2022. (Chapter 2)

- S. Bulusu, V. Gandikota, A. Mazumdar, A. S. Rawat, and P. K. Varshney, “Byzantine Resilient Distributed Clustering with Redundant Data Assignment", *IEEE International Symposium on Information Theory (ISIT)*, 2021. (Chapter 4)

- S. Bulusu, P. Khanduri, P. Sharma, and P. K. Varshney, “On Distributed Stochastic Gradient Descent for Nonconvex Functions in the Presence of Byzantines", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020. (Chapter 3)

- P. Khanduri, S. Bulusu, P. Sharma, and P. K. Varshney, “Byzantine resilient non-convex SVRG with distributed batch gradient computations", *Optimization for Machine Learning (OPTML)*, 2019. (Chapter 3)

- S. Bulusu, Q. Li and P. K. Varshney, “On Convex Stochastic Variance Reduced Gradient for Adversarial Machine Learning", *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2019. (Chapter 3)

Work not Included in the Dissertation

Journal Papers:

- C. Quan, S. Bulusu, B. Geng, and P. K. Varshney, “Ordered Transmission-based Detection in Distributed Networks in the Presence of Byzantines", *submitted to IEEE Trans. Signal Process.*, 2023.
- S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, and D. Song, “Anomalous Example Detection in Deep Learning: A Survey", *IEEE Access*, vol. 8, pp. 132330-132347, 2020.

Under Preparation:

- S. Bulusu, V. Gandikota, A. Grootveld, G. Joseph, and P. K. Varshney, “One-bit Compressed Sensing with Adversarial Noise", *to be submitted to IEEE Trans. Signal Process.*, 2023.

Conference Papers:

- S. Bulusu, V. Gandikota, and P. K. Varshney, “One-bit Compressed Sensing with Local Sparsity", *IEEE International Symposium on Information Theory (ISIT)*, 2023.
- N. Sriranga, A. Trezza, S. Bulusu, D. J. Bucci, and P. K. Varshney, “Online Identification of Recurring Changepoints", *Asilomar Conference on Signals, Systems, and Computers*, 2022.

Other Publications Prior to SU:

- S. Bulusu, N. B. Mehta, and S. Kalyanasundaram, “Rate Adaptation, Scheduling, and Mode Selection in D2D Systems With Partial Channel Knowledge", *IEEE Trans. on Wireless Comm.*, vol. 17, no. 2, pp. 1053-1065, Feb. 2018.
- P. S. Kumar, S. Bulusu, A. P. Kannu, and S. Bhashyam, “Algorithms for change detection with unknown number of affected sensors,", *NCC*, 2013.

Chapter 2
Learning Distributions Generated by Single-Layer ReLU Networks

In this chapter, we propose a robust algorithm for the estimation of the parameters (weight matrix and bias vector) of a single-layer neural network (NN) with rectified linear unit (ReLU) activation. We first introduce a few assumptions and the observations required to design the algorithm. Then, we provide the intuition for the design of the algorithm. Next, we describe the steps involved in the proposed algorithm. We then derive the error bounds for the estimates of the parameters given by the algorithm. We also derive the sample complexity bounds for the proposed approach. Finally, we present the simulation results and conclude the chapter.

2.1 Introduction

Although NNs achieve state-of-the-art performance in many fields, they have one major drawback. They are extremely hard to interpret. Typically, a NN consists of multiple interconnected layers of multiple neurons, each of which applies a mathematical transformation to the received data. We can observe the data at the input of the networks and also at the output. We can also observe the individual changes to each neuron and understand all the singular pieces of their inner workings.

However, it is difficult to fully understand the end-to-end behavior due to the nonlinear transformation that these networks apply. Moreover, it is also very difficult to verify which neurons are important for the result and are not just getting activated due to some relationship learned from the randomness in the training set. Hence, this *black box* nature of the NNs has become an impediment to its adoption in areas where decisions have irreversible consequences.

Therefore, having a theoretical understanding about the inner workings of the NNs is warranted. One possible approach is the estimation of the parameters of the NN under both supervised and unsupervised learning frameworks using mathematical approaches (optimization or statistics). The estimation approaches under the supervised learning framework generally rely on the stochastic gradient descent (SGD)-based algorithm [3, 22, 48, 72, 82]; or the gradient descent (GD)-based approach [19, 34]. Further, some works have considered the sample complexity of a single-layer NN with ReLU activation in the unsupervised learning framework [78, 102]. It is known that when the bias vector is assumed to be a random vector, the column space of the weight matrix can be recovered within an error of $O(d)$ when the output dimension is d [78]. Further, when the bias vector is assumed to be nonnegative, $\tilde{O}(1/\epsilon^2)$ samples and $\tilde{O}(d^2/\epsilon^2)$ iterations are sufficient to estimate the parameters of the ReLU network within an error of ϵ. However, none of the above works have considered the problem of corrupted samples. To the best of our knowledge, the estimation of a neural network in the presence of noise or outliers has been addressed only in the context of supervised learning [5, 42, 47, 79, 95, 116]. This chapter focuses on the case of unsupervised learning with corrupted samples wherein we consider a single layer ReLU network.

The NN parameter estimation with corrupted samples is also related to the area of robust statistics. Mathematically, the problem of learning the distribution using a single layer NN is equivalent to estimating the parameters of a truncated Gaussian distribution. Robust statistics also deals with the estimation of high dimensional distributions like Gaussian, Gaussian product, and Gaussian mixture distributions where a fraction of the samples are arbitrary outliers [32, 33, 52, 57, 64, 68]. However, parameter estimation from truncated Gaussian samples in the noisy setting has not been studied in the robust statistics literature, and we address this literature gap.

We present our major contributions as the following.

- *Sample Complexity:* Our algorithm requires $\tilde{O}\left(\frac{1}{p}\left[\frac{1}{p}+\frac{1}{\epsilon^2}\right]\log\frac{d}{\delta}\right)$ samples to estimate the network parameters within an error of ϵ with probability $1-\delta$ when the probability of a sample being uncorrupted is $p \in (2/3, 1]$ and the output dimension is d. We also characterize the total variation distance between the estimated and true distributions.
- *Lower bound:* We derive a lower bound on the sample complexity which says that at least $\Omega(1/p\epsilon^2)$ samples are required to estimate the parameters up to an error of ϵ.
- *Empirical validation:* We empirically evaluate the performance of our algorithm and show that it is robust to arbitrary outliers. Also, we observe that the performance of our algorithm improves with the probability of a sample being uncorrupted and the number of samples, and it increases slowly as the network output dimension grows. These observations from the empirical results are consistent with our theoretical results.

Overall, the results obtained show that parameter estimation of a single layer ReLU neural network is possible using robust gradient descent even in the presence of outliers.

2.2 Problem Statement

Let the weight matrix of the ReLU NN be denoted by $\boldsymbol{W} \in \mathbb{R}^{d\times m}$ and the bias vector by $\boldsymbol{b} \in \mathbb{R}^d$. The input to the neural network is denoted by the latent variable $\boldsymbol{z} \in \mathbb{R}^m$. We assume that the variable $\boldsymbol{z}$ is drawn from the standard Gaussian distribution. Thus, the output of the network is the random vector $\boldsymbol{x} \in \mathbb{R}^d$ given by

$$\boldsymbol{x} = \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{z}+\boldsymbol{b}) \sim \mathcal{D}(\boldsymbol{W},\boldsymbol{b}), \tag{2.1}$$

where $\mathcal{D}(\boldsymbol{W},\boldsymbol{b})$ denotes the distribution of $\boldsymbol{x}$. Our goal is to estimate the unknown parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ of the distribution $\mathcal{D}(\boldsymbol{W},\boldsymbol{b})$ using the knowledge of a corrupted set of data samples

$\mathcal{X} = \left\{\boldsymbol{x}^{(n)} \in \mathbb{R}^d\right\}_{n=1}^{N}$. Here, we assume that a data sample in $\mathcal{X}$ follows *Huber's p-contamination model* [56], i.e., a sample is drawn from $\mathcal{D}(\boldsymbol{W}, \boldsymbol{b})$ with probability p, and it is an arbitrary outlier drawn from an unknown distribution $\mathcal{D}_{\text{out}}$ with probability $1 - p$. Hence, a given sample $\boldsymbol{x} \in \mathcal{X}$ follows the distribution $\mathcal{D}_p = p\mathcal{D}(\boldsymbol{W}, \boldsymbol{b}) + (1 - p)\mathcal{D}_{\text{out}}$.

We make two observations about the learning problem. Firstly, it is known that when all the samples are from the true distribution, exponentially large number of samples are required to estimate the bias $\boldsymbol{b}$, if it can take any value from the set $\mathbb{R}^d$ [102] as the following.

Claim 1. *For any value $\epsilon > 0$, there exists one-dimensional distributions $\mathcal{D}(1, b_1)$ and $\mathcal{D}(1, b_2)$ such that: (a) $|b_1 - b_2| = \epsilon$; (b) at least $\Omega(\exp{(b_1^2/2)})$ samples are required to distinguish them.*

PROOF: Let us assume $b_1 < 0$ and $b_2 = b_1 - \epsilon$. Firstly, (a) holds. Next, notice that the probability of observing a positive sample from $\mathcal{D}(1, b_1)$ is $\mathbb{P}(\text{ReLU}(z - |b_1|) > 0) = \mathbb{P}(z > |b_1|)$. From the standard Gaussian tail bound in [98], this probability is upper bounded by $\exp{(-b_1^2/2)}$. Same is the case when the sample is from $\mathcal{D}(1, b_2)$. Hence, to distinguish $\mathcal{D}(1, b_1)$ and $\mathcal{D}(1, b_2)$, we need to observe at least one nonzero sample. This requires $\Omega(\exp{(b_1^2/2)})$ samples. ■

Naturally, the requirement on the number of samples would be worse in the presence of arbitrary outliers. Hence, to reduce the number of samples required we assume $\boldsymbol{b}$ to be non-negative. Thus, we assume the following.

Assumption 1. *The entries of $\boldsymbol{b} \in \boldsymbol{R}^d$ are all nonnegative.*

Secondly, the weight matrix $\boldsymbol{W}$ may not be identifiable from the distribution $\mathcal{D}(\boldsymbol{W}, \boldsymbol{b})$. In particular, for any unitary matrix $\boldsymbol{Q} \in \mathbb{R}^{m \times m}$, we have $\mathcal{D}(\boldsymbol{W}, \boldsymbol{b}) = \mathcal{D}(\boldsymbol{W}\boldsymbol{Q}, \boldsymbol{b})$. Since our goal is to learn the distribution, learning either $\boldsymbol{W}$ or $\boldsymbol{W}\boldsymbol{Q}$ is sufficient. Thus, we focus on the learnability of the underlying distribution and not the learnability of the neural network parameters. Specifically, our proposed algorithm estimates $\boldsymbol{W}\boldsymbol{W}^{\mathsf{T}} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b} \in \mathbb{R}^d$.

We tackle the issue of estimating the weight matrix and the bias vector of a NN in the presence of arbitrary outliers using a new formulation which we discuss next.

2.3 Robust Estimation Algorithm

Our algorithm is similar to that in [102] which considers the special case of $p = 1$ and uses SGD. For the general case of $p \leq 1$, we combine the estimation framework with a robust filter to estimate the parameters.

To derive the robust estimation algorithm, we first consider a true sample $\boldsymbol{x} \sim \mathcal{D}(\boldsymbol{W}, \boldsymbol{b})$ whose i-th element is nonzero. We have $\boldsymbol{x}_i = \mathrm{ReLU}(\boldsymbol{W}_i^{\mathsf{T}} \boldsymbol{z} + \boldsymbol{b}_i) \sim \mathcal{N}^+(\boldsymbol{b}_i, \|\boldsymbol{W}_i\|^2)$, where $\mathcal{N}^+(\boldsymbol{b}_i, \|\boldsymbol{W}_i\|^2)$ is the truncated normal distribution obtained by restricting the normal distribution $\mathcal{N}(\boldsymbol{b}_i, \|\boldsymbol{W}_i\|^2)$ to the set of positive real numbers. Hence, estimation of $\boldsymbol{b}_i$ and $\|\boldsymbol{W}_i\|$ is equivalent to estimating the parameters of a one-dimensional truncated normal distribution $\mathcal{N}^+(\boldsymbol{b}_i, \|\boldsymbol{W}_i\|^2)$ using positive samples. Therefore, we estimate the parameters in two steps. In the first step, we estimate the i-th element $\boldsymbol{b}_i$ of the bias vector and the norm $\|\boldsymbol{W}_i\|$ of the i-th row of the weight matrix, for $i \in [d]$. The second step is the estimation of the angle θ_{ij} between the i-th row $\boldsymbol{W}_i \in \mathbb{R}^m$ and the j-th row $\boldsymbol{W}_j \in \mathbb{R}^m$ of the matrix $\boldsymbol{W}$, for $i, j \in [d]$. Then, the (i, j)-th entry of the symmetric matrix $\boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}$ is given by $\|\boldsymbol{W}_i\| \, \|\boldsymbol{W}_j\| \cos(\theta_{ij})$.

The first step of the algorithm estimates the parameters of the univariate distribution given by $\mathcal{N}^+(\boldsymbol{b}_i, \|\boldsymbol{W}_i\|^2)$ using the i-th element of the samples $\mathcal{X}_i^+ = \{\boldsymbol{x}_i : \boldsymbol{x}_i > 0, \boldsymbol{x} \in \mathcal{X}\}$ via maximum likelihood estimation [27]. Then, the parameter estimates are given by $\arg\min_{\mu,\sigma^2} \ell(\mu, \sigma^2)$ where $\ell(\mu, \sigma^2)$ is the expected negative log likelihood that $\boldsymbol{x}_i \sim \mathcal{N}^+(\mu, \sigma^2)$ and the expectation is with respect to the true distribution $\boldsymbol{x}_i \sim \mathcal{N}^+(\boldsymbol{b}_i, \|\boldsymbol{W}_i\|^2)$. Further, from [27], $\ell(\mu, \sigma^2)$ is a convex function of $\boldsymbol{v} = \begin{bmatrix} 1/\sigma^2 & \mu/\sigma^2 \end{bmatrix} \in \mathbb{R}^2$. Thus, we solve the convex optimization problem of maximizing $\ell(\mu, \sigma^2)$ over $\boldsymbol{v}$,

$$\boldsymbol{v}^* = \underset{\boldsymbol{v} \in \mathbb{R}^2}{\arg\min} \, \ell(\boldsymbol{v}). \tag{2.2}$$

The optimization problem in Eq. (2.2) can be solved using GD or SGD which is based on the gradient of $\ell(\boldsymbol{v})$ given by the relation [27]: $\nabla\ell(\boldsymbol{v}) = \boldsymbol{g} - \boldsymbol{h}(\boldsymbol{v})$ where

$$\boldsymbol{g} = \mathbb{E}_{x\sim\mathcal{N}^+\left(\boldsymbol{b}_i,\|\boldsymbol{W}_i\|^2\right)}\left\{\begin{bmatrix} x^2/2 & -x \end{bmatrix}^\mathsf{T}\right\}, \tag{2.3}$$

$$\boldsymbol{h}(\boldsymbol{v}) = \mathbb{E}_{y\sim\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)}\left\{\begin{bmatrix} \frac{y^2}{2} & -y \end{bmatrix}^\mathsf{T}\right\} = \begin{bmatrix} \frac{\sigma^2+\mu^2}{2} + \frac{\mu\sigma\phi(-\mu/\sigma)}{2(1-\Phi(-\mu/\sigma))} & -\mu - \frac{\sigma\phi(-\mu/\sigma)}{1-\Phi(-\mu/\sigma)} \end{bmatrix}^\mathsf{T}, \tag{2.4}$$

where $\mu = \boldsymbol{v}_2/\boldsymbol{v}_1$, $\sigma^2 = 1/\boldsymbol{v}_1$, and $\phi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function and the cumulative distribution function of the standard normal distribution, respectively. The relation in Eq. (2.4) follows directly from the closed form expressions of the first and second moments of the truncated Gaussian distribution $\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)$, which are functions of $\boldsymbol{v}$ only [62].

We observe that $\boldsymbol{g}$ does not depend on $\boldsymbol{v}$ and only depends on the true distribution parameters $(\boldsymbol{b}_i, \|\boldsymbol{W}_i\|^2)$, whereas $\boldsymbol{h}(\boldsymbol{v})$ does not depend on the true distribution. Therefore, the estimation of $\boldsymbol{g}$, which depends on the true distribution, can use the available data samples, and $\boldsymbol{h}(\boldsymbol{v})$ can be computed in closed form using the current iterate of $\mathbf{v}$. In other words, we compute the component of gradient $\boldsymbol{g}$ only once in the GD or SGD algorithm because it does not change across the iterations. This observation motivates us to use the GD algorithm to estimate the parameters instead of SGD. SGD introduces large variance due to the stochasticity and lower accuracy due to the outliers in every iteration whereas GD introduces a small error that does not depend on the algorithm iterate. This key observation and the use of Eq. (2.4) to compute the gradient is the main difference between our algorithm and the algorithm in [102], apart from the robust estimation aspect.

To estimate $\boldsymbol{g}$, we partition $\mathcal{X}_i^+$ into batches of size N_B and compute the batchwise estimate $\tilde{\boldsymbol{g}}^{(b)}$,

$$\tilde{\boldsymbol{g}}^{(b)} = \frac{1}{N_B}\sum_{x\in\mathcal{X}_{i,b}^+}\begin{bmatrix} x^2/2 & -x \end{bmatrix}^\mathsf{T}, \tag{2.5}$$

where $\mathcal{X}_{i,b}^+ \subset \mathcal{X}_i^+$ is the b-th batch of samples. We then combine the batchwise estimates using a well-known filter in the robust statistics literature such as the median or trimmed mean to handle the outliers.

- Median: The median filter possesses the following robustness property. Typically, a median is the value separating the higher half from the lower half of a given set of points. If more than half of a given set of points are in $[-M, M]$ for some $M > 0$, then their median must be in $[-M, M]$.

- Trimmed Mean: The trimmed mean removes the vectors among the set of N_b vectors with relatively large and small values and computes the mean of the remaining vectors. Here, we use a parameter β to indicate the number of vectors to be discarded.

The median filter mitigates the effect of outliers by ensuring that if more than half of the batchwise estimates lie in an interval around the true value $\boldsymbol{g}$, then their median also lies in the same interval. Similarly, the trimmed mean prunes the outliers by removing the vectors with relatively large and small values (controlled by its parameter) and computes the estimate of $\boldsymbol{g}$ as the mean of the remaining vectors. Therefore, we obtain the gradient estimate $\tilde{\boldsymbol{g}} - \boldsymbol{h}(\boldsymbol{v})$ as

$$\tilde{\boldsymbol{g}} - \boldsymbol{h}(\boldsymbol{v}) = \text{filter}\left(\tilde{\boldsymbol{g}}^{(1)}, \tilde{\boldsymbol{g}}^{(2)}, \ldots, \tilde{\boldsymbol{g}}^{\left(|\mathcal{X}_i^+|/N_B\right)}\right) - \boldsymbol{h}(\boldsymbol{v}), \tag{2.6}$$

where the function $\text{filter}(\cdot)$ is either median or trimmed mean, and $\boldsymbol{h}(\boldsymbol{v})$ is given in Eq. (2.4). Using the gradient estimate, the robust GD algorithm updates the k-th iterate $\boldsymbol{v}(k)$ as

$$\boldsymbol{v}(k) = P\left(\boldsymbol{v}(k-1) - \gamma(k-1)\left[\tilde{\boldsymbol{g}} - \boldsymbol{h}(\boldsymbol{v}(k-1))\right]\right), \tag{2.7}$$

where $\gamma(k) > 0$ is the diminishing step size and $P(\cdot)$ projects the iterate into a bounded region $\mathbb{D}_r$ as

$$\mathbb{D}_r = \left\{\boldsymbol{v} \in \mathbb{R}^2 : 1/r \leq \boldsymbol{v}_1 \leq r, 0 \leq \boldsymbol{v}_2 \leq r\right\} \tag{2.8}$$

$$P(\boldsymbol{v}) = \begin{bmatrix} \min\{\max\{\boldsymbol{v}_1, 1/r\}, r\} & \min\{\max\{\boldsymbol{v}_2, 0\}, r\} \end{bmatrix}. \tag{2.9}$$

The projection ensures that the expected negative log-likelihood $\ell(\boldsymbol{v})$ is a strongly convex function of $\boldsymbol{v}$, and the parameter r controls the strong-convexity [27] (see Section 2.4 for more details). The robust GD algorithm is summarized in Algorithm 2. The role of r is further discussed in Section 2.4. This completes the first step of our algorithm based on robust GD which is summarized in Algorithm 2.

Finally, using the estimates obtained via the robust GD algorithm, we estimate $\hat{\theta}_{ij}$ similar to [102] using [101, Lemma 6.7]. Specifically, we have

$$\hat{\theta}_{ij} = \pi - 2\pi \left[\frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(\boldsymbol{x}_i^{(n)} > \hat{\boldsymbol{b}}_i\right) \mathbb{1}\left(\boldsymbol{x}_j^{(n)} > \hat{\boldsymbol{b}}_j\right) \right], \tag{2.10}$$

where $\mathbb{1}(\cdot)$ is the indicator function and $\hat{\boldsymbol{b}}$ is the output of the robust GD algorithm. The overall distribution learning algorithm is given in 1 where $\hat{\boldsymbol{\Sigma}}$ denotes the estimate of $\boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}$.

Algorithm 1: ReLU network estimation

```
   Input: Samples $\mathcal{X} = \{\boldsymbol{x}^{(n)} \in \mathbb{R}^d\}_{n=1}^N$
1  for $i \in [d]$ do
2      $\mathcal{X}_i^+ \leftarrow \{\boldsymbol{x}_i : \boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{x}_i > 0\}$
3      Compute $\hat{\boldsymbol{v}}$ using 2 with input as $\mathcal{X}_i^+$
4      $\hat{\boldsymbol{\Sigma}}_{i,i} \leftarrow 1/\hat{\boldsymbol{v}}_1$
5      $\hat{\boldsymbol{b}}_i \leftarrow \max\{0, \hat{\boldsymbol{v}}_2/\hat{\boldsymbol{v}}_1\}$
6  for $i < j \in [d]$ do
7      Compute $\hat{\theta}_{ij}$ using Eq. (2.10)
8      $\hat{\boldsymbol{\Sigma}}_{i,j} \leftarrow \sqrt{\hat{\boldsymbol{\Sigma}}_{i,i}\hat{\boldsymbol{\Sigma}}_{j,j}} \cos(\hat{\theta}_{ij})$
9      $\hat{\boldsymbol{\Sigma}}_{j,i} \leftarrow \hat{\boldsymbol{\Sigma}}_{i,j}$
   Output: $\hat{\boldsymbol{\Sigma}} \in \mathbb{R}^{d\times d}$ and $\hat{\boldsymbol{b}} \in \mathbb{R}^d$
```

Algorithm 2: Robust GD

```
   Input: Positive samples $\mathcal{X}^+ \subset \mathbb{R}^+$
   Parameters: Iterations $K$, step size $\gamma(k)$, projection parameter $r$, batch size $N_B$
1  $B \leftarrow |\mathcal{X}^+| / N_B$
2  Compute $\{\tilde{\boldsymbol{g}}^{(b)}\}_{b=1}^B$ from $\mathcal{X}^+$ using Eq. (2.5)
3  $\tilde{\boldsymbol{g}} \leftarrow$ filter $\left(\tilde{\boldsymbol{g}}^{(1)}, \tilde{\boldsymbol{g}}^{(2)}, \ldots, \tilde{\boldsymbol{g}}^{(B)}\right)$
4  $\boldsymbol{v}(0) \leftarrow \boldsymbol{0}$
5  for $k = 1, 2, \ldots, K$ do
6      $\mu \leftarrow \frac{\boldsymbol{v}_2(k-1)}{\boldsymbol{v}_1(k-1)}$; $\sigma^2 \leftarrow \frac{1}{\boldsymbol{v}_1(k-1)}$
7      Compute $\boldsymbol{h}(\boldsymbol{v}(k-1))$ using Eq. (2.4)
8      Update $\boldsymbol{v}(k)$ using Eq. (2.7)
   Output: $\boldsymbol{v}(K) \in \mathbb{R}^2$
```

2.4 Error Bounds for Parameter Estimation

This section provides our main result which characterizes the sample complexity for robust estimation of neural network parameters. The analysis of distributed learning in the presence of arbitrary outliers is studied in [24, 112]. They assume that the available samples are split into a fixed number of batches and a constant fraction (<1/2) of batches are outliers. However, our setting assumes a probabilistic model where each data sample can be an outlier with probability $1-p$ and there is no deterministic upper bound on the number of outliers. Consequently, each batch can have a mixture of true samples and outliers, and it is critical to choose the right batch size N_B (see Proposition 2.2 and its discussion for details).

We rely on two propositions to arrive at the main results. The propositions establish the properties of the objective function of the optimization problem in Eq. (2.2) and the error bounds for the robust GD algorithm presented in Algorithm 2.

Proposition 2.1. *There exist positive constants L and η that depend only on the projection parameter $r \geq 1$ of Algorithm 2 such that the objective function $\ell(\boldsymbol{v})$ in Eq.* (2.2) *is an $\eta-$strongly convex and $L-$smooth function of $\boldsymbol{v}$ when $\boldsymbol{v} \in \mathbb{D}_r$ defined in Eq.* (2.8). *Here, L is an increasing function of r whereas η is a decreasing function of r.*

PROOF: See Appendix A.2. ■

Proposition 2.1 proves that the projection parameter r controls the strong-convexity and smoothness parameters of the objective function. If r takes a large value, it leads to a small strong-convexity parameter η and a large smoothness parameter L. We use this property to interpret the role of parameter r in the algorithm performance using the next result that presents the error bounds of the robust GD algorithm. To this end, we make the following assumptions to derive the error bounds:

Assumption 2. *The projection parameter $r \geq 1$ is such that the minimizer of Eq.* (2.2), *$\boldsymbol{v}^* \in \mathbb{D}_r$ where $\mathbb{D}_r$ is defined in Eq.* (2.8).

Assumption 3. *The step size of the robust GD algorithm in Algorithm 2 is fixed across the iterations, i.e., $\gamma(k) = \gamma$, for $k = 1, 2, \ldots, K$.*

A large value of r ensures that the first assumption is satisfied. However, if any prior knowledge about the true parameters is known, the parameter r can accordingly take smaller values. The second assumption is a guideline on how to choose the step size of the robust GD algorithm for the analysis. Under the above assumptions, the error bound for the robust GD algorithm is as follows.

Proposition 2.2. *Consider the robust GD algorithm in Algorithm 2 based on the median filter, which solves Eq.* (2.2) *with input as $\mathcal{X}^+$. Let $p^+ \in (1/2, 1]$ be the probability that a given sample in $\mathcal{X}^+$ follows the true distribution $\mathcal{N}^+(\boldsymbol{v}_2^*/\boldsymbol{v}_1^*, 1/\boldsymbol{v}_1^*)$. Assume that there exist $\epsilon \in (0,1)$, $\delta \in (0, 1/2)$, and $\zeta \in (0, 1-2\delta)$ such that the batch size N_B satisfies*

$$N_B = \tilde{O}\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right) \text{ and } N_B \leq \frac{1}{\log(1/p^+)}\log\frac{2(1-\delta)}{1+\zeta}. \tag{2.11}$$

Then, under Assumptions 1,2,3, the output $\boldsymbol{v}(K)$ of our algorithm satisfies $\|\boldsymbol{v}(K) - \boldsymbol{v}^*\| \leq \epsilon$, with probability at least $1-\delta$ if $|\mathcal{X}^+| = \Omega\left(\frac{N_B}{\zeta^2}\log\frac{1}{\delta}\right)$ and $K = \Omega(\log\frac{1}{\epsilon})$. Here, all the order constants, the step size $\gamma = 1/L$ in Assumption 3, and the linear convergence rate $\frac{L}{\eta+L} < 1$ depend only on the projection parameter r. Also, $\eta, L > 0$ are defined in Proposition 2.1.

PROOF: See Appendix A.3. ■

The above result indicates the role and suitable choices of the parameters: K, $\gamma(k)$, r, and N_B as discussed next. The result states the number of iterations K scales logarithmically with the inverse of the error ϵ. Also, Assumption 3 shows that the result holds when the step size is the same across all the iterations. Finally, r should be large enough to satisfy Assumption 2. However, a large r leads to slower convergence because the rate of convergence is an increasing function of r. Finally, the algorithm gives an upper and lower bound on the batch size N_B. We note that for GD, the estimation error depends on the error in the first term of the gradient in Eq. (2.6), which is estimated using the batchwise gradient estimate in Eq. (2.5). The error in the batchwise gradient estimate is contributed by the outliers and the finite sample error (the difference between the sample moments computed using a finite number of samples from a distribution and the true moment of the distribution). With large batch size, the number of batches without any outliers is also small. Since the outliers are drawn from an arbitrary distribution $\mathcal{D}_{\text{out}}$, even if a batch contains one outlier, the error in the batchwise gradient estimate can be large. This observation explains the upper bound on the batch size which depends on p^+. It is important to note that when there are no outliers (i.e., $p^+ = 1$), there is no upper bound on the batch size. Similarly, if the batch size is small, the batchwise gradient computed using batches without outliers incurs a large finite sample error. This observation intuitively explains the lower bound on the batch size which is independent of p^+. In short, the upper and lower bounds on N_B balances the tradeoff between the error due to the outliers and the finite sample error. Further, we note that the upper and lower bounds can be simultaneously achieved by choosing ϵ to be large enough.

We also note the restriction on p^+ which is not surprising. This is because the median-based methods work only if the number of outliers are smaller than that of the uncorrupted data samples, which naturally restricts the probability of outliers.

We next present our main theorem that discusses the overall complexity of our algorithm.

Theorem 2.1. *Consider the learning algorithm in Algorithm 1 that uses the median-based robust GD. Let $p \in (2/3, 1]$ be the probability that a given sample follows the true distribution $\mathcal{D}\left(\boldsymbol{W}\boldsymbol{W}^\mathsf{T}, \boldsymbol{b}\right)$. Assume that there exist $\epsilon \in (0,1)$, $\delta \in (0, 1/2)$, and $\zeta \in (0, 1-2\delta)$ such that the batch size N_B of Algorithm 2 satisfies*

$$N_B = \tilde{O}\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right) \text{ and } N_B \leq \frac{1}{\log(2/p)}\log\frac{2(1-\delta)}{1+\zeta}. \tag{2.12}$$

Then, under Assumptions 1,2,3, the outputs $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{b}}$ of Algorithm 1 satisfy

$$\left\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{W}\boldsymbol{W}^\mathsf{T}\right\| \leq [\epsilon + (1-p)] \left\|\boldsymbol{W}\right\|^2 \text{ and } \left\|\hat{\boldsymbol{b}} - \boldsymbol{b}\right\| \leq \epsilon \left\|\boldsymbol{W}\right\|, \tag{2.13}$$

with probability at least $1-\delta$ if the number of samples $N = \tilde{O}\left(\frac{1}{p}\left[\frac{1}{p} + \frac{1}{\zeta^2\epsilon^2}\right]\log\frac{d}{\delta}\right)$. The algorithm runs in time $\tilde{O}\left(\frac{d^2}{p}\left[\frac{1}{p} + \frac{1}{\zeta^2\epsilon^2}\right]\log\frac{d}{\delta}\right)$ and space $\tilde{O}\left(\frac{d}{p}\left[\frac{1}{p} + \frac{1}{\zeta^2\epsilon^2}\right]\log\frac{d}{\delta} + d^2\right)$. All the order constants and the step size γ in Assumption3 depend only on the algorithm parameter r.

PROOF: See Appendix A.4. ■

With no outliers ($p = 1$), our result is the same as the existing error bounds from [102, Theorem

1]. Specifically, for SGD in [102] with no outliers, the number of samples $N = \tilde{O}\left(\frac{1}{\epsilon^2}\log\frac{d}{\delta}\right)$ is sufficient to achieve

$$\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}\| \leq \epsilon\|\boldsymbol{W}\|^2 \text{ and } \|\hat{\boldsymbol{b}} - \boldsymbol{b}\| \leq \epsilon\|\boldsymbol{W}\|. \tag{2.14}$$

with probability at least $1-\delta$ for any $\epsilon, \delta \in (0,1)$, In our case, when $p = 1$, there is no upper bound on N_B and we choose $N_B = N = \Omega\left(\frac{1}{\epsilon^2}\log^3\frac{d}{\delta}\right)$ to achieve Eq. (2.14) with probability $1-\delta$. Thus, the time and space complexities of our algorithm are identical to those of SGD in [102]. We next bound the total variation distance between the estimated distribution and the true distribution under the restriction that $\boldsymbol{W}$ is a full-rank square matrix.

Corollary 2.1. *Consider the learning algorithm in Algorithm 1 that uses the median-based robust GD. Suppose that $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ is full-rank with $d > 1$ and let κ be the condition number of $\boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}$. Let $p > 1 - \frac{1}{2\kappa d}$ be the probability that a given sample follows the true distribution $\mathcal{D}\left(\boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}, \boldsymbol{b}\right)$. Assume that there exist $\epsilon \in (\kappa d(1-p), 1/2]$, $\delta \in (0, 1/2)$, and $\zeta \in (0, 1-2\delta)$ such that the batch size N_B of Algorithm 2 satisfies $N_B = \tilde{O}\left(\frac{\kappa^2 d^2}{(\epsilon-\kappa d(1-p))^2}\log\frac{1}{\delta}\right)$ and $N_B \leq \frac{1}{\log(2/p)}\log\frac{2(1-\delta)}{1+\zeta}$. Then, under Assumptions 1,2,3, the outputs $\hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{b}}$ of Algorithm 1 satisfy*

$$\mathrm{TV}\left(\mathcal{D}\left(\hat{\boldsymbol{\Sigma}}^{1/2}, \hat{\boldsymbol{b}}\right), \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right)\right) \leq \epsilon, \tag{2.15}$$

with probability at least $1-\delta$ if the number of samples $N = \tilde{O}\left(\frac{1}{p}\left[\frac{1}{p} + \frac{\kappa^2 d^2}{\zeta^2(\epsilon-\kappa d(1-p))^2}\right]\log\frac{d}{\delta}\right)$. Here, $\mathrm{TV}(\cdot)$ denotes the total variation distance between the argument distributions. All the order constants and the step size γ in Assumption 3 depend only on the algorithm parameter r.

PROOF: See Appendix A.5. ∎

We note that when $d > 1$, we have $1 - \frac{1}{2\kappa d} \geq 1/4$ and the bound on p in Theorem 2.1 is automatically satisfied.

The last result of this section gives a lower bound on sample complexity of the problem of learning ReLU NN. We restrict the analysis to a specific class of ReLU distributions where $\boldsymbol{W}$ is a scaled identity matrix.

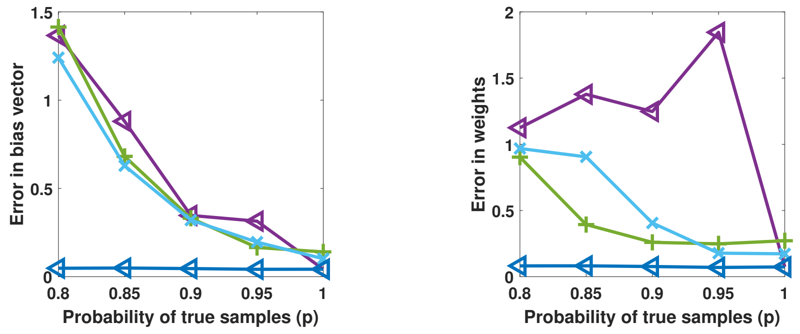
Theorem 2.2. *Consider the ReLU parameter estimation problem with p as the probability that a given sample follows the true distribution. Suppose that the true distribution belongs to $\mathcal{C} = \{\mathcal{D}(\boldsymbol{W}, \boldsymbol{b}) : \boldsymbol{W} = \sigma \boldsymbol{I}, \boldsymbol{b} \in \mathbb{R}^d, \boldsymbol{b}_i > 0 \forall i\}$, where $\sigma = O(1)$. Then, any algorithm that learns $\mathcal{C}$ to satisfy $\left\|\hat{\boldsymbol{b}} - \boldsymbol{b}\right\| \leq \epsilon \left\|\boldsymbol{W}\right\|$ with success probability at least 2/3 requires $\Omega\left(\frac{1}{p\epsilon^2}\right)$ samples.*

PROOF: See Appendix A.6. ■

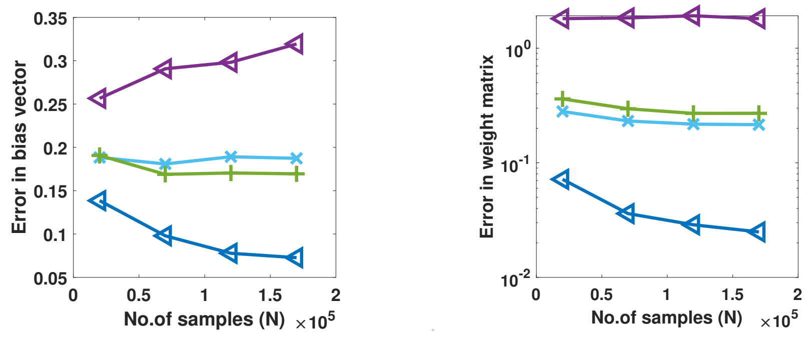
Comparing the sample complexity achieved by our algorithm (Theorem 2.1) and the above lower bound, we can see that the second term of our sample complexity matches the derived bound up to log factors. However, there is a gap between the sample complexity of our algorithm and the lower bound due to the first term that varies as $1/p^2$ (ignoring the log factors). This is an interesting direction for future work to see if there are better bounds.

2.5 Simulation Results

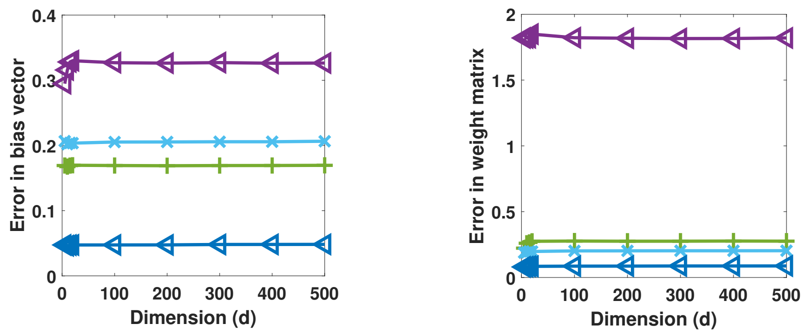
In this section, we provide numerical results to verify the performance of our algorithm. In our simulation setup, the columns of $\boldsymbol{W}$ are chosen as the left singular vectors of random matrices from the standard Gaussian distribution. For $\boldsymbol{b}$, we use a random vector from the standard normal distribution whose negative values are replaced with zeros. The mixture of samples are generated such that a sample comes from $\mathcal{D}(\boldsymbol{W}, \boldsymbol{b})$ with probability p and from $\mathcal{D}_{\text{out}}$ with probability $1-p$. The outlier distribution $\mathcal{D}_{\text{out}} = \mathcal{N}(5, 1)$ and the algorithm hyper-parameters are $r = 3$ and $\gamma(k) = \frac{1}{0.1k}$. We use the batch-splitting approach to compute the gradient, inducing randomization. Also, from our experiments, we observe that the errors flatten certain number of iterations (see Figs. A.1 and A.2) around $1/100$-th of the number of positive output samples which is chosen as the number of GD and SGD iterations K. We compute two error metrics from the estimated parameters and the ground truth, $\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{W}\boldsymbol{W}^{\mathsf{T}}\|_F / \|\boldsymbol{W}\|_F^2$ and $\|\hat{\boldsymbol{b}} - \boldsymbol{b}\|_2 / \|\boldsymbol{W}\|_F$. Further, we compare our algorithm with two other schemes: the oracle schemes (estimation using the true samples only) and schemes without a filter. Our results are shown in Figs. 2.1, 2.2, and Fig. 2.1, and the observations from them are as follows.



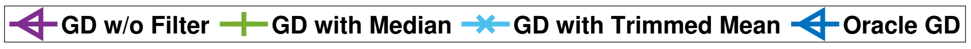
(a) $d = 5$ and $N = 20000$.



(b) $p = 0.95$ and $d = 5$.



(c) $p = 0.95$ and $N = 20000$.



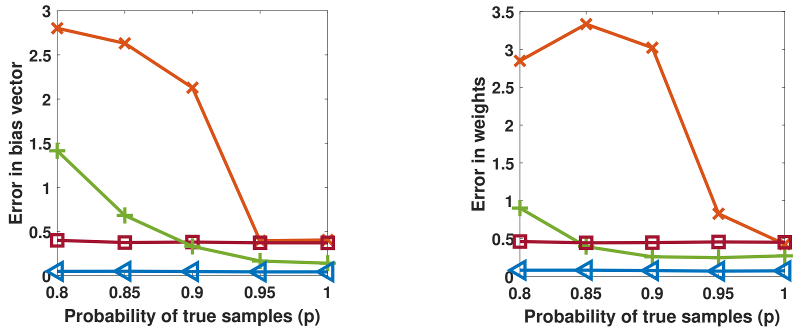
(d)

Fig. 2.1: Comparison of the different GD schemes as a function of p (first row), N (second row), and d (third row).

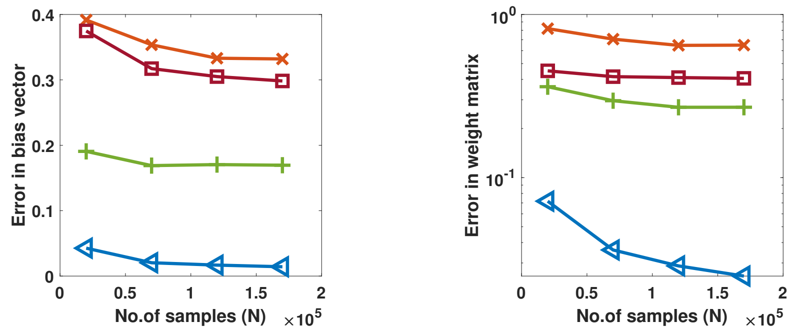
Effect of the filters: From Fig. 2.2, the GD schemes perform better than the corresponding SGD schemes. Also from Figs. 2.1 and 2.2, we infer that using filters along with GD or SGD reduces the effect of the outliers, and the curves are closer to the oracle schemes. We also infer that the median-based approach performs slightly better than the trimmed mean-based approach.

Dependence on the probability of a sample being uncorrupted p: From Figs. 2.1(a) and 2.2(a), the performance of all the schemes except the oracle schemes improves with p because the fraction of outliers in the observed samples decreases with increasing p. However, the schemes without filters show a considerable difference in performance as they are not able to handle the outliers effectively. The performance of oracle schemes does not change with p as they assume the knowledge of true samples. Further, all the schemes converge to the corresponding oracle schemes when $p = 1$.

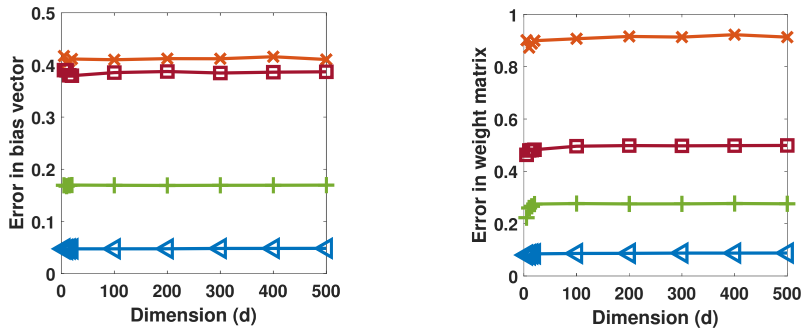
Dependence on the number of samples N and dimension d: Figs. 2.1(b) and 2.2(b) show that the estimation performance of the oracle schemes and the schemes with the filter improves with the number of samples N. However, the schemes without a filter do not always improve with N because the number of outliers also increases with N, which are not handled by the algorithm. In Figs. 2.1(c) and 2.2(c), we varied the dimensions up to 500 and observed that there is a slight increase in the errors as the dimension increases for our proposed schemes as well as oracle schemes. We also observe that the errors increase as d increases for GD without filter due to the presence of arbitrary outliers.



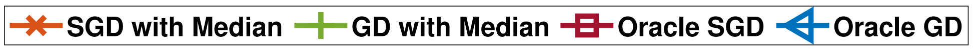
(a) $d = 5$ and $N = 20000$.



(b) $p = 0.95$ and $d = 5$.



(c) $p = 0.95$ and $N = 20000$.



(d)

Fig. 2.2: Comparison of GD and SGD schemes as a function of p (first row), N (second row), and d (third row).

Comparison of runtimes: From Table 2.1, the SGD schemes run faster as SGD utilizes only one sample for the gradient, whereas GD utilizes all the samples. The filter-based schemes have higher computation times than the ones without filters, and the runtimes of the trimmed mean-based schemes are significantly higher than those of the median-based schemes.

Table 2.1: Runtime of various schemes when $p = 0.95$, $N = 20000$, and $d = 5$.

Scheme	**Oracle**	**Without Filter**	**With Median**	**With Trimmed Mean**
GD	16.95 s	17.73 s	34.44 s	60.78 s
SGD	1.24 s	1.60 s	2.11 s	3.62 s

Overall, the median-based GD algorithm is the most effective approach to estimating the NN parameters in the presence of the outliers and is faster than the trimmed mean-based GD algorithm. Also, the median-based scheme is parameter-free and enjoys solid theoretical guarantees.

2.6 Summary

In this chapter, we proposed an algorithm for the estimation of the parameters of a single-layer ReLU neural network from the truncated Gaussian samples where each sample was assumed to be an arbitrary outlier with a fixed probability. Our only assumption was that the bias vector was non-negative. We analyzed the sample and time complexities of the GD-based estimation algorithm combined with median-based filter to handle the outliers. The efficacy of our approach was also demonstrated using numerical experiments.

Chapter 3
Byzantine Resilient Non-Convex SCSG with Distributed Batch Gradient Computations

In this chapter, we propose a robust algorithm based on variance-reduction technique to solve the distributed stochastic optimization problem in the presence of adversarial nodes. We first introduce the assumptions and the definitions required to design the robust algorithm. Next, we describe the steps involved in designing the algorithm along with the description of the crucial *vector median* filter. We then prove the convergence of the proposed algorithm to a first-order stationary point. We also show that the convergence does not depend on the dimension of the problem due to the novel filtering technique. Finally, we present the simulation results and conclude the chapter.

3.1 Introduction

With rapid growth of data centric applications, current machine learning algorithms have to consistently deal with very large size datasets [76]. The use of distributed systems to solve large-scale problems has increased the need for distributed learning architectures to speed up such machine learning algorithms [96]. A distributed learning architecture consists of a central node (CN) and multiple worker nodes (WNs) that jointly perform the learning task [83, 84]. In a distributed network, the computational load at the CN is distributed among the WNs by allowing the WNs to perform the heavy computations at their end [31, 54, 86, 117]. Due to the distributed nature, the network is vulnerable to adversarial attacks on some WNs (*attack on nodes*), also known as Byzantines [69]. Note that the Byzantines may transmit arbitrary values and can adversely affect the convergence performance of the learning algorithm. Hence, robust variants of distributed learning algorithms and sufficient convergence guarantees are warranted. In this chapter, we propose one such robust variant of a distributed learning algorithm where α-fraction of WNs in the network are Byzantines with $\alpha \in [0, 1/2)$.

In [1, 9, 24, 86, 91, 105, 106, 111, 113], the problem of distributed learning in the presence of Byzantines was considered. However, only in [9, 105, 106, 111, 113], the objective function was considered to be nonconvex. In this chapter, we consider a nonconvex optimization problem and develop robust first order algorithms in the presence of Byzantines. Gradient descent (GD) based first order algorithms require computations of gradients of all the available samples from the dataset which is prohibitive for large datasets. To overcome this limitation, stochastic gradient descent (SGD) has been proposed where only a small batch of data samples are used to compute the gradients. Note that the batch size can be as small as one. However, due to the stochasticity, variance of the gradients can be large which leads to slow convergence of SGD. Therefore, variance reduced algorithms have been proposed which lead to improved convergence [63, 71, 87]. In the literature, a number of robust variants of GD and SGD algorithms have been explored which are discussed below.

3.1.1 Related Work

Gradient Descent: In [24], a robust variant of the GD algorithm was proposed to minimize a strongly convex objective function. The authors employed robust mean estimation to counter the presence of Byzantines and provide convergence to a point in the neighborhood of the optimum.

The size of the neighborhood depends on the problem dimension and the number of Byzantines. The analysis was extended in [91] for the case when the number of sample functions available at the WNs is much smaller than the problem dimension. In [111, 113], Byzantine resilient GD algorithms for nonconvex objective function minimization were proposed. In [113], the authors proposed median and trimmed mean based filters for aggregating the gradients sent by the WNs. In [111], the aggregation rules in [113] were used to propose a robust algorithm which escapes from saddle points and incorrect local minima. The authors showed convergence to an approximate local minimum with low iteration complexity.

Moreover, in [9], the authors constructed a distance based filtering rule to identify the Byzantines. This filtering rule was used to propose a robust GD algorithm for nonconvex objective functions. In [105], the authors used coordinate-wise trimmed mean to remove the Byzantines. They showed neighborhood convergence to a local minima for nonconvex objective functions using the GD algorithm. Neighborhood convergence to a local minima for the nonconvex objective functions using the GD algorithm was also given in [106]. However, a score was computed based on the estimated descent of the loss function, and the magnitude of the iterate update. The gradients of the WNs with the highest scores were aggregated to obtain the descent direction for the iterate update. Another line of work [109, 110], considered the decentralized architecture without a CN in the presence of Byzantines. Unlike [9, 91, 105, 106, 109–111, 113], our approach develops a robust variant of the variance reduced algorithm in the presence of Byzantines.

Stochastic Gradient Descent: Furthermore, in [1, 9, 73, 105, 106], the authors proposed robust SGD algorithms in the presence of Byzantines. Specifically, in [1, 73], the authors considered strongly convex and convex objective functions, respectively. In [73], a regularization term was added to the objective function to robustify the algorithm from adversarial attacks. The proposed algorithm was based on the robust mean aggregation of iterate values sent by the WNs. The convergence depends on the problem dimension. The authors showed neighborhood convergence for strongly convex objective functions. In [1], the Byzantines were filtered by computing the *vector median* of the stochastic gradients sent by the WNs for convex objective functions. In this chapter, we utilize the idea of *vector median* to construct our novel filtering rule.

In [104], the geometric median based aggregation rule for the SGD algorithm was provided. The geometric median involves finding a vector that minimizes the sum of the distances to the set of stochastic gradients received in an iteration from all the WNs. However, the convergence rates for nonconvex objective functions were not provided. Furthermore, in [108], the authors proposed a coordinate-wise median based aggregation rule for the distributed SGD algorithm. Although the convergence analysis for nonconvex objective functions was provided, the convergence rate and the explicit complexity bounds were not provided. More recently, in [2, 15], the authors provided convergence rate results for nonconvex objective functions using the SGD algorithm in the presence of Byzantines.

Unlike [1, 2, 9, 15, 73, 104–106, 108] that used SGD based algorithms which suffer from high variance introduced by the stochasticity of the gradients, the stochastic variance-reduced algorithms utilize variance reduction techniques to iteratively reduce variance. The authors in [103] considered one such algorithm called SAGA. The authors assumed that the objective function is strongly convex. The geometric median was considered to mitigate the effect of the Byzantines in the system. The use of geometric median along with SAGA algorithm resulted in the convergence to the neighborhood of a stationary point even though the objective function was assumed to be strongly convex. Furthermore, the authors assumed that the identities of the Byzantines do not vary with time. Also, in [16], the authors provided a robust version of the stochastic variance-reduced gradient (SVRG) algorithm for convex objective functions. However, our algorithm guarantees exact convergence to the stationary point for nonconvex objective functions using the stochastically controlled stochastic gradient (SCSG) algorithm in mean, and also in probability. Further, we show via simulations that our algorithm works well even when the behavioral identities of Byzantines vary with time. Note that the convergence rate in mean is a weaker result as in some instances the algorithm may have the rate that is slower than the rate in mean. However, the convergence rate in probability guarantees that the algorithm always achieves the rate with high probability. Moreover, the convergence rate of our algorithm does not depend on the problem dimension.

Our proposed algorithm is based on a variant of the stochastic variance reduced algorithm called the SCSG algorithm proposed in [71] for nonconvex objective functions. Specifically, the stochastic variance reduced algorithm reduces the variance of stochastic gradients in SGD by adding two more terms to the stochastic gradient. One term is a stochastic gradient and the other one is a batch gradient. The batch gradient is computed by aggregating gradients for a batch of samples once every m steps, and the stochastic gradient is computed at every step. Note that in stochastic variance reduced algorithm, the number of steps m is fixed. However, in SCSG algorithm, the number of steps is given by a geometric random variable (RV).

The following are our major contributions.

3.1.2 Major Contributions

- A novel filtering rule is proposed that is used to identify the honest WNs and prune the Byzantines. It compares the norm of the difference between the batch gradient of a WN and the *vector median* of all the batch gradients to a threshold. This then serves as the basis for the aggregation rule. Note that we design the threshold.

- For the proposed algorithm, we provide the convergence rate guarantees in mean and in probability as a function of $\alpha \in [0, \frac{1}{2})$, the upper bound on the fraction of Byzantines present in the network. Importantly, we also show that with no Byzantines, $\alpha = 0$, the convergence rate of the proposed algorithm is the best known rate for distributed nonconvex optimization (Section 3.4) [61, 114].

- The performance of our algorithm is evaluated via simulation and the results using MNIST and CIFAR10 datasets are presented. We utilize GPU servers to emulate our distributed network with multiple WNs and one GPU server acting as a CN.

Unlike [1, 9, 73, 91, 105, 106, 111, 113] where the authors considered the robust variants of GD and SGD algorithms in the presence of Byzantines, we provide a robust variant of the variance reduced algorithm. This ensures that the proposed algorithm does not have the disadvantages of

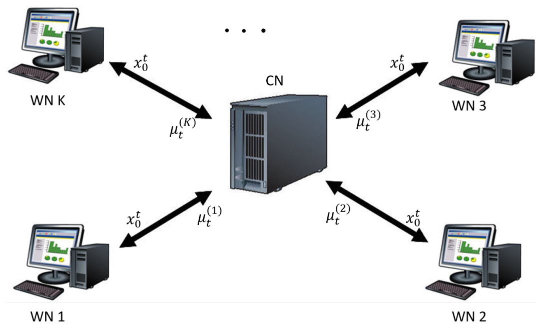


Fig. 3.1: System model illustration with one CN and K WNs.

GD and SGD algorithms like large computational overhead and large variance. Further, unlike [1, 73, 91], we consider nonconvex objective functions which is more attuned to real world problems.

3.2 System Model

Consider the system shown in Fig. 3.1 that has K WNs and a CN. This model is similar to the one in [1]. In this chapter, we focus on distributed algorithms for the empirical risk minimization problems. We have the following setting. We assume that there are K WNs and a CN in the network as illustrated in Fig. 3.1. Each WN has access to one sample indicated by ξ_k, for $k = \{1, \ldots, K\}$. The samples are random variables from an unknown distribution $\mathcal{D}$. Specifically, the goal is to solve the following problem in a distributed setup:

$$\min_{x \in \mathbb{R}^d} f(x) = \mathbb{E}_{\xi \sim \mathcal{D}} f(x; \xi)$$

where the functions $f(\cdot\,; \xi) : \mathbb{R}^d \to \mathbb{R}$, for ξ chosen uniformly randomly from distribution $\mathcal{D}$, and $f(\cdot)$, are assumed to be nonconvex. We assume that all the nodes including CN have access to the stochastic functions from distribution $\mathcal{D}$. We assume that at most α-fraction of WNs are Byzantines, $\alpha \in [0, 1/2)$. CN updates the iterate, $x_{0,t}$ at the beginning of each inner loop and broadcasts it to all WNs in the network. Here, $t \in [T]$ indicates the outer loop (epoch) index of the proposed algorithm. WNs compute the batch gradient, $\mu_t^{(k)}, k \in [K]$ based on the iterate $x_{0,t}$ and transmit the batch gradients to CN.

The set of honest nodes is denoted by $\mathcal{G}$. We assume the following for each honest node $k \in \mathcal{G}$:

Assumption 4 (Gradient Lipschitz continuity, [81]). *All the functions $f(\cdot\,;\xi)$ for any $\xi \sim \mathcal{D}$ and $f(\cdot)$ are assumed to be Lipschitz smooth. A function f is said to be Lipschitz smooth with constant L if its derivatives are Lipschitz continuous with constant L. For any x and y, we have $\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|$, where $L > 0$. Lipschitz Smoothness implies that for any x and y,*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|^2. \tag{3.1}$$

Assumption 5 (Bounded Variance). *For any $\xi \sim \mathcal{D}$ we have $\|\nabla f(x;\xi) - \nabla f(x)\| \leq \mathcal{V}$.*

Remark 3.1. *Assumption 5 is also required in [1] to design the Byzantine filtering strategy. Moreover, for $\alpha = 0$, Assumption 5 can be relaxed to $\mathbb{E}\|\nabla f(x;\xi) - \nabla f(x)\| \leq \mathcal{V}$, which is a standard assumption in stochastic nonconvex optimization.*

In general, for nonconvex optimization problems, it is not feasible to measure the suboptimality of the function value. Therefore, the convergence of the problems is measured in terms of gradient norm square, $\|\nabla f(x)\|^2$. We define an ϵ-stationary point for a nonconvex problem as follows:

Definition 3.1 (ϵ-Stationary Point, [60]). *A point x is called ϵ-stationary if $\|\nabla f(x)\|^2 \leq \epsilon$. Moreover, a stochastic algorithm is said to achieve ϵ-stationarity in t epochs if $\mathbb{E}[\|\nabla f(x_t)\|^2] \leq \epsilon$, where the expectation is over the stochasticity of the algorithm until time instant t.*

3.3 Byzantine SCSG Algorithm

In this section, we present our proposed algorithm and discuss the steps of Algorithm 3. As mentioned earlier, we consider the distributed version of SCSG which is a variant of the stochastic variance reduced algorithm. Here, the WNs compute the batch gradients and share the computed batch gradients with the CN. Note that the algorithm is similar to the original variance reduced algorithm except the fact that the number of inner loop iterations is a geometric RV.

We assume that the algorithm runs for a total of T epochs. At the start of each epoch $t \in [T]$, the CN broadcasts the iterate $x_{0,t}$ to the WNs. The WNs then compute their batch gradients at $x_{0,t}$ and transmit them to the CN. The honest WNs compute and forward their batch gradients, $\frac{1}{B}\sum_{i=1}^{B} \nabla f(x_{0,t}; \xi_{t,i}^{(k)})$, where B indicates the batch size. However, a Byzantine may transmit an arbitrary vector to the CN. The generalization with variable batch sizes at different nodes is straightforward. The Byzantine model where the vector sent by WN k at epoch t to the CN is as follows:

$$\mu_t^{(k)} = \begin{cases} \frac{1}{B}\sum_{i=1}^{B} \nabla f(x_{0,t}; \xi_{t,i}^{(k)}) & \text{if } k \in \mathcal{G} \\ * & \text{if } k \notin \mathcal{G}, \end{cases} \tag{3.2}$$

where $*$ indicates an arbitrary vector sent by the Byzantine.

Next, the CN performs a filtering step after receiving the batch gradients, $\mu_t^{(k)}$, from the WNs. The filtering step consists of the CN computing its estimate of the honest set $\mathcal{G}_t$ at epoch t. Using this estimated set $\mathcal{G}_t$, the CN performs an aggregation of the batch gradients received from the WNs which are present in this estimated set. The aggregation results in the formation of the batch gradient μ_t. Furthermore, the CN runs the inner loop indexed by $n = 1, 2, \ldots, N_t$ of the Byzantine SCSG algorithm, where N_t is chosen randomly using a geometric random variable with parameter $\frac{B}{B+1}$, $N_t \sim \text{Geom}\left(\frac{B}{B+1}\right)$. The CN then performs the update step using the aggregated batch gradient μ_t, and the stochastic gradients computed at $x_{n-1,t}$ and $x_{0,t}$. The Byzantine filtering step is described as follows.

3.3.1 Byzantine Filtering Step

The CN uses the batch gradients sent by the worker nodes, $\{\mu_t^{(k)}\}_{k\in[K]}$, to design the filtering rule. Due to the presence of Byzantines in the network, some of the batch gradients may be faulty or may consist of arbitrary values (see Eq. (3.2)). A key component of the filtering rule is the vector median which is any vector which is close to at least $\frac{K}{2}$ other vectors. Specifically, a vector median

Algorithm 3: Byzantine SCSG with Distributed Batch Gradient Computations

```
Input: $\tilde{x}_0 \in \mathbb{R}^d$, step sizes $(\eta_t)_{t=1}^T$, batch size $B$, Variance Bound $\mathcal{V}$ (Assumption 5),
       $\mathfrak{T}_\mu = 2\mathcal{V}\sqrt{\frac{C}{B}}$ (Lemma A.14) where $C = 2\log\left(\frac{2K}{\delta}\right)$ with $\delta \in (0,1)$ (Theorem 3.1).
1  for t = 1,2, ..., T do
2      $x_{0,t} \leftarrow \tilde{x}_{t-1}$        $\rightarrow$ Push to WNs;
3      for k = 1,2,..., K do
4          Compute $\mu_t^{(k)}$ as in Eq. (3.2)     $\rightarrow$ Push to CN
5      $\mu_t^{\text{med}} \leftarrow \mu_t^{(k)}$ where $k \in [K]$ is any WN such that
         $|\{k' \in [K] : \|\mu_t^{(k')} - \mu_t^{(k)}\| \leq \mathfrak{T}_\mu\}| > K/2$;
6      $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 2\mathfrak{T}_\mu\}$;
7      if $|\mathcal{G}_t| < (1-\alpha)K$ then
8          $\mu_t^{\text{med}} \leftarrow \mu_t^{(k)}$ where $k \in [K]$ is any WN s.t.
             $|\{k' \in [K] : \|\mu_t^{(k')} - \mu_t^{(k)}\| \leq 2\mathcal{V}\}| > K/2$;
9          $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 4\mathcal{V}\}$;
10     $\mu_t = \frac{1}{|\mathcal{G}_t|}\sum_{k\in\mathcal{G}_t}\mu_t^{(k)}$;
11     for $n = 1, 2, \ldots, N_t$, $N_t \sim$ Geom$(\frac{B}{B+1})$ do
12         $v_{n-1,t} = \nabla f(x_{n-1,t};\xi_{n-1,t}) - \nabla f(x_{0,t};\xi_{n-1,t}) + \mu_t$;
13         $x_{n,t} = x_{n-1,t} - \eta_t v_{n-1,t}$;
14     $\tilde{x}_t \leftarrow x_{N_t,t}$;
Output: $\tilde{x}_a$ chosen uniformly randomly from $(\tilde{x}_t)_{t=1}^T$.
```

μ_t^{med} is defined as any vector μ_t^k, for $k \in [K]$ such that $|\{k' \in [K] : \|\mu_t^{(k')} - \mu_t^{(k)}\| \leq \mathfrak{T}_\mu\}| > K/2$; as given in Algorithm 3. The CN uses the vector median to filter out the WNs that it *believes* to be Byzantine and updates the set $\mathcal{G}_t$ as $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 2\mathfrak{T}_\mu\}$.

Note that the above rule is motivated by the fact that for the honest nodes $k \in \mathcal{G}$, the batch gradients will concentrate around the true gradient,$\nabla f(\cdot)$, with high probability. However, the set $\mathcal{G}_t$ can be empty or can have $|\mathcal{G}_t| < (1-\alpha)K$ with non-zero probability. We would want to avoid such scenarios. Therefore, by using the first rule in Algorithm 3, if we obtain $|\mathcal{G}_t|$ less than $(1-\alpha)K$. Then, we increase the threshold from $\mathfrak{T}_\mu$ to $2\mathcal{V}$ in the vector median definition to accommodate more WNs in $\mathcal{G}_t$. In particular, the new vector median definition is the following: $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 4\mathcal{V}\}$. This condition ensures that we always have $|\mathcal{G}_t| \geq (1-\alpha)K$.

Note that the Byzantine filtering rule is similar to that in [1]. However, unlike the filtering rule in [1], we do not maintain a running sum of any statistic to filter Byzantines. Instead, we control the impact of Byzantines by the selection of appropriate batch size. Next, we provide the theoretical guarantees for the proposed algorithm in expectation.

3.4 Convergence Guarantees in Mean

In this section, we present the convergence guarantees in mean as the following:

Theorem 3.1. *Given that Assumptions 4 and 5 are satisfied. For the step size* $\eta_t = \eta = \frac{1}{3LB^{2/3}}$, $B \geq 16$, *and* $\delta \in (0,1)$ *such that* $\mathrm{e}^{\frac{\delta B}{2(1-2\delta)}} \leq \frac{2K}{\delta} \leq \mathrm{e}^{\frac{B}{2}}$, *and* $\delta \leq \frac{1}{25KB}$, *then we have*

$$\begin{aligned}&\mathbb{E}\|\nabla f(\tilde{x}_a)\|^2\\ &\leq \underbrace{\frac{12L\mathbb{E}[f(\tilde{x}_0)-f(\tilde{x}^*)]}{TB^{1/3}}}_{T=O\left(\frac{1}{\epsilon B^{1/3}}\right)} + \underbrace{\frac{32\mathcal{V}^2}{(1-\alpha)^2KB}}_{B=O\left(\frac{1}{\epsilon K}\right)} + \underbrace{\frac{2176\alpha^2\mathcal{V}^2C}{(1-\alpha)^2B}}_{B=O\left(\frac{\alpha^2}{\epsilon}\right)}.\end{aligned} \tag{3.3}$$

PROOF: We relegate the proof to Appendix A.8 ■

Note that when $\alpha = 0$ and $K \leq 1/\epsilon$, our algorithm improves upon the rates achieved by SGD based distributed algorithms [61, 114].

Let $\mathbb{E}G_{\text{comp, SN}}(\epsilon)$ denote the expected number of total gradient computations required at the CN to reach an ϵ-stationary point. Note that the same number of computations are required at individual WNs. From Algorithm 3, we have

$$\mathbb{E}G_{\text{comp, SN}}(\epsilon) = \sum_{t=1}^{T}(B + \mathbb{E}[N_t]) = 2TB, \tag{3.4}$$

where $N_t \sim$ Geom $\left(\frac{B}{B+1}\right)$. Here, T is the total number of iterations required to reach an ϵ-stationary point. Using the above, $\mathbb{E}G_{\text{comp, SN}}(\epsilon)$ can be computed as:

Corollary 3.1. *Under the assumptions as stated in Theorem 3.1, we have the following:*
(i) For $\alpha \in (0, 1/2)$, $\mathbb{E}G_{\textit{comp, SN}}(\epsilon) \leq \tilde{O}\left(\frac{1}{\epsilon^{5/3}K^{2/3}} + \frac{\alpha^{4/3}}{\epsilon^{5/3}}\right)$, *where* $\tilde{O}(\cdot)$ *hides the logarithmic*

factors.

(ii) For $\alpha = 0$, $\mathbb{E}G_{\textit{comp, SN}}(\epsilon) \leq O\left(\frac{1}{\epsilon^{5/3}K^{2/3}}\right)$.

PROOF: We present the proof in Appendix A.10 ■

Note that when Algorithm 3 is run only at the CN, we achieve $\mathbb{E}G_{\text{comp, SN}}(\epsilon) \leq O\left(\frac{1}{\epsilon^{5/3}}\right)$ which is the same as computed in [71].

Remark 3.2. *The convergence rate in mean is a weaker result as it does not guarantee that the algorithm achieves the convergence rate in all instances. A stronger result is to guarantee that the algorithm is able to always achieve the convergence rate with high probability. We discuss this stronger result in the next section.*

3.5 Convergence Guarantees in Probability

In this section, we present stronger convergence guarantees. Specifically, we show that with probability $1-\delta$, our proposed algorithm converges to an ϵ-stationary point with a rate given in Eq. (3.5), where $\delta \in (0, 1)$. We present the modified algorithm for convergence in probability results as follows.

Note that the check in step 7 in Algorithm 3, $|\mathcal{G}_t| < (1-\alpha)K$ is not needed as we know with high probability $|\mathcal{G}_t| \geq (1-\alpha)K$ holds. Therefore, we do not require the update step 8, $\mu_t^{\text{med}} \leftarrow \mu_t^{(k)}$ where $k \in [K]$ is any WN s.t. $|\{k' \in [K] : \|\mu_t^{(k')} - \mu_t^{(k)}\| \leq 2\mathcal{V}\}| > K/2$ and step 9, $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 4\mathcal{V}\}$. The convergence rate in probability is presented in the following theorem.

Theorem 3.2. *Given that Assumptions 4 and 5 are satisfied as in the previous theorem, for the step size* $\eta_t = \eta = \frac{1}{3LB^{2/3}}$, $B \geq 16$, *and* $\delta \in (0, 1)$ *then with probability at least* $1-\delta$, $\overline{x} = \frac{x_1 + \ldots + x_T}{T}$,

Algorithm 4: Byzantine SCSG with Distributed Batch Gradient Computations in Probability

```
Input: $\tilde{x}_0 \in \mathbb{R}^d$, step sizes $(\eta_t)_{t=1}^T$, batch size $B$, Variance Bound $\mathcal{V}$ (Assumption 5),
       $\mathfrak{T}_\mu = 2\mathcal{V}\sqrt{\frac{C}{B}}$ (Lemma A.14) where $C = 2\log\left(\frac{2K}{\delta}\right)$ with $\delta \in (0,1)$ (Theorem 3.2).
1  for t = 1,2, ..., T do
2      $x_{0,t} \leftarrow \tilde{x}_{t-1}$    $\rightarrow$ Push to WNs;
3      for k = 1,2,..., K do
4          Compute $\mu_t^{(k)}$ as in Eq. (3.2)    $\rightarrow$ Push to CN
5      $\mu_t^{\text{med}} \leftarrow \mu_t^{(k)}$ where $k \in [K]$ is any WN such that
         $|\{k' \in [K] : \|\mu_t^{(k')} - \mu_t^{(k)}\| \leq \mathfrak{T}_\mu\}| > K/2$;
6      $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 2\mathfrak{T}_\mu\}$;
7      $\mu_t = \frac{1}{|\mathcal{G}_t|}\sum_{k\in\mathcal{G}_t} \mu_t^{(k)}$;
8      for $n = 1, 2, \ldots, N_t$, $N_t \sim Geom(\frac{B}{B+1})$ do
9          $v_{n-1,t} = \nabla f(x_{n-1,t}; \xi_{n-1,t}) - \nabla f(x_{0,t}; \xi_{n-1,t}) + \mu_t$;
10         $x_{n,t} = x_{n-1,t} - \eta_t v_{n-1,t}$;
11     $\tilde{x}_t \leftarrow x_{N_t,t}$;
Output: $\tilde{x}_a$ chosen uniformly randomly from $(\tilde{x}_t)_{t=1}^T$.
```

we have

$$\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_a)\|^2 \leq \underbrace{\frac{12L\tilde{\mathbb{E}}[f(\tilde{x}_0)-f(\tilde{x}^*)]}{TB^{1/3}}}_{T=O\left(\frac{1}{\epsilon B^{1/3}}\right)} + \underbrace{\frac{16\mathcal{V}^2}{(1-\alpha)^2 KB}}_{B=O\left(\frac{1}{\epsilon K}\right)} + \underbrace{\frac{800\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B}}_{B=O\left(\frac{\alpha^2}{\epsilon}\right)}, \tag{3.5}$$

where $\tilde{\mathbb{E}}$ indicates the expectation with respect to only $\xi_{1,t}, \ldots, \xi_{n,t}$ and N_t.

PROOF: We relegate the proof to Appendix A.11 ■

Similar to the result in mean, the convergence rate in probability when $\alpha = 0$ and $K \leq 1/\epsilon$, our algorithm improves upon the rates achieved by SGD based distributed algorithms [61, 114]. Furthermore, using the definition in Eq. (3.4), $\mathbb{E}G_{\text{comp, SN}}(\epsilon)$ can be computed as

Corollary 3.2. *Under the assumptions as stated in Theorem 3.2, we have the following:*

(i)For $\alpha \in (0, 1/2)$, $\mathbb{E}G_{comp, SN}(\epsilon) \leq \tilde{O}\left(\frac{1}{\epsilon^{5/3}K^{2/3}} + \frac{\alpha^{4/3}}{\epsilon^{5/3}}\right)$, where $\tilde{O}(\cdot)$ subsumes the log *factors.*

(ii) For $\alpha = 0$, $\mathbb{E}G_{comp, SN}(\epsilon) \leq O\left(\frac{1}{\epsilon^{5/3}K^{2/3}}\right)$.

PROOF: We present the proof in Appendix A.13. ■

Moreover, if Algorithm 4 is run only at the CN, we achieve the similar result of $\mathbb{E}G_{\text{comp, SN}}(\epsilon) \leq O\left(\frac{1}{\epsilon^{5/3}}\right)$.

3.6 Simulation Results

In this section, we present the simulation results which characterize the performance of Algorithm 3, and gain insights into its behavior. We have K WNs and a CN in the network. The numerical results are obtained for 150 epochs. We consider two real world datasets. The handwritten digits dataset called MNIST and the color image dataset called CIFAR10 are considered for performance evaluation. To setup the distributed network, we use 11 GeForce GTX 750 Ti GPUs. One GPU is designated as the CN and the other 10 GPUs operate as WNs. We use Open MPI library to communicate between the CN and the WNs in the network and PyTorch for implementation. We fix the learning rate to be 0.01. We fix the batchsizes as 64 for MNIST and 256 for CIFAR10 datasets. We utilize a two layer convolutional neural network for MNIST dataset and ResNet20 for CIFAR10 dataset.

We split the data so that each WN has a subset of data samples that has less overlap with the subset of data samples from other WNs. For example, for the MNIST dataset, WN 1 receives a subset of samples from classes 0 and 1, WN 2 receives samples from classes 1 and 2, and so on. For the CIFAR10 dataset, WN 1 receives a subset of samples from classes Dog and Cat, WN 2 receives samples from classes Dog and Car, and so on. This split ensures that the data is non-i.i.d. Note that all the plots generated in this paper consider the non-i.i.d. data.

Determination of Parameters $\mathcal{V}$ and M: We use a subset of data to estimate the parameters $\mathcal{V}$ and M as follows. We train a NN using the subset of data to compute the stochastic gradient vector and the full batch gradient by accumulating stochastic gradients over multiple epochs through backpropagation. We compute the norm of the difference between the stochastic gradient and the full batch gradient and observe the values of this norm for multiple epochs. We set the value of $\mathcal{V}$

more than the largest value of the norm of the difference. Similarly, we set the value of M more than the largest value of the norm of the stochastic gradient vector.

Byzantine Attack Model

We consider that a Byzantine WN sends a random valued vector to the CN instead of the full gradient as an honest WN does. Note that for the simulation setup we consider that the random values are generated from a Gaussian distribution with mean 5 and variance 1.

Type of Byzantines

We consider two types of Byzantines. In the static case, the behavioral identities of the Byzantines do not change over time. In the dynamic case, the behavioral identities of the Byzantines vary over time.

3.6.1 Benchmarking Schemes

In the following, we discuss the benchmarking schemes used to compare the performance of our proposed algorithm.

Oracle SCSG: In this scheme, all the WNs are assumed to be honest in the network.

SCSG with Median: In this scheme, the median is computed using the gradient vectors received from the WNs at the CN. For this scheme, the aggregation rule is given by $\mu_t = \text{median}\{\mu_t^k\}_{k=1}^K$.

SCSG with Trimmed Mean: In this scheme, the β-trimmed mean is given by $\mu_t = \frac{1}{(1-2\beta)K}\sum_{k\in\mathcal{U}_t}\mu_k^t$ where $\mathcal{U}_t$ is a subset of $\{\mu_t^k\}_{k=1}^K$ obtained by removing the largest and smallest β-fraction of its elements.

SCSG with Phocas [105]: In this scheme, the aggregation is computed as the average of the first $(1-\beta)K$ nearest gradient vectors to the β-trimmed mean given by $\mu_t = \frac{1}{(1-\beta)K}\sum_{k\in\mathcal{W}_t}\mu_k^t$ where $\mathcal{W}_t$ is a subset of $\{\mu_t^k\}_{k=1}^K$ nearest to the β-trimmed mean.

3.6.2 Performance and Comparison Results

In the following, we summarize the results of our experiments using MNIST and CIFAR10 datasets.

MNIST Dataset

In Fig. 3.2(a), the training accuracy of the proposed algorithm for different values of epochs is compared with the performance of the other filtering schemes as mentioned above. For this result, we assume two Byzantines in the distributed setup. Note that initially the median based scheme dominates. However, as the training progresses our proposed algorithm performs better than Phocas, median, and trimmed mean based filtering schemes. Although there are two Byzantines, the proposed scheme attains a training accuracy of 93.8% after 150 epochs.

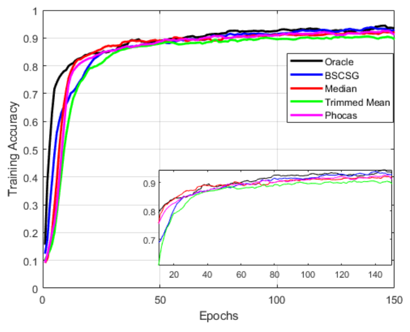
Fig. 3.2(c) benchmarks the test accuracy of the proposed algorithm with other filtering schemes as a function of number of epochs. We observe trends similar to those of the training accuracy. The test accuracy of the proposed algorithm is better than that of Phocas, median, and trimmed mean based filtering schemes. The proposed algorithm achieves a test accuracy of 92.1% in 150 epochs in the presence of two Byzantines.

Effect of Byzantines: We fix the total number of WNs in the system and vary the number of Byzantines under the static and dynamic cases as follows. We vary the number of Byzantines as 2, 4, and 6 to show the effect of the number of Byzantines on the training and test accuracies of our algorithm. Fig. 3.3(a) compares the training accuracy of the proposed algorithm as a function of the number of epochs for different number of Byzantines. In the static case, we observe that as the number of Byzantines increases the training accuracy decreases. In Fig. 3.3(c), the test accuracy of the proposed algorithm for different number of Byzantines is compared. The test accuracy decreases as the number of Byzantines increases as expected. We observe similar trends in Fig. 3.4(a) for the dynamic case as that for the static case for the training accuracy albeit with a further reduction in accuracy values compared to the static case as the identities of the Byzantines vary with time. Furthermore, in Fig. 3.4(c), the test accuracy for different number of Byzantines in the dynamic case is compared. The test accuracy decreases as the number of Byzantines increases as expected. Hence, we observe that the system performance decreases as the number of Byzantines increases.

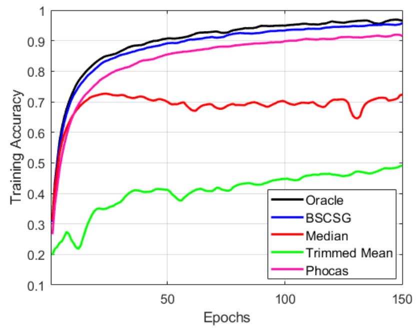
CIFAR Dataset

In Fig. 3.2(b), the training accuracy of the proposed algorithm for different values of epochs is compared with the performance of other filtering schemes as mentioned above for the CIFAR dataset. We assume the number of Byzantines in the distributed setup are two. We observe that our proposed algorithm performs better than Phocas, median, and trimmed mean based filtering schemes. Although there are two Byzantines, the proposed scheme attains a training accuracy of 94% after 150 epochs.

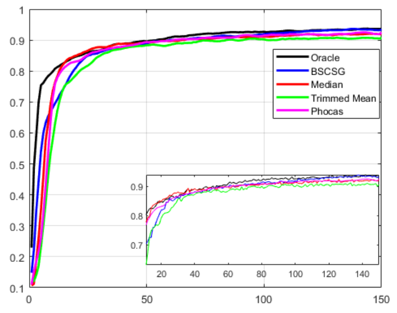
Fig. 3.2(d) benchmarks the test accuracy of the proposed algorithm with other filtering schemes as a function of the number of epochs. We observe trends similar to those of the training accuracy. The test accuracy of the proposed algorithm is better than Phocas, median, and trimmed mean based filtering schemes. The proposed algorithm achieves a test accuracy of 84% in 150 epochs in the presence of Byzantines.



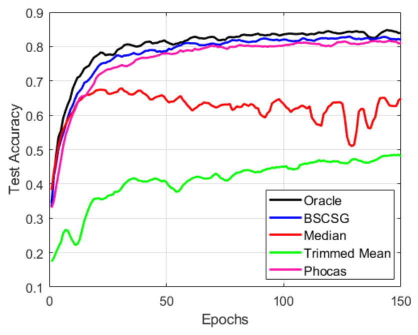
(a) Training accuracy on MNIST.



(b) Training accuracy on CIFAR.



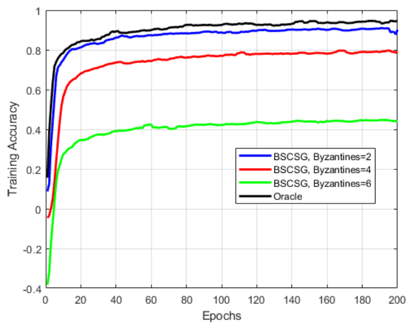
(c) Test accuracy on MNIST.



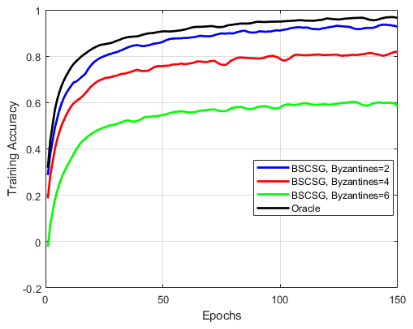
(d) Test accuracy on CIFAR.

Fig. 3.2: Benchmarking of Byzantine SCSG with other filtering schemes and oracle SCSG.

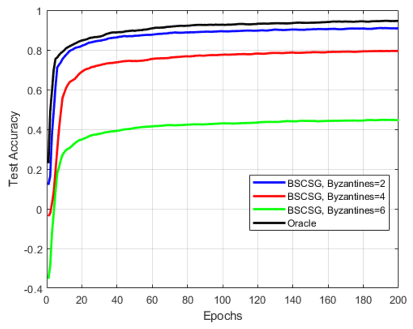
Effect of Byzantines: We fix the total number of WNs in the system and vary the number of Byzantines under the static and dynamic cases as follows. Fig. 3.3(b) compares the training accuracy of the proposed algorithm as a function of the number of epochs for different number of Byzantines in the static case. We observe that the training accuracy decreases as the number of Byzantines increases. In Fig. 3.3(d), we observe that the test accuracy decreases as the number of Byzantines increases for different number of Byzantines as expected for the static case. We observe similar trends in Fig. 3.4(b) for the dynamic case as that in the static case for the training accuracy with reduced accuracy values as the behavioral identities of the Byzantines vary with time. Furthermore, in Fig. 3.4(d), the test accuracy decreases as the number of Byzantines increases in the dynamic case. Hence, we observe that the system performance decreases as the number of Byzantines increases.



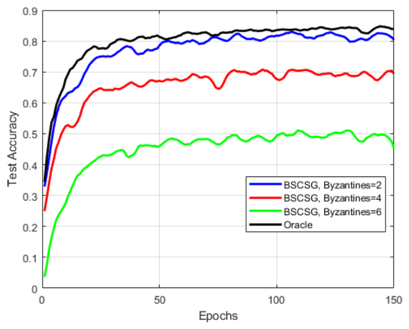
(a) Training accuracy on MNIST.



(b) Training accuracy on CIFAR.

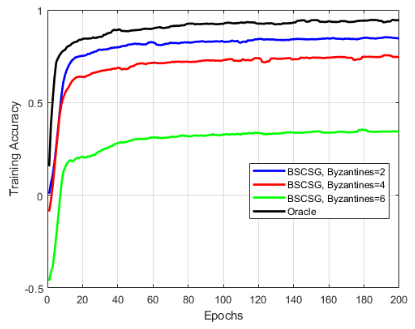


(c) Test accuracy on MNIST.

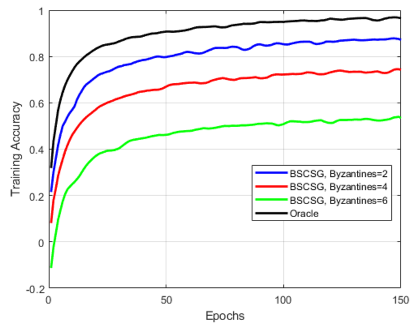


(d) Test accuracy on CIFAR.

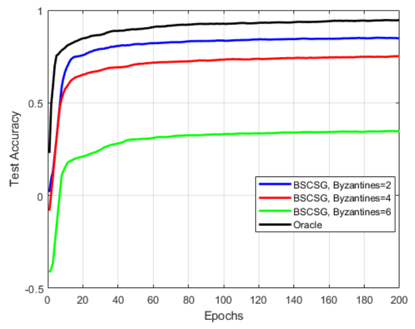
Fig. 3.3: Comparison of Byzantine SCSG by varying number of Byzantines with oracle SCSG in the static scenario.



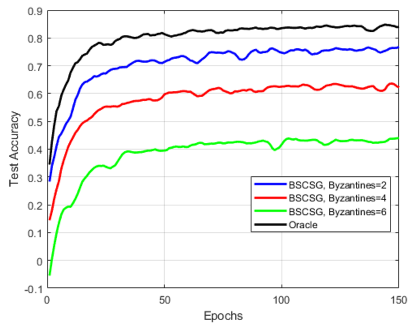
(a) Training accuracy on MNIST.



(b) Training accuracy on CIFAR.



(c) Test accuracy on MNIST.



(d) Test accuracy on CIFAR.

Fig. 3.4: Comparison of Byzantine SCSG by varying number of Byzantines with oracle SCSG in the dynamic scenario.

3.7 Summary

In this chapter, we proposed the Byzantine resilient SCSG algorithm to solve the distributed stochastic nonconvex optimization problem in the presence of Byzantines. The novel filtering rule designed in this work results in the convergence rate being independent of the problem dimension. The effect of Byzantines is captured in the convergence rate results by the presence of an additional term dependent on α, the fraction of Byzantines. We showed that our proposed algorithm outperforms the known convergence rates in the literature in the presence of Byzantines. Further, we provided simulation results using MNIST and CIFAR10 datasets which affirm our theoretical guarantees.

Chapter 4
Robust Distributed Clustering with Redundant Data Assignment

Unlike the previous chapter where the goal was to obtain robust learning algorithms under the supervised learning framework, in this chapter, we propose robust learning algorithms under the unsupervised learning framework. Specifically, we propose robust distributed clustering algorithms that can handle large-scale data across multiple nodes in the presence of faulty nodes. The faulty nodes can either be straggling nodes that fail to respond within a stipulated time or Byzantines that send arbitrary responses. We first introduce the assumptions and the definitions required to design the robust algorithm. Next, the crucial redundant data assignment schemes that enable us to obtain clustering solutions based on the entire dataset even in the presence of stragglers or Byzantines. We then describe the steps involved in designing the algorithm followed by our analyses showing that our proposed algorithms obtain constant factor approximate solution in the presence of stragglers or Byzantines. We also provide various constructions of the data assignment scheme that provide resilience against a large fraction of the faulty nodes. Finally, we present the simulation results and conclude the chapter.

4.1 Introduction

Clustering is one of the basic unsupervised learning tasks used to infer informative patterns in data. The goal of clustering algorithms is to find a subset of data points, called cluster centers, that provide a good representation of the given dataset. The cluster centers provide a partition of the given set of data points that maximize similarity within a group and minimize similarity across the groups. The quality of the clusters is measured using a cost function of which, the k-means and k-median are the most commonly used. The k-median (k-means) clustering problem aims to find a set of k centers that minimize the sum of the distances (sum of the squared distances) of the individual points to their closest cluster center. The cost of clustering using the set of k centers using k-median or k-means is within a constant factor times the cost obtained using the optimal solution [49]. We refer to this as the constant factor approximate solution.

Most widely used centralized clustering algorithms assume that the entire data fits in a single node. However, the centralized clustering algorithms are no longer desirable with the increasing size of the datasets. Hence, there has been a significant interest in designing efficient distributed algorithms for the clustering problem. The goal is to design algorithms that can work with multiple worker nodes having access only to their respective local datasets. Under the data-distributed setup, we assume one central node (CN) and m worker nodes (WNs) such that the dataset P consisting of n data points is partitioned arbitrarily and distributed across the WNs. We denote these partitions by $\{P_1, \ldots, P_m\} \subseteq P$ and assign each of these subsets to a different WN. The individual WNs perform computation on the locally available data points and transmit the obtained results to the CN. The CN then aggregates these results to obtain the final clustering result. Recent works have provided clustering algorithms in such data-distributed setup with provably constant factor approximate solutions [4, 6, 7, 21, 50, 77].

Although the distributed model of computation improves computational efficiency, it makes the system vulnerable to faulty WNs. The faulty WNs may send information with delay, may completely crash, or may send arbitrary (possibly adversarial) information, thereby drastically affecting the quality of the computed solutions. In this chapter, we consider two kinds of faulty

WNs which (i) may send information with delay (or not send anything at all) (*stragglers*), or (ii) may send arbitrary information (*Byzantines*).

Clustering with Stragglers: The stragglers correspond to the WNs that take significantly more time than expected to respond. Several issues could lead to this behavior in the WNs, like power outages, congested communication networks, or software updates running on the WNs. One naïve approach to handling straggling WNs in certain distributed tasks is to ignore them or rely on asynchronous methods. There are established tradeoffs between the loss of information due to ignoring the stragglers and the efficiency of specific tasks such as computing distributed gradients [17, 35, 65, 70, 92, 115]. However, considering the presence of stragglers in distributed clustering has received much less attention.

Clustering with Byzantines: Another challenge in the distributed setup is the presence of adversarial WNs, also known as Byzantines [69]. An adversarial attack usually has the ability to influence the centers in one (or more) of the clusters. Instead of sending the correct result of the computation to the CN, a Byzantine may send arbitrary values. A naïve approach is to rely on simple distributed clustering methods even when Byzantines are present [6, 77]. However, this may lead to extremely poor-quality solutions computed by the distributed clustering algorithm due to the arbitrary information sent by the Byzantines. Another approach is to provide filters to identify and remove the Byzantines in the setup as proposed in the Byzantine machine learning literature [9, 24, 86, 91, 113] (also considered in Chapters 2 and 3).

An alternate solution, that we adopt, is to introduce redundancy in the data distributed to the WNs. This ensures that the information obtained from a subset of WNs is sufficient to compute the desired function on the entire dataset. Multiple coding-based redundant data distribution schemes have been proposed to mitigate the effect of stragglers [46, 70, 85, 92, 99, 100] and Byzantines [28–30, 45] for computing linear functions such as gradient aggregation in first-order optimization methods. However, these techniques do not translate well for clustering tasks where, unlike the prior works, the responses from different WNs may not be related.

In this chapter, we propose a data distribution scheme for distributed clustering problems in the presence of stragglers and Byzantines. The stragglers send the correct information albeit with a delay. Hence, the CN knows the identities of the stragglers. However, in the case of clustering with Byzantines, the formulation deals with a more general scenario where a subset of the WNs are adversarial and can send arbitrary information. Moreover, the identity of these adversarial WNs (Byzantines) is not known to the CN which constitutes the main bottleneck in obtaining Byzantine resilient clustering algorithms. We show that our proposed data distribution scheme allows us to compute provably good-quality cluster centers even in the presence of a relatively large number of stragglers and/or Byzantines[1].

In [14, 43], the assumption was that the WNs had the ability to compute exact solutions to the clustering problem on the local datasets. In this chapter, we consider the case when the WNs can no longer compute the exact solution to the clustering problem. This reduces the computational load at each machine with an increased approximation factor (Sections 4.4 and 4.5).

Moreover, in [14], we assumed that the CN computes the local summaries to evaluate the quality of the data sent by each local machine. Hence, the CN required the access to the entire dataset P and had to estimate the cost of computing a cluster on the local dataset using the summaries sent by each machine. In resource-constrained settings, such assumptions can increase the computational load at the FC. In this chapter, we assume that the CN does not have access to the entire dataset, and hence, can not estimate the cost of computing a cluster on the local dataset P_i using the summaries sent by each machine. Hence, the analyses in [14] can no longer be utilized. The first challenge in this chapter is the computation of coresets by the CN as surrogates of the respective local datasets P_i. These are efficiently computed in a streaming fashion using the sensitivity sampling technique [12]. We utilize these coresets to approximate the cost of clustering using the local datasets P_i at the CN. Another challenge is the filtration step. In [14], the filtering step depended on the cost of clustering computed on the local datasets. However, in this chapter, the filtering is performed utilizing the cost of clustering using the coresets computed by the CN and the local summaries sent by the WNs. Moreover, in [14], the weights for each of the points in the summaries

[1]Unlike the assumption in Chapter 3 where the fraction of Byzantines in the system was assumed to be less than $1/2$.

sent by the WNs were obtained on the respective local datasets P_i. However, crucially in this work, the weights for each of the points in the local summaries sent by the WNs are estimated using the coresets computed by the CN. Therefore, the third challenge is the estimation of these weights. We show that these estimated weights are a constant factor away from their intended value with high probability (Sections 4.5.2 and 4.5.3). Thus, as described next, our work in this chapter is much more general and extends our prior work [14, 43] quite significantly.

4.1.1 Our Results

In this chapter, we provide robust distributed clustering approaches that generate solutions with a cost at most $c \cdot \text{OPT}$, for a small approximation factor $c \geq 1$, where, OPT denotes the cost of the best clustering solution. Our algorithms are resilient to WNs that are either (i) stragglers, or (ii) Byzantines.

We propose a redundant data distribution scheme that allocates a data point to multiple WNs to mitigate the loss of information (or misinformation) that arises due to the presence of some faulty WNs. The following are our major contributions.

- We establish sufficient conditions on the data assignment scheme that enables us to mitigate the effect of stragglers and Byzantines to compute good-quality clusters (Property 1 and Property 2).

- We design robust k-median and k-means clustering approaches that generate a constant factor approximate solution in the presence of stragglers. Our approach also extends to a more general class of squared ℓ_2-fitting problems known as subspace clustering. Theorem 4.2 shows that we can achieve an approximation factor of roughly 3 for k-median clustering. This approach extends naturally to k-means clustering (Theorem 4.4) and gives an approximation factor of almost 10 for k-means. The results for k-means can be improved and generalized to subspace clustering using a slight variant of the above approach which is formalized in Theorem 4.5.

- Byzantines are much harder to handle since their identity is unknown. Using a stronger data assignment scheme compared to its straggler counterpart (Remark 4.1), we obtain k-median (Theorem 4.6) and k-means (Theorem 4.8) approaches that are guaranteed to achieve approximation factors of almost 3 and 10 respectively. We also improve upon the suggested algorithms to make them computationally and storage efficient in Theorems 4.7 and Theorem 4.9.

- Finally, we provide various constructions of the assignment scheme that satisfy the established conditions (Properties 1 and 2) and provide resilience against a large fraction of faulty WNs while incurring little redundancy. The various constructions and the tradeoffs they present are summarized in Table 4.1.

- Simulation results illustrate the excellent performance of our algorithm.

4.2 System Model

Given a dataset with n points $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\} \subseteq \mathbb{R}^d$, distributed among m WNs, the goal in clustering is to find a set of k cluster centers $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\} \subseteq \mathbb{R}^d$ that closely represent the entire dataset. The quality of these centers is usually measured by a cost function cost(P, C). For k-median, the cost function is defined as cost$(P, C) = \sum_{\mathbf{x}\in P} d(\mathbf{x}, C)$, where $d(\mathbf{x}, C) := \min_{\mathbf{c}\in C} d(\mathbf{x}, \mathbf{c})$. The k-means cost function for clustering is given by cost$(P, C) = \sum_{\mathbf{x}\in P} d^2(\mathbf{x}, C)$. If the dataset P is weighted with an associated non-negative weight function $g : P \to \mathbb{R}^+$, the k-median cost for the weighted dataset (P, g) is then defined as cost$(P, g, C) = \sum_{\mathbf{x}\in P} g(\mathbf{x}) d(\mathbf{x}, C)$. The k-means cost for (P, g, C) is defined analogously. Our goal is, therefore, to obtain a set of k centers C that minimizes cost(P, C). For any data point $\mathbf{x} \in P$, and any set of centers C, we denote its cluster center by $C(\mathbf{x}) := \arg\min_{\mathbf{c}\in C} d(\mathbf{x}, \mathbf{c})$. Also, for any point set P, we denote the cluster of P associated with a center $\mathbf{c} \in C$ by cluster$(\mathbf{c}, P) := \{\mathbf{x} \in P | C(\mathbf{x}) = \mathbf{c}\}$.

We consider the data-distributed clustering framework with m WNs $W_1, \ldots, W_m$. Let $P_i \subseteq P$ be the set of points assigned to the WN W_i. To compute the cluster centers in such data-distributed setups, the WNs transmit a summary of their local data to the CN. For the simplicity of presentation, we assumed each WN computes the optimal clustering solution on its assigned data points. Similar to the result of [51], we show that the set of k-centers computed by these WNs summarizes their local dataset. Our results also extend trivially when WNs provide *approximate* clustering solutions as a summary.

In the presence of stragglers, the CN combines the local summaries obtained from non-straggler WNs to obtain the summary of the global dataset which is a constant factor approximate solution. To mitigate the effect of Byzantines, the CN ranks the received local summaries to evaluate the quality of the data summary sent by each local WN. An approximate solution to the clustering problem can then be computed at the CN by aggregating the subset of best summaries.

Problem Statement: In this paper, the main goal is to design data-distributed robust clustering approaches. Specifically, given a dataset P of n points in $\mathbb{R}^d$, and distributed setup with m WNs where at most t WNs are faulty (either stragglers or Byzantines), the goal is to design a clustering approach that generates a solution with the cost at most $c \cdot \text{OPT}$, with a small approximation factor $c \geq 1$ for the k-median and the k-means clustering problems.

Next, we provide some definitions and results that are helpful for the presentation in the rest of this chapter.

4.2.1 Preliminaries

Definition 4.1 ((r, k)-subspace clustering)**.** *Given a dataset $P \subset \mathbb{R}^d$ find a set of k-subspaces (linear or affine) $\mathcal{L} = \{L_i\}_{i=1}^k$, each of dimension r, that minimizes cost$(P, \mathcal{L}) := \sum_{i=1}^n \min_{L \in \mathcal{L}} d^2(\boldsymbol{p}_i, L)$.*

Note that for $r = 0$, this is exactly the k-means problem described above. Another special case, when $k = 1$, is known as principal component analysis (PCA).

For any $\alpha, \beta \geq 1$, we define an α-approximate solution to a clustering problem with cost function defined by cost$(\cdot, \cdot)$ as follows:

Definition 4.2 ((α, β)-approximate solution)**.** *For any $\alpha, \beta > 1$, the set of cluster centers C, $|C| \leq \beta k$, is an (α, β)-approximate solution to the k-center clustering problem if the cost of* *clustering P with C, cost(P, C), is at most α times the cost of clustering with optimal set k-centers, cost$(P, C) \leq \alpha \cdot$ OPT.*

An $(\alpha, 1)$-approximate solution is just called an α-approximate solution.

The quality of the data summaries is captured by the notion of a coreset. Informally, a coreset is a small weighted set of representative points of the dataset that closely approximates the cost of clustering on any set of k centers.

Definition 4.3 (ϵ-coreset)**.** *Let $\epsilon \geq 0$. For a dataset P, an ϵ-coreset with respect to a cost function cost$(\cdot, \cdot)$ is a weighted dataset S with an associated weight function $g : S \to \mathbb{R}^+$ such that, for any set of k centers C, we have*

$$(1 - \epsilon)\textit{cost}(P, C) \leq \textit{cost}(S, g, C) \leq (1 + \epsilon)\textit{cost}(P, C)$$

Using any off-the-shelf α-approximate solution to the clustering problem on an ϵ-coreset of the dataset P yields a good approximate solution on the entire dataset. This fact is formalized by the following Theorem.

Theorem 4.1 ([38])**.** *Let (S, g) be an ϵ-coreset for a dataset P with respect to the cost function* $\mathrm{cost}(\cdot, \cdot)$. *Any α-approximate solution to the clustering problem on input S, is an $\alpha(1 + 3\epsilon)$-approximate solution to the clustering problem on P.*

Next, we present our approach to assign data to different WNs with redundancy.

4.3 Data Assignment

The first step to obtaining robust distributed clustering in the presence of faulty WNs is the initial data assignment to the WNs. Specifically, every data point in the dataset P is mapped to multiple WNs by carefully employing redundancy in the assignment process. Hence, each data point affects the local computation performed on multiple WNs and the final clusters at the CN are obtained by taking into account the contributions of most of the data points in P even though some of the WNs are faulty. We introduce the data assignment scheme along with the resilience properties below. This property enables the aggregation of local computations from the non-straggling or honest WNs at the CN and preserves the relevant information present in the dataset P for the clustering problems. The assignment scheme is utilized to obtain good-quality solutions to k-median and k-means clustering.

4.3.1 Straggler-resilient Data Assignment

Let $A \in \{0,1\}^{m\times n}$ be the binary assignment matrix where the i-th row, $\mathbf{a}_i$, indicates the set $P_i \subseteq P$ of points assigned to WN W_i. Let $\mathcal{R} \subset [m]$ denote the set of non-straggler WNs. We assume that $|\mathcal{R}| \geq m - t$, where $t < m$ denotes an upper bound on the number of stragglers in the system. For any such set of non-straggler WNs $\mathcal{R}$, we require the assignment matrix A to satisfy the following property.

Property 1 ((t,δ)-Straggler resilience property)**.** *Let $\delta > 0$ be a given constant. The assignment matrix $A \in \{0,1\}^{m\times n}$ has (t,δ)-straggler resilience if for every subset of $m-t$ rows $\mathcal{R} \subseteq [m]$, $\exists$ a recovery vector, $\boldsymbol{b} = (b_1, \dots, b_{|\mathcal{R}|})^T \in \mathbb{R}^{|\mathcal{R}|}, b_i > 0, \forall i \in |\mathcal{R}|$, such that*

$$\boldsymbol{1}_n^T \leq \sum_{i\in\mathcal{R}} b_i \boldsymbol{a}_i \leq (1+\delta)\boldsymbol{1}_n^T, \tag{4.1}$$

where $\leq$ indicates coordinate-wise inequality.

We remark that the straggler resilience property is significantly different from that in [92] where the property utilizes the fact that the gradients are related to each other across different WNs. For the straggler resilience property, the recovery vector $\mathbf{b}$ is restricted to be only non-negative.

Utilizing the combinatorial characterization for the assignment scheme given by Property 1, the information received from the non-straggler WNs can be combined to generate close to optimal clustering solutions using the following result.

Lemma 4.1. *Let $P \subset \mathbb{R}^d$ be a dataset distributed across m WNs using a (t,δ)-straggler resilient assignment matrix A that satisfies Property 1. Let $\mathcal{R}$ be the set of $m-t$ non-straggler WNs. For any $\delta > 0$, let $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{R}|}$ be the recovery vector corresponding to $\mathcal{R}$. Then, for any set of k centers $C \subset \mathbb{R}^d$, any weight function $g : P \to \mathbb{R}$,*

$$\mathit{cost}(P,g,C) \leq \sum_{i\in\mathcal{R}} b_i \mathit{cost}(P_i,g,C) \leq (1+\delta)\mathit{cost}(P,g,C).$$

PROOF: See Appendix A.14. ■

4.3.2 Byzantine-resilient Data Assignment

Similar to the straggler resilient data assignment, we propose a modified data assignment to the WNs to mitigate the effect of Byzantines. Let $A \in \{0,1\}^{m\times n}$ denote the binary assignment matrix whose i-th row, $\mathbf{a}_i$, indicates the set $P_i \subseteq P$ of points assigned to WN W_i. Let $\mathcal{R} \subset [m]$ denote the set of honest (non-Byzantine) WNs. We assume that $|\mathcal{R}| \geq m-t$, where $t < m$ denotes an upper bound on the number of Byzantines in the system. For any such set of honest WNs $\mathcal{R}$, we require the assignment matrix A to satisfy the following property.

Property 2 ((t,δ)-Byzantine resilience property)**.** *Let $\delta > 0$ be a given constant. The assignment matrix $A \in \{0,1\}^{m\times n}$ has (t,δ)-Byzantine resilience if $\exists$ a reconstruction coefficient $\rho > 0$, such that for any subset of $m-t$ rows $\mathcal{R} \subseteq [m]$,*

$$\boldsymbol{1}_n^T \leq \rho \sum_{i\in\mathcal{R}} \boldsymbol{a}_i \leq (1+\delta)\boldsymbol{1}_n^T, \tag{4.2}$$

where $\leq$ indicates coordinate-wise inequality.

Remark 4.1. *The Byzantine-resilience property is much stronger than the straggler resilience property introduced in Property 1. For straggler resilience, it is sufficient to have some non-negative linear combination of the rows (corresponding to the non-straggler WNs) that is close to the all-ones vector. However, for Byzantine resilience, we need all these linear combinations to be* *uniform and non-negative. Further, we also need this reconstruction factor to be the same across all subsets of Byzantines.*

The information received from the honest WNs is combined using the following lemma to generate a close-to-optimal clustering solution.

Lemma 4.2. *Let $\mathcal{T} \subseteq [m]$ be any set of $m - t$ indices. Let ρ be the reconstruction coefficient of the (t, δ)-Byzantine resilient assignment matrix. Then, for any set of centers C, we have cost$(P, C) \leq \sum_{i \in \mathcal{T}} \rho$ cost$(P_i, C) \leq (1 + \delta)$cost$(P, C)$.*

PROOF: The proof is analogous to that of Lemma 4.1 and follows based on the combinatorial characterization for the assignment scheme enforced by Property 2. ■

4.4 Straggler Resilient Clustering

In this section, we present straggler resilient clustering techniques using the redundant data distribution scheme described above. In Section 4.4.1, we present the k-median clustering algorithm that is extended in a straightforward manner to the k-means setting in Section 4.4.2. This algorithm is improved in Section 4.4.3, where we present a general algorithm for the (r, k)-subspace clustering.

4.4.1 Straggler-Resilient Distributed k-median Clustering

The dataset P is distributed across m WNs using a (t, δ)-straggler resilient assignment matrix A that satisfies Property 1. Each non-straggling WN sends a set of weighted k-median centers of their local datasets which when aggregated at the CN gives a summary for the entire dataset. Hence, the weighted k-median clustering on this summary at the CN provides a good-quality solution for the entire dataset P. In Algorithm 5, we provide the above-discussed steps in detail.

Algorithm 5: Straggler-resilient distributed k-median clustering

```
Input: A collection of $n$ data points $P \subset \mathbb{R}^d$
1 Allocate $P$ to $m$ WNs according to a $(t, \delta)$-straggler resilient matrix $A$.
2 Let $P_i \subset P$ be the set of points assigned to WN $W_i$.
3 Each WN $W_i$ computes a $k$-median solution, $Y_i$, on set $P_i$.
4 Define $g_i : Y_i \to \mathbb{R}$ as $g_i(\mathbf{y}) = |\text{cluster}(\mathbf{y}, P_i)|$, for every $y \in Y_i$.
5 CN collects $\{(Y_i, g_i)\}_{i \in \mathcal{R}}$ from the non-straggling WNs, for some $\mathcal{R} \subseteq [m], |\mathcal{R}| \geq m - t$.
6 Let $Y = \bigcup_{i \in \mathcal{R}} Y_i$. Using the recovery vector $\mathbf{b}$, define $g : Y \to \mathbb{R}$ such that
    $g(\mathbf{y}) = b_i g_i(\mathbf{y}), \forall \mathbf{y} \in Y_i$ and $i \in \mathcal{R}$[2]
Output: $\hat{C}$, the $k$-median solution on $(Y, g)$.
```

Before we state the theorem that quantifies the quality of the clustering solution $\hat{C}$ provided by Algorithm 5 on the entire dataset P, we present the following lemma where we show that the cost incurred by the weighted dataset Y is close to the cost incurred by P for any set of k centers C, which is necessary to prove the theorem.

Lemma 4.3. *For k-median clustering, for any set of k-centers $C \subset \mathbb{R}^d$, we have*

$$\textit{cost}(P, C) - \sum_{i \in \mathcal{R}} b_i \textit{cost}(P_i, Y_i) \leq \textit{cost}(Y, g, C) \leq 2(1+\delta)\textit{cost}(P, C).$$

PROOF: Presented in Appendix A.15. ■

Theorem 4.2. *Let C^* be the optimal set of k-median centers for dataset P. Then, Algorithm 5 on dataset P returns a set of centers $\hat{C}$ such that cost$(P, \hat{C}) \leq 3(1+\delta)$cost$(P, C^*)$.*

PROOF: Utilizing the lower bound from Lemma 4.3 with $C = \hat{C}$, we have

$$\begin{aligned} \text{cost}(P, \hat{C}) &\leq \text{cost}(Y, g, \hat{C}) + \sum_{i \in \mathcal{R}} b_i \text{cost}(P_i, Y_i) \overset{(a)}{\leq} \text{cost}(Y, g, C^*) + \sum_{i \in \mathcal{R}} b_i \text{cost}(P_i, C^*) \\ &\overset{(b)}{\leq} 2(1+\delta)\text{cost}(P, C^*) + (1+\delta)\text{cost}(P, C^*), \end{aligned} \tag{4.3}$$

[2]In general, if $\mathbf{y} \in Y_{i_1} \cap Y_{i_r} \cap \ldots \cap Y_{i_r}$, then $g(\mathbf{y}) = \sum_{j=1}^{r} b_{i_j} g_{i_j}(\mathbf{y})$.

where (a) follows from the fact that $\hat{C}$ and Y_i are the optimal set of centers for the weighted dataset (Y, g) and the partial dataset P_i, respectively. For (b), we utilize the upper bound in Lemma 4.3 and Lemma 4.1 with $C = C^*$. ■

Note that the summary computed at the CN uses the weighted set (Y, g) which is constructed only from the information sent by the non-straggling WNs. Also, the data assignment scheme initially used to distribute the data satisfies the Property 1. Hence, from Theorem 4.2, we observe that the CN is able to construct a good summary of the entire dataset P despite the presence of the stragglers. Moreover, this summary is sufficient to generate a good quality k-median clustering solution corresponding to P, i.e., the summary generates a constant factor approximate solution.

Remark 4.2. *Suppose the honest WNs and the CN are unable to compute the exact k-median clustering solution as required in Step 3 and the last step of Algorithm 5, but instead produce an α-approximate solution (such as in [18]), then this slight variant of Algorithm 5 returns a set of k-centers $\hat{C}$ such that cost$(P, \hat{C}) \leq \alpha(1+\delta)(2+\alpha)$cost$(P, C^*)$, even in the presence of t stragglers (Theorem 4.3).*

Theorem 4.3. *Let C^* be the optimal set of k-median centers for dataset P. Then, Algorithm 5 on dataset P returns a set of centers $\hat{C}$ such that cost$(P, \hat{C}) \leq \alpha(1+\delta)(2+\alpha)$cost$(P, C^*)$.*

PROOF: See Appendix A.16. ■

4.4.2 Straggler-Resilient Distributed k-means Clustering

Observe that the above-described algorithms for distributed k-median clustering in the presence of stragglers can be generalized to other classical cost functions to yield algorithms such as for the k-means clustering algorithm. The key observation that we use to extend the above-described algorithms is that the distance function $d^2(\cdot, \cdot)$ satisfies a scaled version of the triangle inequality, i.e., for any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^d$,

$$d^2(\mathbf{a}, \mathbf{b}) \leq 2(d^2(\mathbf{a}, \mathbf{c}) + d^2(\mathbf{b}, \mathbf{c})). \tag{4.4}$$

We use a strategy similar to Algorithm 5 to compute the k-means clustering solution in the presence of stragglers. We observe that if each local WN can compute an exact (or approximate) k-means solution on their local datasets, then it can be suitably combined using the recovery vector to obtain a constant factor approximate k-means solution to the global dataset. Algorithm 6, does exactly this, where in Step 3 of Algorithm 5, each WN W_i sends a k-means solution Y_i corresponding to P_i weighted accordingly.

Algorithm 6: Straggler-resilient distributed k-means clustering

```
Input: A collection of $n$ data points $P \subset \mathbb{R}^d$
1 Allocate $P$ to $m$ WNs according to a $(t, \delta)$-straggler resilient matrix $A$.
2 Assign the set of points $P_i \subset P$ to WN $W_i$.
3 Each WN $W_i$ computes $\alpha$-approximate $k$-means solution $Y_i$ on set $P_i$. Let $g_i : Y_i \to \mathbb{R}$ as
    $g_i(\mathbf{y}) = |\text{cluster}(\mathbf{y}, P_i)|$, for every $y \in Y_i$.
4 CN collects $\{(Y_i, g_i)\}_{i\in\mathcal{R}}$ from the non-straggling WNs.
5 Let $Y = \bigcup_{i\in\mathcal{R}} Y_i$. Using the recovery vector b, define $g : Y \to \mathbb{R}$ such that
    $g(\mathbf{y}) = b_i g_i(\mathbf{y}), \forall \mathbf{y} \in Y_i$ and $i \in \mathcal{R}$.
Output: $\hat{C}$, the $\alpha$-approximate $k$-means solution on $(Y, g)$.
```

The performance guarantees of Algorithm 6 can be stated as follows:

Theorem 4.4. *Let C^* be the optimal set of k-means centers for dataset P. Then, Algorithm 6 on dataset P returns a set of centers $\hat{C}$ such that cost$(P, \hat{C}) \leq 2\alpha(3 + 2\alpha)(1 + \delta)$cost$(P, C^*)$.*

PROOF: The proof is very similar to that of Theorem 4.2, and can be found in Appendix A.17. ■

4.4.3 Straggler-Resilient Distributed (r,k)-Subspace Clustering

Note that the approximation factor of over 10 obtained using Algorithm 6 is quite prohibitive. We observe that Algorithm 6 succeeds because the weighted centers (Y_i, g_i) sent by the local WNs W_i are in fact a coreset of the local dataset P_i in a weak sense [3]. We leverage this observation to present

[3]the cost of clustering using the weighted set (Y, g) is close to the cost of clustering using the entire dataset P albeit with an offset (Lemma A.21)

a variant of Algorithm 6 that is computationally more efficient and also guarantees an improved approximation factor. In Algorithm 7, each WN sends a δ-coreset of its local dataset instead of an exact k-means solution. We now show that this small change can yield a better approximation factor and more general results.

In this section, we present a straggler resilient algorithm for a general class of squared ℓ_2 fitting problems, known as (r, k) subspace clustering problems where the goal is to find k subspaces each of dimension at most r that best fit the data. Note that the subspace clustering problem covers both the k-means and the principal component analysis (PCA) problems as special cases.

Algorithm 7: Straggler-resilient distributed (r, k)-subspace clustering

```
  Input: A collection of $n$ data points $P \subset \mathbb{R}^d$.
1 Allocate $P$ to $m$ WNs according to a $(t, \delta)$-straggler resilient matrix $A$.
2 Assign the set of points $P_i \subset P$ to WN $W_i$
3 Each WN $W_i$ computes $\delta$-coreset $(Y_i, g_i)$ of $P_i$.
4 CN Collects $\{(Y_i, g_i)\}_{i\in\mathcal{R}}$ from the non-straggling WNs
5 Let $Y = \bigcup_{i\in\mathcal{R}} Y_i$. Using the recovery vector $\mathbf{b}$, define $g : Y \to \mathbb{R}$ such that
  $g(\mathbf{y}) = b_i g_i(\mathbf{y}), \forall \mathbf{y} \in Y_i$ and $i \in \mathcal{R}$
  Output: $\hat{C}$, the set of $r$-subspaces that is an $\alpha$-approximate solution to the
          $(r, k)$-subspace clustering on input $(Y, g)$.
```

In the following lemma, we show that the cost incurred by the aggregated weighted dataset (Y, g) is close to the cost incurred by P for any set of k centers C. In other words, (Y, g) is a coreset of P.

Lemma 4.4. *Let $\delta \in (0, 1)$. For any set of k-centers $C \subset \mathbb{R}^d$, we have*

$$(1-\delta)\mathit{cost}(P, C) \leq \mathit{cost}(Y, g, C) \leq (1+3\delta)\mathit{cost}(P, C).$$

PROOF: The proof is relegated to Appendix A.18. ■

The following result quantifies the quality of the clustering solution $\hat{C}$ provided by Algorithm 7 on the entire dataset P.

Theorem 4.5. *Let $\delta \in (0,1)$. Let C^* be the optimal solution for (r,k)-subspace clustering on dataset P. Then, Algorithm 7 on dataset P returns a set of k subspaces $\hat{C}$ such that cost$(P,\hat{C}) \leq \alpha(1+4\delta)$cost$(P,C^*)$.*

PROOF: From the bounds in Lemma 4.4, we have

$$\begin{aligned}\text{cost}(P,\hat{C}) &\overset{(a)}{\leq} \frac{\text{cost}(Y,g,\hat{C})}{1-\delta} \overset{(b)}{\leq} \frac{\alpha}{1-\delta}\text{cost}(Y,g,C^*) \\ &\overset{(c)}{\leq} \frac{\alpha(1+3\delta)}{1-\delta}\text{cost}(P,C^*) \leq \alpha(1+4\delta)\text{OPT}, \end{aligned} \tag{4.5}$$

where (a) and (c) follow from Lemma 4.4, and (b) follows from the fact that CN computes an α-approximate k-means clustering on (Y,g). ■

Coreset constructions for various clustering algorithms with squared ℓ_2 cost were considered in [36,93]. There is a long line of work that has focused on constructing coresets for subspace clustering and for the k-means problems [36–38,94]. Prior to the work of [39], the size of the coresets was dependent on the dimension of the problem d. However, in [38], first coresets of dimension independent sizes were provided. They constructed ϵ-coresets of size $O(k/\epsilon)$ and $\tilde{O}(k^3/\epsilon^4)$ for subspace and k-means clustering, respectively. Later, [40,90] improved the coreset sizes to poly(k/ϵ) for the subspace clustering problem and was further reduced to $\tilde{O}(k/\epsilon^4)$ for k-means and k-median problems by [26]. The current state-of-the-art coreset sizes are $\tilde{O}(k\epsilon^{-2} \cdot \min\{\epsilon^{-z}, k\})$ where $z = 2$ for k-means and $z = 1$ for k-median problems as given in [25].

Therefore, Algorithm 7 obtains an approximation factor of $(1+4\delta)$ when each WN communicates $\tilde{O}(k/\epsilon^4)$ points to the CN. Whereas, in Algorithm 6 each WN communicates k points to obtain an approximation factor of $10(1+\delta)$. The observation indicates that with an increase in communication, we can obtain better accuracy.

4.5 Byzantine Resilient Clustering

4.5.1 Byzantine Resilient Distributed k-Median Clustering

In this section, we design distributed clustering methods that are robust to the presence of Byzantines. Since the Byzantines can send arbitrary information, naive clustering algorithms may lead to solutions that may be of poor quality (illustrated in Section 4.7).

We first present a simple solution that assumes sufficient storage and computational power of the FC. Note that such an assumption is not unrealistic as central servers are usually quite powerful. However, the proposed algorithm is quite computationally and storage intensive which makes it prohibitive for practical applications with limited resources. We later propose techniques to address this difficulty by incurring slightly larger approximation factors.

The dataset P is distributed among the m WNs using the assignment matrix A which satisfies Property 2. The basic idea of the proposed algorithm is that each honest WN sends a set of k-median centers of their respective data subsets. Next, the FC combines the set of k-median centers from all the WNs and computes the respective cost of clustering on them to gauge the quality of the centers sent by each WN. The FC then computes a good-quality[4] clustering solution for the entire dataset by filtering out the summaries with larger cost. We present the aforementioned steps in detail in Algorithm 8.

[4] good approximation factor

Algorithm 8: Byzantine-resilient distributed k-median clustering

```
Input: A collection of $n$ vectors $P \subset \mathbb{R}^d$.
1 Allocate $P$ to $m$ WNs according to a $(t, \delta)$-Byzantine resilient matrix $A$.
2 Assign the set of points $P_i \subset P$ to WN $W_i$
3 Each honest WN $W_i$ computes an $\alpha$-approximate $k$-median solution $Y_i$ on set $P_i$
4 Each honest WN $W_i$ sends the set of points $Y_i$ to CN
5 Byzantine WNs send an arbitrary set of $k$ points.
6 CN computes & arranges received point sets in non-decreasing order of cost$(P_i, Y_i)$.
7 Without loss of generality, assume cost$(P_1, Y_1) \leq$ cost$(P_2, Y_2) \leq \ldots \leq$ cost$(P_m, Y_m)$.
8 For each point $\mathbf{y} \in Y_i$, CN computes weight $g_i(\mathbf{y}) = |$cluster$(\mathbf{y}, P_i)|$.
9 Let $Y = \bigcup_{i \in [m-t]} Y_i$. Using $\rho$, define $g : Y \to \mathbb{R}$ such that $g(\mathbf{y}) = \rho g_i(\mathbf{y}), \forall \mathbf{y} \in Y_i$
Output: $\hat{C}$, the $\alpha$-approximate $k$-median solution on $(Y, g)$.
```

We present the following intermediate result, which shows that the cost incurred by the weighted summary Y_i of WN W_i on any set of k centers C is bounded by the cost of clustering the local dataset P_i with C, and the *quality* of the summary Y_i. Note that these bounds hold irrespective of the WN W_i being honest or Byzantine and rely on the fact that the CN can correctly compute the weights $g_i(\mathbf{y})$.

Lemma 4.5. *For any $i \in [m]$, the weighted point set (Y_i, g_i) satisfies*

$$\textit{cost}(P_i, C) - \textit{cost}(P_i, Y_i) \leq \textit{cost}(Y_i, g_i, C) \leq \textit{cost}(P_i, C) + \textit{cost}(P_i, Y_i).$$

PROOF: The proof is relegated to Appendix A.19. ■

Lemma 4.5 shows that the cost of clustering the weighted data subset (Y_i, g_i) (where summary Y_i is obtained from W_i and weight function g_i is computed at the CN), with any set of k centers C, cost(Y_i, g_i, C) deviates from cost(P_i, C) by an additive term of cost(P_i, Y_i). The latter term quantifies the quality of the summary Y_i obtained from W_i. We assume that this quantity can be computed (or approximated) by the CN. This information is then used to filter out the summaries that contribute to large cost of clustering. From these observations, we get our main result that evaluates the quality of the clustering solution, $\hat{C}$, obtained by Algorithm 8 on the entire dataset P.

Theorem 4.6. *Let C^* be the optimal solution to the k-median problem on point set P. Then, Algorithm 8 on dataset P returns a set of k-centers $\hat{C}$ such that cost$(P, \hat{C}) \leq 3\alpha(1+\delta)$cost$(P, C^*)$, even in the presence of t Byzantines.*

PROOF: The proof is relegated to Appendix A.20. ■

4.5.2 Improved Byzantine Resilient Distributed k-Median Clustering

Recall that in the previous section, we assumed that the CN can compute the local summaries to evaluate the quality of the data sent by each local WN. In particular, we required the CN to have access to the entire dataset P. The CN needs them to estimate the cost of computing cluster P_i using Y_i sent by the WN W_i (last step in Algorithm 8). This assumption is generally reasonable since in most applications, the CN is quite powerful and has access to the entire dataset. However, in resource-constrained settings such assumptions that increase the computational load at the CN can be rather restrictive.

In this section, we discuss a simple technique to relax this assumption. For any $\delta \in (0, 1)$, let $(\tilde{P}_i, w_i)$ denote a δ-coreset computed by the CN of dataset $P_i, i \in [m]$. One possible approach to compute this efficiently in a streaming fashion is by using the uniform sampling where a δ-coreset of a set of n points is computed by only storing poly$(k\epsilon^{-1})$ points as given in [10]. Another possible approach is to use importance sampling techniques of [11, 12]. Specifically, the algorithm of [12] only stores a small set of points, and computes a good coreset using only the stored points. They show that to compute a δ-coreset of a set of n points, it is sufficient to only store $O(\delta^{-2}dk \log k)$ points. Therefore, the CN will only need to store $O(mdk \log k)$ points in total. Using standard dimension reduction techniques, we can without loss of generality, assume that $d = O(\log n)$.

To improve Algorithm 8, the coreset $(\tilde{P}_i, w_i)$ computed on each dataset P_i (using sensitivity sampling [12]) is utilized to approximate the cost of clustering pointset P_i with $Y_i, i \in [m]$.

Furthermore, the weights $g_i(y)$ for each $y \in Y_i$ are also estimated using only the coreset points. This reduces the computational load at the CN. In particular, estimating cost(P_i, Y_i), takes only $O(k^2 \log |P_i|)$ time instead of $O(k|P_i|)$. In the following, we show that we still obtain a good approximation for k-median clustering in the presence of Byzantines using $(\tilde{P}_i, w_i)$ instead of P_i. Furthermore, this coreset computation at the CN can be done while assigning them to the WNs. The modified algorithm for distributed k-median clustering in the presence of Byzantines is presented in Algorithm 9. For the simplicity of presentation, we assume that WNs and the CN can compute the exact (i.e., $\alpha = 1$) k-median solution on a small dataset. The results extend trivially when in Step 4 and last step, the WNs and the CN compute an α-approximate solution.

Algorithm 9: Computationally-efficient Byzantine-resilient distributed k-median clustering

```
Input: A collection of n vectors $P \subset \mathbb{R}^d$
1  Allocate $P$ to $m$ WNs according to $A$ with Property 2.
2  CN computes $\delta$-coreset $(\tilde{P}_i, w_i)$ from the streaming data with respect to each $P_i$
3  Assign the set of points $P_i \subset P$ to WN $W_i$
4  Each honest WN $W_i$ computes $k$-median solution $Y_i$ on set $P_i$
5  Each honest WN $W_i$ sends the set of points $Y_i$ to CN
6  Byzantine WNs send an arbitrary set of $k$ points.
7  CN computes & arranges received point sets in non-decreasing order of cost$(\tilde{P}_i, w_i, Y_i)$.
8  Without loss of generality, assume
   cost$(\tilde{P}_1, w_1, Y_1) \leq$ cost$(\tilde{P}_2, w_2, Y_2) \leq \ldots \leq$ cost$(\tilde{P}_m, w_m, Y_m)$.
9  For each point $\mathbf{y} \in Y_i$, CN computes weight $\tilde{g}_i(\mathbf{y}) = \sum_{p \in \text{cluster}(\mathbf{y}, \tilde{P}_i)} w_i(p)$.
10 Let $Y = \bigcup_{i \in [m-t]} Y_i$. Using $\rho$, define $\tilde{g} : Y \to \mathbb{R}$ such that $\tilde{g}(\mathbf{y}) = \rho \tilde{g}_i(\mathbf{y}), \forall \mathbf{y} \in Y_i$
Output: $\hat{C}$, the $k$-median solution on $(Y, \tilde{g})$.
```

Theorem 4.7. *Let $\delta \in (0, 1)$. Let C^* be the optimal solution to the k-median problem on point set P. Then, Algorithm 9 returns a set of k-centers $\hat{C}$ such that cost$(P, \hat{C}) \leq \left(\frac{2}{1-\gamma} + \frac{1}{1-\delta}\right)(1 + 3\delta)$cost$(P, C^*)$, even in the presence of t Byzantines with probability $1 - \frac{1}{k}$ by choosing $\gamma = \frac{1}{k}$.*

We now briefly sketch the proof of Theorem 4.7. The formal proof is presented in Appendix A.21.

The two main differences in Algorithm 9 compared to Algorithm 8 are

1. The filtering of Byzantines in Step 7 is done with respect to cost$(\tilde{P}_i, w_i, Y_i)$ instead of cost(P_i, Y_i). Since $(\tilde{P}_i, w_i)$ is a δ-coreset of P_i, we incur at most a factor of $(1+\delta)$ in cost by making this change.

2. For any $i \in [m-t]$, and $\mathbf{y} \in Y_i$, the quantity $g_i(\mathbf{y}) = |\text{cluster}(\mathbf{y}, P_i)|$ is computed using the coreset $\tilde{P}_i$ instead of the actual pointset P_i. We show that by adopting a particular sensitivity-based i.i.d. sampling technique of coreset construction, the estimate of $\tilde{g}_i$ is at most some $(1+\gamma)$ factor away from its intended value with very high probability, for some appropriately chosen value of γ.

We now formalize the above two statements in the following Lemmas and Observations.

Observation 1. *Let $\delta \in (0,1)$. For any $i \in [m]$ and any set of k centers C, we have*

$$|cost(\tilde{P}_i, w_i, C) - cost(P_i, C)| \leq \delta cost(P_i, C)$$

The observation follows from the fact that $(\tilde{P}_i, w_i)$ is a δ-coreset of P_i

Lemma 4.6. *Let $\gamma \geq \frac{1}{k}$. For any $i \in [m]$, and $\boldsymbol{y} \in Y_i$,*

$$\Pr[|\tilde{g}_i(\boldsymbol{y}) - g_i(\boldsymbol{y})| \geq \gamma\, g_i(\boldsymbol{y})] \leq \frac{1}{k}$$

Lemma 4.6 therefore ensures the following:

Observation 2. *Let $\gamma \geq \frac{1}{k}$. For any $i \in [m]$ and $\boldsymbol{y} \in Y_i$, let $g_i(\boldsymbol{y}) := |cluster(\boldsymbol{y}, P_i)|$. Then for any set of k centers C,*

$$cost(Y_i, g_i, C) = \sum_{\boldsymbol{y} \in Y_i} g_i(\boldsymbol{y}) d(\boldsymbol{y}, C) \leq \sum_{\boldsymbol{y} \in Y_i} \frac{1}{1-\gamma} \tilde{g}_i(\boldsymbol{y}) d(\boldsymbol{y}, C) = \frac{1}{1-\gamma} cost(Y_i, \tilde{g}_i, C),$$

with probability at least $1 - 1/k$.

Using the two observations listed above, we get an equivalent of Lemma 4.5.

Lemma 4.7. *Let $\delta, \gamma \in (0,1)$. For any $i \in [m]$, the weighted point set $(Y_i, \tilde{g}_i)$ satisfies*

$$(1-\gamma)\textit{cost}(P_i, C) - \frac{1-\gamma}{1-\delta}\textit{cost}(\tilde{P}_i, w_i, Y_i) \leq \textit{cost}(Y_i, \tilde{g}_i, C) \leq (1+\delta)\textit{cost}(P_i, C) + \textit{cost}(\tilde{P}_i, w_i, Y_i).$$

The proof of Theorem 4.7 then follows similarly as the proof of Theorem 4.6 using the adjusted Lemma 4.7 instead of Lemma 4.5.

PROOF: Proof of Theorem 4.7 is relegated to Appendix A.21 ■

4.5.3 Byzantine Resilient k-means Clustering

Similar to Algorithm 6 for straggler resilient k-means clustering, a simple modification can be made to Algorithm 8 to obtain a Byzantine-resilient distributed k-means algorithm (Algorithm 10) with performance guarantees given in Theorem 4.8.

Algorithm 10: Byzantine-resilient distributed k-means clustering

```
   Input: A collection of $n$ vectors $P \subset \mathbb{R}^d$
1  Allocate $P$ to $m$ WNs according to $A$ with Property 2.
2  Assign the set of points $P_i \subset P$ to WN $W_i$
3  Each honest WN $W_i$ computes $k$-means solution $Y_i$ on set $P_i$
4  Each honest WN $W_i$ sends the set of points $Y_i$ to CN
5  Byzantine WNs send an arbitrary set of $k$ unweighted points.
6  CN computes & arranges received point sets in non-decreasing order of cost$(P_i, Y_i)$.
7  Without loss of generality, assume cost$(P_1, Y_1) \leq$ cost$(P_2, Y_2) \leq \ldots \leq$ cost$(P_m, Y_m)$.
8  For each point $\mathbf{y} \in Y_i$, CN computes weight $g_i(\mathbf{y}) = |\text{cluster}(\mathbf{y}, P_i)|$.
9  Let $Y = \bigcup_{i\in[m-t]} Y_i$. Using $\rho$, define $g : Y \to \mathbb{R}$ such that $g(\mathbf{y}) = \rho g_i(\mathbf{y}), \forall \mathbf{y} \in Y_i$
   Output: $\hat{C}$, the $k$-means solution on $(Y, g)$.
```

Theorem 4.8. *Let C^* be the optimal solution to the k-means problem on point set P. Then, Algorithm 10 returns a set of k-centers $\hat{C}$ such that cost$(P, \hat{C}) \leq 10\alpha(1+\delta)$cost$(P, C^*)$, even in* *the presence of t Byzantines.*

Moreover, similar to Algorithm 9, the CN can reduce its computational and storage costs by computing δ-coresets for k-means clustering of each local data set. Coresets obtained by i.i.d. sensitivity sampling of the data in a streaming fashion require $O(\delta^{-2}mdk\log k)$ points to be stored in total.

Algorithm 11: Computationally-efficient Byzantine-resilient distributed k-means clustering

```
Input: A collection of $n$ vectors $P \subset \mathbb{R}^d$
1  Allocate $P$ to $m$ WNs according to $A$ with Property 2
2  CN computes weighted coreset $(\tilde{P}_i, w_i)$ from the streaming data with respect to each $P_i$
3  Assign the set of points $P_i \subset P$ to WN $W_i$
4  Each honest WN $W_i$ computes $k$-means solution $Y_i$ on set $P_i$
5  Each honest WN $W_i$ sends the set of points $Y_i$ to CN
6  Byzantine WNs send an arbitrary set of $k$ points.
7  CN computes & arranges received point sets in non-decreasing order of cost$(\tilde{P}_i, w_i, Y_i)$.
8  Without loss of generality, assume
   cost$(\tilde{P}_1, w_1, Y_1) \leq$ cost$(\tilde{P}_2, w_2, Y_2) \leq \ldots \leq$ cost$(\tilde{P}_m, w_m, Y_m)$.
9  For each point $\mathbf{y} \in Y_i$, CN computes weight $\tilde{g}_i(\mathbf{y}) = \sum_{\mathbf{p} \in \text{cluster}(\mathbf{y}, \tilde{P}_i)} w_i(\mathbf{p})$.
10 Let $Y = \bigcup_{i \in [m-t]} Y_i$. Using $\rho$, define $\tilde{g} : Y \to \mathbb{R}$ such that $\tilde{g}(\mathbf{y}) = \rho \tilde{g}_i(\mathbf{y}), \forall \mathbf{y} \in Y_i$
Output: $\hat{C}$, the $k$-means solution on $(Y, \tilde{g})$.
```

Theorem 4.9. *Let $\delta \in (0,1)$. Let C^* be the optimal solution to the k-means problem on point set P. Then, Algorithm 11 returns a set of k-centers $\hat{C}$ such that cost$(P, \hat{C}) \leq \left(\frac{8}{1-\gamma} + \frac{2}{1-\delta}\right)(1 + 3\delta)$cost$(P, C^*)$, even in the presence of t Byzantines with probability $1 - \frac{1}{k}$ by choosing $\gamma = \frac{1}{k}$.*

The proofs of both Theorem 4.8, and Theorem 4.9 are analogous to the k-median proof. In fact, they are the same except for the use of scaled triangular inequality (Eq. (4.4)) instead of standard triangle inequality used in the proofs for their k-median counterparts.

4.6 Construction of Data Assignment Matrix

In this section, we provide the approach for the construction of the assignment matrix in the presence of stragglers and Byzantines. Since Property 2 for Byzantine resilience is stronger than Property 1, we will focus only on the construction of assignment matrices A that satisfy the former. The straggler resilience property of those matrices will follow from the definition.

Let n be the number of data points in P, and m be the number of WNs. Let $\mathcal{B} \subset [m]$, $|\mathcal{B}| < t$ denote the set of Byzantines, and let $\mathcal{R} = [m] \setminus \mathcal{B}$ be the set of non-Byzantines. For the simplicity of presentation, we assume $n = m$. In the random Byzantine model, we assume that each WN W_i, for $i \in [m]$ behaves as a Byzantine independently with some fixed (known) probability p_t. We now present the construction of various assignment matrices $A \in \{0,1\}^{m \times m}$ that satisfy Property 2, and hence Property 1 as well.

The two parameters of importance when constructing an assignment matrix are the load per WN and the fraction of faulty WNs that can be tolerated. Increasing the redundancy increases both, the fault tolerance and the computational load on individual WNs. For each of the constructions provided below, we analyze the tradeoffs between these two parameters namely, the load per WN ($\ell = \max_i |P_i|$) and the fraction of Byzantines (t/m) that can be tolerated.

4.6.1 Randomized Construction for Random Byzantines

We present a randomized construction of the assignment matrix that satisfies Property 2. For the construction of the matrix, we assume a random Byzantine (or straggler) model, where every WN acts as a Byzantine (or straggler) independently with probability p_t. Hence, the local computation from each WN is received at the CN with probability $1 - p_t$.

For some ℓ (to be chosen later), the (i, j)-th entry of the assignment matrix, based on the random construction discussed above, is defined as

$$A_{i,j} = \begin{cases} 1 & \text{with probability } p_a = \frac{l}{m} \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

For an appropriate choice of ℓ (and, hence, p_a), we show that the random matrix A satisfies Property 2 with high probability.

Theorem 4.10. *For any $\delta > 0$, the randomized assignment matrix in* (4.6) *with $\ell = \frac{6(2+\delta)^2}{\delta^2} \cdot \frac{\log(n\sqrt{2})}{1-p_t}$ satisfies Property 2 with probability at least $1 - \frac{1}{n}$ under the random Byzantine (or straggler) model.*

PROOF: Proof is relegated to Appendix A.22. ■

For $m = O(n)$, our construction assigns $O(\log n)$ data points to each WN and is resilient to a constant fraction of random Byzantines (or stragglers).

4.6.2 Explicit Construction for Random Byzantines

Fractional Repetition Codes (FRC) have been well-studied in [20] for straggler resilient gradient computations. In this section, we show that the FRC scheme also satisfies Property 2 for random Byzantines with high probability, and hence provides redundant data assignment for Byzantine-resilient clustering problems.

For simplicity, let us assume that we have m data points and m WNs. In FRC, the m data points are partitioned into groups of size s (assume that s divides m), and each group of data points is replicated across s WNs. The assignment matrix A for this scheme is given by

$$A = \begin{pmatrix} \mathbf{1}_{s\times s} & \mathbf{0}_{s\times s} & \mathbf{0}_{s\times s} & \dots & \mathbf{0}_{s\times s} \\ \mathbf{0}_{s\times s} & \mathbf{1}_{s\times s} & \mathbf{0}_{s\times s} & \dots & \mathbf{0}_{s\times s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{s\times s} & \mathbf{0}_{s\times s} & \mathbf{0}_{s\times s} & \dots & \mathbf{1}_{s\times s} \end{pmatrix}, \tag{4.7}$$

where $\mathbf{1}_{s\times s}$ denotes an $s \times s$ matrix of all 1's.

Let $A_{\mathcal{R}}$ of size $|\mathcal{R}| \times m$ denote the submatrix of honest WNs obtained by removing t rows from A uniformly at random. We now show that the random matrix $A_{\mathcal{R}}$ satisfies Property 2 with high probability.

Theorem 4.11. *For any $\delta > 0$, the FRC based assignment matrix A with $\ell = s = O(\log m)$, satisfies Property 2 with probability at least $1 - O(\frac{1}{m})$ under the random Byzantine model, and provides resilience against $t = O(m)$ Byzantines.*

PROOF: The proof is relegated to Appendix A.23. ■

Theorem 4.11 provides good tradeoff between the load per WN $\ell = O(\log m)$, and the number of Byzantines tolerated, $t = O(m)$. However, the performance guarantees hold only in the random Byzantine model. In the adversarial Byzantine model, any subset $\mathcal{B} \subset [m]$ can be Byzantines. This is a much stronger yet practical model for Byzantines. We now give two constructions - one randomized and one explicit construction that provide relatively worse tradeoffs in the adversarial Byzantine model.

4.6.3 Random Construction for Adversarial Byzantines

In this section, we show that a random Bernoulli assignment matrix satisfies Property 2 under the adversarial Byzantine model albeit with slightly degraded tradeoffs between ℓ and t.

Consider an $m \times m$ random Bernoulli assignment matrix A where each entry $A_{i,j}$ is set to 1 independently with some probability p, and 0 otherwise.

Theorem 4.12. *For any $\delta > 0$, the Bernoulli assignment matrix A with $p = O(\frac{1}{\log m})$, satisfies Property 2 with probability at least $1 - O(\frac{1}{m})$ under the adversarial Byzantine model, and is resilient to $t = O(\frac{m}{\log^2 m})$ Byzantines.*

PROOF: Proof is relegated to Appendix A.24. ■

Alternatively, Theorem 4.12 can be stated in terms of a random construction that is resilient to t arbitrary Byzantines with an expected load of $O(\frac{mt}{m-t} \log m)$. Note that Theorem 4.12 provides lesser redundancy in the regime when $t = o(m)$ compared to the naïve solution of distributing all the points to all the WNs.

4.6.4 Explicit Construction for Adversarial Byzantines

We now present an explicit construction of an assignment matrix that satisfies Property 2 for the adversarial Byzantine model. The construction is based on expander graphs which were recently used to construct explicit data assignment schemes for gradient coding [46, 85].

Let $G = (V, E)$ be a connected d-regular graph on m vertices and let $\mathcal{A}_G$ denote its adjacency matrix. Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$ be the m real eigenvalues of $\mathcal{A}_G$. Define the expansion parameter of graph G as $\lambda = \max\{|\lambda_2|, |\lambda_m|\}$. We denote such d-regular graphs on n vertices with expansion parameter λ as (n, d, λ)-expanders.

The double cover of a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ on n vertices, is a bipartite graph $G = (L \cup R, E)$, on $2n$ vertices with $L = R = V$. There is an edge $(u, v) \in L \times R$ in G if and only if $(u, v) \in \tilde{E}$.

To construct our assignment matrix, we consider a bipartite graph $G = (L \cup R, E)$ that is a double cover of an (n, d, λ)-expander. The $m \times m$ assignment matrix A is obtained from G by setting $A_{u,v} = 1$ if and only if there is an edge between $(u, v) \in G$ for any $u \in R$ and, $v \in L$. We now show that the assignment matrix A obtained from G satisfies Property 2 for any set of t Byzantines.

Theorem 4.13. *For any $\delta > 0$, the assignment matrix A satisfies Property 2 under adversarial Byzantine model with $t = \sqrt{\log m / \log\log m}$, and $\ell = O(\log m)$.*

The proof, presented formally in Appendix A.25, follows from the fact that if $\tilde{G}$ is an expander graph, then its double cover G satisfies the expander Mixing Lemma [55].

Theorem 4.14 (Expander Mixing Lemma [55]). *For any sets S and T in a (n, d, λ)-expander, we have $|E(S, T) - \frac{d}{n}|S||T|| \leq \lambda\sqrt{|S||T|}$, where, $E(S, T)$ denotes the number of edges between sets S and T.*

Using Expander Mixing Lemma, we can show that no vertex in L is incident to a large fraction of vertices in any t subset of R. This in turn translates to the fact that no column of A has a large number of 1's in any subset of t rows of A. Therefore, removing any t rows of A keeps all the column weights within a fixed range.

The existence of graphs with appropriate expansion properties then completes the proof. We use the constructions of (n, d, λ)-expanders of [8], to get data assignment schemes that are resilient to $O(\sqrt{\log m})$ Byzantines with an overhead of $O(\log m)$ data points per WN.

Theorem 4.15 ([8]). *There exists a polynomial time algorithm to construct* $(n, d, \lambda) = (2^\ell, \ell - 1, \sqrt{\ell \log^3 \ell})$.

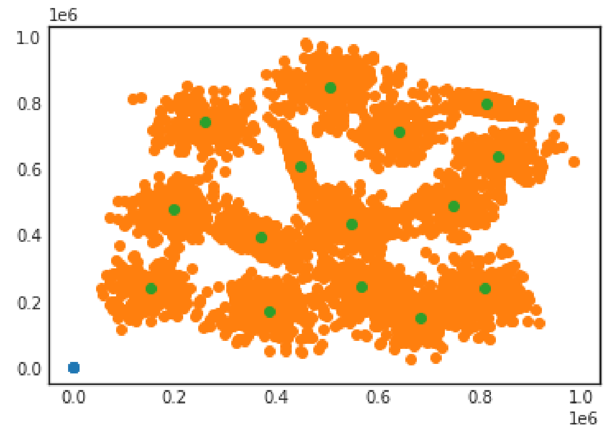
	Construction	Byzantine model	# Byzantines	Load per node
Thm 4.10	Random	Random	$O(m)$	$\frac{6(2+\delta)^2}{\delta^2} \cdot \frac{\log(n\sqrt{2})}{1-p_t}$
Thm 4.11	Explicit	Random	$O(m)$	$O(\log m)$
Thm 4.12	Random	Adversarial	$o(m)$	$O(\frac{mt}{m-t} \log m)$
Thm 4.13	Explicit	Adversarial	$\sqrt{\log m / \log(\log m)}$	$O(\log m)$

Table 4.1: Summary of constructions of data assignment schemes

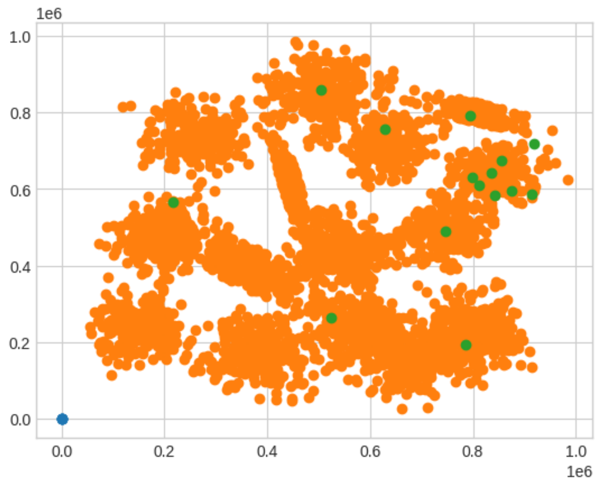
Next, we empirically evaluate the performance of our algorithms and show that they are robust to Byzantines (or stragglers).

4.7 Simulation Results

In this section, we demonstrate the performance of our distributed k-median clustering algorithms that are resilient to stragglers and Byzantines, respectively. We consider the synthetic Gaussian dataset [41] with $n = 5000$ two-dimensional points that are distributed among $m = 10$ WNs.



(a) Ground Truth.



(b) Performance with no redundancy.

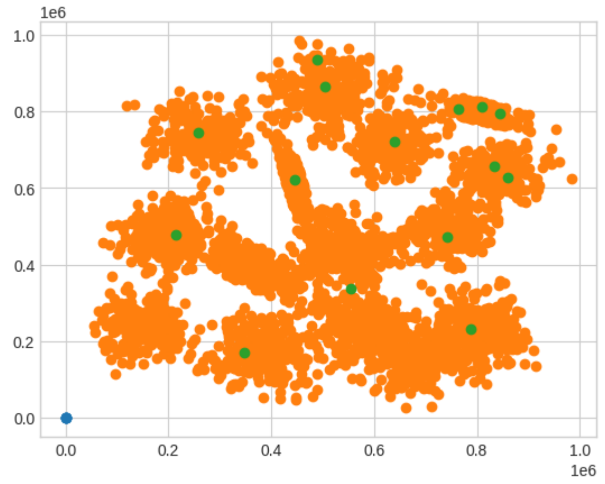
Fig. 4.1: Performance of the proposed Straggler-resilient k-median algorithm with no redundancy.

4.7.1 Straggler-resilient Clustering

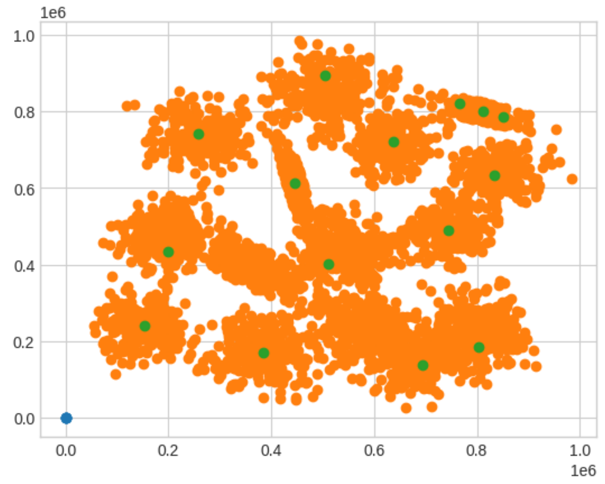
In this section, we illustrate the performance of our straggler-resilient distributed k-median algorithm and benchmark it with the non-redundant data assignment scheme. We consider $t = 3$ randomly chosen stragglers. We present the results in Figures 4.1(a), 4.1(b), 4.2(a), and 4.2(b).

We plot the ground truth using the centroids provided in the dataset in Fig. 4.1(a) with k-median clustering, for $k = 15$. In Fig. 4.1(b), we present the results by ignoring the local computations from the stragglers, i.e., Algorithm 5 is used without any redundant data assignment. We randomly partition the $n = 5000$ data points among $m = 10$ WNs. The non-straggler WNs send their respective k-median centers to the FC. Then, the FC runs a k-median algorithm on the $k(m - t)$ centers obtained from the non-straggler WNs. From Fig. 4.1(b), the set of poor quality k-centers obtained from this scheme is noticeable.

In Fig. 4.2(a), the result obtained by using Algorithm 5 is shown. We choose the assignment matrix randomly with $p = \Pr[A_{i,j} = 1] = 0.1$. Hence, using this assignment matrix ensures that each WN receives 500 data points on an average which results in a non-redundant data assignment. Lastly, in Fig. 4.2(b), we show the effect of increasing the value of p to 0.2. Therefore, the redundancy in the data assignment increases which results in each WN receiving about 1000 data points.



(a) Performance with redundancy $p = 0.1$.



(b) Performance with redundancy $p = 0.2$.

Fig. 4.2: Performance of the proposed Straggler-resilient k-median algorithm.

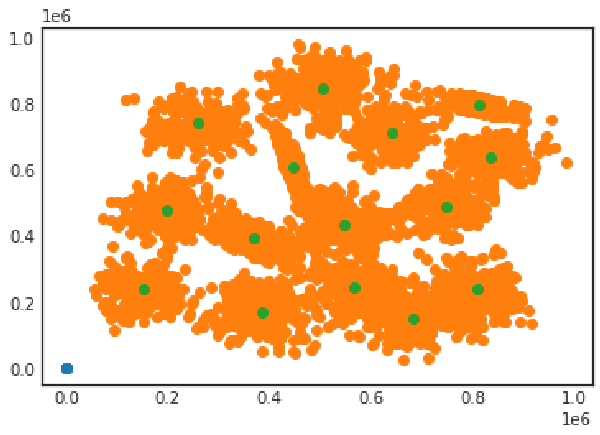
We observe that the results are very close to the ground truth clustering presented in Fig. 4.1(a).

4.7.2 Byzantine-resilient Clustering

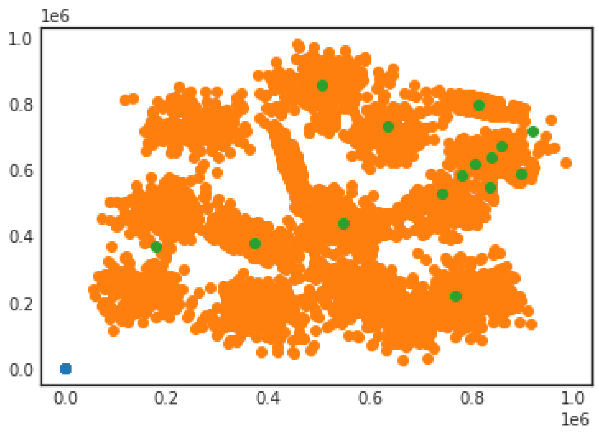
In this section, we illustrate the performance of our Byzantine-resilient distributed k-median algorithm and benchmark it with the non-redundant data assignment scheme. We consider $t = 3$ randomly chosen Byzantines. We present the results in Figures 4.3(a), 4.3(b), 4.4(a), and 4.4(b).

We plot the ground truth using the centroids provided in the dataset in Fig. 4.3(a) with k-median clustering, for $k = 15$. In Fig. 4.3(b), we present the results by ignoring the local computations from the Byzantines, i.e., Algorithm 8 is used without any redundant data assignment. We randomly partition the $n = 5000$ data points among $m = 10$ WNs. The honest WNs send their respective k-median centers to the FC. Then, the FC runs a k-median algorithm on the $(m - t)$ centers obtained from the honest WNs. From Fig. 4.3(b), the set of poor quality k-centers obtained from this scheme is noticeable.

In Fig. 4.4(a), the result obtained by using Algorithm 8 is shown. We choose the assignment matrix randomly with $p = \Pr[A_{i,j} = 1] = 0.1$. Hence, using this assignment matrix ensures that each WN receives 500 data points on an average which results in a non-redundant data assignment.

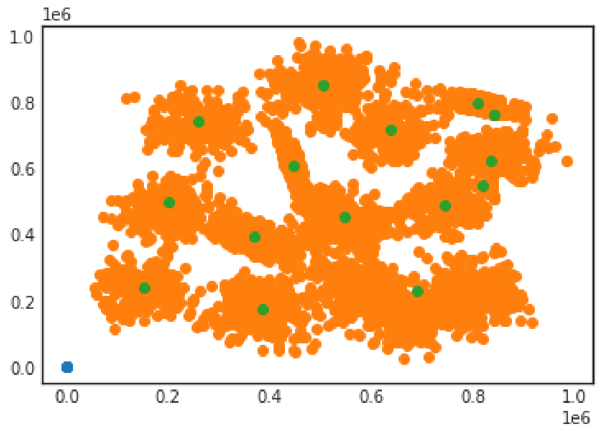


(a) Ground Truth.

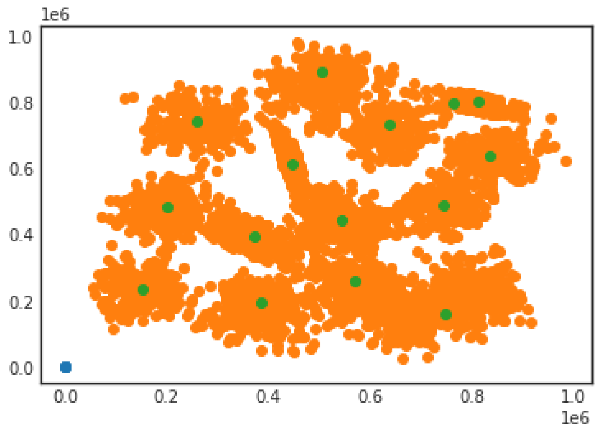


(b) Performance with no Redundancy.

Fig. 4.3: Performance of the proposed Byzantine-resilient k-median algorithm with no redundancy.



(a) Performance with Redundancy $p = 0.1$.



(b) Performance with Redundancy $p = 0.2$.

Fig. 4.4: Performance of the proposed Byzantine-resilient k-median algorithm.

Lastly, in Fig. 4.4(b), we show the effect of increasing the value of p to 0.2. Therefore, the redundancy in the data assignment increases which results in each WN receiving about 1000 data points. We observe that the results are very close to the ground truth clustering presented in Fig. 4.3(a).

4.8 Summary

In this chapter, we provided $O(1)$-approximate solutions for the distributed k-median and k-means clustering problems in the presence of stragglers. These solutions were obtained under the assumption that the WNs can not compute exact clustering solutions. These algorithms were then extended to the case where Byzantines were present in the system. Note that the approach for k-means (Algorithm 6 and Algorithm 10) used in this work can be generalized to obtain straggler-resilient and Byzantine-resilient algorithms for a larger class of ℓ_2 fitting problems such as (r, k)-subspace clustering solutions. We also provided computationally efficient constant factor approximate solutions for the distributed clustering problems in the presence of Byzantines. Further, we provided simulation results which affirm our theoretical guarantees.

CHAPTER 5
CONCLUSION AND FUTURE DIRECTIONS

5.1 Summary

In this dissertation, we studied machine learning algorithms under both supervised and unsupervised learning frameworks in the presence of two types of attacks: 1) *attack on data*, and 2) *attack on nodes*. We proposed learning algorithms that are robust under both types of attacks.

Firstly, we considered the problem of estimating the parameters (weight matrix and bias vector) of a neural network with rectified linear unit as the activation function under the unsupervised learning framework. Specifically, we considered the estimation procedure in the presence of arbitrary outliers (*attack on data*) where in the given set of data samples a fraction of the samples are arbitrary outliers, and the rest are the output samples of a single-layer neural network. We designed our robust estimation procedure by combining the gradient descent algorithm with the median filter to mitigate the effect of the arbitrary outliers. We then proceeded to compute the bounds on the number of samples needed by the algorithm along with its running time to perform the estimation of the parameters. Our theoretical and simulation results provided insights into the training complexity of the neural network in terms of the probability of outliers and problem dimension.

Secondly, we considered the problem of distributed optimization with the worker-server architecture under the supervised learning framework where a fraction ($< 1/2$) of the nodes are adversarial (*attack on nodes*). We proposed a robust variant of the learning algorithm to solve stochastic nonconvex optimization. The algorithm design consisted of the combination of distributed variance-reduction and the filtering technique called *vector median* used to identify and prune the Byzantines. We showed that our algorithm converges to a first-order stationary point and the convergence rate does not depend on the problem dimension. We evaluated the performance of the proposed algorithm and presented the simulation results using MNIST and CIFAR10 datasets.

Thirdly, under the unsupervised learning framework, we considered the problem of distributed clustering for any set of adversarial nodes (*attack on nodes*). We proposed a redundant data assignment scheme that enabled us to obtain global information about the entire dataset even in the presence of adversarial nodes. We proposed robust clustering algorithms that generate a constant factor approximate solution in the presence of adversarial nodes. Moreover, we provided several constructions for the data assignment scheme which provided resilience against a large fraction of the adversarial nodes. We also provided simulation results that corroborated the excellent performance of our proposed algorithms.

Next, we discuss some of the future directions of the work presented in this dissertation.

5.2 Future Directions

5.2.1 Robust Learning of Multi-layer Neural Network

The problem of learning multi-layer neural networks has received a lot of attention. It was only recently in [23] that a polynomial-time algorithm for learning a multi-layer neural network with ReLU activation was proposed. However, the problem of learning a multi-layer neural network in the presence of adversarial attacks has received less attention. Further, differential privacy (DP) is the most widely accepted notion of data privacy in theory and practice. Under DP, a single sample is not allowed to have significant impact on the output distribution of a learning algorithm that operates on a dataset. DP can be considered to provide *robustness* against data leakage. Also, there has been significant amount of work in robust statistics showing that robustness and privacy are indeed related [67, 74, 75]. However, there has not been any significant work on the algorithms for learning neural networks that ensure robustness against adversarial attacks and privacy of the data. Extending our work to learn the parameters of a multi-layer neural network ensuring privacy and robustness against adversarial attacks will bring us closer to understanding inner workings of a neural network.

5.2.2 Byzantine-resilient Decentralized Optimization

To reduce the computational bottleneck of a central server that performs a majority of the computations in a distributed system, a decentralized setup was proposed. Typically, in a decentralized setup, all the devices exchange information to perform local tasks and achieve a global objective in the absence of a central server. The connectivity of all the nodes in the decentralized setting is represented by a mixing matrix. There have been several works on decentralized optimization in the literature with different types of mixing matrices [53, 88] where all the participating devices are assumed to be honest. However, robust decentralized optimization in the presence of adversarial nodes that also satisfies privacy constraints has received less attention and is a research direction worth pursuing. For stochastic decentralized optimization problem, the design of efficient algorithms that ensure privacy and robustness against adversarial attacks is a research direction worth pursuing.

5.2.3 Robust Fair Clustering

There are several examples of the unsavory behavior of the learning algorithms related to unsupervised learning tasks like gender stereotypes in *word2vec* embeddings. It is crucial to ensure *robustness* against over-represented classes dominating under-represented classes. This can be ensured through algorithmic fairness. One direction to ensure fairness is the problem of data summarization through the lens of algorithmic fairness where the goal is to output a small but representative subset of a dataset such that some fairness constraint is satisfied. Under the fairness consideration, each class should receive a similar service guarantee. Alternatively, each cluster should contain approximately the same proportion of samples from each protected class as they appear in the dataset. One typical approach to solving this is *Colorful k-Center Clustering* which includes a covering constraint which ensures that at least a given number of points from each group is covered by the clusters [59]. The existing algorithms for the clustering problem with fairness constraint run in super-quadratic time. This drawback has been addressed in [66] with a linear time algorithm. Although efficient algorithms have been considered for clustering, fairness and *robustness* against adversarial attacks perspective has not been explored. Hence, designing fair clustering algorithms that are robust to adversarial attacks with running time linear in the size of the data is another research direction worth pursuing.

APPENDIX A

APPENDIX: PROOFS OF VARIOUS RESULTS

A.1 Toolbox

This section summarizes a list of well-known results used in the proofs of our results.

Lemma A.1 (Strong Convexity with Truncation ([27, Lemma 4])). *Consider the truncated normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S})$ whose support is $\mathcal{S} \subset \mathbb{R}^d$. Let $\boldsymbol{H}(\boldsymbol{v}) \in \mathbb{R}^{(d+d^2)\times(d+d^2)}$ denotes the Hessian matrix of the expected negative log-likelihood function computed with respect to $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ \operatorname{vec}\left\{\boldsymbol{\Sigma}^{-1}\right\} \end{bmatrix} \in \mathbb{R}^{d+d^2}$ where $\operatorname{vec}\{\cdot\}$ is the vectorization operator. Then, $\boldsymbol{H}(\boldsymbol{v})$ satisfies*

$$\boldsymbol{H}(\boldsymbol{v}) \succeq C \left[\int_{\mathcal{S}} \frac{1}{\sqrt{2\pi|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{u}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{u}-\boldsymbol{\mu})\right) d\boldsymbol{u}\right]^4 \frac{\lambda_{\min}}{\max\left\{4, 16\left\|\boldsymbol{\mu}\right\|^2 + \sqrt{\lambda_{\min}}\right\}}\boldsymbol{I}, \tag{A.1}$$

where $C > 0$ is a universal constant, and $\lambda_{\min} = [\min_{1\le i\le j\le d} \min\{\lambda_i\lambda_j, \lambda_i\}]$ with $\lambda_1, \lambda_2, \ldots, \lambda_d$ being the eigenvalues of $\boldsymbol{\Sigma}$.

Lemma A.2 (Smooth and strongly convex functions ([13, Lemma 3.11])). *Let the function $l : \mathbb{R}^d \to \mathbb{R}$ be L-smooth and η-strongly convex. Then, for all $\boldsymbol{u}, \boldsymbol{w} \in \mathbb{R}^d$, the following holds:*

$$(\nabla l(\boldsymbol{u}) - \nabla l(\boldsymbol{w}))^{\mathsf{T}}(\boldsymbol{u}-\boldsymbol{w}) \ge \frac{\eta L}{\eta + L}\left\|\boldsymbol{u}-\boldsymbol{w}\right\|^2 + \frac{1}{\eta+L}\left\|\nabla l(\boldsymbol{u}) - \nabla l(\boldsymbol{w})\right\|^2. \tag{A.2}$$

Lemma A.3 (Hoeffding's inequality ([97, Theorem 2.2.2]))**.** *Let* $\{x_i \in \{0,1\}\}_{n=1}^N$ *be a set of iid Bernoulli random variables with mean* p*. Then, for any* $\epsilon \in (0,1)$*, we have*

$$\mathbb{P}\left(\sum_{n=1}^{N} x_i \geq (p-\epsilon)n\right) \geq 1 - e^{-2\epsilon^2 N}. \tag{A.3}$$

Lemma A.4 (Concentration of empirical mean and covariance)**.** *Consider a set of iid random variables* $\left\{x^{(n)} \sim \mathcal{N}^+(\mu,\sigma^2)\right\}_{n=1}^N$*. Let the empirical mean and variance computed using* $N = \mathcal{O}\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\Psi}\right)\log^2\left(\frac{1}{\delta}\right)\right)$ *be* $\hat{\mu} = \frac{1}{N}\sum_{n=1}^N x^{(n)}$ *and* $\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^N \left(x^{(n)} - \hat{\mu}\right)^2$*, respectively. Then, we have*

$$\left|\hat{\mu} - \mathbb{E}\left\{x^{(1)}\right\}\right| \leq \epsilon \quad \textit{and} \quad \left|\hat{\sigma}^2 - \mathbb{E}\left\{\left(x^{(1)} - \mathbb{E}\left\{x^{(1)}\right\}\right)^2\right\}\right| \leq \epsilon, \tag{A.4}$$

with probability exceeding $1-\delta$ *where* $\Psi = \int_{x>0} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx$*.*

PROOF: The proof is similar to the proof of ([27, Lemma 5]), and hence, it is omitted. ■

Lemma A.5 (Angle estimated from the sign of inner products ([102, Lemma 2]))**.** *Suppose that* $x \sim \mathcal{D}(\boldsymbol{W},\boldsymbol{b})$ *and that* $\boldsymbol{b} \in \mathbb{R}^d$ *is non-negative, for all* $i \neq j \in [d]$*,*

$$\mathbb{P}_{x\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\boldsymbol{x}_i > \boldsymbol{b}_i \textit{ and } \boldsymbol{x}_j > \boldsymbol{b}_j] = \frac{\pi - \theta_{ij}}{2\pi}, \tag{A.5}$$

where θ_{ij} *is the angle between vectors* $\boldsymbol{W}_i$ *and* $\boldsymbol{W}_j$*.*

Lemma A.6 (Error bound of angle estimation ([102, Lemma 3]))**.** *Let* $x \sim \mathcal{D}(\boldsymbol{W},\boldsymbol{b})$*, where* $\boldsymbol{b} \in \mathbb{R}^d$ *is non-negative. Suppose* $\hat{\boldsymbol{b}} \in \mathbb{R}^d$ *is non-negative such that* $\left|\boldsymbol{b}_i - \hat{\boldsymbol{b}}_i\right| \leq \epsilon\,\|\boldsymbol{W}_i\|$*, for all* $i \in [d]$*. Then, the following relation holds for any* $i,j \in [d]$

$$\left|\mathbb{P}_{x\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\boldsymbol{x}_i > \boldsymbol{b}_i \textit{ and } \boldsymbol{x}_j > \boldsymbol{b}_j] - \mathbb{P}_{x\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\boldsymbol{x}_i > \hat{\boldsymbol{b}}_i \textit{ and } \boldsymbol{x}_j > \hat{\boldsymbol{b}}_j]\right| \leq \epsilon. \tag{A.6}$$

A.2 Proof of Proposition 2.1

Invoking Lemma A.1 with $d = 1$, we obtain that there exists a universal constant $C > 0$ such that

$$\nabla^2 \ell(\boldsymbol{v}) \succeq C \left[1 - \Phi\left(-\frac{\boldsymbol{v}_2}{\sqrt{\boldsymbol{v}_1}}\right)\right]^4 \frac{\min\{1/\boldsymbol{v}_1, 1/\boldsymbol{v}_1^2\}}{\max\left\{4, 16\boldsymbol{v}_2^2/\boldsymbol{v}_1^2 + \sqrt{\min\{1/\boldsymbol{v}_1, 1/\boldsymbol{v}_1^2\}}\right\}} \boldsymbol{I} \tag{A.7}$$

$$\succeq C \left[1 - \Phi\left(r^{3/2}\right)\right]^4 \frac{r^{-2}}{\max\{4, 16r^4 + \sqrt{r}\}} \boldsymbol{I}, \tag{A.8}$$

where we use the assumption that $\boldsymbol{v} \in \mathbb{D}_r$. Hence, $\ell(\boldsymbol{v})$ is η-strongly convex where η is decreasing function of r.

We next prove the smoothness of $\ell(\boldsymbol{v})$ to complete the proof. We start by considering the Hessian of ℓ. Since $\nabla \ell(\boldsymbol{v}) = \boldsymbol{g} - \boldsymbol{h}(\boldsymbol{v})$, we have

$$\nabla^2 \ell(\boldsymbol{v}) = -\nabla \boldsymbol{h}(\boldsymbol{v}) = \nabla \mathbb{E}_{y \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1}, \frac{1}{\boldsymbol{v}_1}\right)} \left\{ \begin{bmatrix} -y^2/2 & y \end{bmatrix}^\mathsf{T} \right\} \tag{A.9}$$

$$\preceq -\left\{ \frac{1}{2} \frac{\partial}{\partial \boldsymbol{v}_1} \left[\mathbb{E}_{y \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1}, \frac{1}{\boldsymbol{v}_1}\right)} \{y^2\} \right] - \frac{\partial}{\partial \boldsymbol{v}_2} \left[\mathbb{E}_{y \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1}, \frac{1}{\boldsymbol{v}_1}\right)} \{y\} \right] \right\} \boldsymbol{I}, \tag{A.10}$$

we use fact that the matrix $\nabla^2 \ell(\boldsymbol{v})$ is a positive semi-definite matrix from Eq. (A.7), and thus, its largest eigenvalue is upper bounded by the sum of its eigenvalues. We simplify each term in Eq. (A.10) as follows. The first term of Eq. (A.10) is bounded as

$$\frac{\partial}{\partial \boldsymbol{v}_1} \left[\mathbb{E}_{y \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1}, \frac{1}{\boldsymbol{v}_1}\right)} \{y^2\} \right] = \frac{\partial}{\partial \boldsymbol{v}_1} \left[\frac{\int_{y>0} y^2 \sqrt{\frac{\boldsymbol{v}_1}{2\pi}} \exp\left(-\frac{\boldsymbol{v}_1}{2} y^2 + \boldsymbol{v}_2 y\right) dy}{\int_{z>0} \sqrt{\frac{\boldsymbol{v}_1}{2\pi}} \exp\left(-\frac{\boldsymbol{v}_1}{2} z^2 + \boldsymbol{v}_2 z\right) dz} \right] \tag{A.11}$$

$$= \mathbb{E}_{y \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1}, \frac{1}{\boldsymbol{v}_1}\right)} \left\{ -\frac{y^4}{2} - y^2 \mathbb{E}_{z \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1}, \frac{1}{\boldsymbol{v}_1}\right)} \left\{ -\frac{z^2}{2} \right\} \right\} \tag{A.12}$$

$$\geq -\frac{1}{2} \mathbb{E}_{y \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1}, \frac{1}{\boldsymbol{v}_1}\right)} \{y^4\}. \tag{A.13}$$

Similarly, the second term of Eq. (A.10) is bounded as

$$\frac{\partial}{\partial \boldsymbol{v}_2}\left[\mathbb{E}_{y\sim\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)}\{y\}\right] = \frac{\partial}{\partial \boldsymbol{v}_2}\left[\frac{\int_{y>0} y\sqrt{\frac{\boldsymbol{v}_1}{2\pi}}\exp\left(-\frac{\boldsymbol{v}_1}{2}y^2+\boldsymbol{v}_2 y\right)dy}{\int_{z>0}\sqrt{\frac{\boldsymbol{v}_1}{2\pi}}\exp\left(-\frac{\boldsymbol{v}_1}{2}z^2+\boldsymbol{v}_2 z\right)dz}\right] \tag{A.14}$$

$$= \mathbb{E}_{y\sim\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)}\left\{y^2 + y\mathbb{E}_{z\sim\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)}\{-z\}\right\} \tag{A.15}$$

$$\leq \left[\mathbb{E}_{y\sim\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)}\{y^2\}\right]. \tag{A.16}$$

Substituting Eq. (A.13) and Eq. (A.16) into Eq. (A.10), we arrive at

$$\nabla^2\ell(\boldsymbol{v}) \preceq \left[\mathbb{E}_{y\sim\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)}\{y^4/2\} + \left(\mathbb{E}_{y\sim\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)}\{y\}\right)^2\right]\boldsymbol{I}. \tag{A.17}$$

Here, the moments of the distribution $\mathcal{N}^+\left(\frac{\boldsymbol{v}_2}{\boldsymbol{v}_1},\frac{1}{\boldsymbol{v}_1}\right)$ are finite and depend only on $\boldsymbol{v}$ which belongs to the bounded region $\mathbb{D}_r$ whose volume increases with r. Thus, we conclude that $\ell(\boldsymbol{v})$ is an $L-$smooth function where $L > 0$ is an increasing function of r. Hence, the proof is completed.

□

A.3 Proof of Proposition 2.2

To bound the estimation error of the robust GD algorithm, we use a constant step size $\gamma(k) = 1/L$ where $L > 0$ is defined in Proposition 2.1. The estimation error $\|\boldsymbol{v}(k+1) - \boldsymbol{v}^*\|$ in the $(k+1)$-th iteration is bounded as follows using Eq. (2.7),

$$\|\boldsymbol{v}(k+1) - \boldsymbol{v}^*\| = \left\|P\left(\boldsymbol{v}(k) - \frac{1}{L}\left[\tilde{\boldsymbol{g}} - \boldsymbol{h}(\boldsymbol{v}(k))\right]\right) - P(\boldsymbol{v}^*)\right\| \tag{A.18}$$

$$\leq \left\|\boldsymbol{v}(k) - \frac{1}{L}\left[\tilde{\boldsymbol{g}} - \boldsymbol{h}(\boldsymbol{v}(k))\right] - \boldsymbol{v}^*\right\| \tag{A.19}$$

$$\leq \left\|\boldsymbol{v}(k) - \frac{1}{L}\nabla\ell(\boldsymbol{v}(k)) - \boldsymbol{v}^*\right\| + \frac{1}{L}\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\|, \tag{A.20}$$

where Eq. (A.19) follows from the non-expansiveness of the projection $P(\cdot)$ defined in Eq. (2.9) , and Eq. (A.20) follows because $\nabla\ell(\boldsymbol{v}(k)) = \boldsymbol{g} - \boldsymbol{h}(\boldsymbol{v}(k))$, where $\boldsymbol{g}$ and $\boldsymbol{h}(\boldsymbol{v}(k))$ are defined in

Eq. (2.3) and Eq. (2.4) , respectively. We further upper bound the first term in Eq. (A.20) as

$$\left\|\boldsymbol{v}(k)-\frac{1}{L}\nabla\ell(\boldsymbol{v}(k))-\boldsymbol{v}^*\right\|^2 = \|\boldsymbol{v}(k)-\boldsymbol{v}^*\|^2+\frac{1}{L^2}\|\nabla\ell(\boldsymbol{v}(k))\|^2-2\frac{1}{L}\left(\boldsymbol{v}(k)-\boldsymbol{v}^*\right)^{\mathsf{T}}\nabla\ell(\boldsymbol{v}(k)) \tag{A.21}$$

$$\leq\left(1-\frac{2\eta}{\eta+L}\right)\|\boldsymbol{v}(k)-\boldsymbol{v}^*\|^2+\left(\frac{1}{L^2}-\frac{2}{L(\eta+L)}\right)\|\nabla\ell(\boldsymbol{v}(k))\|^2 \tag{A.22}$$

$$\leq\left(\frac{L}{\eta+L}\right)^2\|\boldsymbol{v}(k)-\boldsymbol{v}^*\|^2, \tag{A.23}$$

where we use Lemma A.2, Proposition 2.1 to bound the last term of Eq. (A.21) and the fact that $\nabla\ell(\boldsymbol{v}^*)=\mathbf{0}$ to obtain Eq. (A.22). Also, Eq. (A.23) is due to $\eta\leq L$ which follows from the definitions of strong-convexity and smoothness. Further, combining Eq. (A.20) and Eq. (A.23), we obtain

$$\|\boldsymbol{v}(K)-\boldsymbol{v}^*\|=\left(\frac{L}{\eta+L}\right)\|\boldsymbol{v}(K-1)-\boldsymbol{v}^*\|+\frac{1}{L}\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\| \tag{A.24}$$

$$\leq\left(\frac{L}{\eta+L}\right)^K\|\boldsymbol{v}(0)-\boldsymbol{v}^*\|+\frac{1}{L}\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\|\sum_{k=0}^{K-1}\left(\frac{L}{\eta+L}\right)^k \tag{A.25}$$

$$\leq\left(\frac{L}{\eta+L}\right)^K\sqrt{(r-1/r)^2+r^2}+\left(\frac{\eta+L}{\eta L}\right)\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\|, \tag{A.26}$$

where we use the fact that $\boldsymbol{v}(0),\boldsymbol{v}^*\in\mathbb{D}_r$. Thus, the rest of the proof is devoted to bounding the error $\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\|$ in the gradient computation to arrive at the desired result.

To bound the error in the gradient, we recall from Eq. (2.3) and Eq. (2.6) that

$$\|\tilde{\boldsymbol{g}}-\boldsymbol{g}\|=\left\|\operatorname{med}\left\{\tilde{\boldsymbol{g}}^{(1)},\tilde{\boldsymbol{g}}^{(2)},\ldots,\tilde{\boldsymbol{g}}^{(B)}\right\}-\begin{bmatrix}\left(\sigma_{\text{trun}}^{*2}-\mu_{\text{trun}}^{*2}\right)/2 & -\mu_{\text{trun}}^*\end{bmatrix}\right\|, \tag{A.27}$$

where $\tilde{\boldsymbol{g}}^{(b)}$ is defined in Eq. (2.5), and we define $B=|\mathcal{X}^+|/N_B$, and μ_{trun}^* and $\sigma_{\text{trun}}^{*2}$ are the mean and variance of the true distribution $\mathcal{N}^+\left(\frac{\boldsymbol{v}_2^*}{\boldsymbol{v}_1^*}, \frac{1}{\boldsymbol{v}_2^*}\right)$, respectively. Then, we have

$$\mathbb{P}\left\{\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\| \le \epsilon\right\} \ge \mathbb{P}\left\{\sum_{b=1}^{B} \mathbb{1}\left(\left\|\tilde{\boldsymbol{g}}^{(b)} - \boldsymbol{g}\right\| \le \epsilon\right) \ge B/2\right\}. \tag{A.28}$$

Further, from Lemma A.4, for any $\epsilon > 0$ and $0 \le \delta < 1/2$, if $N_B = \tilde{O}\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\Psi}\right)\log^2\left(\frac{1}{\delta}\right)\right)$,

$$\mathbb{P}\left\{\left\|\tilde{\boldsymbol{g}}^{(b)} - \boldsymbol{g}\right\| \le \epsilon \middle| \forall x \in \mathcal{X}^{+(b)}, x \sim \mathcal{N}^+\left(\frac{\boldsymbol{v}_2^*}{\boldsymbol{v}_1^*}, \frac{1}{\boldsymbol{v}_2^*}\right)\right\} \ge 1 - \delta. \tag{A.29}$$

Here, $\Psi = \int_{x>0}\sqrt{\frac{\boldsymbol{v}_2^*}{2\pi}}\exp\left(-\frac{1}{2}\left(x - \frac{\boldsymbol{v}_2^*}{\boldsymbol{v}_1^*}\right)^2\boldsymbol{v}_2^*\right)dx$, and Ψ can be bounded using a function of r since $\boldsymbol{v}^* \in \mathbb{D}_r$. Therefore, if $N_B = \tilde{O}\left(\frac{1}{\epsilon^2}\log^2\left(\frac{1}{\delta}\right)\right)$,

$$\mathbb{P}\left\{\left\|\tilde{\boldsymbol{g}}^{(b)} - \boldsymbol{g}\right\| \le \epsilon\right\} \ge \left(p^+\right)^{N_B}(1 - \delta). \tag{A.30}$$

Consequently, the random variable $\sum_{b=1}^{B}\mathbb{1}\left(\left\|\tilde{\boldsymbol{g}}^{(b)} - \boldsymbol{g}\right\| \le \epsilon\right)$ first-order stochastically dominates the binomial random variable $T \sim \text{binom}\left(B, \left(p^+\right)^{N_B}(1 - \delta)\right)$. Hence, from Eq. (A.28),

$$\mathbb{P}\left\{\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\| \le \epsilon\right\} \ge \mathbb{P}\left\{T \ge B/2\right\}. \tag{A.31}$$

Further, we choose

$$N_B \le \frac{1}{\log p^+}\log\left(\frac{1 + \zeta}{2(1 - \delta)}\right), \tag{A.32}$$

so that $\left(p^+\right)^{N_B}(1 - \delta) \ge 1/2 + \zeta/2$, for any $\zeta \in (0, 1 - 2\delta)$. We note that such a choice exists for any $p^+ > 1/2$. Then, the Hoeffding's inequality leads to the following:

$$\begin{aligned}
\mathbb{P}\left\{\|\tilde{\boldsymbol{g}} - \boldsymbol{g}\| \le \epsilon\right\} &\ge \mathbb{P}\left\{T \ge B/2\right\} && \text{(A.33)}\\
&\ge 1 - \exp\left(-2B(\left(p^+\right)^{N_B}(1 - \delta) - 1/2)^2\right) && \text{(A.34)}\\
&\ge 1 - \exp\left(-B\zeta^2/2\right). && \text{(A.35)}
\end{aligned}$$

Combining Eq. (A.26) and Eq. (A.35), we arrive at

$$\|\boldsymbol{v}(k) - \boldsymbol{v}^*\| \leq \left(\frac{L}{\eta + L}\right)^K \sqrt{(r - 1/r)^2 + r^2} + \left(\frac{\eta + L}{\eta L}\right) \epsilon, \tag{A.36}$$

with probability exceeding $1 - \delta$ if $B = \Omega\left(\frac{1}{\zeta^2} \log\left(\frac{1}{\delta}\right)\right)$. Hence, if we choose the number of iterations as

$$K \geq \left[\log\left(1 + \frac{\eta}{L}\right)\right]^{-1} \log\left[\epsilon^{-1}\sqrt{(r - 1/r)^2 + r^2}\right] = \Omega\left(\log 1/\epsilon\right), \tag{A.37}$$

we arrive at the desired result. Hence, the proof is complete. □

A.4 Proof of Theorem 2.1

To prove the overall error bound, we first bound the error in the bias and the diagonal entries of $\hat{\boldsymbol{\Sigma}}$ using Proposition 2.2. Then, we bound the error in the angle estimation using Lemmas A.5, A.6. Finally, we combine the two error bounds to compute the overall estimation error.

To apply Proposition 2.2, we need p^+ which is the probability of a true sample in the set of positive i-th elements of the observed samples, $\mathcal{X}_i^+$. We have

$$\begin{aligned}
p^+ &= \mathbb{P}\left\{\boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right) | \boldsymbol{x}_i > 0, \boldsymbol{x} \in \mathcal{X}\right\} && \text{(A.38)}\\
&= \frac{\mathbb{P}\left\{\boldsymbol{x}_i > 0, \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right) | \boldsymbol{x} \in \mathcal{X}\right\}}{\mathbb{P}\left\{\boldsymbol{x}_i > 0, \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right) | \boldsymbol{x} \in \mathcal{X}\right\} + \mathbb{P}\left\{\boldsymbol{x}_i > 0, \boldsymbol{x} \sim \mathcal{D}_{\text{out}} | \boldsymbol{x} \in \mathcal{X}\right\}} && \text{(A.39)}\\
&\geq \frac{p\mathbb{P}\left\{\boldsymbol{x}_i > 0 | \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right), \boldsymbol{x} \in \mathcal{X}\right\}}{p\mathbb{P}\left\{\boldsymbol{x}_i > 0 | \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right), \boldsymbol{x} \in \mathcal{X}\right\} + \mathbb{P}\left\{\boldsymbol{x} \sim \mathcal{D}_{\text{out}} | \boldsymbol{x} \in \mathcal{X}\right\}} && \text{(A.40)}\\
&= \frac{p\mathbb{P}\left\{\boldsymbol{x}_i > 0 | \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right), \boldsymbol{x} \in \mathcal{X}\right\}}{p\mathbb{P}\left\{\boldsymbol{x}_i > 0 | \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right), \boldsymbol{x} \in \mathcal{X}\right\} + (1 - p)}. && \text{(A.41)}
\end{aligned}$$

We notice that the above lower bound is an increasing function of $\mathbb{P}\left\{\boldsymbol{x}_i > 0 | \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right), \boldsymbol{x} \in \mathcal{X}\right\}$

and because $\boldsymbol{b}$ is nonnegative, $\mathbb{P}\left\{\boldsymbol{x}_i > 0 | \boldsymbol{x} \sim \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right), \boldsymbol{x} \in \mathcal{X}\right\} \geq 1/2$. Consequently,

$$p^+ \geq \frac{p}{2-p} \geq \frac{p}{2}. \tag{A.42}$$

Thus, if $p > 2/3$, we have $p^+ > 1/2$, and from Proposition 2.2, we obtain for any $i \in [d]$,

$$\left|\hat{\boldsymbol{b}}_i - \boldsymbol{b}_i\right| \leq \epsilon \left\|\boldsymbol{W}_i\right\| \tag{A.43}$$

$$\left|\hat{\boldsymbol{\Sigma}}_{ii} - \left\|\boldsymbol{W}_i\right\|^2\right| \leq \epsilon \left\|\boldsymbol{W}_i\right\|^2, \tag{A.44}$$

holds with probability $1-\delta$ if $K = \Omega\left(\log 1/\epsilon\right)$, $\left|\mathcal{X}_i^+\right| = \Omega\left(\frac{1}{\zeta^2\epsilon^2}\log^3\frac{1}{\delta}\right)$ and $N_B = \tilde{O}\left(\frac{1}{\epsilon^2}\log^2\frac{1}{\delta}\right)$ and $N_B \leq \frac{1}{\log(p/2)}\log\frac{1+\zeta}{2(1-\delta)}$. We next invoke Hoeffding's inequality (Lemma A.3) to compute the $N = |\mathcal{X}|$ from $\left|\mathcal{X}_i^+\right|$. For this, we have

$$\mathbb{P}\left(\boldsymbol{x}_i > 0 | \boldsymbol{x} \in \mathcal{X}\right) \geq \mathbb{P}\left(\boldsymbol{x}_i > 0 | \boldsymbol{x} \sim \mathcal{D}(\boldsymbol{W}, \boldsymbol{b}), \boldsymbol{x} \in \mathcal{X}\right) \mathbb{P}\left(\boldsymbol{x} \sim \mathcal{D}(\boldsymbol{W}, \boldsymbol{b}), \boldsymbol{x} \in \mathcal{X}\right) \geq p/2. \tag{A.45}$$

Using Hoeffding's inequality (Lemma A.3), we obtain for any $N^+ = \Omega\left(\frac{1}{\zeta^2\epsilon^2}\log^3\frac{1}{\delta}\right) > 0$,

$$\mathbb{P}\left\{\left|\mathcal{X}_i^+\right| \geq N^+\right\} = \mathbb{P}\left\{\sum_{n=1}^{N} \mathbb{1}(\boldsymbol{x}_i^{(n)} > 0) \geq N^+ \middle| \boldsymbol{x}^n \in \mathcal{X}, n = 1, 2, \ldots, N\right\} \tag{A.46}$$

$$\geq 1 - \exp\left[-2\left(\frac{p}{2} - \frac{N^+}{N}\right)^2 N\right] \tag{A.47}$$

$$\geq 1 - \exp\left[-Np^2/2 + 2pN^+\right] \geq 1 - \delta, \tag{A.48}$$

if $N = \Omega\left(1/p^2 \log(1/\delta) + 2N^+/p\right) = \Omega\left(\frac{1}{p^2}\log\frac{1}{\delta} + \frac{1}{p\zeta^2\epsilon^2}\log^3\frac{1}{\delta}\right)$.

Next, we bound the error in the estimate $\hat{\theta}_{ij}$. For this, we define the functions $\beta_{ij} : \mathbb{R}^d \times \mathbb{R}^d \to \{0, 1\}$ for any pair $(i, j) \in [d] \times [d]$ as follows:

$$\beta_{ij}(\boldsymbol{x}; \hat{\boldsymbol{b}}) = \mathbb{1}(\boldsymbol{x}_i > \hat{\boldsymbol{b}}_i \text{ and } \boldsymbol{x}_j > \hat{\boldsymbol{b}}_j). \tag{A.49}$$

Then, we deduce the following relations:

$$\frac{1}{2\pi}\left|\hat{\theta}_{ij} - \theta_{ij}\right| = \left|\frac{1}{N}\sum_{n=1}^{N}\beta_{ij}(\boldsymbol{x}^{(n)};\hat{\boldsymbol{b}}) - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\boldsymbol{b})]\right| \tag{A.50}$$

$$\leq \left|\frac{1}{N}\sum_{n=1}^{N}\beta_{ij}(\boldsymbol{x}^{(n)};\hat{\boldsymbol{b}}) - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_p(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})]\right| + \left|\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_p(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\boldsymbol{b})]\right|. \tag{A.51}$$

Here, we simplify the first term of Eq. (A.51) using the Hoeffding's inequality (Lemma A.3) as

$$\mathbb{P}\left\{\left|\frac{1}{N}\sum_{n=1}^{N}\beta_{ij}(\boldsymbol{x}^{(n)};\hat{\boldsymbol{b}}) - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_p(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})]\right| \leq \epsilon\right\} \geq 1 - 2e^{-2\epsilon^2 N}, \tag{A.52}$$

for any $\epsilon > 0$. Further, we bound the second term of Eq. (A.51) as

$$\begin{aligned}
&\left|\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_p(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\boldsymbol{b})]\right| \\
&\quad = \left|p\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})] + (1-p)\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{out}}}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\boldsymbol{b})]\right| && \text{(A.53)}\\
&\quad = p\left|\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\boldsymbol{b})]\right| \\
&\qquad + (1-p)\left|\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{out}}}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})] - \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\boldsymbol{b})]\right| && \text{(A.54)}\\
&\quad \leq p\epsilon + (1-p) \leq \epsilon + (1-p), && \text{(A.55)}
\end{aligned}$$

with probability at least $1 - 2\delta$. Here, we use Eq. (A.43) and Lemma A.6 to obtain the first term. The second term follows from the fact that both $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{out}}}[\beta_{ij}(\boldsymbol{x};\hat{\boldsymbol{b}})]$ and $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{W},\boldsymbol{b})}[\beta_{ij}(\boldsymbol{x};\boldsymbol{b})]$ belongs to $[0,1]$, and thus, their difference lies in $[-1,1]$. Substituting Eq. (A.52) and Eq. (A.55) into Eq. (A.51), and since $\cos(\cdot)$ has Lipschitz constant 1, we derive that

$$\mathbb{P}\left\{\left|\cos(\hat{\theta}_{ij}) - \cos(\theta_{ij})\right| \leq \epsilon + (1-p)\right\} \geq 1 - 3\delta, \tag{A.56}$$

when $N = \Omega\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right) + \Omega\left(\frac{1}{p^2}\log\frac{1}{\delta} + \frac{1}{p\zeta^2\epsilon^2}\log^3\frac{1}{\delta}\right) = \Omega\left(\frac{1}{p^2}\log\frac{1}{\delta} + \frac{1}{p\zeta^2\epsilon^2}\log^3\frac{1}{\delta}\right)$.

Finally, we combine the error in angle and row norm estimation of $\boldsymbol{W}\boldsymbol{W}^\intercal$ to obtain the final bound. Without loss of generality, suppose that $\cos(\theta_{ij}) \geq 0$. Then, from Eq. (A.44) and Eq. (A.56), we have with probability at least $1 - 3\delta$

$$\begin{aligned}
\hat{\boldsymbol{\Sigma}}(i,j) &= \sqrt{\hat{\boldsymbol{\Sigma}}(i,i)\hat{\boldsymbol{\Sigma}}(j,j)}\cos(\hat{\theta}_{ij}) && \text{(A.57)}\\
&\leq (1+\epsilon)^2 \|\boldsymbol{W}_i\| \|\boldsymbol{W}_j\| \left[\cos(\theta_{ij}) + \epsilon + 1 - p\right] && \text{(A.58)}\\
&= \boldsymbol{W}_i^\intercal \boldsymbol{W}_j + \left[(2\epsilon+\epsilon^2)\cos(\theta_{ij}) + (1+\epsilon)^2(\epsilon + 1 - p)\right] \|\boldsymbol{W}_i\| \|\boldsymbol{W}_j\| && \text{(A.59)}\\
&\leq \boldsymbol{W}_i^\intercal \boldsymbol{W}_j + \left[3\epsilon\cos(\theta_{ij}) + 4\epsilon + (1+3\epsilon)(1-p)\right] \|\boldsymbol{W}_i\| \|\boldsymbol{W}_j\| && \text{(A.60)}\\
&\leq \boldsymbol{W}_i^\intercal \boldsymbol{W}_j + \left[10\epsilon + (1-p)\right] \|\boldsymbol{W}_i\| \|\boldsymbol{W}_j\|. && \text{(A.61)}
\end{aligned}$$

Similarly, we derive a lower bound for $\hat{\boldsymbol{\Sigma}}(i,j)$ which leads to the following:

$$\begin{aligned}
\left\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{W}\boldsymbol{W}^\intercal\right\|^2 &= \sum_{i,j=1}^{d} \left|\hat{\boldsymbol{\Sigma}}(i,j) - \boldsymbol{W}_i^\intercal \boldsymbol{W}_j\right|^2 && \text{(A.62)}\\
&\leq \sum_{i,j=1}^{d} \left[10\epsilon + (1-p)\right]^2 \|\boldsymbol{W}_i\|^2 \|\boldsymbol{W}_j\|^2 && \text{(A.63)}\\
&\leq \left[10\epsilon + (1-p)\right]^2 \|\boldsymbol{W}\|^4, && \text{(A.64)}
\end{aligned}$$

with probability at least $1 - 3d^2\delta$. We note that we obtain the probability using the union bound over all the $\binom{d}{2} \leq d^2$ pairs of (i,j). Thus, the sample complexity of Algorithm 1 is given by $N = \Omega\left(\frac{1}{p^2}\log\frac{d}{\delta} + \frac{1}{p\zeta^2\epsilon^2}\log^3\frac{d}{\delta}\right)$.

Next, we complete the proof by analyzing the time complexity. The complexity of the first for-loop containing the robust GD is given by $O(dKN\log(B))$ due to the median computation, where $B = \Omega\left(\frac{1}{\zeta^2}\log\frac{1}{\delta}\right)$ is the number of batches from Proposition 2.2. The complexity of the second for-loop is $O(d^2N)$, and the overall time complexity is $\Omega\left(\frac{d^2}{p^2}\log\frac{d}{\delta} + \frac{d^2}{p\zeta^2\epsilon^2}\log^3\frac{d}{\delta}\right)$. Further, the space complexity is defined as the space required to store N samples and the covariance estimate, the space complexity is $O(Nd + d^2)$. Hence, the proof of Theorem 2.1 is completed. $\square$

A.5 Proof of Corollary 2.1

From the proof of [102, Corollary 1], it is easy to show that under the assumptions of Theorem 2.1 with $\tilde{\epsilon} = \epsilon$,

$$\mathrm{TV}\left(\mathcal{D}\left(\hat{\boldsymbol{\Sigma}}^{1/2}, \hat{\boldsymbol{b}}\right), \mathcal{D}\left(\boldsymbol{W}, \boldsymbol{b}\right)\right) \leq \sqrt{(\tilde{\epsilon}+1-p)^2\kappa^2 d^2/2 + \tilde{\epsilon}^2\kappa d} \leq (\tilde{\epsilon}+1-p)\kappa d, \tag{A.65}$$

when $0 < \tilde{\epsilon}+1-p \leq 1/2$. Defining ϵ in the corollary to be $(\tilde{\epsilon}+1-p)\kappa d$, we arrive at the desired result. □

A.6 Proof of Theorem 2.2

To prove the result, we first observe that for any two distributions from $\mathcal{C}$ denoted by $\mathcal{D}\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right)$ and $\mathcal{D}\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right)$, there exist two outlier distributions $\mathcal{D}_{\text{out}}$ and $\tilde{\mathcal{D}}_{\text{out}}$ given by

$$\mathcal{D}_{\text{out}} = \frac{p}{1-p}\left[\mathcal{D}\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right) - \mathcal{D}\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right)\right] \mathbb{1}\left(\mathcal{D}\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right) \geq \mathcal{D}\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right)\right) \tag{A.66}$$

$$\tilde{\mathcal{D}}_{\text{out}} = \frac{p}{1-p}\left[\mathcal{D}\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right) - \mathcal{D}\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right)\right] \mathbb{1}\left(\mathcal{D}\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right) < \mathcal{D}\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right)\right). \tag{A.67}$$

Then, the distribution of the available data samples satisfies

$$p\mathcal{D}\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right) + (1-p)\mathcal{D}_{\text{out}} = p\mathcal{D}\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right) + (1-p)\tilde{\mathcal{D}}_{\text{out}}, \tag{A.68}$$

which ensures that no algorithm can distinguish between $\mathcal{D}\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right)$ and $\mathcal{D}\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right)$. Further, let $\boldsymbol{b}$ and $\tilde{\boldsymbol{b}}$ be such that the total variation distance between $\mathcal{D}^N\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right)$ and $\mathcal{D}^N\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right)$ satisfies

$$\frac{1-p}{p} = \mathrm{TV}\left(\mathcal{D}^N\left(\boldsymbol{b}, \sigma^2\boldsymbol{I}\right), \mathcal{D}^N\left(\tilde{\boldsymbol{b}}, \sigma^2\boldsymbol{I}\right)\right), \tag{A.69}$$

where $\mathcal{D}^N$ denotes the joint distribution of N iid samples from $\mathcal{D}$. From the results of the data processing inequality for f-divergence: $\mathrm{TV}(f(X), f(Y)) \leq \mathrm{TV}(X, Y)$ for any function f and random variables X, Y over the same space, and therefore, we obtain

$$\frac{1-p}{p} \leq \mathrm{TV}\left(\mathcal{N}^N\left(\boldsymbol{b}, \sigma^2 \boldsymbol{I}\right), \mathcal{N}^N\left(\tilde{\boldsymbol{b}}, \sigma^2 \boldsymbol{I}\right)\right) \leq \frac{N\left\|\boldsymbol{b}-\tilde{\boldsymbol{b}}\right\|^2}{2\sigma^2}. \tag{A.70}$$

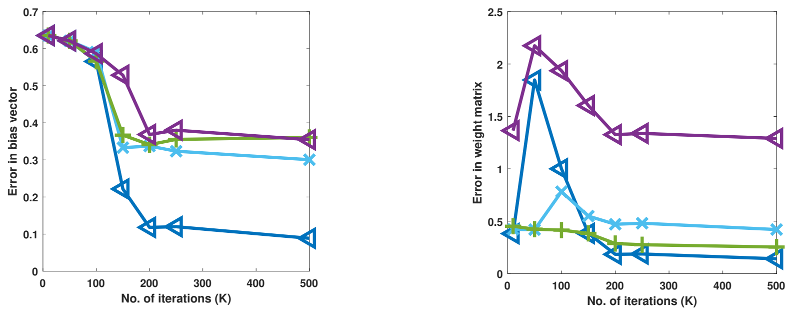
Hence, we derive

$$\left\|\boldsymbol{b}-\tilde{\boldsymbol{b}}\right\| \geq \sqrt{\frac{2\sigma^2(1-p)}{pN}} \geq \sqrt{\frac{2\sigma^2}{pN}}(1-\sqrt{p}). \tag{A.71}$$

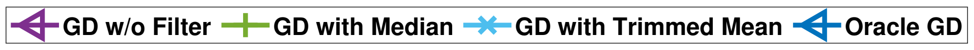
This result implies that when $\|\boldsymbol{W}\|^{-1}\left\|\boldsymbol{b}-\tilde{\boldsymbol{b}}\right\| = \Omega\left(\frac{1}{\sqrt{pN}}(1-\sqrt{p})\right)$, the two distributions cannot be distinguished in the worst case in the presence of outliers. Further, from the results on estimation error from uncorrupted data due to [102, Theorem 2], we have $\|\boldsymbol{W}\|^{-1}\left\|\boldsymbol{b}-\tilde{\boldsymbol{b}}\right\| = \Omega\left(\frac{1}{\sqrt{N}}\right)$. Combining the two results, we have $\|\boldsymbol{W}\|^{-1}\left\|\boldsymbol{b}-\tilde{\boldsymbol{b}}\right\| = \Omega\left(\frac{1}{\sqrt{pN}}\right)$. □

A.7 Additional Simulation Results: Errors vs number of iterations

Figs. A.1 and A.2 show how the estimation error of different schemes varies with K. From our experiments, we observe that the errors first decrease with the number of iterations and then after a certain number of iterations, the errors flatten. Based on this observation, we chose the number of iterations. Note that we discard the zero entries and only consider the set of positive samples $\mathcal{X}^+$ for the number of iterations as they do not convey any information about the row norms of $\mathbf{W}$ and bias vector $\mathbf{b}$. We observe that the parameter estimation errors computed using Oracle SGD and Oracle GD schemes decrease as the number of samples increases. Also, the estimation errors computed using our proposed schemes with median and trimmed mean based filters perform better than GD without filter. Hence, the filters mitigate the effect of arbitrary outliers and having more samples reduces the estimation errors. Further, we observe that the errors first decrease with the number of iterations and then after a certain number of iterations, the errors flatten.

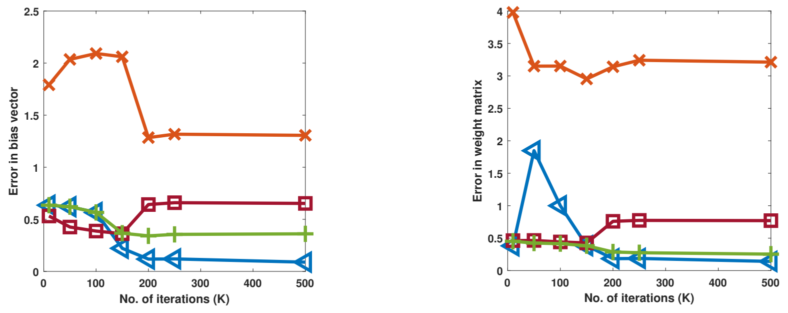


(a)

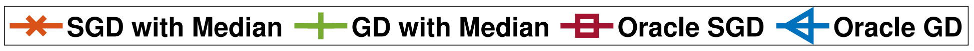


(b)

Fig. A.1: Comparison of the different GD schemes as a function of K for $p = 0.95$, d = 5, and N = 20000.



(a)



(b)

Fig. A.2: Comparison of GD and SGD schemes as a function of K for $p = 0.95$, d = 5, and N = 20000.

We consider a more general model where the nodes can choose different batch sizes, B_t, for $t = 1, 2, \dots, T$. For varying batch sizes, Algorithm 3 and Algorithm 4 remain the same except for the Byzantine filtering constant $\mathfrak{T}_\mu$ which is evaluated inside the epochs and is a function of B_t. Our proof follows the structure similar to that in [71] with a few major differences. The problem considered in [71] was a finite sum problem with all the gradient computations designated at the CN. The key novelty in our proofs lie in proving the boundedness of the norm square of the error term e_t (see Lemma A.16 and Lemma A.20).

A.8 Proof of Theorem 3.1

Noting that f is Lipschitz smooth from Eq. (3.1), taking the expectation with respect to $\xi_{n,t}$ on both sides and applying Lemma A.7 to the last term, we have

$$\begin{aligned} \mathbb{E}_{\xi_{n,t}} f(x_{n+1,t}) \leq & f(x_{n,t}) - \eta_t(1 - L\eta_t)\|\nabla f(x_{n,t})\|^2 \\ & - \eta_t\langle e_t, \nabla f(x_{n,t})\rangle + \frac{L^3\eta_t^2}{2}\|x_{n,t} - x_{0,t}\|^2 + L\eta_t^2\|e_t\|^2, \end{aligned} \tag{A.72}$$

where $e_t = \mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})$. Let $\mathbb{E}_t$ denote the expectation with respect to all $\xi_{1,t}, \xi_{2,t}, \dots$ given N_t. As $\xi_{1,t}, \xi_{2,t}, \dots$ are independent of N_t, $\mathbb{E}_t$ is equivalent to expectation with respect to $\xi_{1,t}, \xi_{2,t}, \dots$, we have

$$\begin{aligned} \mathbb{E}_t f(x_{n+1,t}) \leq & \mathbb{E}_t f(x_{n,t}) - \eta_t(1 - L\eta_t)\mathbb{E}_t\|\nabla f(x_{n,t})\|^2 \\ & - \eta_t\mathbb{E}_t\langle e_t, \nabla f(x_{n,t})\rangle + \frac{L^3\eta_t^2}{2}\mathbb{E}_t\|x_{n,t} - x_{0,t}\|^2 + L\eta_t^2\|e_t\|^2. \end{aligned}$$

Next, taking $n = N_t$, denoting $\mathbb{E}_{N_t}$ as the expectation with respect to N_t, rearranging the terms and utilizing Fubini's theorem [71], we get

$$\begin{aligned}
\eta_t(1 - L\eta_t)\mathbb{E}_{N_t}\mathbb{E}_t\|\nabla f(x_{N_t,t})\|^2 \leq{}& \mathbb{E}_{N_t}\mathbb{E}_t f(x_{N_t,t}) \\
&- \mathbb{E}_{N_t}\mathbb{E}_t f(x_{N_t+1,t}) - \eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle e_t, \nabla f(x_{N_t,t})\rangle \\
&+ \frac{L^3\eta_t^2}{2}\mathbb{E}_{N_t}\mathbb{E}_t\|x_{N_t,t} - x_{0,t}\|^2 + L\eta_t^2\|e_t\|^2 \\
=&\frac{1}{B_t}[f(x_{0,t}) - \mathbb{E}_t\mathbb{E}_{N_t} f(x_{N_t,t})] - \eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle e_t, \nabla f(x_{N_t,t})\rangle \\
&+ \frac{L^3\eta_t^2}{2}\mathbb{E}_{N_t}\mathbb{E}_t\|x_{N_t,t} - x_{0,t}\|^2 + L\eta_t^2\|e_t\|^2
\end{aligned}$$

where the last equality follows from Lemma A.10, Lemma A.11 and Fubini's theorem [71]. Further, taking expectation over all the randomness and using $x_{N_t,t} = \tilde{x}_t$ and $\tilde{x}_{t-1} = x_{0,t}$, we obtain

$$\begin{aligned}
\eta_t(1 - L\eta_t)\mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 \leq{}& \frac{1}{B_t}\mathbb{E}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] \\
&- \eta_t\mathbb{E}\langle e_t, \nabla f(\tilde{x}_t)\rangle + \frac{L^3\eta_t^2}{2}\mathbb{E}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 + L\eta_t^2\mathbb{E}\|e_t\|^2 \\
\overset{(a)}{=}{}& \frac{1}{B_t}\mathbb{E}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] + \frac{1}{B_t}\mathbb{E}\langle e_t, \tilde{x}_t - \tilde{x}_{t-1}\rangle \\
&+ \frac{L^3\eta_t^2}{2}\mathbb{E}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 + \eta_t(1 + L\eta_t)\mathbb{E}\|e_t\|^2,
\end{aligned}$$

where (a) is from Lemma A.8 and from Lemma A.9, we have

$$\begin{aligned}
\eta_t(1 - L\eta_t)\mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 \overset{(b)}{\leq}{}& \frac{1}{B_t}\mathbb{E}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] \\
&+ \left(\frac{1}{2\eta_t B_t}\left(-\frac{1}{B_t} + \eta_t^2 L^2\right)\right)\mathbb{E}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 \\
&- \frac{1}{B_t}\mathbb{E}\langle\nabla f(\tilde{x}_t), \tilde{x}_t - \tilde{x}_{t-1}\rangle + \frac{\eta_t}{B_t}\mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 \\
&+ \frac{\eta_t}{B_t}\mathbb{E}\|e_t\|^2 + \frac{L^3\eta_t^2}{2}\mathbb{E}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 + \eta_t(1 + L\eta_t)\mathbb{E}\|e_t\|^2.
\end{aligned}$$

Rearranging the terms yield

$$\begin{aligned}
&\eta_t \left(1 - L\eta_t - \frac{1}{B_t}\right) \mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 \\
&\quad + \left(\frac{1 - \eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2}{2\eta_t B_t^2}\right) \mathbb{E}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 \\
&\quad \le \frac{1}{B_t}\mathbb{E}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] + \frac{1}{B_t}\mathbb{E}\langle \nabla f(\tilde{x}_t), \tilde{x}_{t-1} - \tilde{x}_t\rangle \\
&\qquad\qquad + \eta_t \left(1 + L\eta_t + \frac{1}{B_t}\right) \mathbb{E}\|e_t\|^2.
\end{aligned}$$

Applying Young's inequality $\langle a, b\rangle \le \frac{\beta}{2}\|a\|^2 + \frac{1}{2\beta}\|b\|^2$ to $\mathbb{E}\langle \nabla f(\tilde{x}_t), \tilde{x}_{t-1} - \tilde{x}_t\rangle$ with $\beta = \frac{1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2}{\eta_t B_t}$, $a = \tilde{x}_{t-1} - \tilde{x}_t$ and $b = \nabla f(\tilde{x}_t)$, and from Lemma A.16, we have

$$\begin{aligned}
&\eta_t \left(1 - L\eta_t - \frac{1}{B_t} - \frac{1}{2\eta_t B_t \beta}\right) \mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 \\
&\quad \le \frac{1}{B_t}\mathbb{E}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] + \eta_t \left(1 + L\eta_t + \frac{1}{B_t}\right) \\
&\quad \times \left(\frac{4\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{272\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B_t}\right).
\end{aligned} \tag{A.73}$$

We need an appropriate choice of η_t to ensure that $1 - \eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2 > 0$ which implies $\frac{1}{2(1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2)} > \frac{1}{2}$. Furthermore, we choose η_t such that the following holds

$$1 - L\eta_t - \frac{1}{B_t} - \frac{1}{2(1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2)} \ge \frac{1}{4}.$$

Therefore, choice of η_t should satisfy the following conditions:

1. $\frac{1}{2} < \frac{1}{2(1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2)} \le \frac{5}{8}$,
2. $L\eta_t \le \frac{1}{16}$, and $\frac{1}{B_t} \le \frac{1}{16}$.

From above, condition (1) implies that $\eta_t^2 L^2 B_t + \eta_t^3 L^3 B_t^2 \le \frac{1}{5}$. Therefore, η_t should also satisfy $\eta_t^2 L^2 B_t \le \frac{1}{10}$ and $\eta_t^3 L^3 B_t^2 \le \frac{1}{10}$. Hence, we have $\eta_t \le \frac{1}{10^{1/2} L B_t^{1/2}}$ and $\eta_t \le \frac{1}{10^{1/3} L B_t^{2/3}}$. Furthermore, from condition (2), we have $\eta_t \le \frac{1}{16L}$ and $B_t \ge 16$. Therefore, we can choose $\eta_t \le \frac{1}{3LB_t^{2/3}}$

as we have

$$\frac{1}{3LB_t^{2/3}} \leq \min\left\{\frac{1}{16L}, \frac{1}{10^{1/3}LB_t^{2/3}}, \frac{1}{10^{1/2}LB_t^{1/2}}\right\}$$

for $B_t \geq 16$. This choice of η_t ensures the following

$$1 - L\eta_t - \frac{1}{B_t} - \frac{1}{2(1-\eta_t^2L^2B_t - \eta_t^3L^3B_t^2)} \geq \frac{1}{4}. \tag{A.74}$$

Substituting η_t and B_t above, we get

$$1 + \eta_t L + \frac{1}{B_t} \leq 1 + \frac{1}{3B_t^{2/3}} + \frac{1}{B_t} \leq 2. \tag{A.75}$$

Next, replacing Eq. (A.74) and Eq. (A.75) in Eq. (A.73), and substituting $\eta_t = \frac{1}{3LB_t^{2/3}}$ yields

$$\begin{aligned}\mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 \leq & \frac{12L\mathbb{E}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)]}{B_t^{1/3}} \\ & + \frac{32\mathcal{V}^2}{(1-\alpha)^2KB_t} + \frac{2176\alpha^2\mathcal{V}^2C}{(1-\alpha)^2B_t}.\end{aligned}$$

Summing over $t = 1, 2, \dots, T$ and choosing $\tilde{x}_a$ using Algorithm 3, we obtain

$$\begin{aligned}&\mathbb{E}\|\nabla f(\tilde{x}_a)\|^2 \\ &\quad \leq \frac{12L\mathbb{E}[f(\tilde{x}_0) - f(\tilde{x}^*)] + 32\mathcal{V}^2(1-\alpha)^{-2}K^{-1}\sum_{t=1}^T B_t^{\frac{-2}{3}}}{\sum_{t=1}^T B_t^{\frac{1}{3}}} \\ &\qquad + \frac{2176\alpha^2\mathcal{V}^2C(1-\alpha)^{-2}\sum_{t=1}^T B_t^{\frac{-2}{3}}}{\sum_{t=1}^T B_t^{\frac{1}{3}}}.\end{aligned}$$

For $B_t = B$, we have

$$\begin{aligned}
&\mathbb{E}\|\nabla f(\tilde{x}_a)\|^2 \\
&\leq \underbrace{\frac{12L\mathbb{E}[f(\tilde{x}_0) - f(\tilde{x}^*)]}{TB^{1/3}}}_{T=O\left(\frac{1}{\epsilon B^{1/3}}\right)} + \underbrace{\frac{32\mathcal{V}^2}{(1-\alpha)^2 KB}}_{B=O\left(\frac{1}{\epsilon K}\right)} + \underbrace{\frac{2176\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B}}_{B=O\left(\frac{\alpha^2}{\epsilon}\right)}.
\end{aligned}$$

A.9 Useful Lemmas for Proof in Appendix A.8

In this section, we present the following lemmas which are used in the proof of Appendix A.8.

Lemma A.7. *For the gradient $v_{n,t}$ computed by Algorithm 3, we have $\mathbb{E}_{\xi_{n,t}}\|v_{n,t}\|^2 \leq L^2\|x_{n,t} - x_{0,t}\|^2 + 2\|\nabla f(x_{n,t})\|^2 + 2\|e_t\|^2$.*

PROOF: From the definition of v_n^t we have:

$$v_n^t = \nabla f(x_n^t; \xi_n^t) - \nabla f(x_0^t; \xi_n^t) + \mu_t.$$

where $\mu_t = \frac{1}{|\mathcal{G}_t|}\sum_{k\in\mathcal{G}_t} \mu_t^{(k)}$. We define

$$e_t = \mu_t - \nabla f(x_0^t) = \mu_t - \nabla f(\tilde{x}_{t-1}).$$

This implies that we have

$$\mathbb{E}_{\xi_n^t} v_n^t = \nabla f(x_n^t) + e_t.$$

Now taking $\mathbb{E}_{\xi_n^t}\|v_n^t\|^2$ and using $\mathbb{E}\|Z\|^2 = \mathbb{E}\|Z - \mathbb{E}Z\|^2 + \|\mathbb{E}Z\|^2$, we have

$$\begin{aligned}
&\mathbb{E}_{\xi_n^t}\|v_n^t\|^2 \\
&= \mathbb{E}_{\xi_n^t}\|v_n^t - \mathbb{E}_{\xi_n^t} v_n^t\|^2 + \|\mathbb{E}_{\xi_n^t} v_n^t\|^2 \\
&= \mathbb{E}_{\xi_n^t}\|\nabla f(x_n^t;\xi_n^t) - \nabla f(x_0^t;\xi_n^t) - (\nabla f(x_n^t) - \nabla f(x_0^t))\|^2 \\
&\qquad\qquad\qquad + \|\nabla f(x_n^t) + e_t\|^2 \\
&\overset{(a)}{\le} \mathbb{E}_{\xi_n^t}\|\nabla f(x_n^t;\xi_n^t) - \nabla f(x_0^t;\xi_n^t) - (\nabla f(x_n^t) - \nabla f(x_0^t))\|^2 \\
&\qquad\qquad\qquad + 2\|\nabla f(x_n^t)\|^2 + 2\|e_t\|^2 \\
&\overset{(b)}{\le} \mathbb{E}_{\xi_n^t}\|\nabla f(x_n^t;\xi_n^t) - \nabla f(x_0^t;\xi_n^t)\|^2 + 2\|\nabla f(x_n^t)\|^2 + 2\|e_t\|^2 \\
&\overset{(c)}{\le} L^2\|x_n^t - x_0^t\|^2 + 2\|\nabla f(x_n^t)\|^2 + 2\|e_t\|^2.
\end{aligned}$$

where (a) follows from the definition of v_n^t and Lemma A.13, (b) follows from variance inequality and (c) follows from the Gradient Lipschitz continuity of $f(\cdot\,;\xi_n^t)$. ■

Lemma A.8. *For the error term e_t defined in the proof of Lemma A.7, we have the following equality*

$$\eta_t \mathbb{E}\langle e_t, \nabla f(\tilde{x}_t)\rangle = \frac{1}{B_t}\mathbb{E}\langle e_t, \tilde{x}_{t-1} - \tilde{x}_t\rangle - \eta_t \mathbb{E}\|e_t\|^2. \tag{A.76}$$

PROOF: Consider the term $M_n^t = \langle e_t, x_n^t - x_0^t\rangle$. From the definition of M_n^t we have

$$M_{n+1}^t - M_n^t = \langle e_t, x_{n+1}^t - x_n^t\rangle = -\eta_t\langle e_t, v_n^t\rangle.$$

Taking expectation w.r.t. ξ_n^t, we have

$$\begin{aligned}
\mathbb{E}_{\xi_n^t}(M_{n+1}^t - M_n^t) &= -\eta_t\langle e_t, \mathbb{E}_{\xi_n^t} v_n^t\rangle \\
&\overset{(a)}{=} -\eta_t\langle e_t, \nabla f(x_n^t)\rangle - \eta_t\|e_t\|^2,
\end{aligned}$$

where (a) follows from the definition of v_n^t. Denoting by $\mathbb{E}_t$ the expectation w.r.t. all $\xi_1^t, \xi_2^t, \dots$ given N_t. Since $\xi_1^t, \xi_2^t, \dots$ are independent of N_t, $\mathbb{E}_t$ is equivalent to expectation w.r.t. $\xi_1^t, \xi_2^t, \dots$. We have

$$\mathbb{E}_t(M_{n+1}^t - M_n^t) = -\eta_t \langle e_t, \mathbb{E}_t \nabla f(x_n^t)\rangle - \eta_t \|e_t\|^2.$$

Taking $n = N_t$ and expectation w.r.t. N_t as $\mathbb{E}_{N_t}$ we have

$$\mathbb{E}_{N_t}\mathbb{E}_t(M_{N_t+1}^t - M_{N_t}^t) = -\eta_t \langle e_t, \mathbb{E}_{N_t}\mathbb{E}_t \nabla f(x_{N_t}^t)\rangle - \eta_t \|e_t\|^2.$$

Using Fubini's theorem, Lemma A.10, Lemma A.11 and using the fact $x_{N_t}^t = \tilde{x}_t$ and $\tilde{x}_{t-1} = x_0^t$, we have

$$\frac{1}{B_t}\mathbb{E}_{N_t}\mathbb{E}_t\langle e_t, \tilde{x}_t - \tilde{x}_{t-1}\rangle = -\eta_t \langle e_t, \mathbb{E}_{N_t}\mathbb{E}_t \nabla f(\tilde{x}_t)\rangle - \eta_t \|e_t\|^2.$$

Taking expectation w.r.t. the whole past yields the statement of the lemma. ■

Lemma A.9. *The inner product term in Eq.* (A.76) *is bounded as follows*

$$\begin{aligned} 2\eta_t \mathbb{E}\langle e_t, \tilde{x}_t - \tilde{x}_{t-1}\rangle \le & \left(-\frac{1}{B_t} + \eta_t^2 L^2\right)\mathbb{E}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 \\ & - 2\eta_t \mathbb{E}\langle \nabla f(\tilde{x}_t), \tilde{x}_t - \tilde{x}_{t-1}\rangle + 2\eta_t^2 \mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 + 2\eta_t^2 \mathbb{E}\|e_t\|^2. \end{aligned}$$

PROOF: We have from the update equation $x_{n+1}^t = x_n^t - \eta_t v_n^t$, we have

$$\begin{aligned} \mathbb{E}_{\xi_n^t}\|x_{n+1}^t - x_0^t\|^2 &= \mathbb{E}_{\xi_n^t}\|x_n^t - \eta_t v_n^t - x_0^t\|^2 \\ &= \|x_n^t - x_0^t\|^2 + \eta_t^2 \mathbb{E}_{\xi_n^t}\|v_n^t\|^2 - 2\eta_t \langle \mathbb{E}_{\xi_n^t} v_n^t, x_n^t - x_0^t\rangle \\ &\overset{(a)}{\le} (1 + \eta_t^2 L^2)\|x_n^t - x_0^t\|^2 - 2\eta_t \langle \nabla f(x_n^t), x_n^t - x_0^t\rangle \\ &\qquad - 2\eta_t \langle e_t, x_n^t - x_0^t\rangle + 2\eta_t^2 \|\nabla f(x_n^t)\|^2 + 2\eta_t^2 \|e_t\|^2 \end{aligned} \tag{A.77}$$

where (a) follows from Lemma A.7 and the definition of v_n^t. Denoting by $\mathbb{E}_t$ the expectation w.r.t. all $\xi_1^t, \xi_2^t, \ldots$ given N_t. Since $\xi_1^t, \xi_2^t, \ldots$ are independent of N_t, $\mathbb{E}_t$ is equivalent to expectation w.r.t. $\xi_1^t, \xi_2^t, \ldots$. We have

$$\begin{aligned}\mathbb{E}_t\|x_{n+1}^t - x_0^t\|^2 \le (1+\eta_t^2L^2)\mathbb{E}_t\|x_n^t - x_0^t\|^2 - 2\eta_t\mathbb{E}_t\langle\nabla f(x_n^t), x_n^t - x_0^t\rangle \\ - 2\eta_t\mathbb{E}_t\langle e_t, x_n^t - x_0^t\rangle + 2\eta_t^2\mathbb{E}_t\|\nabla f(x_n^t)\|^2 + 2\eta_t^2\|e_t\|^2.\end{aligned}$$

Now taking $n = N_t$ and taking expectation $\mathbb{E}_{N_t}$ w.r.t. N_t we have

$$\begin{aligned}2\eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle e_t, \tilde{x}_t - \tilde{x}_{t-1}\rangle &\le (1+\eta_t^2L^2)\mathbb{E}_{N_t}\mathbb{E}_t\|x_{N_t}^t - x_0^t\|^2 - \mathbb{E}_{N_t}\mathbb{E}_t\|x_{N_t+1}^t - x_0^t\|^2 \\ &\quad - 2\eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle\nabla f(\tilde{x}_t), \tilde{x}_t - \tilde{x}_{t-1}\rangle + 2\eta_t^2\mathbb{E}_{N_t}\mathbb{E}_t\|\nabla f(\tilde{x}_t)\|^2 \\ &\quad + 2\eta_t^2\|e_t\|^2 \\ &\overset{(a)}{=} \left(-\frac{1}{B_t} + \eta_t^2L^2\right)\mathbb{E}_{N_t}\mathbb{E}_t\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 \\ &\quad - 2\eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle\nabla f(\tilde{x}_t), \tilde{x}_t - \tilde{x}_{t-1}\rangle + 2\eta_t^2\mathbb{E}_{N_t}\mathbb{E}_t\|\nabla f(\tilde{x}_t)\|^2 \\ &\quad + 2\eta_t^2\|e_t\|^2.\end{aligned}$$

where (a) follows from Lemma A.10, Lemma A.11 and Fubini's theorem. Finally, rearranging the terms and taking expectation w.r.t. the whole past yields the lemma. ■

Lemma A.10. *Given $N \sim$ Geom(Γ) for $\Gamma > 0$. For any sequence $D_0, D_1, \ldots$ with $\mathbb{E}|D_N| < \infty$, we have*

$$\mathbb{E}[D_N - D_{N+1}] = \left(\frac{1}{\Gamma} - 1\right)(D_0 - \mathbb{E}D_N)$$

PROOF: Proof follows from [71]. ■

Lemma A.11. *For step size $\eta_t \le \frac{1}{3LB_t^{2/3}}$, we have the following*

i) $\mathbb{E}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 < \infty$,

ii) $\mathbb{E}(f(\tilde{x}_t) - f(\tilde{x}^*)) < \infty$, *iii)* $\mathbb{E}\|\nabla f(\tilde{x}_t)\|^2 < \infty,$

iv) $\mathbb{E}|\langle e_t, \tilde{x}_t - \tilde{x}_{t-1}\rangle| < \infty,$

and v) $\mathbb{E}|\langle e_t, \nabla f(\tilde{x}_t)\rangle| < \infty.$

PROOF: The lemma is proven using induction and follows the same structure as the proof in [71]. The second inequality Eq. (A.72) in the proof of theorem yields

$$\begin{aligned}\mathbb{E}_{\xi_n^t} f(x_{n+1}^t) \le f(x_n^t) &- \eta_t(1 - L\eta_t)\|\nabla f(x_n^t)\|^2 \\ &- \eta_t\langle e_t, \nabla f(x_n^t)\rangle + \frac{L^3\eta_t^2}{2}\|x_n^t - x_0^t\|^2 + L\eta_t^2\|e_t\|^2, \end{aligned} \tag{A.78}$$

using Young's inequality $\langle a, b\rangle \le \frac{1}{2\beta}\|a\|^2 + \frac{\beta}{2}\|b\|^2$ for any $\beta > 0$, on $-\eta_t\langle e_t, \nabla f(x_n^t)\rangle$ with $\beta = \frac{1}{2}$ we get:

$$-\eta_t\langle e_t, \nabla f(x_n^t)\rangle \le \eta_t\|e_t\|^2 + \frac{\eta_t}{4}\|\nabla f(x_n^t)\|^2.$$

Moreover, using the fact that $\eta_t \le \frac{1}{3LB_t^{2/3}} \le \frac{1}{4L}$ since $B_t \ge 16$ and rearranging the terms in Eq. (A.78) we have

$$\begin{aligned}\eta_t\left(1 - L\eta_t - \frac{1}{4}\right)\|\nabla f(x_n^t)\|^2 &\le f(x_n^t) - \mathbb{E}_{\xi_n^t} f(x_{n+1}^t) + \frac{L^3\eta_t^2}{2}\|x_n^t - x_0^t\|^2 + \eta_t(1 + L\eta_t)\|e_t\|^2 \\ \eta_t\left(1 - \frac{1}{4} - \frac{1}{4}\right)\|\nabla f(x_n^t)\|^2 &\le f(x_n^t) - \mathbb{E}_{\xi_n^t} f(x_{n+1}^t) + \frac{L^3\eta_t^2}{2}\|x_n^t - x_0^t\|^2 + \eta_t\left(1 + \frac{1}{4}\right)\|e_t\|^2 \\ \eta_t\|\nabla f(x_n^t)\|^2 &\le 2\big(f(x_n^t) - \mathbb{E}_{\xi_n^t} f(x_{n+1}^t)\big) + L^3\eta_t^2\|x_n^t - x_0^t\|^2 + \frac{5\eta_t}{2}\|e_t\|^2. \end{aligned} \tag{A.79}$$

Now using the first inequality Eq. (A.77) in Proof of Lemma A.9, we have

$$\begin{aligned}\mathbb{E}_{\xi_n^t}\|x_{n+1}^t - x_0^t\|^2 \le (1 + \eta_t^2L^2)\|x_n^t - x_0^t\|^2 &- 2\eta_t\langle\nabla f(x_n^t), x_n^t - x_0^t\rangle \\ &- 2\eta_t\langle e_t, x_n^t - x_0^t\rangle + 2\eta_t^2\|\nabla f(x_n^t)\|^2 + 2\eta_t^2\|e_t\|^2, \end{aligned}$$

using Young's inequality $\langle a, b\rangle \le \frac{\beta}{2}\|a\|^2 + \frac{1}{2\beta}\|b\|^2$ for any $\beta > 0$, on $-2\eta_t\langle\nabla f(x_n^t), x_n^t - x_0^t\rangle$ and

$-2\eta_t\langle e_t, x_n^t - x_0^t\rangle$ with $\beta = 8\eta_t B_t$ we get:

$$-2\eta_t\langle \nabla f(x_n^t), x_n^t - x_0^t\rangle \leq 8\eta_t^2 B_t\|\nabla f(x_n^t)\|^2 + \frac{1}{8B_t}\|x_n^t - x_0^t\|^2$$
$$-2\eta_t\langle e_t, x_n^t - x_0^t\rangle \leq 8\eta_t^2 B_t\|e_t\|^2 + \frac{1}{8B_t}\|x_n^t - x_0^t\|^2,$$

Therefore, we get:

$$\begin{aligned}\mathbb{E}_{\xi_n^t}\|x_{n+1}^t - x_0^t\|^2 &\leq \left(1 + \eta_t^2 L^2 + \frac{1}{4B_t}\right)\|x_n^t - x_0^t\|^2 + (2\eta_t^2 + 8\eta_t^2 B_t)\|\nabla f(x_n^t)\|^2 \\ &\quad + (2\eta_t^2 + 8\eta_t^2 B_t)\|e_t\|^2 \\ &\overset{(a)}{\leq} \left(1 + \frac{13}{36B_t}\right)\|x_n^t - x_0^t\|^2 + 10\eta_t^2 B_t\|\nabla f(x_n^t)\|^2 + 10\eta_t^2 B_t\|e_t\|^2 \qquad \text{(A.80)}\end{aligned}$$

where (a) used the fact that we $\eta_t L \leq \frac{1}{3B_t^{2/3}}$. Now plugging Eq. (A.79) into Eq. (A.80) we get:

$$\begin{aligned}\mathbb{E}_{\xi_n^t}\|x_{n+1}^t - x_0^t\|^2 &\leq \left(1 + \frac{13}{36B_t}\right)\|x_n^t - x_0^t\|^2 + 10\eta_t^2 B_t\|e_t\|^2 \\ &\quad + 20\eta_t B_t\big(f(x_n^t) - \mathbb{E}_{\xi_n^t} f(x_{n+1}^t)\big) + 10\eta_t^3 L^3 B_t\|x_n^t - x_0^t\|^2 \\ &\quad + 25\eta_t^2 B_t\|e_t\|^2 \\ &\leq \left(1 + \frac{13}{36B_t} + 10\eta_t^3 L^3 B_t\right)\|x_n^t - x_0^t\|^2 \\ &\quad + 20\eta_t B_t\big(f(x_n^t) - \mathbb{E}_{\xi_n^t} f(x_{n+1}^t)\big) + 35\eta_t^2 B_t\|e_t\|^2 \\ &\overset{(a)}{\leq} \left(1 + \frac{711}{972B_t}\right)\|x_n^t - x_0^t\|^2 \\ &\quad + 20\eta_t B_t\big(f(x_n^t) - \mathbb{E}_{\xi_n^t} f(x_{n+1}^t)\big) + 35\eta_t^2 B_t\|e_t\|^2 \qquad \text{(A.81)}\end{aligned}$$

where (a) follows from using $\eta_t L \leq \frac{1}{3B_t^{2/3}}$. Let us assume

$$L_n^t = 20\eta_t B_t\mathbb{E}\big(f(x_n^t) - f(\tilde{x}^*)\big) + \mathbb{E}\|x_n^t - x_0^t\|^2$$

Taking expectation over Eq. (A.81) we get:

$$L_{n+1}^t \leq \left(1 + \frac{711}{972 B_t}\right) L_n^t + 35\eta_t^2 B_t \mathbb{E}\|e_t\|^2$$

Denoting $\gamma = \frac{711}{972} < 1$ we have:

$$\begin{aligned} L_{n+1}^t &\leq \left(1 + \frac{\gamma}{B_t}\right) L_n^t + 35\eta_t^2 B_t \mathbb{E}\|e_t\|^2 \\ L_{n+1}^t + \frac{35\eta_t^2 B_t^2 \mathbb{E}\|e_t\|^2}{\gamma} &\overset{(a)}{\leq} \left(1 + \frac{\gamma}{B_t}\right) L_n^t \\ &\quad + \left(1 + \frac{\gamma}{B_t}\right) \frac{35\eta_t^2 B_t^2 \mathbb{E}\|e_t\|^2}{\gamma} \\ L_{n+1}^t + \frac{35\eta_t^2 B_t^2 \mathbb{E}\|e_t\|^2}{\gamma} &\leq \left(1 + \frac{\gamma}{B_t}\right) \left(L_n^t + \frac{35\eta_t^2 B_t^2 \mathbb{E}\|e_t\|^2}{\gamma}\right) \end{aligned}$$

where (a) follows from adding and subtracting $\frac{35\eta_t^2 B_t^2 \mathbb{E}\|e_t\|^2}{\gamma}$ on both sides. this implies that we have:

$$L_n^t \leq \left(1 + \frac{\gamma}{B_t}\right)^n \left(L_0^t + \frac{35\eta_t^2 B_t^2 \mathbb{E}\|e_t\|^2}{\gamma}\right)$$

Since we have: $N_t \sim \text{Geom}\left(\frac{B_t}{B_t+1}\right)$, and assuming N_t can be 0 we have

$$\mathbb{P}[N_t = n] = \frac{1}{B_t + 1} \left(\frac{B_t}{B_t + 1}\right)^n \leq \left(\frac{B_t}{B_t + 1}\right)^n.$$

Now the term:

$$\begin{aligned} \mathbb{E}\left[\left(1 + \frac{\gamma}{B_t}\right)^{N_t}\right] &\leq \sum_{n \geq 0} \left(\frac{B_t + \gamma}{B_t} \times \frac{B_t}{B_t + 1}\right)^n \\ &= \sum_{n \geq 0} \left(\frac{B_t + \gamma}{B_t + 1}\right)^n \\ &\overset{(a)}{=} \frac{B_t + 1}{1 - \gamma}. \end{aligned}$$

(a) follows since $\gamma = \frac{711}{972} < 1$. This implies that:

$$\mathbb{E}L_{N_t}^t \leq \frac{B_t + 1}{1-\gamma}\big(L_0^t + 70\eta_t^2 B_t \mathbb{E}\|e_t\|^2\big).$$

which is finite since $\mathbb{E}\|e_t\| < \infty$ is finite by Lemma A.16 as well as the filtering rule of Algorithm 3. The induction hypothesis implies that $\mathbb{E}L_{N_t} < \infty$.

All the claims follow. ■

Lemma A.12 ([1] Lemma 2.4)**.** *Let the sequence of random variables $X_1, X_2, \ldots, X_N \in \mathbb{R}^d$ represent a random process such that we have $\mathbb{E}[X_n|X_1, \ldots, X_{n-1}] = 0$ and $\|X_n\| \leq M$. Then, $\mathbb{P}[\|X_1 + \ldots + X_N\|^2 \leq 2\log(2/\delta)M^2N] \geq 1-\delta$.*

Lemma A.13. *For $X_1, X_2, \ldots, X_n \in \mathbb{R}^d$, we have $\|X_1 + \ldots + X_n\|^2 \leq n\|X_1\|^2 + \ldots + n\|X_n\|^2$.*

Lemma A.14. *For any $t \in [T]$ and for all $k \in \mathcal{G}$, we define Event A as follows:*

1. *$\|\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\| \leq \mathcal{V}\sqrt{\frac{C}{B_t}}$.*
2. *$\|\mu_t^{(k)} - \mu_t^{med}\| \leq 4\mathcal{V}\sqrt{\frac{C}{B_t}}$ and $\|\mu_t^{med} - \nabla f(\tilde{x}_{t-1})\| \leq 3\mathcal{V}\sqrt{\frac{C}{B_t}}$, where $C = 2\log\left(\frac{2K}{\delta}\right)$,*

with probability at least $1-\delta$.

PROOF: *For (1):* Consider the RV, $\nabla f(\tilde{x}_{t-1}; \xi_{t,i}^{(k)}) - \nabla f(\tilde{x}_{t-1})$. From Assumption 5, we have $\|\nabla f(\tilde{x}_{t-1}; \xi_{t,i}^{(k)}) - \nabla f(\tilde{x}_{t-1})\| \leq \mathcal{V}$. We apply Lemma A.12 on the summation with $\|\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\| = \|\frac{1}{B_t}\sum_{i=1}^{B_t}\Big(\nabla f(\tilde{x}_{t-1}; \xi_{t,i}^{(k)}) - \nabla f(\tilde{x}_{t-1})\Big)\|$ to obtain the result.

For (2): Proof follows from the above result. ■

Lemma A.15. *For any $t \in [T]$ and for all $k \in \mathcal{G}$, we have*

1. *$\|\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\| \leq \mathcal{V}$.*
2. *Furthermore, we have $\|\mu_t^{(k)} - \mu_t^{med}\| \leq 4\mathcal{V}$ and $\|\mu_t^{med} - \nabla f(\tilde{x}_{t-1})\| \leq 3\mathcal{V}$.*

PROOF: *For (1):* Proof follows from the definition of $\mu_t^{(k)}$ and applying triangle inequality along with Assumption 5.

For (2): Proof follows from the above result. ■

Lemma A.16. *The error term $\mathbb{E}\|e_t\|^2$ is bounded as*

$$\mathbb{E}\|e_t\|^2 \leq \frac{4\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{272\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B_t},$$

provided that the parameters δ and B_t in Algorithm 3 satisfy the conditions $\mathrm{e}^{\frac{\delta B_t}{2(1-2\delta)}} \leq \frac{2K}{\delta} \leq \mathrm{e}^{\frac{B_t}{2}}$ and $\delta \leq \frac{1}{25KB_t}$.

PROOF: From the definition of the error term, e_t, we obtain

$$\mathbb{E}\|e_t\|^2 = \mathbb{E}\left\|\frac{1}{|\mathcal{G}_t|}\sum_{k\in\mathcal{G}_t}\left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\right)\right\|^2.$$

For the proof, we define the following three events and their complements, respectively.

Definition A.1. *The three events and their complements are:*

1. **Event A (Event $A!$):** *Recall the event defined in Lemma A.14 as Event A. Note that we have $\mathbb{P}[\text{Event } A] \geq 1-\delta$. Hence, the probability of the complement of Event A called Event $A!$ is given by $\mathbb{P}[\text{Event } A!] \leq \delta$.*

2. **Event $\bar{A}$ (Event $\bar{A}!$):** *The Event $\bar{A}$ consists of the set $\mathcal{G} \subset \mathcal{G}_t$. Therefore, the complement Event $\bar{A}!$ consists of the set $\mathcal{G} \not\subset \mathcal{G}_t$.*

3. **Event $R1$ (Event $R2$):** *Let Event $R1$ denotes the event that Rule 1 executes. The complement of Event $R1$, Event $R2$ denotes the event that Rule 2 executes.*

We provide the definitions of Rule 1 and Rule 2 below:

Rule 1: $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 2\mathfrak{T}_\mu\}$, where the vector median is given by $\mu_t^{\text{med}} = \mu_t^{(k)}$ for $k \in [K]$ is any WN such that $|\{k' \in [K] : \|\mu_t^{(k')} - \mu_t^{(k)}\| \leq \mathfrak{T}_\mu\}| > K/2$. However, $|\mathcal{G}_t| < (1-\alpha)K$ then Rule 2 is used.

Rule 2: $\mathcal{G}_t = \{k \in [K] : \|\mu_t^{(k)} - \mu_t^{\text{med}}\| \leq 4\mathcal{V}\}$, where the vector median is modified as $\mu_t^{\text{med}} = \mu_t^{(k)}$ where $k \in [K]$ is any WN s.t. $|\{k' \in [K] : \|\mu_t^{(k')} - \mu_t^{(k)}\| \leq 2\mathcal{V}\}| > K/2$.

Relationship between events:

- From Definition of Event A in Definition A.1, Event $A \subset$ Event $\bar{A}$. Furthermore, from Lemma A.14, we obtain

$$\mathbb{P}[\text{Event } \bar{A}] \geq \mathbb{P}[\text{Event } A] \geq 1 - \delta. \tag{A.82}$$

- From above, we get Event $A! \supset$ Event $\bar{A}!$. Furthermore, from Lemma A.14, we obtain

$$\mathbb{P}[\text{Event } \bar{A}!] \leq \mathbb{P}[\text{Event } A!] \leq \delta. \tag{A.83}$$

- From the definitions of Event $R1$ and Event A in Definition A.1, Event $A \subset$ Event $R1$. Hence, Lemma A.14 yields

$$\mathbb{P}[\text{Event } R1] \geq \mathbb{P}[\text{Event } A] \geq 1 - \delta. \tag{A.84}$$

- Note that Event $R2$ is the complement of Event $R1$. Therefore, Event $R2 \subset$ Event $A!$. Hence, Lemma A.14 yields

$$\mathbb{P}[\text{Event } R2] \leq \mathbb{P}[\text{Event } A!] \leq \delta. \tag{A.85}$$

Using law of total expectation, we have

$$\begin{aligned}
&\mathbb{E}\|e_t\|^2 \\
&= \mathbb{P}[\text{Event}\bar{A}]\mathbb{E}\left[\|e_t\|^2 | \text{Event}\bar{A}\right] + \mathbb{P}[\text{Event}\bar{A}!]\mathbb{E}\left[\|e_t\|^2 | \text{Event}\bar{A}!\right] \\
&\overset{(a)}{\leq} \mathbb{E}\left[\|e_t\|^2 | \text{Event } \bar{A}\right] + \delta\, \mathbb{E}\left[\|e_t\|^2 | \text{Event } \bar{A}!\right]
\end{aligned} \tag{A.86}$$

where we use Eq. (A.82) and Eq. (A.83) in (a). Considering the term $\mathbb{E}\left[\|e_t\|^2|\text{Event } \bar{A}\right]$ from above, we get

$$\begin{aligned}
&\mathbb{E}\left[\|e_t\|^2|\text{Event } \bar{A}\right] \\
&\overset{(b)}{\leq} \frac{2}{(1-\alpha)^2K^2}\Bigg(\mathbb{E}\left[\left\|\sum_{k\in\mathcal{G}}\left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\right)\right\|^2\Bigg|\text{Event } \bar{A}\right] \\
&+ \underbrace{\mathbb{E}\left[\left\|\sum_{k\in\mathcal{G}_t\backslash\mathcal{G}}\left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\right)\right\|^2\Bigg|\text{Event } \bar{A}\right]}_{I_{\mathcal{G}_t\backslash\mathcal{G}}}\Bigg),
\end{aligned} \tag{A.87}$$

where we use $|\mathcal{G}_t| \geq (1-\alpha)K$ and $\mathcal{G} \subset \mathcal{G}_t$ under Event $\bar{A}$, and Lemma A.13. Consider the first term in Eq. (A.87), noting that $f(\tilde{x}_{t-1};\xi_{t,i}^{(k)})$ are chosen uniformly independently across samples $i \in [B]$ and WNs $k \in [K]$, we get

$$\begin{aligned}
&\mathbb{E}\left[\left\|\sum_{k\in\mathcal{G}}\left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\right)\right\|^2\Bigg|\text{ Event } \bar{A}\right] \\
&\overset{(a)}{=} \frac{1}{B_t^2}\mathbb{E}\left[\sum_{k\in\mathcal{G}}\sum_{i=1}^{B_t}\left\|\nabla f(\tilde{x}_{t-1};\xi_{t,i}^{(k)}) - \nabla f(\tilde{x}_{t-1})\right\|^2\Bigg|\text{Event } \bar{A}\right] \\
&+ \frac{1}{B_t^2}\mathbb{E}\Bigg[\sum_{(k,i)\neq(k',i'),k,k'\in\mathcal{G}}\langle\nabla f(\tilde{x}_{t-1};\xi_{t,i}^{(k)}) - \nabla f(\tilde{x}_{t-1}) \\
&,\nabla f(\tilde{x}_{t-1};\xi_{t,i'}^{(k')}) - \nabla f(\tilde{x}_{t-1})\rangle\Bigg|\text{ Event } \bar{A}\Bigg],
\end{aligned}$$

as the second term in (a) is zero under Event $\bar{A}$ and using Assumption 5 and $|\mathcal{G}| \leq K$, we have

$$\mathbb{E}\left[\left\|\sum_{k\in\mathcal{G}}\left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\right)\right\|^2\Bigg|\text{ Event } \bar{A}\right] \overset{(a)}{\leq} \frac{K\mathcal{V}^2}{B_t}, \tag{A.88}$$

Considering the second term in Eq. (A.87),

noting that WNs $k \in \mathcal{G}_t\backslash\mathcal{G}$ under Event $\bar{A}$ can come from either Rule 1 or Rule 2, and rewriting

$\mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}]$ using the law of total expectation yields

$$\begin{aligned}
&\mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}] \\
&\overset{(a)}{=} \mathbb{P}[\text{Event } R1 \big| \text{Event } \bar{A}]\ \mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}, \text{Event } R1] \\
&\qquad + \mathbb{P}[\text{Event } R2 \big| \text{Event } \bar{A}]\ \mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}, \text{Event } R2] \\
&\overset{(b)}{\leq} \mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}, \text{Event } R1] \\
&\qquad\qquad + \left(\frac{\delta}{1-\delta}\right)\ \mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}, \text{Event } R2].
\end{aligned} \tag{A.89}$$

where the fact Event $R2$ is the complement of Event $R1$ is used in (a). Next, (b) follows from

$$\begin{aligned}
\mathbb{P}[\text{Event } R2 \big| \text{Event } \bar{A}] &= \frac{\mathbb{P}[\text{Event } R2 \cap \text{Event } \bar{A}]}{\mathbb{P}[\text{Event } \bar{A}]} \\
&\overset{(a)}{\leq} \frac{\mathbb{P}[\text{Event } R2]}{\mathbb{P}[\text{Event } \bar{A}]} \overset{(b)}{\leq} \frac{\delta}{1-\delta}.
\end{aligned}$$

where (a) follows from [Event $R2 \cap$ Event $\bar{A}$] $\subset$ Event $R2$, and (b) follows from Eq. (A.85) and Eq. (A.82). Next, considering the first term in Eq. (A.89), we obtain

$$\begin{aligned}
&\mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}, \text{Event } R1] \\
&\overset{(a)}{\leq} 2\alpha K \mathbb{E}\left[\sum_{k\in\mathcal{G}_t\backslash\mathcal{G}} \left\|\mu_t^{(k)} - \mu_t^{\text{med}}\right\|^2 \middle| \text{Event } \bar{A}, \text{Event } R1\right] \\
&+ 2\alpha K \mathbb{E}\Bigg[\underbrace{\sum_{k\in\mathcal{G}_t\backslash\mathcal{G}} \left\|\mu_t^{\text{med}} - \nabla f(\tilde{x}_{t-1})\right\|^2}_{J_{\mathcal{G}_t\backslash\mathcal{G}}} \Bigg| \text{Event } \bar{A}, \text{Event } R1\Bigg] \\
&\overset{(b)}{\leq} 2\alpha^2 K^2 \left[\frac{16\mathcal{V}^2 C}{B_t}\right] + 2\alpha^2 K^2 \left[\frac{18\mathcal{V}^2 C}{B_t}\right] = \frac{68\alpha^2 K^2 \mathcal{V}^2 C}{B_t}.
\end{aligned} \tag{A.90}$$

where adding and subtracting μ_t^{med}, and applying Lemma A.13 and $|\mathcal{G}_t\backslash\mathcal{G}| \leq \alpha K$ in (a). (b) is obtained as below

$$\begin{aligned}\mathbb{E}[J_{\mathcal{G}_t\setminus\mathcal{G}}\big|\text{Event } \bar{A}, \text{Event } R1] = {} & \mathbb{P}[\text{Event } A\big|\text{Event } \bar{A}, \text{Event } R1] \\ & \qquad \mathbb{E}[J_{\mathcal{G}_t\setminus\mathcal{G}}\big|\text{Event } A, \text{Event } \bar{A}, \text{Event } R1] \\ & + \mathbb{P}[\text{Event } A!\big|\text{Event } \bar{A}, \text{Event } R1] \\ & \qquad \mathbb{E}[J_{\mathcal{G}_t\setminus\mathcal{G}}\big|\text{Event } A!, \text{Event } \bar{A}, \text{Event } R1] \\ & \overset{(d)}{\leq} \mathbb{E}[J_{\mathcal{G}_t\setminus\mathcal{G}}\big|\text{Event } A, \text{Event } \bar{A}, \text{Event } R1] \\ & + \left(\frac{\delta}{1-2\delta}\right)\mathbb{E}[J_{\mathcal{G}_t\setminus\mathcal{G}}\big|\text{Event } A!, \text{Event } \bar{A}, \text{Event } R1] \\ & \overset{(e)}{\leq} \alpha K\frac{9\mathcal{V}^2C}{B_t} + \left(\frac{\delta}{1-2\delta}\right)\alpha K 9\mathcal{V}^2 \overset{(f)}{\leq} \alpha K\frac{18\mathcal{V}^2C}{B_t},\end{aligned}$$

where (d) follows from

$$\begin{aligned}\mathbb{P}[\text{Event } A!\big|\text{Event } \bar{A}, \text{Event } R1] &\leq \frac{\mathbb{P}[\text{Event } A!]}{\mathbb{P}[\text{Event } \bar{A}, \text{Event } R1]} \\ &\leq \frac{\delta}{1-2\delta},\end{aligned}$$

where we use $[\text{Event } A! \cap \text{Event } R1 \cap \text{Event } \bar{A}] \subset \text{Event } A!$ and the last inequality follows from the fact that $\mathbb{P}[\text{Event } \bar{A} \cap \text{Event } R1] = \mathbb{P}[\text{Event } \bar{A}] + \mathbb{P}[\text{Event } R1] - \mathbb{P}[\text{Event } \bar{A} \cup \text{Event } R1] \geq 1-\delta+1-\delta-1 = 1-2\delta$. Moreover, we apply Lemma A.14 and Lemma A.15 in (e). Finally, (f) follows from choosing δ such that $\mathrm{e}^{\frac{\delta B_t}{2(1-2\delta)}} \leq \frac{2K}{\delta}$. Considering the second term in Eq. (A.89) yields

$$\begin{aligned}\mathbb{E}[I_{\mathcal{G}_t\setminus\mathcal{G}}\big|\text{Event } \bar{A}, \text{Event } R2] \overset{(a)}{\leq} {} & 2\alpha K\mathbb{E}\left[\sum_{k\in\mathcal{G}_t\setminus\mathcal{G}}\left\|\mu_t^{(k)} - \mu_t^{\text{med}}\right\|^2\middle|\text{Event } \bar{A}, \text{Event } R2\right] \\ & + 2\alpha K\mathbb{E}\left[\sum_{k\in\mathcal{G}_t\setminus\mathcal{G}}\left\|\mu_t^{\text{med}} - \nabla f(\tilde{x}_{t-1})\right\|^2\middle| \text{Event } \bar{A}, \text{Event } R2\right] \\ & \overset{(b)}{\leq} 2\alpha^2K^2\left[16\mathcal{V}^2\right] + 2\alpha^2K^2\left[9\mathcal{V}^2\right] = 50\alpha^2K^2\mathcal{V}^2 < 68\alpha^2K^2\mathcal{V}^2. \quad \text{(A.91)}\end{aligned}$$

where adding and subtracting μ_t^{med}, and applying Lemma A.13 and $|\mathcal{G}_t \backslash \mathcal{G}| \leq \alpha K$ in (a). We use the fact that under Event $R2$ WNs $k \in \mathcal{G}_t \backslash \mathcal{G}$ satisfy statement (a) in Lemma A.15 in (b). Substituting Eq. (A.90) and Eq. (A.91) in Eq. (A.89) yields

$$\begin{aligned}\mathbb{E}[I_{\mathcal{G}_t\backslash\mathcal{G}} \big| \text{Event } \bar{A}] &\leq \frac{68\alpha^2 K^2 \mathcal{V}^2 C}{B_t} + \left(\frac{\delta}{1-\delta}\right) 68\alpha^2 K^2 \mathcal{V}^2 \\ &\overset{(a)}{\leq} \frac{136\alpha^2 K^2 \mathcal{V}^2 C}{B_t},\end{aligned} \tag{A.92}$$

where (a) follows from choosing δ such that $\mathrm{e}^{\frac{\delta B_t}{2(1-\delta)}} \leq \frac{2K}{\delta}$. Substituting Eq. (A.88) and Eq. (A.92) in Eq. (A.87), we use the bound on the first term in Eq. (A.86) as follows:

$$\begin{aligned}\mathbb{E}\left[\|e_t\|^2 | \text{Event } \bar{A}\right] &\leq \frac{2}{(1-\alpha)^2 K^2}\left(\frac{K\mathcal{V}^2}{B_t} + \frac{136\alpha^2 K^2 \mathcal{V}^2 C}{B_t}\right) \\ &= \frac{2\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{272\alpha^2 \mathcal{V}^2 C}{(1-\alpha)^2 B_t}.\end{aligned} \tag{A.93}$$

Considering the second term in Eq. (A.86)

$$\begin{aligned}\delta\, \mathbb{E}\left[\|e_t\|^2 | \text{Event } \bar{A}!\right] &\overset{(a)}{\leq} \frac{2\delta}{(1-\alpha)^2 K}\Bigg(\mathbb{E}\Bigg[\sum_{k\in\mathcal{G}_t} \left\|\mu_t^{(k)} - \mu_t^{\text{med}}\right\|^2 \Bigg| \text{ Event } \bar{A}!\Bigg] \\ &\qquad + \mathbb{E}\Bigg[\sum_{k\in\mathcal{G}_t} \left\|\mu_t^{\text{med}} - \nabla f(\tilde{x}_{t-1})\right\|^2 \Bigg| \text{ Event } \bar{A}!\Bigg]\Bigg) \\ &\leq \frac{2\delta}{(1-\alpha)^2 K}\left(16K\mathcal{V}^2 + 9K\mathcal{V}^2\right) = \delta \frac{50\mathcal{V}^2}{(1-\alpha)^2},\end{aligned} \tag{A.94}$$

where adding and subtracting μ_t^{med}, applying Lemma A.13, and using $|\mathcal{G}_t| \geq (1-\alpha)K$ in the denominator and $|\mathcal{G}_t| \leq K$ in the numerator in (a). Substituting Eq. (A.93) and Eq. (A.94) in Eq. (A.86), we have

$$\mathbb{E}\|e_t\|^2 \leq \frac{2\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{272\alpha^2 \mathcal{V}^2 C}{(1-\alpha)^2 B_t} + \delta \frac{50\mathcal{V}^2}{(1-\alpha)^2},$$

choosing $\delta \leq \frac{1}{25KB_t}$, we obtain the final result. ■

A.10 Proof of Corollary 3.1

To guarantee that we get an ϵ-accurate solution, we need $T = O\left(\frac{1}{\epsilon B^{1/3}}\right)$ iterations for the first term in Eq. (3.3). For the second term in Eq. (3.3), we need $B_K = O\left(\frac{1}{\epsilon K}\right)$ batch size. We need $B_\alpha = O\left(\frac{\alpha^2}{\epsilon}\right)$ batch size for the third term in Eq. (3.3) to account for the Byzantine workers. Hence, the total number of gradient computations, $\mathbb{E}G_{\text{comp, CN}}$ required at the CN (and at the individual WNs) on an average are of the order of $\mathbb{E}G_{\text{comp, CN}}(\epsilon) \le TB_K + TB_\alpha \le O\left(\frac{1}{\epsilon^{5/3}K^{2/3}} + \frac{\alpha^{4/3}}{\epsilon^{5/3}}\right)$. Also, the expected number of gradient computations across the network denoted by, $\mathbb{E}G_{\text{comp, NW}}$, are of the order of $\mathbb{E}G_{\text{comp, NW}}(\epsilon) \le O\left(\frac{K^{1/3}}{\epsilon^{5/3}} + \frac{K\alpha^{4/3}}{\epsilon^{5/3}}\right)$.

Moreover, if $\alpha = 0$, we get the expected computational complexity and the expected number of gradient computations across the network as $\mathbb{E}G_{\text{comp, CN}}(\epsilon) \le O\left(\frac{1}{\epsilon^{5/3}K^{2/3}}\right)$ and $\mathbb{E}G_{\text{comp, NW}}(\epsilon) \le O\left(\frac{K^{1/3}}{\epsilon^{5/3}}\right)$.

A.11 Proof of Theorem 3.2

Using the smoothness of function f and Lemma A.7, we have:

$$\begin{aligned} \mathbb{E}_{\xi_{n,t}} f(x_{n+1,t}) \le f(x_{n,t}) - \eta_t(1 - L\eta_t)\|\nabla f(x_{n,t})\|^2 \\ - \eta_t\langle e_t, \nabla f(x_{n,t})\rangle + \frac{L^3\eta_t^2}{2}\|x_{n,t} - x_{0,t}\|^2 + L\eta_t^2\|e_t\|^2. \end{aligned} \tag{A.95}$$

Denoting by $\mathbb{E}_t$ the expectation w.r.t. all $\xi_{1,t}, \xi_{2,t}, \ldots$ given N_t. Since $\xi_{1,t}, \xi_{2,t}, \ldots$ are independent of N_t, $\mathbb{E}_t$ is equivalent to expectation w.r.t. $\xi_{1,t}, \xi_{2,t}, \ldots$. Taking $n = N_t$ and denoting by $\mathbb{E}_{N_t}$ expectation w.r.t. N_t, we get

$$\begin{aligned} \mathbb{E}_{N_t}\mathbb{E}_t f(x_{N_t+1,t}) \le \mathbb{E}_{N_t}\mathbb{E}_t f(x_{N_t,t}) - \eta_t(1 - L\eta_t)\mathbb{E}_{N_t}\mathbb{E}_t\|\nabla f(x_{N_t,t})\|^2 + L\eta_t^2\|e_t\|^2 \\ - \eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle e_t, \nabla f(x_{N_t,t})\rangle + \frac{L^3\eta_t^2}{2}\mathbb{E}_{N_t}\mathbb{E}_t\|x_{N_t,t} - x_{0,t}\|^2 \end{aligned}$$

Rearranging the terms and using Lemma A.10, Lemma A.19 and Fubini's theorem [71], we have

$$\begin{aligned}\eta_t(1-L\eta_t)\mathbb{E}_{N_t}\mathbb{E}_t\|\nabla f(x_{N_t,t})\|^2 &= \frac{1}{B_t}[f(x_{0,t})-\mathbb{E}_t\mathbb{E}_{N_t}f(x_{N_t,t})]-\eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle e_t,\nabla f(x_{N_t,t})\rangle\\ &\quad+\frac{L^3\eta_t^2}{2}\mathbb{E}_{N_t}\mathbb{E}_t\|x_{N_t,t}-x_{0,t}\|^2+L\eta_t^2\|e_t\|^2\\ &\overset{(a)}{=}\frac{1}{B_t}[\mathbb{E}_t\mathbb{E}_{N_t}f(x_{0,t})-\mathbb{E}_t\mathbb{E}_{N_t}f(x_{N_t,t})]+L\eta_t^2\|e_t\|^2\\ &\quad-\eta_t\mathbb{E}_{N_t}\mathbb{E}_t\langle e_t,\nabla f(x_{N_t,t})\rangle+\frac{L^3\eta_t^2}{2}\mathbb{E}_{N_t}\mathbb{E}_t\|x_{N_t,t}-x_{0,t}\|^2\end{aligned}$$

Note that in (a), $\mathbb{E}_t\mathbb{E}_{N_t}f(x_{0,t})=f(x_{0,t})$. Denoting the expectation as $\tilde{\mathbb{E}}[\cdot]$, and using $x_{N_t,t}=\tilde{x}_t$ and $\tilde{x}_{t-1}=x_{0,t}$, and applying Lemma A.17 we get

$$\begin{aligned}\eta_t(1-L\eta_t)\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2 &\le \frac{1}{B_t}\tilde{\mathbb{E}}[f(\tilde{x}_{t-1})-f(\tilde{x}_t)]+\frac{1}{B_t}\tilde{\mathbb{E}}\langle e_t,\tilde{x}_t-\tilde{x}_{t-1}\rangle\\ &\quad+\frac{L^3\eta_t^2}{2}\tilde{\mathbb{E}}\|\tilde{x}_t-\tilde{x}_{t-1}\|^2+\eta_t(1+L\eta_t)\|e_t\|^2\\ &\le\frac{1}{B_t}\tilde{\mathbb{E}}[f(\tilde{x}_{t-1})-f(\tilde{x}_t)]+\left(\frac{1}{2\eta_tB_t}\left(-\frac{1}{B_t}+\eta_t^2L^2\right)\right)\tilde{\mathbb{E}}\|\tilde{x}_t-\tilde{x}_{t-1}\|^2\\ &\quad-\frac{1}{B_t}\tilde{\mathbb{E}}\langle\nabla f(\tilde{x}_t),\tilde{x}_t-\tilde{x}_{t-1}\rangle+\frac{\eta_t}{B_t}\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2+\frac{\eta_t}{B_t}\|e_t\|^2\\ &\quad+\frac{L^3\eta_t^2}{2}\tilde{\mathbb{E}}\|\tilde{x}_t-\tilde{x}_{t-1}\|^2+\eta_t(1+L\eta_t)\|e_t\|^2\end{aligned}$$

where the last inequality follows from Lemma A.18. Rearranging the terms, we get

$$\begin{aligned}&\eta_t\left(1-L\eta_t-\frac{1}{B_t}\right)\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2+\left(\frac{1-\eta_t^2L^2B_t-\eta_t^3L^3B_t^2}{2\eta_tB_t^2}\right)\tilde{\mathbb{E}}\|\tilde{x}_t-\tilde{x}_{t-1}\|^2\\ &\le\frac{1}{B_t}\tilde{\mathbb{E}}[f(\tilde{x}_{t-1})-f(\tilde{x}_t)]+\frac{1}{B_t}\tilde{\mathbb{E}}\langle\nabla f(\tilde{x}_t),\tilde{x}_{t-1}-\tilde{x}_t\rangle\\ &\quad+\eta_t\left(1+L\eta_t+\frac{1}{B_t}\right)\|e_t\|^2.\end{aligned}$$

Using the Young's inequality $\langle a, b\rangle \le \left[\frac{\beta}{2}\|a\|^2 + \frac{1}{2\beta}\|b\|^2\right]$ to $\tilde{\mathbb{E}}\langle \nabla f(\tilde{x}_t), \tilde{x}_{t-1} - \tilde{x}_t\rangle$ with $\beta = \frac{1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2}{\eta_t B_t}$, $a = \tilde{x}_{t-1} - \tilde{x}_t$ and $b = \nabla f(\tilde{x}_t)$, and Lemma A.20, we get

$$\begin{aligned}
&\eta_t\left(1 - L\eta_t - \frac{1}{B_t} - \frac{1}{2\eta_t B_t \beta}\right)\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2 \\
&\qquad\le \frac{1}{B_t}\tilde{\mathbb{E}}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] + \eta_t\left(1 + L\eta_t + \frac{1}{B_t}\right)\|e_t\|^2 \\
&\eta_t\left(1 - L\eta_t - \frac{1}{B_t} - \frac{1}{2\eta_t B_t\beta}\right)\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2 \le \frac{1}{B_t}\tilde{\mathbb{E}}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] \\
&+ \eta_t\left(1 + L\eta_t + \frac{1}{B_t}\right)\left(\frac{2\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{100\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B_t}\right).
\end{aligned} \tag{A.96}$$

Choosing η_t such that we have $1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2 > 0$, this implies that we have $\frac{1}{2(1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2)} > \frac{1}{2}$. Further, we choose η_t such that, we have

$$1 - L\eta_t - \frac{1}{B_t} - \frac{1}{2(1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2)} \ge \frac{1}{4}.$$

Therefore, we choose η_t to ensure the following.

1. $\frac{1}{2} < \frac{1}{2(1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2)} \le \frac{5}{8}$,

2. $L\eta_t \le \frac{1}{16}$, and $\frac{1}{B_t} \le \frac{1}{16}$.

Condition (1) above implies $\eta_t^2 L^2 B_t + \eta_t^3 L^3 B_t^2 \le \frac{1}{5}$. Further ensuring η_t such that $\eta_t^2 L^2 B_t \le \frac{1}{10}$ and $\eta_t^3 L^3 B_t^2 \le \frac{1}{10}$. This implies that $\eta_t \le \frac{1}{10^{1/2} L B_t^{1/2}}$ and $\eta_t \le \frac{1}{10^{1/3} L B_t^{2/3}}$. Further from Condition (2), we get $\eta_t \le \frac{1}{16L}$ and $B_t \ge 16$. The above discussion implies that we must have $B_t \ge 16$ and we can choose $\eta_t \le \frac{1}{3LB_t^{2/3}}$ as we have

$$\frac{1}{3LB_t^{2/3}} \le \min\left\{\frac{1}{16L}, \frac{1}{10^{1/3} L B_t^{2/3}}, \frac{1}{10^{1/2} L B_t^{1/2}}\right\}.$$

This choice of η_t ensures that

$$1 - L\eta_t - \frac{1}{B_t} - \frac{1}{2(1-\eta_t^2 L^2 B_t - \eta_t^3 L^3 B_t^2)} \ge \frac{1}{4}. \tag{A.97}$$

Now, replacing η_t and B_t, we get

$$1 + \eta_t L + \frac{1}{B_t} \leq 1 + \frac{1}{3B_t^{2/3}} + \frac{1}{B_t} \leq 2. \tag{A.98}$$

Now replacing Eq.(A.97) and Eq. (A.98) in Eq. (A.96), and replacing $\eta_t = \frac{1}{3LB_t^{2/3}}$ we get

$$\begin{aligned}
\frac{\eta_t}{4}\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2 &\leq \frac{1}{B_t}\tilde{\mathbb{E}}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)] \\
&\quad + 2\eta_t \left(\frac{2\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{100\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B_t} \right) \\
\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2 &\leq \frac{12L\tilde{\mathbb{E}}[f(\tilde{x}_{t-1}) - f(\tilde{x}_t)]}{B_t^{1/3}} \\
&\quad + \frac{16\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{800\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B_t}.
\end{aligned}$$

For $B_t = B$ and summing over $t = 1, 2, \ldots, T$ and choosing x_a using Algorithm 4, we obtain

$$\begin{aligned}
&\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_a)\|^2 \\
&\leq \frac{12L\mathbb{E}[f(\tilde{x}_0) - f(\tilde{x}^*)] + 16\mathcal{V}^2(1-\alpha)^{-2}K^{-1}\sum_{t=1}^T B_t^{-2/3}}{\sum_{t=1}^T B_t^{1/3}} \\
&\quad + \frac{800\alpha^2\mathcal{V}^2 C(1-\alpha)^{-2}\sum_{t=1}^T B_t^{-2/3}}{\sum_{t=1}^T B_t^{1/3}}.
\end{aligned}$$

Replacing $B_t = B$, we get

$$\begin{aligned}
&\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_a)\|^2 \\
&\leq \underbrace{\frac{12L\mathbb{E}[f(\tilde{x}_0) - f(\tilde{x}^*)]}{TB^{1/3}}}_{T=O\left(\frac{1}{\epsilon B^{1/3}}\right)} + \underbrace{\frac{16\mathcal{V}^2}{(1-\alpha)^2 KB}}_{B=O\left(\frac{1}{\epsilon K}\right)} + \underbrace{\frac{800\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B}}_{B=O\left(\frac{\alpha^2}{\epsilon}\right)}.
\end{aligned}$$

A.12 Useful Lemmas for Proof in Appendix A.11

In this section, we present the following lemmas which are used in the proof of Appendix A.11.

Lemma A.17. *For the error term, e_t, we have the following*

$$\eta_t \tilde{\mathbb{E}}\langle e_t, \nabla f(\tilde{x}_t)\rangle = \frac{1}{B_t}\tilde{\mathbb{E}}\langle e_t, \tilde{x}_{t-1} - \tilde{x}_t\rangle - \eta_t\|e_t\|^2. \tag{A.99}$$

Lemma A.18. *The inner product term in Eq.* (A.99) *satisfies the following inequality*

$$\begin{aligned}2\eta_t\tilde{\mathbb{E}}\langle e_t, \tilde{x}_t - \tilde{x}_{t-1}\rangle &\le \left(-\frac{1}{B_t} + \eta_t^2 L^2\right)\tilde{\mathbb{E}}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 \\ &- 2\eta_t\tilde{\mathbb{E}}\langle \nabla f(\tilde{x}_t), \tilde{x}_t - \tilde{x}_{t-1}\rangle + 2\eta_t^2\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2 + 2\eta_t^2\tilde{\mathbb{E}}\|e_t\|^2.\end{aligned}$$

Lemma A.19. *For step size $\eta_t \le \frac{1}{3LB_t^{2/3}}$, we have:*
i) $\tilde{\mathbb{E}}\|\tilde{x}_t - \tilde{x}_{t-1}\|^2 < \infty$, ii) $\tilde{\mathbb{E}}(f(\tilde{x}_t) - f(\tilde{x}^*)) < \infty$,
iii) $\tilde{\mathbb{E}}\|\nabla f(\tilde{x}_t)\|^2 < \infty$, iv) $\tilde{\mathbb{E}}|\langle e_t, \tilde{x}_t - \tilde{x}_{t-1}\rangle| < \infty$,
and v) $\tilde{\mathbb{E}}|\langle e_t, \nabla f(\tilde{x}_t)\rangle| < \infty$.

Note that the missing proofs of Lemmas A.17, A.18, and A.19 are similar to those of the proofs of Lemmas A.8, A.9, and A.11 and therefore have been skipped.

Lemma A.20. *Choosing B_t in Algorithm 4 then $\|e_t\|^2$ is bounded as*

$$\|e_t\|^2 \le \frac{2\mathcal{V}^2}{(1-\alpha)^2 K B_t} + \frac{100\alpha^2\mathcal{V}^2 C}{(1-\alpha)^2 B_t}.$$

PROOF: From the definition of e_t and noting that we are in Event $\overline{A}$ where $\mathcal{G} \subset \mathcal{G}_t$, we can write $\|e_t\|^2$ as

$$\begin{aligned}\|e_t\|^2 \le \frac{2}{(1-\alpha)^2K^2}&\left(\left\|\sum_{k\in\mathcal{G}}\left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\right)\right\|^2\right. \\ &\left.+\underbrace{\left\|\sum_{k\in\mathcal{G}_t\setminus\mathcal{G}}\left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1})\right)\right\|^2}_{I_{\mathcal{G}_t\setminus\mathcal{G}}}\right)\end{aligned} \tag{A.100}$$

where the inequality follows from the fact that $|\mathcal{G}_t| \ge (1-\alpha)K$ and from Lemma A.13. Now considering the two terms separately under Event $\bar{A}$, first consider the terms for $k \in \mathcal{G}$

$$\left\| \sum_{k \in \mathcal{G}} \left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1}) \right) \right\|^2 \leq \frac{KC\mathcal{V}^2}{B_t}, \tag{A.101}$$

where inequality follows from Lemma A.19. Now, consider the second term in Eq. (A.100), we get

$$\begin{aligned} I_{\mathcal{G}_t \setminus \mathcal{G}} &= \left\| \sum_{k \in \mathcal{G}_t \setminus \mathcal{G}} \left(\mu_t^{(k)} - \nabla f(\tilde{x}_{t-1}) \right) \right\|^2 \\ &\overset{(a)}{\leq} 2\alpha K \sum_{k \in \mathcal{G}_t \setminus \mathcal{G}} (\|\mu_t^{(k)} - \mu_t^{\text{med}}\|^2 + \|\mu_t^{\text{med}} - \nabla f(\tilde{x}_{t-1})\|^2) \\ &\overset{(b)}{\leq} 2\alpha K \left[\alpha K \frac{16\mathcal{V}^2 C}{B_t} \right] + 2\alpha K \left[\alpha K \frac{9\mathcal{V}^2 C}{B_t} \right] = \frac{50\alpha^2 K^2 \mathcal{V}^2 C}{B_t}. \end{aligned} \tag{A.102}$$

where (a) follows from Lemma A.13 and $|\mathcal{G}_t \setminus \mathcal{G}| \leq \alpha K$, (b) follows by adding and subtracting μ_t^{med} and applying Lemma A.13. Now replacing Eq. (A.102) and Eq. (A.101) in Eq. (A.100) yields the result. ■

A.13 Proof of Corollary 3.2

To guarantee that we get an ϵ-accurate solution, we need $T = O\left(\frac{1}{\epsilon B^{1/3}}\right)$ iterations for the first term in Eq. (3.5). For the second term in Eq. (3.5), we need $B_K = O\left(\frac{1}{\epsilon K}\right)$ batch size. We need $B_\alpha = O\left(\frac{\alpha^2}{\epsilon}\right)$ batch size for the third term in Eq. (3.5) to account for the Byzantine workers. Hence, the total number of gradient computations, $\mathbb{E}G_{\text{comp, CN}}$ required at the CN (and at the individual WNs) on an average are of the order of $\mathbb{E}G_{\text{comp, CN}}(\epsilon) \leq TB_K + TB_\alpha \leq O\left(\frac{1}{\epsilon^{5/3}K^{2/3}} + \frac{\alpha^{4/3}}{\epsilon^{5/3}}\right)$. Also, the expected number of gradient computations across the network denoted by, $\mathbb{E}G_{\text{comp, NW}}$, are of the order of $\mathbb{E}G_{\text{comp, NW}}(\epsilon) \leq O\left(\frac{K^{1/3}}{\epsilon^{5/3}} + \frac{K\alpha^{4/3}}{\epsilon^{5/3}}\right)$. Moreover, if $\alpha = 0$, we get the expected computational complexity and the expected number of gradient computations across the network as $\mathbb{E}G_{\text{comp, CN}}(\epsilon) \leq O\left(\frac{1}{\epsilon^{5/3}K^{2/3}}\right)$ and $\mathbb{E}G_{\text{comp, NW}}(\epsilon) \leq O\left(\frac{K^{1/3}}{\epsilon^{5/3}}\right)$.

A.14 Proof of Lemma 4.1

Here, we prove the result for the $d^2(\cdot,\cdot)$ cost function, and the proof extends similarly to $d(\cdot,\cdot)$ as well. The proof is independent of the choice of the distance function, and we only use the properties of the assignment matrix. First note that,

$$\begin{aligned}\sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, g, C) &= \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{p}\in P_i} g(\mathbf{p}) d^2(\mathbf{p}, C) \\ &= \sum_{i\in\mathcal{R}} b_i \sum_{j\in[n]} A_{i,j} g(\mathbf{p}_j)\, d^2(\mathbf{p}_j, C) \\ &= \sum_{j\in[n]} g(\mathbf{p}_j)\, d^2(\mathbf{p}_j, C) \sum_{i\in\mathcal{R}} b_i A_{i,j}. \end{aligned} \tag{A.103}$$

From Property 1 we know that for any $j \in [n]$, $\sum_{i\in\mathcal{R}} b_i A_{i,j} \le 1+\delta$. By combining this fact with (A.103), we obtain that

$$\begin{aligned}\sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, g, C) &= \sum_{j\in[n]} g(\mathbf{p}_j)\, d^2(\mathbf{p}_j, C) \sum_{i\in\mathcal{R}} b_i A_{i,j} \\ &\le (1+\delta) \sum_{j\in[n]} g(\mathbf{p}_j)\, d^2(\mathbf{p}_j, C) \\ &= (1+\delta)\cdot \text{cost}(P, g, C)\end{aligned}$$

Similarly, Property 1 ensures that for any $j \in [n]$, $\sum_{i\in\mathcal{R}} b_i A_{i,j} \ge 1$. Utilizing this fact in (A.103) gives us the desired lower bound as follows.

$$\begin{aligned}\sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, g, C) &= \sum_{j\in[n]} g(\mathbf{p}_j)\, d^2(\mathbf{p}_j, C) \sum_{i\in\mathcal{R}} b_i A_{i,j} \\ &\ge \sum_{j\in[n]} g(\mathbf{p}_j)\, d^2(\mathbf{p}_j, C) \\ &= \text{cost}(P, C).\end{aligned}$$

A.15 Proof of Lemma 4.3

We prove each part of the inequality separately. First, we prove the upper bound on cost(Y, g, C) followed by the lower bound.

Upper Bound: We first show that for any set of k-centers $C \subset \mathbb{R}^d$, and for any $i \in [m]$, cost$(Y_i, g_i, C) \leq (1+\alpha)$cost$(P_i, C)$ which ensures that the weighted k-centers (Y_i, g_i) are a good representation of the partial dataset P_i. Consider the following

$$\begin{aligned}
\text{cost}(Y_i, g_i, C) &= \sum_{\mathbf{y} \in Y_i} g_i(\mathbf{y}) d(\mathbf{y}, C) \\
&= \sum_{\mathbf{y} \in Y_i} |\text{cluster}(\mathbf{y}, P_i)| d(\mathbf{y}, C) \qquad \text{(by definition of } g_i\text{)} \\
&= \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} d(\mathbf{y}, C). \qquad \text{(A.104)}
\end{aligned}$$

For any $\mathbf{x} \in \mathbb{R}^d$, recall that $C(\mathbf{x})$ denotes its closest center in C. From the above equality, we have

$$\begin{aligned}
\text{cost}(Y_i, g_i, C) &= \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} d(\mathbf{y}, C(\mathbf{y})) \\
&\overset{(a)}{\leq} \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} d(\mathbf{y}, C(\mathbf{x})) \\
&\overset{(b)}{\leq} \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} (d(\mathbf{x}, \mathbf{y}) + d(\mathbf{x}, C(\mathbf{x}))) \\
&= \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} d(\mathbf{x}, \mathbf{y}) + \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, C(\mathbf{x})) \\
&= \text{cost}(P_i, Y_i) + \text{cost}(P_i, C) \\
&\overset{(c)}{\leq} (1+\alpha)\text{cost}(P_i, C), \qquad \text{(A.105)}
\end{aligned}$$

where (a) follows from the definition of $C(\mathbf{x})$ and (b) follows from triangular inequality. (c) follows from the fact that the k-centers Y_i on the partial dataset P_i is an α-approximate solution, and therefore, $\text{cost}(P_i, Y_i) \leq \alpha\, \text{cost}(P_i, C)$. Next, we have

$$\begin{aligned}\text{cost}(Y, g, C) &= \sum_{i\in\mathcal{R}} \text{cost}(Y_i, b_i \cdot g_i, C)\\ &= \sum_{i\in\mathcal{R}} b_i \text{cost}(Y_i, g_i, C)\\ &\leq (1+\alpha)\sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, C) \leq (1+\alpha)(1+\delta)\text{cost}(P, C), \qquad \text{(A.106)}\end{aligned}$$

where the first inequality follows from (A.105) and the second inequality follows from Lemma 4.1.

Lower Bound: From Lemma 4.1 for any set of k-centers C, we have

$$\begin{aligned}\text{cost}(P, C) &\leq \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, C)\\ &= \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{x}\in P_i} d(\mathbf{x}, C(\mathbf{x})). \qquad \text{(A.107)}\end{aligned}$$

From the definition of cluster centers, we know that for any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and for any set of k-centers C, $d(\mathbf{x}, C(\mathbf{x})) \leq d(\mathbf{x}, C(\mathbf{y}))$. Applying this observation in (A.107), we get

$$\begin{aligned}\text{cost}(P, C) &\leq \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{x}\in P_i} d(\mathbf{x}, C(\mathbf{x}))\\ &\leq \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{x}\in P_i} d(\mathbf{x}, C(Y_i(\mathbf{x}))), \qquad \text{(A.108)}\end{aligned}$$

where $Y_i(\mathbf{x})$ is the cluster center in Y_i closest to $\mathbf{x} \in P_i$. Using triangular inequality, we obtain

$$\begin{aligned}
\text{cost}(P, C) &\le \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{x}\in P_i} (d(\mathbf{x}, Y_i(\mathbf{x})) + d(Y_i(\mathbf{x}), C(Y_i(\mathbf{x})))) \\
&= \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) + \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{x}\in P_i} d(Y_i(\mathbf{x}), C(Y_i(\mathbf{x}))) \\
&= \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) + \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{x}\in Y_i} |\text{cluster}(\mathbf{y}, P_i)| d(\mathbf{y}, C(\mathbf{y})) \\
&= \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) + \sum_{i\in\mathcal{R}} b_i \text{cost}(Y_i, g_i, C) \\
&= \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) + \sum_{i\in\mathcal{R}} \text{cost}(Y_i, b_i \cdot g_i, C) \\
&= \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) + \text{cost}(Y, g, C). \qquad \text{(A.109)}
\end{aligned}$$

Combining the upper and the lower bounds with $\alpha = 1$, we obtain the final result.

A.16 Proof of Theorem 4.3

Utilizing the lower bound from Lemma 4.3 with $C = \hat{C}$, we have

$$\begin{aligned}
\text{cost}(P, \hat{C}) &\le \text{cost}(Y, g, \hat{C}) + \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, Y_i) \\
&\overset{(a)}{\le} \alpha\, \text{cost}(Y, g, C^*) + \alpha \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, C^*) \\
&\overset{(b)}{\le} \alpha(1+\alpha)(1+\delta)\text{cost}(P, C^*) + \alpha(1+\delta)\text{cost}(P, C^*) \\
&= \alpha(1+\delta)(2+\alpha)\text{cost}(P, C^*), \qquad \text{(A.110)}
\end{aligned}$$

where (a) follows from the fact that $\hat{C}$ and Y_i are the α-approximate set of centers for the weighted dataset (Y, g) and the partial dataset P_i, respectively. For (b), we utilize the upper bound in Lemma 4.3 and Lemma 4.1 with $C = C^*$.

A.17 Proof of 4.4

Lemma A.21. *For the k-means clustering, for any set of k-centers $C \subset \mathbb{R}^d$, we have*

$$\frac{1}{2}\mathit{cost}(P, C) - \sum_{i \in \mathcal{R}} b_i \mathit{cost}(P_i, Y_i) \leq \mathit{cost}(Y, g, C) \leq (2 + 2\alpha)(1 + \delta)\mathit{cost}(P, C).$$

PROOF: [Proof of Lemma A.21] We split the proof into two parts. The first part involves the upper bound and in the second part, we prove the lower bound.

Upper Bound: We first show that for any set of k-centers $C \subset \mathbb{R}^d$, for any $i \in [m]$, $\text{cost}(Y_i, g_i, C) \leq (2 + 2\alpha)\text{cost}(P_i, C)$ which ensures that the weighted k-centers (Y_i, g_i) are a good representation of the partial dataset P_i. Consider the following:

$$\begin{aligned} \text{cost}(Y_i, g_i, C) &= \sum_{\mathbf{y} \in Y_i} g_i(\mathbf{y}) d^2(\mathbf{y}, C) \\ &= \sum_{\mathbf{y} \in Y_i} |\text{cluster}(\mathbf{y}, P_i)| d^2(\mathbf{y}, C) \\ &= \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} d^2(\mathbf{y}, C). \end{aligned} \tag{A.111}$$

For any $\mathbf{x} \in \mathbb{R}^d$, recall that $C(\mathbf{x})$ denotes its closest center in C. From the above equality, we have

$$\begin{aligned}
\text{cost}(Y_i, g_i, C) &= \sum_{\mathbf{y}\in Y_i} \sum_{\mathbf{x}\in \text{cluster}(\mathbf{y}, P_i)} d^2(\mathbf{y}, C(\mathbf{y})) \\
&\overset{(a)}{\leq} \sum_{\mathbf{y}\in Y_i} \sum_{\mathbf{x}\in \text{cluster}(\mathbf{y}, P_i)} d^2(\mathbf{y}, C(\mathbf{x})) \\
&\overset{(b)}{\leq} \sum_{\mathbf{y}\in Y_i} \sum_{\mathbf{x}\in \text{cluster}(\mathbf{y}, P_i)} (2d^2(\mathbf{x}, \mathbf{y}) + 2d^2(\mathbf{x}, C(\mathbf{x}))) \\
&= \sum_{\mathbf{y}\in Y_i} \sum_{\mathbf{x}\in \text{cluster}(\mathbf{y}, P_i)} 2d^2(\mathbf{x}, \mathbf{y}) + \sum_{\mathbf{x}\in P_i} 2d^2(\mathbf{x}, C(\mathbf{x})) \\
&= 2\text{cost}(P_i, Y_i) + 2\text{cost}(P_i, C) \\
&\overset{(c)}{\leq} (2 + 2\alpha)\text{cost}(P_i, C), \qquad \text{(A.112)}
\end{aligned}$$

where (a) follows from the definition of $C(\mathbf{x})$ and (b) follows from scaled triangular inequality. (c) follows from the fact that Y_i is a set of α-approximate k centers on the partial dataset P_i, $\text{cost}(P_i, Y_i) \leq \alpha\ \text{cost}(P_i, C)$. Next, we have

$$\begin{aligned}
\text{cost}(Y, C, g) &= \sum_{i\in\mathcal{R}} \text{cost}(Y_i, C, b_i \cdot g_i) \\
&= \sum_{i\in\mathcal{R}} b_i \text{cost}(Y_i, C, g_i) \\
&\leq (2 + 2\alpha) \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, C) \leq (2 + 2\alpha)(1 + \delta)\text{cost}(P, C), \qquad \text{(A.113)}
\end{aligned}$$

where the first inequality follows from (A.112) and the second inequality follows from Lemma 4.1.

Lower Bound: From Lemma 4.1 for any set of k-centers C, we have

$$\begin{aligned}
\text{cost}(P, C) &\leq \sum_{i\in\mathcal{R}} b_i \text{cost}(P_i, C) \\
&= \sum_{i\in\mathcal{R}} b_i \sum_{\mathbf{x}\in P_i} d^2(\mathbf{x}, C(\mathbf{x})). \qquad \text{(A.114)}
\end{aligned}$$

From the definition of cluster centers, we know that for any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and for any set of k-centers, $d^2(\mathbf{x}, C(\mathbf{x})) \leq d^2(\mathbf{x}, C(\mathbf{y}))$. Applying this observation in (A.114), we get

$$\begin{aligned} \text{cost}(P, C) &\leq \sum_{i \in \mathcal{R}} b_i \sum_{\mathbf{x} \in P_i} d^2(\mathbf{x}, C(\mathbf{x})) \\ &\leq \sum_{i \in \mathcal{R}} b_i \sum_{\mathbf{x} \in P_i} d^2(\mathbf{x}, C(Y_i(\mathbf{x}))), \end{aligned} \tag{A.115}$$

where $Y_i(\mathbf{x})$ is the cluster center in Y_i closest to $\mathbf{x} \in P_i$. Using scaled triangular inequality, we obtain

$$\begin{aligned} \text{cost}(P, C) &\leq \sum_{i \in \mathcal{R}} b_i \sum_{\mathbf{x} \in P_i} (2d^2(\mathbf{x}, Y_i(\mathbf{x})) + 2d^2(Y_i(\mathbf{x}), C(Y_i(\mathbf{x})))) \\ &= \sum_{i \in \mathcal{R}} 2b_i \text{cost}(P_i, Y_i) + \sum_{i \in \mathcal{R}} b_i \sum_{\mathbf{x} \in P_i} 2d^2(Y_i(\mathbf{x}), C(Y_i(\mathbf{x}))) \\ &= \sum_{i \in \mathcal{R}} 2b_i \text{cost}(P_i, Y_i) + \sum_{i \in \mathcal{R}} b_i \sum_{\mathbf{x} \in Y_i} |\text{cluster}(\mathbf{y}, P_i)| 2d^2(\mathbf{y}, C(\mathbf{y})) \\ &= \sum_{i \in \mathcal{R}} 2b_i \text{cost}(P_i, Y_i) + \sum_{i \in \mathcal{R}} 2b_i \text{cost}(Y_i, C, g_i) \\ &= \sum_{i \in \mathcal{R}} 2b_i \text{cost}(P_i, Y_i) + \sum_{i \in \mathcal{R}} 2\text{cost}(Y_i, C, b_i \cdot g_i) \\ &= \sum_{i \in \mathcal{R}} 2b_i \text{cost}(P_i, Y_i) + 2\text{cost}(Y, C, g). \end{aligned} \tag{A.116}$$

Combining the upper and the lower bounds, we obtain the final result. ■

PROOF: [Proof of Theorem 4.4] Utilizing the lower bound from Lemma A.21 with $C = \hat{C}$, we have

$$\begin{aligned}
\text{cost}(P, \hat{C}) &\leq 2\text{cost}(Y, \hat{C}, g) + \sum_{i \in \mathcal{R}} 2b_i \text{cost}(P_i, Y_i) \\
&\overset{(a)}{\leq} 2\alpha \text{cost}(Y, C^*, g) + \ \alpha \sum_{i \in \mathcal{R}} 2b_i \text{cost}(P_i, C^*) \\
&\overset{(b)}{\leq} 2\alpha(2 + 2\alpha)(1 + \delta)\text{cost}(P, C^*) + 2\alpha(1 + \delta)\text{cost}(P, C^*) \\
&= 2\alpha(3 + 2\alpha)(1 + \delta)\text{cost}(P, C^*), \qquad \text{(A.117)}
\end{aligned}$$

where (a) follows from the fact that $\hat{C}$ and Y_i are the α-approximate set of centers for the weighted dataset (Y, g) and the partial dataset P_i, respectively. For (b), we utilize the upper bound in Lemma A.21 and Lemma 4.1 with $C = C^*$. ■

A.18 Proof of Lemma 4.4

For any $i \in \mathcal{R}$, note that the weighted point set (Y_i, g_i) is an δ-coreset of the partial dataset P_i. Hence, from the Definition 4.3, we have that for any set of k-centers $C \subset \mathbb{R}^d$,

$$(1 - \delta)\text{cost}(P_i, C) \leq \text{cost}(Y_i, g_i, C) \leq (1 + \delta)\text{cost}(P_i, C). \qquad \text{(A.118)}$$

For $Y = \cup_{i \in \mathcal{R}} Y_i$ and any set of k-centers C, we have

$$\begin{aligned}
\text{cost}(Y, g, C) &= \sum_{\mathbf{y} \in Y} g(\mathbf{y}) d^2(\mathbf{y}, C) \\
&= \sum_{i \in \mathcal{R}} b_i \sum_{\mathbf{y} \in Y_i} g_i(\mathbf{y}) d^2(\mathbf{y}, C) \\
&= \sum_{i \in \mathcal{R}} b_i \text{cost}(Y_i, g_i, C). \qquad \text{(A.119)}
\end{aligned}$$

Combining (A.119) and (A.118), we get

$$(1-\delta)\sum_{i\in\mathcal{R}} b_i\text{cost}(P_i, C) \le \text{cost}(Y, g, C) \le (1+\delta)\sum_{i\in\mathcal{R}} b_i\text{cost}(P_i, C). \tag{A.120}$$

Now using the above inequality and Lemma 4.1, we have

$$\begin{aligned}\text{cost}(Y, C, g) &\ge (1-\delta)\sum_{i\in\mathcal{R}} b_i\text{cost}(P_i, C)\\ &\ge (1-\delta)\text{cost}(P, C),\end{aligned} \tag{A.121}$$

and

$$\begin{aligned}\text{cost}(Y, C, g) &\le (1+\delta)\sum_{i\in\mathcal{R}} b_i\text{cost}(P_i, C)\\ &\le (1+\delta)(1+\delta)\text{cost}(P, C)\\ &\le (1+3\delta)\text{cost}(P, C) \qquad \text{for any } \delta \le 1.\end{aligned} \tag{A.122}$$

Combining the upper and the lower bounds, we obtain the final result.

A.19 Proof of Lemma 4.5

We prove both sides of the inequality separately.

Upper Bound: Using the definitions of cost(Y_i, g_i, C), and $g_i(\mathbf{y}) = |\text{cluster}(\mathbf{y}, P_i)|$, we get

$$\text{cost}(Y_i, g_i, C) = \sum_{\mathbf{y}\in Y_i}\ \sum_{x\in\text{cluster}(\mathbf{y}, P_i)} d(\mathbf{y}, C(\mathbf{y})) \tag{A.123}$$

$$\le \sum_{\mathbf{y}\in Y_i}\ \sum_{\mathbf{x}\in\text{cluster}(\mathbf{y}, P_i)} d(\mathbf{y}, C(\mathbf{x})). \tag{A.124}$$

Applying triangular inequality, we obtain

$$\text{cost}(Y_i, g_i, C) \leq \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} (d(\mathbf{x}, \mathbf{y}) + d(\mathbf{x}, C(\mathbf{x}))). \tag{A.125}$$

Splitting the summation into two terms, simplifying further, and utilizing the definition of cost$(\cdot, \cdot)$ yields the final result as the following.

$$\begin{aligned} \text{cost}(Y_i, g_i, C) &\leq \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, P_i)} d(\mathbf{x}, \mathbf{y}) + \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, C(\mathbf{x})) \\ &= \text{cost}(P_i, Y_i) + \text{cost}(P_i, C). \end{aligned} \tag{A.126}$$

Lower Bound: For any machine $i \in [m]$, we have

$$\text{cost}(P_i, C) = \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, C(\mathbf{x})).$$

Let $Y_i(x)$ be the cluster center in Y_i that is closest to $\mathbf{x} \in P_i$. Then, we get

$$\text{cost}(P_i, C) \leq \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, C(Y_i(\mathbf{x}))),$$

applying triangular inequality, we have

$$\text{cost}(P_i, C) \leq \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, Y_i(\mathbf{x})) + \sum_{\mathbf{x} \in P_i} d(Y_i(\mathbf{x}), C(Y_i(\mathbf{x}))).$$

simplifying further, and utilizing the definitions of cost(P_i, Y_i) and cost(Y_i, g_i, C), we obtain the final result.

$$\begin{aligned} \text{cost}(P_i, C) &\leq \text{cost}(P_i, Y_i) + \sum_{\mathbf{y} \in Y_i} |\text{cluster}(\mathbf{y}, P_i)| d(\mathbf{y}, C(\mathbf{y})) \\ &= \text{cost}(P_i, Y_i) + \text{cost}(Y_i, g_i, C), \end{aligned}$$

A.20 Proof of Theorem 4.6

Let $\hat{C}$ be the set of k-centers returned by Algorithm 8. From Lemma 4.2, we have

$$\text{cost}(P, \hat{C}) \leq \sum_{i=1}^{m-t} \rho \, \text{cost}(P_i, \hat{C}),$$

utilizing the result from Lemma 4.5 with $C = \hat{C}$, we get

$$\text{cost}(P, \hat{C}) \leq \sum_{i=1}^{m-t} \rho \, \text{cost}(P_i, \hat{C}) \leq \sum_{i=1}^{m-t} \rho \, \text{cost}(P_i, Y_i) + \sum_{i=1}^{m-t} \rho \, \text{cost}(Y_i, g_i, \hat{C}).$$

Next, we note that for every Byzantine in $j \in [m-t]$, there is an honest machine $i \in \mathcal{R}$ with a higher cost, $\text{cost}(P_i, Y_i) \leq \text{cost}(P_j, Y_j)$, which yields the following.

$$\text{cost}(P, \hat{C}) \leq \sum_{i \in \mathcal{R}} \rho \, \text{cost}(P_i, Y_i) + \sum_{i=1}^{m-t} \rho \, \text{cost}(Y_i, g_i, \hat{C}).$$

Since Y_i is an α approximate k-median solution on the partial dataset P_i, we have $\text{cost}(P_i, Y_i) \leq \alpha \, \text{cost}(P_i, C^*)$. Hence, we have

$$\text{cost}(P, \hat{C}) \leq \alpha \sum_{i \in \mathcal{R}} \rho \, \text{cost}(P_i, C^*) + \sum_{i=1}^{m-t} \rho \, \text{cost}(Y_i, g_i, \hat{C}).$$

We apply the result from Lemma 4.2 to the first term. Utilizing the definition of the cost function on a weighted point set, $\text{cost}(Y, g, \hat{C})$ and the α approximate solution $\hat{C}$ of the weighted dataset (Y, g) in the second term, we obtain

$$\text{cost}(P, \hat{C}) \leq \alpha(1+\delta)\text{cost}(P, C^*) + \alpha \, \text{cost}(Y, g, C^*).$$

From the definition of the cost function, cost(Y, g, C^*), we get

$$\begin{aligned}\text{cost}(P, \hat{C}) &\leq \alpha(1+\delta)\text{cost}(P, C^*) + \alpha \sum_{i=1}^{m-t} \text{cost}(Y_i, \rho g_i, C^*) \\ &\leq \alpha(1+\delta)\text{cost}(P, C^*) + \alpha \sum_{i=1}^{m-t} \rho \,\text{cost}(Y_i, g_i, C^*).\end{aligned}$$

Next, applying the result from Lemma 4.5 to the second term above, we have

$$\text{cost}(P, \hat{C}) \leq \alpha(1+\delta)\text{cost}(P, C^*) + \alpha \sum_{i=1}^{m-t} \rho \,\text{cost}(P_i, Y_i) + \alpha \sum_{i=1}^{m-t} \rho \,\text{cost}(P_i, C^*).$$

For the second term above, using a similar manipulation as before, we obtain

$$\text{cost}(P, \hat{C}) \leq \alpha(1+\delta)\text{cost}(P, C^*) + \alpha \sum_{i \in \mathcal{R}} \rho \,\text{cost}(P_i, C^*) + \alpha \sum_{i=1}^{m-t} \rho \,\text{cost}(P_i, C^*),$$

applying Lemma 4.2 to the second and third terms, we obtain

$$\begin{aligned}\text{cost}(P, \hat{C}) &\leq \alpha(1+\delta)\text{cost}(P, C^*) + \alpha(1+\delta)\text{cost}(P, C^*) + \alpha(1+\delta)\text{cost}(P, C^*) \\ &= 3\alpha(1+\delta)\text{cost}(P, C^*)\end{aligned}$$

A.21 Proof of Theorem 4.7

PROOF: [Proof of Lemma 4.6] The weight $\tilde{g}_i(\mathbf{y})$ can be written as

$$\begin{aligned}\tilde{g}_i(\mathbf{y}) &= \sum_{p \in \text{cluster}(\mathbf{y}, \tilde{P}_i)} w_i(p) \\ &= \sum_{p \in \text{cluster}(\mathbf{y}, P_i)} w_i(p) \mathbb{1}(p \in \tilde{P}_i). \qquad \text{(A.127)}\end{aligned}$$

Applying expectation on both sides where the randomness is due to the sampling while constructing the coreset [12], we obtain

$$\mathbb{E}[\tilde{g}_i(\mathbf{y})] = \sum_{p \in \text{cluster}(\mathbf{y}, P_i)} w_i(p)\mathbb{P}(p \in \tilde{P}_i).$$

From [12], we know that $w_i(p)\mathbb{P}(p \in \tilde{P}_i) = 1$. Therefore, we have

$$\mathbb{E}[\tilde{g}_i(\mathbf{y})] = \sum_{p \in \text{cluster}(\mathbf{y}, P_i)} 1 = g_i(\mathbf{y}). \tag{A.128}$$

From Chernoff's inequality, we have $\mathbb{P}(|g_i(\mathbf{y}) - \tilde{g}_i(\mathbf{y})| \leq \gamma g_i(\mathbf{y})) \geq 1 - e^{-2\gamma^2 g_i(\mathbf{y})^2 |P_i|}$, for a given $i \in [m-t]$ and $\mathbf{y} \in Y_i$.

Taking union bound over all $i \in [m-t]$ and $\mathbf{y} \in Y_i$, and setting $\gamma^2 \geq \frac{\log k}{(\min_{i,y} g_i(\mathbf{y})^2 |P_i|) \log(k(m-t))}$, the above inequality holds with probability at least $1 - \frac{1}{k}$. We assume that a cluster includes itself, thus ensuring that $g_i(\mathbf{y}) \geq 1$. Note that an upper bound for $g_i(\mathbf{y})$ and $|P_i|$ is n. Therefore, $\gamma^2 \geq \frac{\log k}{n^3 \log(k(m-t))}$. Thus, we choose $\gamma = 1/k$ which satisfies the inequality. ■

PROOF: [Proof of Lemma 4.7]

We prove the upper and the lower bound separately.

Upper bound: Expanding the definition of cost$(Y_i, \tilde{g}_i, C)$ and $\tilde{g}_i(\mathbf{y})$ for any $\mathbf{y} \in Y_i$, we have

$$\begin{aligned}
\text{cost}(Y_i, \tilde{g}_i, C) &= \sum_{\mathbf{y} \in Y_i} \tilde{g}_i(\mathbf{y}) d(\mathbf{y}, C(\mathbf{y})) \\
&\leq \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, \tilde{P}_i)} w_i(\mathbf{x}) d(\mathbf{y}, C(\mathbf{x})) \\
&\overset{(a)}{\leq} \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, \tilde{P}_i)} w_i(\mathbf{x}) d(\mathbf{y}, \mathbf{x}) + \sum_{\mathbf{y} \in Y_i} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{y}, \tilde{P}_i)} w_i(\mathbf{x}) d(\mathbf{x}, C(\mathbf{x})) \\
&= \text{cost}(\tilde{P}_i, w_i, Y_i) + \text{cost}(\tilde{P}_i, w_i, C) \\
&\overset{(b)}{\leq} \text{cost}(\tilde{P}_i, w_i, Y_i) + (1+\delta)\text{cost}(P_i, C).
\end{aligned} \tag{A.129}$$

The inequality (a) follows from triangular inequality, and (b) follows from the fact that $(\tilde{P}_i, w_i)$ is a δ-coreset of P_i as mentioned in Observation 1.

Lower bound: For any machine $i \in [m]$ with any set of centers C, we have

$$\text{cost}(P_i, C) = \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, C(\mathbf{x})).$$

Let $Y_i(x)$ be the cluster center in Y_i that is closest to $x \in P_i$. Then, we get

$$\text{cost}(P_i, C) \leq \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, C(Y_i(\mathbf{x}))),$$

applying triangular inequality, we have

$$\text{cost}(P_i, C) \leq \sum_{\mathbf{x} \in P_i} d(\mathbf{x}, Y_i(\mathbf{x})) + \sum_{\mathbf{x} \in P_i} d(Y_i(\mathbf{x}), C(Y_i(\mathbf{x}))).$$

For any $\mathbf{y} \in Y_i$, define $g_i(\mathbf{y}) := |\text{cluster}(\mathbf{y}, P_i)|$. Simplifying further, and utilizing the definitions of $\text{cost}(P_i, Y_i)$ and $\text{cost}(Y_i, g_i, C)$, we obtain

$$\begin{aligned} \text{cost}(P_i, C) &\leq \text{cost}(P_i, Y_i) + \sum_{\mathbf{y} \in Y_i} |\text{cluster}(\mathbf{y}, P_i)| d(\mathbf{y}, C(\mathbf{y})) \\ &= \text{cost}(P_i, Y_i) + \text{cost}(Y_i, g_i, C). \end{aligned} \tag{A.130}$$

Now, using Observation 2, we know that with probability at least $1 - 1/k$,

$$\text{cost}(Y_i, g_i, C) \leq \frac{1}{1-\gamma} \text{cost}(Y_i, \tilde{g}_i, C).$$

Plugging this back in Equation A.130, and rearranging the terms, we get that

$$\begin{aligned} \text{cost}(Y_i, \tilde{g}_i, C) &\geq (1-\gamma)\text{cost}(P_i, C) - (1-\gamma)\text{cost}(P_i, Y_i) \\ &\geq (1-\gamma)\text{cost}(P_i, C) - \frac{(1-\gamma)}{1-\delta}\text{cost}(\tilde{P}_i, w_i, Y_i), \end{aligned}$$

where the last inequality follows from the fact that $(\tilde{P}_i, w_i)$ is a δ-coreset of P_i (Observation 1). ■

PROOF: [Proof of Theorem 4.7]

We need to show that cost$(P, \hat{C}) \leq \alpha$cost(P, C^*), for some $\alpha \geq 1$. Starting from the LHS, using the lower bound from Lemma 4.7, we get

$$\begin{aligned}\text{cost}(P, \hat{C}) &\leq \sum_{i=1}^{m-t} \rho\, \text{cost}(P_i, \hat{C}) \\ &\leq \underbrace{\frac{\rho}{1-\delta} \sum_{i=1}^{m-t} \text{cost}(\tilde{P}_i, w_i, Y_i)}_{(Term1)} + \underbrace{\frac{\rho}{1-\gamma} \sum_{i=1}^{m-t} \text{cost}(Y_i, \tilde{g}_i, \hat{C})}_{(Term2)} . \qquad \text{(A.131)}\end{aligned}$$

We now bound each of the terms in Equation A.131 separately.

Term 1

$$\begin{aligned}\frac{\rho}{1-\delta} \sum_{i=1}^{m-t} \text{cost}(\tilde{P}_i, w_i, Y_i) &\overset{(a)}{\leq} \frac{\rho}{1-\delta} \sum_{i \in \mathcal{R}} \text{cost}(\tilde{P}_i, w_i, Y_i) \\ &\overset{(b)}{\leq} \frac{\rho(1+\delta)}{1-\delta} \sum_{i \in \mathcal{R}} \text{cost}(P_i, Y_i) \\ &\overset{(c)}{\leq} \frac{\rho(1+\delta)}{1-\delta} \sum_{i \in \mathcal{R}} \text{cost}(P_i, C^*) \\ &\overset{(d)}{\leq} \frac{(1+\delta)^2}{1-\delta} \text{cost}(P, C^*). \qquad \text{(A.132)}\end{aligned}$$

Since for every Byzantine in the first $[m-t]$ machines, there will exist an honest machine with higher cost, (a) follows. (b) follows from the fact that $(\tilde{P}_i, w_i)$ is a δ-coreset of P_i. The optimality of the centers Y_i on P_i computed at the honest machines implies (c). Finally, (d) follows from the property of the assignment matrix shown in Lemma 4.2.

Term 2

$$\begin{aligned}\frac{\rho}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(Y_i,\tilde{g}_i,\hat{C}) &\overset{(a)}{=} \frac{1}{1-\gamma}\text{cost}(Y,\tilde{g},\hat{C})\\ &\overset{(b)}{\leq} \frac{1}{1-\gamma}\text{cost}(Y,\tilde{g},C^*)\\ &\overset{(c)}{\leq} \frac{\rho}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(Y_i,\tilde{g}_i,C^*) \qquad \text{(A.133)}\end{aligned}$$

(a) and (c) follow from the definitions of Y and $\tilde{g}$, and the optimality of the k-centers $\hat{C}$ on (Y,g) implies (b).

Now using the upper bound from Lemma 4.7, continuing from Equation A.133, we get

$$\begin{aligned}\frac{\rho}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(Y_i,\tilde{g}_i,\hat{C}) &\leq \frac{\rho}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(Y_i,\tilde{g}_i,C^*)\\ &\leq \underbrace{\frac{\rho(1+\delta)}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(P_i,C^*)}_{\text{Term 21}} + \underbrace{\frac{\rho}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(\tilde{P}_i,w_i,Y_i)}_{\text{Term 22}} \qquad \text{(A.134)}\end{aligned}$$

(A.135)

Term 21, by the property of the assignment matrix is equivalent to

$$\frac{\rho(1+\delta)}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(P_i,C^*) = \frac{(1+\delta)^2}{1-\gamma}\,\text{cost}(P,C^*) \qquad \text{From Lemma 4.2}$$

Also, observe that Term 22 is just a scaled version of Term 1 simplified above in Equation A.132. Therefore,

$$\frac{\rho}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(\tilde{P}_i,w_i,Y_i) \leq \frac{(1+\delta)^2}{1-\gamma}\,\text{cost}(P,C^*)$$

Plugging these two inequalities back in Equation A.134, we get that Term 2 is bounded by

$$\frac{\rho}{1-\gamma}\sum_{i=1}^{m-t}\text{cost}(Y_i, \tilde{g}_i, \hat{C}) \leq \frac{2(1+\delta)^2}{1-\gamma}\,\text{cost}(P, C^*) \tag{A.136}$$

Finally, combining Equation A.132 and Equation A.136 in Equation A.131 we get

$$\begin{aligned}\text{cost}(P, \hat{C}) &\leq (1+\delta)^2\left(\frac{1}{1-\delta}+\frac{2}{1-\gamma}\right)\text{cost}(P, C^*)\\ &\leq (1+3\delta)\left(\frac{1}{1-\delta}+\frac{2}{1-\gamma}\right)\text{cost}(P, C^*) \qquad \text{(for any } \delta \in (0,1]).\end{aligned}$$

■

A.22 Proof of Theorem 4.10

Recall that $\mathcal{R} \subseteq [m]$ denotes the set of stragglers. Then, for any $i \in [m]$, we have

$$\mathbb{P}(i \in \mathcal{R}) = 1 - p_t. \tag{A.137}$$

Next, we argue that for any $\delta > 0$, we can choose $p_a = \frac{\ell}{m}$ large enough to ensure Property 1 with high probability. First, we analyze the weight of each of the column in the random matrix. For $i \in [m]$ and $j \in [n]$, define an event $E_{i,j}$ as follows

$$E_{i,j} = \begin{cases} 1 & \text{if } i \in \mathcal{R} \text{ and } A_{i,j} = 1\\ 0 & \text{otherwise.}\end{cases} \tag{A.138}$$

Note that for any fixed $j \in [n]$, $\{E_{i,j}\}_{i\in[m]}$ is a collection of m independent events. Further, it follows from (4.6) and (A.137) that

$$\mathbb{P}(E_{i,j} = 1) = p_a(1 - p_t).$$

Note that

$$\mathbb{E}\left[\sum_{i=1}^{m} E_{i,j}\right] = mp_a(1-p_t) = \ell(1-p_t).$$

It then follows from standard Chernoff bound that for any $\gamma \in (0,1)$, we have

$$\mathbb{P}(\left|\sum_{i=1}^{m} E_{i,j} - \ell(1-p_t)\right| \geq \gamma(1-p_t)) \leq 2e^{-\frac{\gamma^2 \ell(1-p_t)}{3}}. \tag{A.139}$$

Specifically, if we choose $\gamma = \frac{\delta}{2+\delta}$ and $\ell = \frac{6\log(n\sqrt{2})}{\gamma^2(1-p_t)}$, then with probability at least $1 - \frac{1}{n^2}$ the following holds for a given $j \in [n]$

$$1 \leq \frac{1}{(1-\gamma)\ell(1-p_t)} \sum_{i=1}^{m} E_{i,j} \leq 1+\delta.$$

Now, taking a union bound over all $j \in [n]$, we have with probability at least $1 - \frac{1}{n}$,

$$1 \leq \frac{1}{(1-\gamma)\ell(1-p_t)} \sum_{i=1}^{m} E_{i,j} \leq 1+\delta, \forall j \in [n]. \tag{A.140}$$

Recall that to establish Property 1, we need to show that there exists a non-negative vector $\mathbf{b} \in \mathbb{R}^{|\mathcal{R}|}$ such that

$$\mathbf{b}^T A_{\mathcal{R}} = (a_1, \ldots, a_n),$$

where $\mathbf{b} = \frac{1}{(1-\gamma)\ell(1-p_t)} \cdot (1, \ldots, 1)$ as a candidate. Note that for this choice of $\mathbf{b}$, we have

$$\mathbf{b}^T B_{\mathcal{R}} = \frac{1}{(1-\gamma)\ell(1-p_t)} \cdot \left(\sum_{i=1}^{m} E_{i,1}, \ldots, \sum_{i=1}^{m} E_{i,n}\right).$$

It follows from (A.140) that with probability at least $1 - \frac{1}{n}$, each of the coordinates of $\mathbf{b}^T B_{\mathcal{R}}$ falls in the interval $[1, 1+\delta]$. This completes the proof.

A.23 Proof of Theorem 4.11

Recall that $\mathcal{R} \subseteq [m]$ indicates the set of honest nodes. Then, for any $i \in [m]$, we have

$$\Pr\{i \in \mathcal{R}\} = 1 - p_t. \tag{A.141}$$

Next, we show that the proposed construction satisfies Property 2 with high probability.

Consider the block of $\mathbf{B}_i = \mathbf{1}_{s\times s}$, of A for any $i \in [m/s]$. First we show that for any block and a random set $\mathcal{R}$ of honest machines, the weights of every column concentrates around it expected values.

For any block $i \in [m/s]$ and row in block $j \in [s]$, we define an event $F_{i,j}$ as follows:

$$F_{i,j} = \begin{cases} 1 & \text{if row } j \text{ in block } i \in \mathcal{R} \\ 0 & \text{otherwise.} \end{cases} \tag{A.142}$$

From (A.141), we know that

$$\Pr\{F_{i,j} = 1\} = 1 - p_t. \tag{A.143}$$

Therefore, for any fixed block i of s rows, we have

$$\mathbb{E}\left[\sum_{j=1}^{s} F_{i,j}\right] = s(1 - p_t). \tag{A.144}$$

Utilizing Chernoff bound, for any $\gamma \in (0, 1)$, we have

$$\Pr\left\{\left|\sum_{j=1}^{s} F_{i,j} - s(1 - p_t)\right| \geq \gamma s(1 - p_t)\right\} \leq 2e^{-\frac{\gamma^2 s(1-p_t)}{3}}. \tag{A.145}$$

So, with high probability, the random set of Byzantines leave about $s(1-p_t)(1\pm\gamma)$ rows unaffected in each block. So summing over the rows in block i of $A_{\mathcal{R}}$, we get that with probability at least

$1 - e^{-\Omega(s(1-p_t))}$,

$$s(1-p_t)(1-\gamma)\mathbf{1}_s^T \leq \sum_{j\in[s]} F_{i,j}\mathbf{B}_{i,j} \leq s(1-p_t)(1+\gamma)\mathbf{1}_s^T.$$

where, $\mathbf{B}_{i,j}$ denotes the j-th row in the i-th block $\mathbf{B}_i$.

Setting $\gamma = \frac{\delta}{2+\delta}$, then with high probability the following holds for a given $j \in [m]$.

$$\mathbf{1}_s^T \leq \frac{1}{(1-\gamma)s(1-p_t)} \sum_{j\in[s]} F_{i,j}\mathbf{B}_{i,j} \leq (1+\delta)\mathbf{1}_s^T. \tag{A.146}$$

Taking union bound over all blocks $i \in [m/s]$, we have with the probability at least $1-\frac{m}{s}e^{-\Omega(s(1-p_t))}$,

$$\mathbf{1}_s^T \leq \frac{1}{(1-\gamma)s(1-p_t)} \sum_{j\in[s]} F_{i,j}\mathbf{B}_{i,j} \leq (1+\delta)\mathbf{1}_s^T,\ \forall i \in [m/s]. \tag{A.147}$$

The result then follows from the fact that all the blocks are in mutually exclusive rows of A. Setting $s = O(\log m)$ for a constant p_t, we see that the assignment scheme satisfies Property 2 with probability at least $1 - O(1/m)$ and $\rho = \frac{1}{(1-\gamma)s(1-p_t)}$, where $\gamma = \frac{\delta}{2+\delta}$.

A.24 Proof of Theorem 4.12

The proof follows from the observation that on deleting any set of t rows, the column weights in $A_{\mathcal{R}}$ are almost preserved with high probability.

Let $\mathcal{B} \subset [m]$ denote a fixed set of t Byzantines. the rows of A indexed by $B \subset [m]$, the expected weight of a fixed column j is $p(m-t)$. Therefore, from standard Chernoff bounds it follows that

$$\Pr[|\mathrm{wt}(A'_j) - p(m-t)| \geq \gamma p(m-t)] \leq e^{-\frac{\gamma^2}{3}p(m-t)},$$

where $\mathrm{wt}(A'_j)$ denotes the number of non-zero entries in the j-th column of $A_{\mathcal{R}}$ - the submatrix of A obtained from deleting the rows in B.

By a union bound over all $\binom{m}{t}$ subsets of rows and all n columns of A, we get that with probability at least

$$1 - n \cdot m^t \cdot e^{-\frac{\gamma^2}{3}p(m-t)},$$

all columns of A will have weight in the range $[(1-\gamma)p(m-t), (1+\gamma)p(m-t)]$. Therefore, setting $\rho = (1-\gamma)p(m-t)$, we get that for any set of B of t rows,

$$\mathbf{1}_n^T \le \rho \sum_{i \in [m] \setminus B} \mathbf{a}_i \le (1+\delta)\mathbf{1}_n^T$$

for $\delta = \frac{2\gamma}{1-\gamma}$.

Setting $p = O(1/\log m)$, the result follows for any $t = O(m/\log^2 m)$, with probability at least $1 - 1/m$.

A.25 Proof of Theorem 4.13

Let $G = (L \cup R, E)$ be the double cover of a c-regular expander graph on m vertices with expansion $\lambda = \max\{|\lambda_2|, |\lambda_n|\}$

We construct the $m \times m$ assignment matrix A from G by setting $A_{u,v} = 1$ if there is an edge between $(u, v) \in G$ for any $u \in R$ and, $v \in L$. Note that each column of A has weight exactly c. Also, any set of t Byzantines will now correspond to a set of t vertices in R. We show that removing any set of t vertices from R does not reduce the individual degrees of any vertex $v \in L$ by a lot. This implies that the column weight in $A_{\mathcal{R}}$ is almost preserved.

Using Expander Mixing Lemma, we get that for any vertex $v \in L$, and any set of t vertices $B \subset R$,

$$\begin{aligned} |E(\{v\}, B)| &\le \frac{c}{m}t + \lambda\sqrt{t} \\ &= c\left(\frac{t}{m} + \frac{\lambda}{c}\sqrt{t}\right). \end{aligned}$$

Therefore, for $\left(\frac{t}{m} + \frac{\lambda}{c}\sqrt{t}\right) = \gamma$, all vertices $v \in L$ are connected to at most $c\gamma$ machines in any set of t machines in R. So on deleting any set of t vertices in R all the vertices $v \in L$ will have degree deg$(v) \in [(1-\gamma)c, c]$.

Therefore, setting $\rho = \frac{1}{c(1-\gamma)}$, we satisfy $\sum_{i \in \mathcal{R}} \mathbf{a}_i \leq \frac{1}{1-\gamma}\mathbf{1}_n^T = (1+\delta)\mathbf{1}_n^T$, for $\gamma = \frac{\delta}{1+\delta}$.

Using the expander constructions in [8], we get an assignment scheme that is resilient to any set of $t = O(\sqrt{\log m / \log\log m})$ Byzantines with an overhead of $O(\log m)$ tasks per machine.

References

[1] D. Alistarh, Z. Allen-Zhu, and J. Li, “Byzantine stochastic gradient descent,” in *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 4613–4623. 27, 28, 29, 30, 31, 32, 34, 35, 105

[2] Z. Allen-Zhu, F. Ebrahimian, J. Li, and D. Alistarh, “Byzantine-resilient non-convex stochastic gradient descent,” *arXiv preprint arXiv:2012.14368*, 2020. 29

[3] Z. Allen-Zhu, Y. Li, and Z. Song, “A convergence theory for deep learning via over-parameterization,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 242–252. 10

[4] P. Awasthi, M. Balcan, and C. White, “General and robust communication-efficient algorithms for distributed clustering,” *CoRR*, vol. abs/1703.00830, 2017. 46

[5] A. Bakshi, R. Jayaram, and D. P. Woodruff, “Learning two layer rectified neural networks in polynomial time,” in *Conference on Learning Theory*. PMLR, 2019, pp. 195–268. 10

[6] M.-F. F. Balcan, S. Ehrlich, and Y. Liang, “Distributed k-means and k-median clustering on general topologies,” *Advances in neural information processing systems*, vol. 26, 2013. 46, 47

[7] A. Bhaskara and M. Wijewardena, “Distributed clustering via lsh based data partitioning,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 570–579. 46

[8] Y. Bilu and N. Linial, “Lifts, discrepancy and nearly optimal spectral gap,” *Combinatorica*, vol. 26, no. 5, pp. 495–519, 2006. 72, 139

[9] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, “Machine learning with adversaries: Byzantine tolerant gradient descent,” in *Advances in Neural Information Processing Systems 30*, 2017, pp. 119–129. 27, 28, 29, 30, 47

[10] V. Braverman, V. Cohen-Addad, H.-C. S. Jiang, R. Krauthgamer, C. Schwiegelshohn, M. B. Toftrup, and X. Wu, “The power of uniform sampling for coresets,” in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 462–473. 63

[11] V. Braverman, D. Feldman, H. Lang, and D. Rus, “Streaming coreset constructions for m-estimators,” in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. 63

[12] V. Braverman, D. Feldman, H. Lang, A. Statman, and S. Zhou, “Efficient coreset constructions via sensitivity sampling,” in *Asian Conference on Machine Learning*. PMLR, 2021, pp. 948–963. 48, 63, 130

[13] S. Bubeck *et al.*, “Convex optimization: Algorithms and complexity,” *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015. 81

[14] S. Bulusu, V. Gandikota, A. Mazumdar, A. S. Rawat, and P. K. Varshney, “Byzantine resilient distributed clustering with redundant data assignment,” in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 2143–2148. 48, 49

[15] S. Bulusu, P. Khanduri, P. Sharma, and P. K. Varshney, “On distributed stochastic gradient descent for nonconvex functions in the presence of byzantines,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 3137–3141. 29

[16] S. Bulusu, Q. Li, and P. K. Varshney, “On convex stochastic variance reduced gradient for adversarial machine learning,” in *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2019, pp. 1–5. 29

[17] B. Buyukates, E. Ozfatura, S. Ulukus, and D. Gündüz, “Gradient coding with dynamic clustering for straggler mitigation,” in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6. 47

[18] J. Byrka, K. Sornat, and J. Spoerhase, “Constant-factor approximation for ordered k-median,” in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 2018, pp. 620–631. 57

[19] Y. Cao and Q. Gu, “Tight sample complexity of learning one-hidden-layer convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 32, 2019. 10

[20] Z. Charles, D. Papailiopoulos, and J. Ellenberg, “Approximate gradient coding via sparse random graphs,” *arXiv preprint arXiv:1711.06771*, 2017. 69

[21] J. Chen, H. Sun, D. Woodruff, and Q. Zhang, “Communication-optimal distributed clustering,” *Advances in Neural Information Processing Systems*, vol. 29, pp. 3727–3735, 2016. 46

[22] S. Chen, A. R. Klivans, and R. Meka, “Learning deep RELU networks is fixed-parameter tractable,” in *IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 696–707. 10

[23] ——, “Learning deep relu networks is fixed-parameter tractable,” in *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022, pp. 696–707. 78

[24] Y. Chen, L. Su, and J. Xu, “Distributed statistical machine learning in adversarial settings: Byzantine gradient descent,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017. 16, 27, 47

[25] V. Cohen-Addad, K. G. Larsen, D. Saulpic, and C. Schwiegelshohn, "Towards optimal lower bounds for k-median and k-means coresets," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1038–1051. 60

[26] V. Cohen-Addad, D. Saulpic, and C. Schwiegelshohn, "A new coreset framework for clustering," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 169–182. 60

[27] C. Daskalakis, T. Gouleakis, C. Tzamos, and M. Zampetakis, "Efficient statistics, in high dimensions, from truncated samples," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018, pp. 639–649. 4, 13, 14, 15, 81, 82

[28] D. Data and S. Diggavi, "On byzantine-resilient high-dimensional stochastic gradient descent," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 2628–2633. 47

[29] D. Data, L. Song, and S. Diggavi, "Data encoding methods for byzantine-resilient distributed optimization," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 2719–2723. 47

[30] D. Data and S. Diggavi, "Byzantine-resilient high-dimensional federated learning," *arXiv e-prints*, pp. arXiv–2006, 2020. 47

[31] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012. 27

[32] I. Diakonikolas, G. Kamath, D. Kane, J. Li, A. Moitra, and A. Stewart, "Robust estimators in high-dimensions without the computational intractability," *SIAM Journal on Computing*, vol. 48, no. 2, pp. 742–864, 2019. 4, 10

[33] I. Diakonikolas and D. M. Kane, "Recent advances in algorithmic high-dimensional robust statistics," *arXiv preprint arXiv:1911.05911*, 2019. 4, 10

[34] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1675–1685. 10

[35] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed sgd," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 803–812. 47

[36] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011, pp. 569–578. 60

[37] D. Feldman, M. Monemizadeh, C. Sohler, and D. P. Woodruff, "Coresets and sketches for high dimensional subspace approximation problems," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '10. USA: Society for Industrial and Applied Mathematics, 2010, p. 630–649. 60

[38] D. Feldman, M. Schmidt, and C. Sohler, "Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering," in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '13. USA: Society for Industrial and Applied Mathematics, 2013, p. 1434–1453. 52, 60

[39] ——, "Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering," *SIAM Journal on Computing*, vol. 49, no. 3, pp. 601–657, 2020. 60

[40] Z. Feng, P. Kacham, and D. Woodruff, "Dimensionality reduction for the sum-of-distances metric," in *International conference on machine learning*. PMLR, 2021, pp. 3220–3229. 60

[41] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition*, vol. 39, no. 5, pp. 761–775, 2006. 72

[42] S. Frei, Y. Cao, and Q. Gu, "Agnostic learning of a single neuron with gradient descent," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5417–5428, 2020. 10

[43] V. Gandikota, A. Mazumdar, and A. S. Rawat, "Reliable distributed clustering with redundant data assignment," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 2556–2561. 48, 49

[44] W. Gao, A. V. Makkuva, S. Oh, and P. Viswanath, "Learning one-hidden-layer neural networks under general input distributions," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1950–1959. 3

[45] A. Ghosh, R. K. Maity, S. Kadhe, A. Mazumdar, and K. Ramchandran, "Communication-efficient and byzantine-robust distributed learning," in *2020 Information Theory and Applications Workshop (ITA)*. IEEE, 2020, pp. 1–28. 47

[46] M. Glasgow and M. Wootters, "Approximate gradient coding with optimal decoding," *arXiv preprint arXiv:2006.09638*, 2020. 47, 71

[47] S. Goel, S. Karmalkar, and A. Klivans, "Time/accuracy tradeoffs for learning a ReLU with respect to gaussian marginals," *Advances in Neural Information Processing Systems*, vol. 32, 2019. 10

[48] S. Goel, A. Klivans, and R. Meka, "Learning one convolutional layer with overlapping patches," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1783–1791. 10

[49] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical computer science*, vol. 38, pp. 293–306, 1985. 46

[50] S. Guha, Y. Li, and Q. Zhang, "Distributed partial clustering," in *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '17. Association for Computing Machinery, 2017, p. 143–152. 46

[51] ——, "Distributed partial clustering," *ACM Transactions on Parallel Computing (TOPC)*, vol. 6, no. 3, pp. 1–20, 2019. 51

[52] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: The approach based on influence functions*. John Wiley & Sons, 2011, vol. 196. 10

[53] S. Han, "Systematic design of decentralized algorithms for consensus optimization," *IEEE Control Systems Letters*, vol. 3, no. 4, pp. 966–971, 2019. 79

[54] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. Ganger, and E. P. Xing, "More effective distributed ml via a stale synchronous parallel parameter server," in *Advances in neural information processing systems*, 2013, pp. 1223–1231. 27

[55] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bulletin of the American Mathematical Society*, vol. 43, no. 4, pp. 439–561, 2006. 71

[56] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73 – 101, 1964. 12

[57] ——, *Robust statistics*. John Wiley & Sons, 2004, vol. 523. 10

[58] IHS, "Internet of things (iot) connected devices installed base world-wide from 2015 to 2025," 2016. 1

[59] X. Jia, K. Sheth, and O. Svensson, "Fair colorful k-center clustering," in *International Conference on Integer Programming and Combinatorial Optimization*, 2020, pp. 209–222. 80

[60] B. Jiang and S. Zhang, "Iteration bounds for finding the ϵ-stationary points for structured nonconvex optimization," *arXiv preprint arXiv:1410.4066*, 2014. 32

[61] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Advances in Neural Information Processing Systems*, 2018, pp. 2525–2536. 30, 35, 37

[62] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous univariate distributions, volume 1*. John wiley & sons, 1995, vol. 289. 14

[63] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013, pp. 315–323. 27

[64] D. M. Kane, "Robust learning of mixtures of Gaussians," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 1246–1258. 10

[65] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," *Advances in Neural Information Processing Systems*, vol. 30, 2017. 47

[66] M. Kleindessner, P. Awasthi, and J. Morgenstern, "Fair k-center clustering for data summarization," in *International Conference on Machine Learning*, 2019, pp. 3448–3457. 80

[67] P. Kothari, P. Manurangsi, and A. Velingker, "Private robust estimation by stabilizing convex relaxations," in *Conference on Learning Theory*. PMLR, 2022, pp. 723–777. 79

[68] K. A. Lai, A. B. Rao, and S. Vempala, "Agnostic estimation of mean and covariance," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, 2016, pp. 665–674. 10

[69] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982. 27, 47

[70] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2018. 47

[71] L. Lei, C. Ju, J. Chen, and M. I. Jordan, "Non-convex finite-sum optimization via scsg methods," in *Advances in Neural Information Processing Systems*, 2017, pp. 2348–2358. 27, 30, 36, 94, 95, 101, 102, 113

[72] Q. Lei, J. Lee, A. Dimakis, and C. Daskalakis, "SGD learns one-layer networks in WGANs," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5799–5808. 10

[73] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1544–1551. 28, 29, 30, 31

[74] X. Liu, W. Kong, S. Kakade, and S. Oh, "Robust and differentially private mean estimation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3887–3901, 2021. 79

[75] X. Liu, W. Kong, and S. Oh, "Differential privacy and robust statistics in high dimensions," in *Conference on Learning Theory*. PMLR, 2022, pp. 1167–1246. 79

[76] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017. 26

[77] G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley, "Fast distributed k-center clustering with outliers on massive data," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15, 2015, p. 1063–1071. 46, 47

[78] A. Mazumdar and A. S. Rawat, "Representation learning and recovery in the ReLU model," *arXiv preprint arXiv:1803.04304*, 2018. 10

[79] A. Mukherjee and R. Muthukumar, "Guarantees on learning depth-2 neural networks under a data-poisoning attack," *arXiv preprint arXiv:2005.01699*, 2020. 10

[80] U. Nations, "World population prospects: The 2015 revisionpopulation database," 2016. 1

[81] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2003, vol. 87. 32

[82] S. Oymak, "Stochastic gradient descent learns state equations with nonlinear activations," in *Conference on Learning Theory*. PMLR, 2019, pp. 2551–2579. 10

[83] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006. 27

[84] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, pp. 20–27. 27

[85] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient coding from cyclic mds codes and expander graphs," *IEEE Transactions on Information Theory*, vol. 66, no. 12, pp. 7475–7489, 2020. 47, 71

[86] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in neural information processing systems*, 2011, pp. 693–701. 27, 47

[87] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 48. PMLR, 20–22 Jun 2016, pp. 314–323. 27

[88] F. Saadatniaki, R. Xin, and U. A. Khan, "Decentralized optimization over time-varying directed graphs with row and column-stochastic matrices," *IEEE Transactions on Automatic Control*, vol. 65, no. 11, pp. 4769–4780, 2020. 79

[89] B. Safaei, A. M. H. Monazzah, M. B. Bafroei, and A. Ejlali, "Reliability side-effects in internet of things application layer protocols," in *2017 2nd International Conference on System Reliability and Safety (ICSRS)*, 2017, pp. 207–212. 1

[90] C. Sohler and D. P. Woodruff, "Strong coresets for k-median and subspace approximation: Goodbye dimension," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018, pp. 802–813. 60

[91] L. Su and J. Xu, "Securing distributed machine learning in high dimensions," *arXiv preprint arXiv:1804.10140*, 2018. 27, 28, 30, 31, 47

[92] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 06–11 Aug 2017, pp. 3368–3376. 47, 53

[93] K. Varadarajan and X. Xiao, "A near-linear algorithm for projective clustering integer points," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2012, pp. 1329–1342. 60

[94] ——, "On the sensitivity of shape fitting problems," in *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 18, Dagstuhl, Germany, 2012, pp. 486–497. 60

[95] S. Vempala and J. Wilmes, "Gradient descent for one-hidden-layer neural networks: Polynomial convergence and SQ lower bounds," in *Conference on Learning Theory*. PMLR, 2019, pp. 3115–3117. 10

[96] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–33, 2020. 27

[97] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*. Cambridge university press, 2018, vol. 47. 82

[98] M. J. Wainwright, *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019. 12

[99] H. Wang, Z. Charles, and D. Papailiopoulos, "Erasurehead: Distributed gradient descent without delays using approximate gradient coding," *arXiv preprint arXiv:1901.09671*, 2019. 47

[100] S. Wang, J. Liu, and N. Shroff, "Fundamental limits of approximate gradient coding," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–22, 2019. 47

[101] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*. Cambridge university press, 2011. 16

[102] S. Wu, A. G. Dimakis, and S. Sanghavi, "Learning distributions generated by one-layer ReLU networks," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8107–8117, 2019. 10, 12, 13, 14, 16, 19, 20, 82, 91, 92

[103] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020. 29

[104] C. Xie, O. Koyejo, and I. Gupta, "Generalized byzantine-tolerant SGD," *arXiv preprint arXiv:1802.10116*, 2018. 29

[105] ——, "Phocas: dimensional byzantine-resilient stochastic gradient descent," *arXiv preprint arXiv:1805.09682*, 2018. 27, 28, 29, 30, 39

[106] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6893–6901. 27, 28, 29, 30

[107] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015. 2

[108] H. Yang, X. Zhang, M. Fang, and J. Liu, "Byzantine-resilient stochastic gradient descent for distributed learning: A lipschitz-inspired coordinate-wise median approach," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 5832–5837. 29

[109] Z. Yang and W. Bajwa, "Bridge: Byzantine-resilient decentralized gradient descent," *ArXiv*, vol. abs/1908.08098, 2019. 28

[110] Z. Yang and W. U. Bajwa, "Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 611–627, 2019. 28

[111] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, "Defending against saddle point attack in byzantine-robust distributed learning," in *ICML*, 2019. 27, 28, 30

[112] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*, 2018, pp. 5650–5659. 16

[113] ——, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of

Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 5650–5659. 27, 28, 30, 47

[114] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," in *International Conference on Machine Learning*, 2019, pp. 7184–7193. 30, 35, 37

[115] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020. 47

[116] X. Zhang, Y. Yu, L. Wang, and Q. Gu, "Learning one-hidden-layer ReLU networks via gradient descent," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 1524–1534. 10

[117] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *Advances in neural information processing systems*, 2010, pp. 2595–2603. 27

VITA

NAME OF AUTHOR: Saikiran Bulusu

MAJOR: Electrical and Computer Engineering

EDUCATION:

M.Tech.	2012	Indian Institute of Technology Madras, Chennai, India
B.Tech.	2009	Jawaharlal Nehru Technological University, Hyderabad, India

AWARDS AND HONORS:

NSF travel grant, ISIT, 2023

Student travel award, Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS), 2022