

Ehsan Lari

Distributed Learning with Enhanced Efficiency, Robustness and Privacy

Thesis for the Degree of Philosophiae Doctor PhD

Trondheim, May 2025

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems



Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

© Ehsan Lari

ISBN (printed version)
ISBN (electronic version)
ISSN 1503-8181 (printed version)
ISSN 2703-8084 (online version)

Doctoral theses at NTNU, 2025:

Printed by NTNU Grafisk senter

در صحن چمن روی دل افروز خوش است
خوش باش وز دی گلو که امروز خوش است
خیام نیشابوری

بر چهره گل نسیم نوروز خوش است
از دی که گذشت هر چه گویی خوش نیست

To my dear parents, Behrooz and Shahla.

*Thanks for the lifetime supply of love, support, and forgivable loans.
This thesis is a small token of appreciation for your unwavering belief in
me and your constant encouragement.*

Abstract

Distributed machine learning for Internet of things (IoT) and cyber-physical systems (CPS) applications face increasing demands for efficiency, privacy, and robustness. This thesis focuses on addressing these challenges through two primary approaches, i.e., fully-distributed learning algorithms and federated learning (FL) algorithms, which form the main contributions of the work.

This thesis proposes machine learning algorithms for distributed and federated learning in the context of IoT and CPS applications. Developing distributed algorithms for artificial intelligence is necessary as centralized data processing may be unfeasible due to computational and communication costs and privacy concerns. The thesis addresses challenges in fully-distributed learning and federated learning (FL) settings, focusing on resilience against attacks, robustness to communication noise, and privacy preservation.

The main contributions of the thesis can be grouped into the following categories:

- resilience of partial-sharing-based online federated learning against model-poisoning attacks
- noise-robust and resource-efficient federated learning
- privacy-preserving distributed nonnegative matrix factorization
- distributed maximum consensus with noisy communication links

In the context of federated learning, we analyze the resilience of the partial-sharing-based online FL (PSO-Fed) algorithm to model-poisoning attacks. We show that PSO-Fed outperforms other communication-efficient FL algorithms against model-poisoning attacks without introducing additional computational burdens on the clients. Theoretical analysis and simulations demonstrate PSO-Fed's convergence properties and robustness against attacks, as well as revealing an optimal stepsize in the presence of model-poisoning attacks.

To address communication noise in federated learning, we propose a novel noise-robust and resource-efficient algorithm called RERCE-Fed. This algorithm introduces key modifications to counteract the adverse effects of communication noise and improve performance through continued local updates. Theoretical analysis confirms convergence in both mean and mean-square senses, with numerical results validating RERCE-Fed's effectiveness.

In the context of privacy-preserving distributed learning, we develop a distributed nonnegative matrix factorization (PPDNMF) algorithm. This algorithm ensures secure information exchange between neighboring agents using the Paillier cryptosystem, protecting local data from internal and external eavesdroppers.

Finally, we introduce a noise-robust distributed maximum consensus (RD-MC) algorithm for estimating the maximum value within multi-agent ad-hoc networks with noisy communication links. RD-MC redefines the conventional maximum consensus problem as a distributed optimization problem, employing techniques to enhance robustness against noise.

Overall, this thesis lays a solid foundation for the development of secure, efficient, and privacy-preserving distributed and federated learning algorithms for IoT applications, offering critical solutions to key challenges in deploying smart and collaborative systems in emerging IoT domains.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for the fulfillment of requirements for the degree of Doctor of Philosophy.

The doctoral work started in January 2020 at the Department of Electronic Systems, NTNU, Trondheim, Norway. The work has been supervised by Professor Stefan Werner and Dr. Reza Arablouei.

The doctoral work was funded by the Research Council of Norway.

Contents

Abstract	v
Preface	vii
Acknowledgements	ix
List of Tables	xv
List of Figures	xx
List of Algorithms	xxi
Abbreviations and Symbols	xxiii
1 Introduction	1
1.1 Motivation and Scope	1
1.2 Objectives	2
1.3 List of Publications	3
1.4 Structure and Contributions	4
2 From Distributed to Federated Learning over Networks	5

2.1	Background and Evolution	5
2.2	Foundations of Distributed Learning	8
2.2.1	Advantages of Distributed Learning	9
2.2.2	Challenges and Limitations of Distributed Learning	9
2.3	Federated Learning: A Shifting Frontier	10
2.3.1	Advantages of Federated Learning	12
2.3.2	Challenges of Federated Learning	13
2.4	Online Federated Learning	13
2.5	Partial Sharing of Information	14
2.6	Alternating Direction Method of Multipliers	16
2.7	Summary	17
3	Resilience of Partial Sharing Online FL to Model Poisoning Attacks	19
3.1	Motivation	20
3.2	Proposed Method	21
3.2.1	Partial-Sharing-Based Online FL (PSO-Fed)	22
3.2.2	Model-Poisoning Attack Model	24
3.3	Theoretical Results	25
3.3.1	Mean Convergence	26
3.3.2	Mean-Square Convergence	27
3.3.3	Steady-State Mean Square Error	28
3.3.4	Optimal Stepsize	29
3.4	Numerical Results	30
3.5	Summary	37
4	Resource-Efficient FL Robust to Communication Errors	39
4.1	Motivation	39

4.2	Proposed Method	41
4.2.1	Federated Weighted Least-Squares Regression	41
4.2.2	Dual Variable Elimination	43
4.2.3	Communication Noise	43
4.3	Resource-efficient FL over Noisy Channels	44
4.3.1	Random Scheduling	44
4.3.2	RERCE-Fed	44
4.3.3	RERCE-Fed with Continual Local Updates	44
4.4	Theoretical Results	47
4.4.1	Mean Convergence	48
4.4.2	Mean-Square Convergence	48
4.4.3	Steady-State Mean-Square Error	49
4.5	Numerical Results	50
4.5.1	Performance of RERCE-Fed	51
4.5.2	Performance of RERCE-Fed with Continual Local Updates	53
4.5.3	Comparisons of Theory and Experiment	55
4.6	Summary	57
5	Privacy-Preserving Nonnegative Matrix Factorization	59
5.1	Motivation	59
5.2	Proposed Method	61
5.2.1	Estimating the Left Factor	64
5.2.2	Estimating the Right Factor	64
5.2.3	Convergence Analysis	65
5.3	Privacy-Preserving Distributed NMF	65
5.3.1	Paillier Cryptosystem	65
5.3.2	Privacy Preservation	66

5.4	Numerical Results	68
5.5	Summary	71
6	Distributed Maximum Consensus over Noisy Links	73
6.1	Motivation	73
6.2	Background	74
6.3	Proposed Method	76
6.3.1	Distributed Maximum Consensus Algorithm	76
6.3.2	Noise-Robust Distributed Maximum Consensus Algorithm	78
6.4	Numerical Results	79
6.5	Summary	83
7	Conclusion and Future Work	85
7.1	Summary	85
7.2	Future Directions	86
	Bibliography	88
A	Publication 1	105
B	Publication 2	111
C	Publication 3	125
D	Publication 4	131
E	Publication 5	137
F	Publication 6	155
G	Publication 7	161

List of Tables

2.1	Comparison of fully-distributed learning and federated learning . . .	17
-----	---	----

List of Figures

1.1	Thesis contributions and organization diagram.	4
2.1	Conventional distributed learning structure.	8
2.2	Conventional federated learning structure.	11
2.3	Online federated learning.	14
2.4	Partial sharing of information.	15
3.1	PSO-Fed algorithm.	22
3.2	Model-poisoning attack model.	24
3.3	Steady-state test MSE for different algorithms with different numbers of Byzantine clients $ \mathcal{S}_B $, attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 1$	31
3.4	Steady-state test MSE of PSO-Fed for different numbers of shared elements M with different numbers of Byzantine clients $ \mathcal{S}_B $, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$	31
3.5	Network-wide average MSE of PSO-Fed for different values of attack strengths σ_B^2 and Byzantine clients $ \mathcal{S}_B $, number of shared elements $M = 1$ and attack probability $p_a = 0.2$	32
3.6	Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of Byzantine clients $ \mathcal{S}_B $, number of shared elements $M = 1$ and attack strength $\sigma_B^2 = 0.25$	33

3.7	Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of shared elements $M \in \{1, 5\}$, number of Byzantine clients $ \mathcal{S}_B = 5$ and attack strength $\sigma_B^2 = 0.25$	34
3.8	Network-wide average MSE of PSO-Fed for different values of stepsize μ , attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 0.25$	34
3.9	Network-wide average MSE of PSO-Fed for different values of stepsize μ , numbers of Byzantine clients $ \mathcal{S}_B = 5$ and attack probability $p_a = 0.25$	35
3.10	Effect of small stepsize approximation (SSA) on network-wide average MSE of PSO-Fed, numbers of Byzantine clients $ \mathcal{S}_B \in \{0, 10\}$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.25$	36
3.11	Effect of number of shared elements M on \mathcal{E}_ω and \mathcal{E}_ϕ in (3.37) for $ \mathcal{S}_B \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$	36
4.1	NMSE of (4.5)-(4.6) and (4.10) for $\mathcal{C} = K = 100$	51
4.2	NMSE of (4.5)-(4.6) with $\mathcal{C} = 4$ and (4.10) with $\mathcal{C} \in \{4, 75, 90\}$	52
4.3	NMSE of (4.10) with $\mathcal{C} = 4$ and RERCE-Fed (4.12) for different numbers of participating clients $\mathcal{C} \in \{4, 10, 25\}$	52
4.4	NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 4$ and different uplink and downlink noise variances $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$	53
4.5	NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 10$ and different uplink and downlink noise variances $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$	54
4.6	NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 25$ and different uplink and downlink noise variances $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$	54
4.7	Squared-norm of bias estimate of RERCE-Fed (4.12) with $K = 6$, $L = 6$, $\mathcal{C} = 3$, and $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 = 10^{-4}$ for different numbers of MC runs $\mathcal{M} \in \{10, 10^2, 10^3, 10^4, 10^5\}$	55
4.8	NMSE of RERCE-Fed (4.12) with $\mathcal{C} = 3$ for different uplink noise variances.	56

4.9	NMSE of RERCE-Fed (4.12) with $\mathcal{C} = 3$ for different downlink noise variances.	56
4.10	NMSE of RERCE-Fed (4.12) for different numbers of participating clients $\mathcal{C} \in \{2, 3, 4, 5, 6\}$ and different link noise variances. . .	57
5.1	Illustration of nonnegative matrix factorization.	60
5.2	The Paillier cryptosystem flow diagram.	65
5.3	The information exchange between agents k and $\ell \in \mathcal{N}_k$ in PPDNMF.	68
5.4	NMSE (5.15) of PPDNMF and centralized algorithm versus the BCD iteration index for different values of N_{\max} on synthetic data.	69
5.5	NMSE (5.15) of PPDNMF and centralized algorithm versus the BCD iteration index for different values of N_{\max} on MIT-CBCL database.	70
5.6	The original and reconstructed faces #1 and #2429 from the MIT-CBCL database.	70
6.1	An illustration of the naive maximum consensus (naive-MC) algorithm.	75
6.2	The considered network with an arbitrary topology and $K = 20$ agents.	80
6.3	The impact of noise on the performance of Naive-MC (6.2), D-MC algorithm (6.7) and RD-MC algorithm (6.9) with window size $\mathcal{C} = 3$ and noise variance $\sigma^2 = 0.1$	80
6.4	The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 1$ and different noise variances $\sigma^2 \in \{0.0001, 0.01, 0.1\}$	81
6.5	The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 2$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$	81
6.6	The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$	82

6.7	The considered network with linear topology and $K = 20$ agents.	82
6.8	The impact of network connectivity on the network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ in the presence of link noise with variance $\sigma^2 = 0.1$	83

List of Algorithms

1	Conventional Distributed Learning.	9
2	Conventional Federated Learning.	12
3	PSO-Fed.	23
4	RERCE-Fed.	45
5	RERCE-Fed with continual local updates.	46
6	PPDNMF.	67
7	RD-MC.	79

Abbreviations and Symbols

Abbreviations

ADMM	Alternating direction method of multipliers
BCD	Block-coordinate descent
CPS	Cyber-physical system
DL	Distributed learning
DMC	Distributed maximum consensus
DNMF	Distributed non-negative matrix factorization
FL	Federated learning
GD	Gradient descent
IID	Independent and identically distributed
IoT	Internet of Things
MC	Maximum consensus
MSD	Mean square deviation
MSE	Mean square error
NMF	Non-negative matrix factorization
Non-IID	Non-independent and identically distributed
Online-Fed	Online federated learning
P2P	Peer-to-peer communication
PP	Privacy-preserving
PS	Partial-sharing-based
PSO-Fed	Partial-sharing-based online federated learning
RFF	Random Fourier features
SGD	Stochastic gradient descent
SNR	Signal-to-noise ratio

Symbols

\mathcal{D}_k	Dataset of client k
\mathcal{S}_n	Set of clients at iteration n
\mathcal{S}_B	Set of Byzantine clients
\mathcal{E}	Mean square error
$\mathbf{w}_{k,n}$	Local model of client k at iteration n
$\tilde{\mathbf{w}}_{k,n}$	Local model of client k at iteration n after noise perturbation
\mathbf{w}_n	Global model of server at iteration n
$\tilde{\mathbf{w}}_n$	Global model of server at iteration n after noise perturbation
\mathbf{w}^*	The optimal solution
K	Number of agents (clients) in the network
L	Size of model parameter vector
M	Number of shared elements of model parameter vector
$\mathbf{S}_{k,n}$	Selection matrix for local model of client k at iteration n
\mathbf{I}	Identity matrix
\mathbf{O}	Zero matrix
$\mathbf{1}$	Vector of ones
$\mathbf{0}$	Vector of zeros
\mathcal{G}	Graph
\mathcal{N}	Set of agents (clients)
\mathcal{N}_k	Set of neighboring agents (clients) of agent (client) k
$\mathcal{J}_k(\cdot)$	Local loss function of agent (client) k
$\mathcal{L}(\cdot)$	Augmented Lagrangian function
$\mathbb{E}[\cdot]$	Statistical expectation operator
$\mathcal{N}(\cdot, \cdot)$	Gaussian distribution
$\mathcal{U}(\cdot, \cdot)$	Uniform distribution
$\ \cdot\ _F$	Frobenius norm of its matrix argument
$\ \cdot\ _2$	Euclidean norm of its vector argument
\otimes	Kronecker product
\otimes_b	Block Kronecker product
$(\cdot)^\top$	Transpose operator
$\text{tr}(\cdot)$	Trace operator
$\text{diag}\{\cdot\}$	Diagonalization operator
$\text{bdiag}\{\cdot\}$	Block diagonalization operator
$\text{col}\{\cdot\}$	Columnization operator
$\text{bcol}\{\cdot\}$	Block columnization operator
$\text{vec}\{\cdot\}$	Vectorization operator
$\text{bvec}\{\cdot\}$	Block vectorization operator
$\text{bvec}^{-1}\{\cdot\}$	Inverse block vectorization operator

Chapter 1

Introduction

1.1 Motivation and Scope

The proliferation of Internet of things (IoT) devices is reshaping our world into an interconnected network of smart objects. These devices, ranging from household appliances and wearable gadgets to industrial sensors and autonomous vehicles, are becoming increasingly common in both personal and commercial settings. This expansion is driven by reasons such as enhanced computing power, and widespread access to high-speed Internet. As more devices join this ocean of devices, they generate vast amounts of data, providing valuable insights into human behavior, environmental conditions, and industrial processes. The potential benefits of IoT technology are immense, spanning across various fields including healthcare, education, entertainment, social life, energy management, and smart cities [1–3]. However, this growth also presents significant challenges in energy efficiency, privacy, security, and data management [4, 5].

In a distributed IoT setting, devices are dispersed across various locations often operating with different communication and computational capabilities and limited energy resources. This gives new opportunities to gather and leverage data for a variety of applications. These devices can also learn from their collected data by extracting valuable information from the data. Distributed learning enables IoT systems to make data-driven decisions, generate insights, uncover patterns, detect anomalies, and make smart inferences that can be used to optimize various systems such as autonomous vehicles [6].

Distributed learning in IoT applications poses several technical challenges, e.g., communication efficiency, additive noise in the communication links, and chal-

lenges related to privacy preservation. Since IoT devices often operate with limited bandwidth and energy resources, communication efficiency is paramount for distributed learning applications. Transmitting large amounts of raw data from dispersed devices to their neighbors or a central server can quickly overwhelm the network. This necessitates the development of methods to transmit only the most relevant information. Additionally, the additive noise in the communication links causes unreliability in IoT systems. Hence, we require robust learning algorithms that are capable of mitigating the impact of noise-corrupted communication. Furthermore, the highly sensitive nature of data collected by IoT devices, ranging from personal health metrics to industrial process details, underscores the critical need for privacy-preserving distributed learning algorithms [7–10].

In addition to the challenges mentioned earlier, various types of adversaries can compromise system integrity, trustworthiness, and performance in distributed learning. The adversaries in a distributed setting can be categorized into external eavesdroppers and internal malicious agents. External eavesdroppers can intercept communications and internal adversaries can disrupt the network. Internal malicious agents can further be categorized into Byzantine agents and honest-but-curious agents. Byzantine agents may arbitrarily deviate from the rules of the network but honest-but-curious agents follow the rules but attempt to infer private information from the exchanged messages. Attacks in distributed networks can take multiple forms, including data-poisoning, model-poisoning attacks, eavesdropping, inference attacks and denial of service (DoS) attacks. This highlights the pressing need for algorithms that are capable of mitigating the impacts of adversarial attacks on a network [11–13].

This thesis aims to develop novel distributed and federated learning algorithms to solve real-world learning tasks that leverage signal processing techniques and encryption algorithms to address the critical challenges of resilience against adversarial attacks, communication efficiency, robustness against noise, and privacy preservation, along with minimizing computational load on devices.

1.2 Objectives

The objectives of this thesis can be summarized as follows:

- O1** To analyze the effect of sharing only a fraction of the model parameter on the resilience against model-poisoning attacks and assess the performance of the network under the strain of such attack scenarios.

- O2** To improve the communication-efficiency and noise-robustness of the network by developing distributed and federated algorithms that exhibit such properties without incurring additional computational and communication load on the network.
- O3** To propose a strategy that fits well into the distributed setting and ensures the privacy of participating agents in the presence of external and internal adversaries.

1.3 List of Publications

The author of this thesis conducted the following research studies in accordance with the research objectives described in section 1.2. The results contained in this thesis are published in seven papers. They include five conferences and two journals. Below is the list of the publications:

- P1** E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, “On the resilience of online federated learning to model poisoning attacks through partial sharing,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2024, pp. 9201–9205.
- P2** E. Lari, R. Arablouei, V. C. Gogineni, and S. Werner, “Resilience in Online Federated Learning: Mitigating Model-Poisoning Attacks via Partial Sharing,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 11, pp. 388–400, 2025.
- P3** E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, “Resource-efficient federated learning robust to communication errors,” in *Proc. IEEE Stat. Signal Process. Workshop*, 2023, pp. 265–269.
- P4** E. Lari, V. C. Gogineni, R. Arablouei, and S. Werner, “Continual local updates for federated learning with enhanced robustness to link noise,” in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2023, pp. 1199–1203.
- P5** E. Lari, R. Arablouei, V. C. Gogineni, and S. Werner, “Noise-Robust and Resource-Efficient ADMM-based Federated Learning for WLS Regression,” submitted to *Elsevier Signal Processing*.
- P6** E. Lari, R. Arablouei, and S. Werner, “Privacy-Preserving Distributed Nonnegative Matrix Factorization,” in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 1022–1026.
- P7** E. Lari, R. Arablouei, N. K. D. Venkategowda, and S. Werner, “Distributed Maximum Consensus over Noisy Links,” in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 2247–2251.

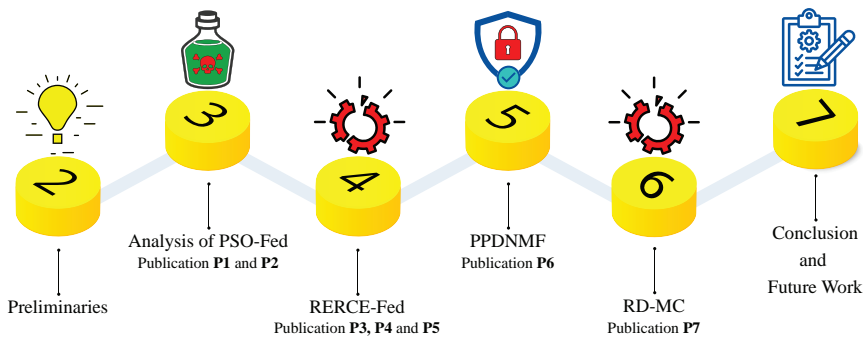


Figure 1.1: Thesis contributions and organization diagram.

1.4 Structure and Contributions

The rest of this thesis is organized as follows. Chapter 2 provides an overview of distributed and federated learning settings and some preliminary concepts relevant to solving distributed optimization problems within the remainder of this thesis. The main technical contributions of this thesis are presented across four chapters, with each chapter addressing one of the research objectives outlined in section 1.2 as follows:

- Chapter 3 analyzes the effect of partial sharing on the resilience of online FL (PSO-Fed) against model-poisoning attacks. It proposes an attack model that fits well into a federated setting and uncovers the existence of an optimal stepsize for PSO-Fed when a model-poisoning attack is present in the system.
- Chapter 4 proposes an ADMM-based FL algorithm to solve a weighted least square problem that is both noise-robust and communication-efficient. Furthermore, it allows the non-participating clients to continue their local updates to improve the overall performance.
- Chapter 5 develops a privacy-preserving distributed non-negative matrix factorization. It employs the Paillier cryptosystem to ensure the preservation of privacy and prevent sensitive information leakage.
- Chapter 6 designs a noise-robust distributed maximum consensus algorithm. It proposes to eliminate the dual variable and employ a moving average to mitigate the adverse effect of the additive noise in the communication links.

Finally, chapter 7 concludes the thesis and proposes future research directions. A diagram illustrating the thesis organization and contributions is given in Figure 1.1.

Chapter 2

From Distributed to Federated Learning over Networks

This chapter provides the background information required for the rest of the thesis. This chapter begins with an overview of the historical developments of learning algorithms, as detailed in section 2.1. Furthermore, section 2.1 motivates the transition from centralized learning frameworks to decentralized and distributed settings. In section 2.2, we state the fundamental principles of distributed learning, highlighting its abilities and challenges. In section 2.3, we introduce federated learning along with its benefits and challenges. In section 2.4, we introduce online federated learning, which is suitable for handling streaming data in a federated setup. In section 2.5, we introduce partial sharing of information as a method to lower the communication load in a distributed setup. We introduce the alternating direction method of multipliers, which is a first-order iterative algorithm to solve distributed optimization problems in section 2.6.

2.1 Background and Evolution

The evolution of learning algorithms has spanned several decades, marked by foundational breakthroughs and the emergence of new paradigms. Traditionally, machine learning relied on centralized frameworks, where all data was collected and processed on a server. This approach offered several benefits, such as efficient data processing and model training, and simple management and deployment. However, centralized learning presents significant challenges like data privacy and security, e.g., sensitive data must be transferred and stored centrally, which increases exposure to breaches and might violate privacy regulations. In addition, centralized learning suffers from communication load where large volumes of data

must be transferred to a central server, which can create bottlenecks and inefficiencies, especially with geographically distributed sources. Moreover, using a centralized setting might cause scalability limitations. In fact, scaling centralized systems requires increasingly powerful hardware, which can be costly and less flexible compared to distributed solutions [14].

To address these limitations, the field has shifted toward distributed and decentralized learning paradigms. Data and computation are spread across multiple nodes or devices. Each participant trains a model locally and sends updates (not raw data) to a central server for aggregation. This approach enhances scalability and computational efficiency, as workloads are shared [15].

A pioneering work in the field of distributed estimation was published in 1982 by Borkar and Varaiya [16]. In 1984, Tsitsiklis produced a significant piece of work for his PhD thesis focusing on decentralized and distributed decision-making and computation. He began by addressing the problem of whether a group of distributed but isolated decision-makers, each possessing different yet related information, can reach similar decisions without any communication. He then explored a scenario in which these decision-makers are allowed to communicate and share their decisions. In this setting, he demonstrated that they are guaranteed to reach a consensus. This conclusion was achieved by solving an optimization problem aimed at minimizing a global objective function [17]. Later in 1986, Tsitsiklis, Bertsekas, and Athans published another significant work that analyzed the convergence of various asynchronous distributed algorithms [18].

In 1989, Bertsekas and Tsitsiklis published a survey that introduced different types of distributed algorithms that can be solved via asynchronous updates. In their work, they showed that although a number of distributed algorithms fail to converge under an asynchronous setting, a fair number of algorithms still manage to converge under such a scenario [19]. In a separate work [20], Bertsekas and Tsitsiklis examined algorithms that are particularly amenable to parallelization. They also investigated the convergence properties of these algorithms, the rates of convergence, and the communication and synchronization challenges associated with their implementation.

Two decades later, the exploration of consensus and cooperation within distributed systems was undertaken in the 2000s by Olfati-Saber, Fax, and Murray, as published in their various works [21–25]. In [26], the authors studied the alignment problem that involved reaching a consensus in a distributed setting. These works laid the foundation for the theoretical framework for posing and solving consensus problems for distributed dynamic settings.

Another major and significant advancement in the field was marked by works on multi-agent adaptive networks by Sayed [27–29]. Multi-agent adaptive networks consist of collections of agents or nodes that interact and cooperate to address problems, such as distributed estimation, learning, and decision-making. These agents or nodes adapt their behaviors based on local data and information exchanged with their neighboring agents. Several distributed strategies can be employed to find solutions to these aforementioned problems. Three prominent strategies include incremental strategies [30–33], consensus strategies [34, 35], and diffusion strategies [36–38].

As an example, in [39], the authors studied distributed estimation algorithms based on diffusion protocols that enhance cooperation among adaptive nodes, where each node possesses local learning capabilities to generate estimates for a specific parameter, sharing information solely with neighboring nodes, thereby forming peer-to-peer protocols. This distributed and cooperative algorithm can respond to changes in the environment and demonstrates improved performance in transient and steady-state mean-square error compared to traditional non-cooperative methods.

The research on adaptive networks has significantly influenced the field of distributed learning. Specifically, it demonstrates that, under certain conditions, distributed strategies can achieve performance similar to that of centralized solutions, even in sparsely connected networks [40, 41]. Moreover, analytical studies have revealed how factors such as network topology, combination policies, and algorithm parameters affect the transient and steady-state performance behavior of adaptive networks, e.g., the convergence rate and steady-state error [40, 41].

In addition to various works on multi-agent adaptive networks, the topic of adaptive filters was studied in [42–46], which significantly influenced the field of distributed learning. One notable contribution is the energy conservation approach, which offers a robust analytical framework for both the understanding and design of adaptive filters [47]. This methodology effectively addresses the stochastic challenges that are characteristic of adaptive algorithms, facilitating a more straightforward analysis of their behavior and stability. We later utilize the energy conservation framework to analyze our algorithms.

In the following sections, we will present an overview of distributed and federated learning. This overview will establish a foundational basis for the subsequent chapters of this thesis.

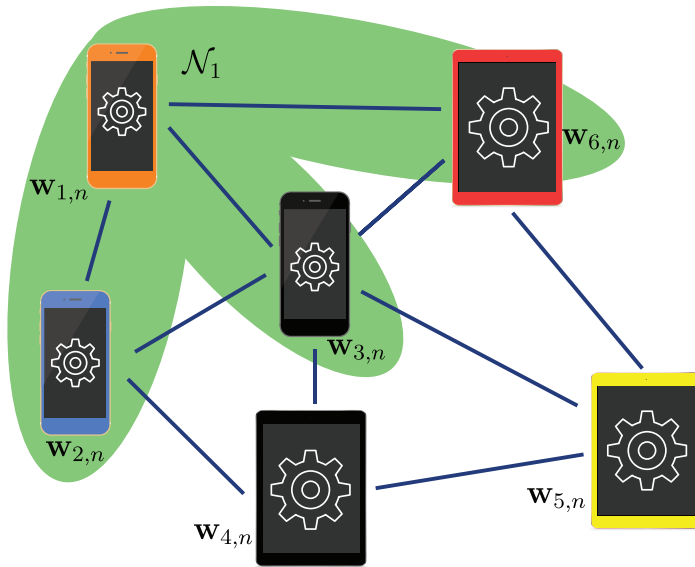


Figure 2.1: Conventional distributed learning structure.

2.2 Foundations of Distributed Learning

Distributed learning (DL) refers to a machine learning (ML) framework where the training process is distributed across multiple agents. It distinguishes itself from traditional centralized ML structures by allowing multiple agents to collaboratively train an ML model without the need for a central server or authority to orchestrate the process. In distributed learning, each participating agent k at iteration n maintains its own local dataset and performs local computations, sharing only its local trained model $w_{k,n}$ with its neighboring agents $\ell \in \mathcal{N}_k$ and contributing to the global learning objective.

In contrast to centralized approaches where data is located in a single place, fully-distributed learning allows data to remain distributed across multiple agents, where these agents themselves can be diverse entities such as individual devices, different organizations, or even distributed among various geographical locations.

The learning process typically involves iterative rounds of local computations followed by peer-to-peer communication among neighboring agents. A conventional distributed learning structure is illustrated in Figure 2.1 and summarized in the form of a pseudo-code in Algorithm 1. We can observe from Figure 2.1 that peer-to-peer communication enables direct interaction with neighboring agents without the need for any intermediary server. This figure also depicts the set of neighbors

Algorithm 1: Conventional Distributed Learning.

- 1 **Parameters:** penalty parameter ρ , set of agents \mathcal{N}
 - 2 **Initialization:** local models $\mathbf{w}_{k,0}$
 - 3 **For** $n = 1, \dots$, *Until Convergence*
 - 4 **Local Update at Agent k :**
 - 5 Receive $\mathbf{w}_{\ell,n}$ from the neighbors $\ell \in \mathcal{N}_k$.
 - 6 Update the local model.
 - 7 Send $\mathbf{w}_{k,n+1}$ to the neighbors $\ell \in \mathcal{N}_k$.
 - 8 **EndFor**
-

of agent 1 that consists of $\mathcal{N}_1 \in \{2, 3, 6\}$. This communication model offers several advantages and exhibits some challenges and limitations. We delve into these topics in the following.

2.2.1 Advantages of Distributed Learning

DL structure exhibits several key advantages, including:

- Ability to handle large datasets: Distributed learning can process datasets that are too large for a single agent. The distribution of data is handled via partitioning a dataset across multiple agents [48]. In addition, data can be distributed in nature, e.g., data from distributed sensor networks.
- Improved processing speed through parallel computation: Depending of factors such as application and topology, distributing a dataset across multiple agents, can reduce the training time significantly [48].
- Fault tolerance and system reliability: A distributed system can continue functioning even if some agents or communication links between agents fail or become unavailable. This redundancy ensures that the learning process is not halted due to faulty agents [49].

2.2.2 Challenges and Limitations of Distributed Learning

DL faces some challenges and limitations, including [50]:

- Model update sharing among agents: A key part of the learning process in a distributed structure is the model exchange between agents. This can create an additional communication load on the agents.

- **Privacy concerns:** Although the goal of distributed learning is to omit the need to transfer raw data, it is not immune to privacy leakage concerns. The exchange of model updates can potentially reveal sensitive information about the underlying data through various inference attacks. To mitigate these risks, privacy-preserving techniques such as differential privacy [51] and homomorphic encryption [52] have been proposed in the literature. These methods aim to add noise to shared information or perform computations on encrypted data. However, implementing these techniques often involves a trade-off between privacy guarantees and model performance.
- **Dealing with stragglers and asynchronous updates:** Stragglers can be defined as agents that are significantly slower in completing their local computations or communicating updates [53, 54]. These types of agents can pose challenges in the form of slowing down the overall learning process. To address these challenges, several strategies have been developed. One approach is to ignore delayed updates from stragglers. Another solution is to perform asynchronous updates, where we allow the agents to perform updates at their own pace without waiting for slower participants. However, these methods introduce challenges in terms of algorithm convergence.

2.3 Federated Learning: A Shifting Frontier

Federated Learning (FL) represents a distinctive approach to distributed learning. The concept of FL was first introduced by Google researchers in 2016, notably in [55] by McMahan et al. This foundational study coined the term "Federated Learning" and described a system where numerous clients, such as mobile devices, collaboratively train a machine learning model under the coordination of a central server, all while keeping the training data decentralized and private on devices. The initial goal was to facilitate privacy-preserving machine learning, especially for applications like Google Keyboard (Gboard), where user data remains on the device, and only model updates are shared, rather than raw data [56]. FL is characterized by employing a central server to coordinate the collaborative learning process. Unlike fully-distributed learning, FL maintains a structure where a server orchestrates the training across multiple edge devices, called clients, within a network.

The FL literature frequently employs the FedAvg algorithm [55] as a standard benchmark. FedAvg begins with the server broadcasting its aggregated global model to a randomly chosen subset of clients $k \in \mathcal{S}_n$ during iteration n , typically over a wireless network. These clients then perform local training to refine their local models $\mathbf{w}_{k,n+1}$ before sending the updates back to the server. The

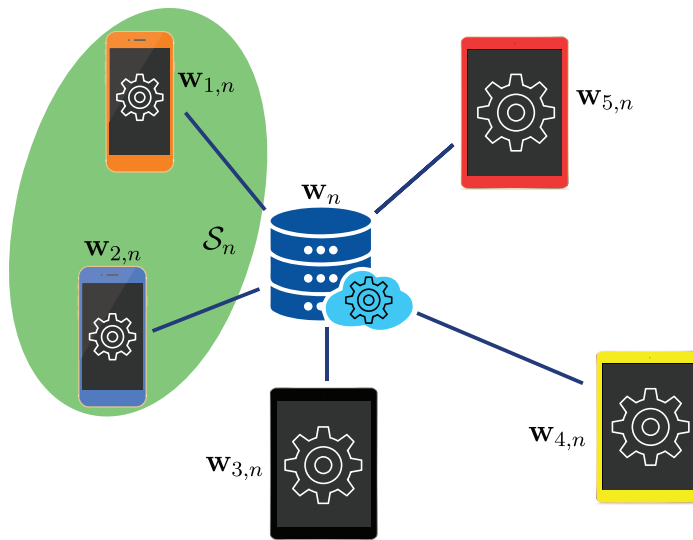


Figure 2.2: Conventional federated learning structure.

server aggregates these local models into a new global model, often by employing a weighted average, and repeats this process until a specific convergence criterion is met. The FedAvg algorithm is summarized in the form of a pseudo-code in Algorithm 2. A conventional FL setting is illustrated in Figure 2.2. We can observe from Figure 2.2 that the only connections in a federated network are between the clients and the central server. In addition, we can observe that in this figure, the set of clients that are communicating to the server is $\mathcal{S}_n \in \{1, 2\}$.

FL can be categorized into two types based on the participating entities and the scale of training: cross-silo FL and cross-device FL [57]. In cross-silo FL, a small number of trusted participants, typically organizations or institutions such as hospitals, banks, and research laboratories, engage in the training process. Each silo has access to a large local dataset and possesses strong computational and networking resources. The environment is stable, characterized by high client availability and reliable connections. This type of FL is particularly useful in scenarios where data privacy, governance, and regulatory compliance are essential, such as in healthcare and finance. On the other hand, cross-device FL involves a vast number of participants, including smartphones, IoT devices, and edge sensors. In this case, each device contributes a small amount of data and has limited computational power and intermittent connectivity. The environment is dynamic, with devices frequently joining and leaving the network. This approach emphasizes efficient communication, privacy (such as through differential privacy), and resilience to

Algorithm 2: Conventional Federated Learning.

- 1 **Parameters:** penalty parameter ρ , number of clients K , set of clients \mathcal{S}
 - 2 **Initialization:** global model \mathbf{w}_0 , local models $\mathbf{w}_{k,0}$
 - 3 **For** $n = 1, \dots$, *Until Convergence*
 - 4 The server randomly selects a subset \mathcal{S}_n of its clients and sends the aggregated global model \mathbf{w}_n to them.
 - 5 **Local Update at Client k :**
 - 6 **If** $k \in \mathcal{S}_n$
 - 7 Receive \mathbf{w}_n from the server.
 - 8 Update the local model.
 - 9 Send $\mathbf{w}_{k,n+1}$ to the server.
 - 10 **EndIf**
 - 11 **Aggregation at the Server:**
 - 12 The server receives $\mathbf{w}_{k,n+1}$ from $k \in \mathcal{S}_n$ and aggregates them via
 - 13
$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} \mathbf{w}_{k,n+1}.$$
 - 14 **EndFor**
-

unreliable devices [58].

2.3.1 Advantages of Federated Learning

FL exhibits several key advantages, such as [59, 60]:

- **Enhanced privacy and security:** FL keeps raw data on client devices, eliminating the need for centralized data storage and reducing the risk of data breaches. This facilitates security in sensitive sectors such as healthcare and finance. Additionally, differential privacy can be incorporated into the FL framework, adding noise to model updates before sharing them to provide guarantees against the extraction of individual data points.
- **Reduced communication compared to DL:** In FL, only model updates are transmitted between clients and the central server. Moreover, FL can also employ various communication-efficient techniques to further reduce the communication load.
- **Leveraging computational power of clients:** FL performs model training locally on clients and harnesses the computational power of these clients. In this way, FL reduces the computational load on the central server, which aggregating

updates from clients is its primary task.

2.3.2 Challenges of Federated Learning

In addition to challenged mentioned for DL, FL faces some other challenges, such as [60, 61]:

- **Data heterogeneity:** The heterogeneity in data arises from the non-independent and identically distributed (non-IID) nature of data or imbalances among client datasets used in training the global model. Diversity in data distribution across participants can significantly impact model convergence and performance.
- **System heterogeneity:** Device heterogeneity refers to the disparities in storage capacity, energy resources, computational power, and communication capabilities among participating clients. Variations in device characteristics can affect the efficiency of the FL process.
- **Personalization:** There is a need to balance the global model performance with local adaptations for different data distributions and various clients. In essence, there is a trade-off between a generally applicable model and one that performs well for individual clients.

2.4 Online Federated Learning

In an IoT network, where devices continuously generate or sense data, real-time processing and learning can be essential. However, conventional FL is typically batch-oriented, meaning that it relies on predefined data collections that are processed at certain intervals. While this approach works well for scenarios with static data, it struggles in dynamic environments where new data constantly arrives and the relevance of older data diminishes. Consequently, developing online algorithms is essential, as real-time data processing is often crucial in real-world dynamic settings. Therefore, Online FL (Online-Fed) addresses these limitations by enabling continuous learning from streaming data [62, 63].

In Online-Fed, clients receive data streams and perform local training at each iteration using only the most recent data. After updating their local models, clients share their models with the central server, which then aggregates the local models to form a new global model. This global model is then being broadcasted to all clients by the server and replaces their previous local models. To improve communication efficiency in an Online FL setup, the server is allowed to randomly select only a subset of the clients in every iteration. During a global iteration n , the set of randomly chosen client indices is denoted by \mathcal{S}_n . All clients can be selected



Figure 2.3: Online federated learning.

with the probability $p_c = \frac{c}{K}$, where $c = |\mathcal{S}_n|$ denotes the cardinality of the set \mathcal{S}_n . Subsequently, the server shares the global model \mathbf{w}_n with the selected clients $k \in \mathcal{S}_n$. The selected clients then update their local models $\mathbf{w}_{k,n}$. The procedure of Online-Fed is illustrated in Figure 2.3. From Figure 2.3, it can be observed that during iteration n , the set of randomly selected clients is $\mathcal{S}_n = \{1, 3, 4\}$ and the rest of the clients are not communicating with the server. Furthermore, data continuously streams at each client over time. It is essential to note that local training utilizing the most recent data continues regardless of the server's selection of the corresponding client. The selection process solely influences the communication aspect of the learning procedure. This reflects in Figure 2.3 by the fact the clients $\{2, 5, 6\}$ still receive data.

2.5 Partial Sharing of Information

To further improve communication efficiency, one can share only a part of the model instead of sharing it entirely. This approach is particularly beneficial when the model is large and sharing the full model can be costly. However, the drawback of partial sharing is slower convergence and an increased error compared to the scenario where the full model is shared [64].

To share only a fraction of the model vector, in every global iteration n , an $L \times L$ diagonal selection matrix $\mathbf{S}_{k,n}$ specifies the model parameters that will be exchanged between clients and the server. The main diagonal of $\mathbf{S}_{k,n}$ has D ones and $L - D$ zeros. The positions of ones in $\mathbf{S}_{k,n}$ specify which parameters to be exchanged. The D model parameters can either be selected in a stochastic manner or sequentially. Figure 2.4 illustrates how partial sharing works. It is observed from Figure 2.4 that, in iteration n , only the first element of the model parameter vector of

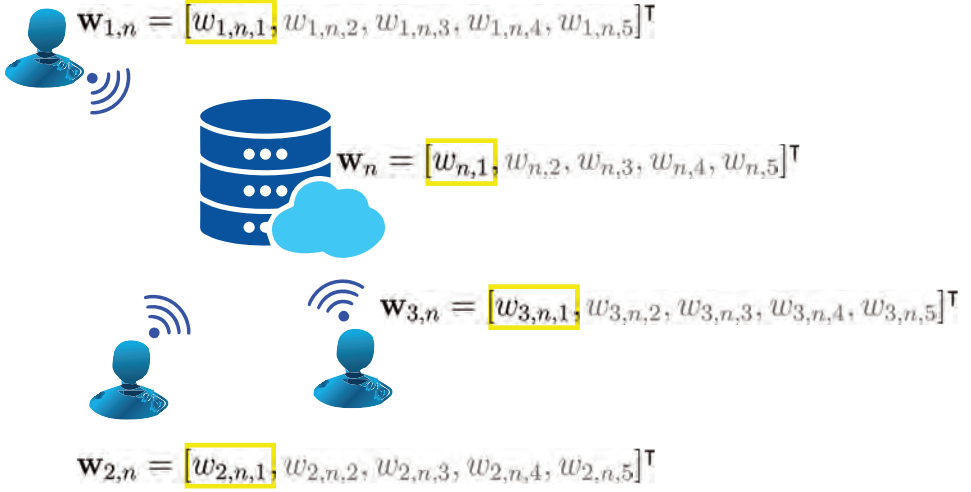


Figure 2.4: Partial sharing of information.

size $L = 5$ is shared. Hence, the selection matrix $\mathbf{S}_{k,n}$ can be written as

$$\mathbf{S}_{k,n} = \text{diag} \{[1, 0, 0, 0, 0]\}. \quad (2.1)$$

The rest of the vector comes from the clients or the server themselves. In other words, clients only share certain parts of their models, and the server does not have access to the unshared parameters. As a result, when aggregating parameters, the server substitutes the unshared parameters with the previously aggregated values. The same process applies to local aggregation. It is important to note that this approach leads to approximations that can impact performance, as analyzed for example in [64].

Therefore, the global model update \mathbf{w}_{n+1} can be written as

$$\begin{aligned} \mathbf{w}_{n+1} &= \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} \mathbf{w}_{k,n+1} \\ &= \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} \mathbf{S}_{k,n+1} \mathbf{w}_{k,n+1} + (\mathbf{I} - \mathbf{S}_{k,n+1}) \mathbf{w}_n. \end{aligned} \quad (2.2)$$

As we can observe from (2.2), in the partial sharing process, the $(\mathbf{I} - \mathbf{S}_{k,n+1}) \mathbf{w}_{k,n}$ portion is substituted with $(\mathbf{I} - \mathbf{S}_{k,n+1}) \mathbf{w}_n$ for aggregation at the server. Similarly, to perform the local model update, the same procedure happens at the client's side.

2.6 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) originated in the mid-1970s when it was introduced by Glowinski, Marrocco, Gabay, and Mercier as an extension of the method of multipliers developed in the late 1960s [65, 66]. ADMM was created to overcome the limitations of earlier augmented Lagrangian methods, particularly in addressing large-scale and structured optimization problems where variables can be updated independently.

While the fundamental concepts behind ADMM trace back even further to the 1950s, its modern form gained popularity in the 1980s and 1990s as theoretical foundations and convergence properties were established. The algorithm remained relatively unexplored until the 2010s, when advancements in distributed computing and the emergence of large-scale data challenges in fields such as machine learning, signal processing, and image reconstruction showcased ADMM's strengths in breaking down complex problems and enabling parallel computation. Nowadays, ADMM is often employed to solve large-scale optimization problems in a distributed manner [48].

ADMM addresses optimization problems of the form

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \\ \text{s. t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} = \mathbf{0}, \end{aligned} \quad (2.3)$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{p \times n}$, $\mathbf{B} \in \mathbb{R}^{p \times m}$, $\mathbf{c} \in \mathbb{R}^p$, and f and g are convex functions [48].

To solve this problem, ADMM employs the augmented Lagrangian of (2.3) that can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{u}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{u}^\top (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) \\ + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2, \end{aligned} \quad (2.4)$$

where \mathbf{u} denotes the vector of dual variables or Lagrange multipliers, and $\rho > 0$ is the penalty parameter.

ADMM solves the optimization problem (2.3) iteratively by minimizing the augmented Lagrangian (2.4) for \mathbf{x} and \mathbf{z} and a dual variable update alternatively as

$$\mathbf{x}_{n+1} = \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{z}_n, \mathbf{u}_n) \quad (2.5a)$$

$$\mathbf{z}_{n+1} = \min_{\mathbf{z}} \mathcal{L}(\mathbf{x}_{n+1}, \mathbf{z}, \mathbf{u}_n) \quad (2.5b)$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \rho(\mathbf{A}\mathbf{x}_{n+1} + \mathbf{B}\mathbf{z}_{n+1} - \mathbf{c}), \quad (2.5c)$$

Table 2.1: Comparison of fully-distributed learning and federated learning

Characteristic	Fully-distributed learning	Federated learning
Architecture	Multiple agents without a central server	Central server coordinating with multiple clients
Data Distribution	Data remains on individual agents	Data remains on clients
Communication	Agents communicate only with their neighbors	Clients communicate only with the central server
Scalability	Scalable	Scalable
Privacy	Prone to internal and external adversarial attacks	Prone to internal and external adversarial attacks
Fault Tolerance	Robust to individual node failures	Single point of failure (central server)
Coordination	Might require consensus algorithms	Simpler coordination through central server
Model Aggregation	Local aggregation	Global aggregation at the central server
Application	Ad-hoc networks	Mobile devices

where the subscript n denotes the iteration number for ADMM.

One of the key advantages of ADMM is its ability to decompose large problems into subproblems that can be solved in a parallel and distributed manner. This makes ADMM a suitable candidate for solving distributed optimization problems in this thesis.

2.7 Summary

This chapter presented background information to help comprehend the remainder of the thesis and centralize the definition of concepts used in the following chapters. A comparison between different characteristics of fully-distributed and federated learning is presented in Table 2.1. In chapter 3, the effect of partial sharing on the resilience of online FL against model-poisoning attacks is analyzed. Chapter 4 develops a noise-robust and communication-efficient ADMM-based FL. In chapter 5, a privacy-preserving distributed non-negative matrix factorization is proposed. Finally, chapter 6 proposes a noise-robust distributed maximum consensus algorithm.

Chapter 3

Resilience of Partial Sharing Online FL to Model Poisoning Attacks

This chapter, which summarizes the results of publication **P1** and its extension **P2**, proposes a thorough analysis of the impact of partial sharing on the resilience of online federated learning algorithm (PSO-Fed) against model-poisoning attacks. The PSO-Fed algorithm aims to alleviate the communication burden by enabling clients to exchange only partial model estimates with the server during each update round. This approach not only reduces communication load but also fortifies the algorithm against model-poisoning attacks. To delve deeper into this phenomenon, our analysis focuses on evaluating the performance of the PSO-Fed algorithm in the presence of Byzantine clients, who may maliciously manipulate their local models by introducing noise before sharing them with the server. Our study demonstrates that PSO-Fed sustains convergence in both mean and mean-square senses even under the adversarial influence of model-poisoning attacks. Additionally, our work entails deriving the theoretical mean square error (MSE) of PSO-Fed, establishing its correlation with various parameters including stepsize, attack probability, number of Byzantine clients, client participation rate, partial-sharing ratio, and noise variance. Furthermore, we ascertain the existence of an optimal stepsize for PSO-Fed when encountering model-poisoning attacks. The outcomes of our comprehensive numerical experiments substantiate our theoretical affirmations and underscore the superior resilience of PSO-Fed in mitigating Byzantine attacks, surpassing other leading related algorithms.

3.1 Motivation

Byzantine clients in network environments are those that behave unpredictably or with malicious intent. They have the ability to disrupt normal operations by sending misleading information or not adhering to established protocols. This type of behavior presents significant risks to the FL process and could potentially result in disruption or reduced performance [67, 68]. The presence of Byzantine clients can compromise the integrity and reliability of the global model in FL. They might employ deceptive tactics, such as submitting false or inconsistent gradients or model weights during updates or even initiating denial-of-service (DoS) attacks, all of which have the potential to undermine the integrity and effectiveness of the FL process [49, 69, 70]. Various strategies have been suggested to address the impact of Byzantine clients in FL. These strategies include using resilient aggregate statistics, assigning trust scores to individual clients, and utilizing historical information to recover from Byzantine attacks [71–74].

Byzantine clients can launch various types of adversarial attacks such as data-poisoning, model-poisoning, and label-poisoning [75, 76]. Data-poisoning attacks involve the injection of malicious data into the training dataset [77, 78]. Evasion attacks involve manipulating input data to undermine the generalizability of the learned model while evading detection [79]. Backdoor attacks involve embedding hidden trigger mechanisms within the model that can cause targeted malicious behavior [80]. Inference-time attacks, like membership inference, seek to reveal if a particular data point was utilized during training, which raises concerns about privacy [81, 82].

Network infrastructure can also be the target of distributed denial-of-service (DoS) attacks and man-in-the-middle attacks, which can disrupt services or intercept communications [83]. In addition to these types of attacks, model-poisoning attacks where Byzantine clients manipulate model updates during training as well as defense strategies against model-poisoning attacks in FL are discussed in [13]. These defense strategies can be categorized into evaluation methods for local model updates and aggregation methods for the global model. Evaluation methods involve the server analyzing client submissions without accessing local client data, which creates difficulties for their practicality. In aggregation methods, the server modifies the local model updates using specific criteria or employs appropriate statistical techniques. Utilizing robust aggregation methods like median or trimmed mean can reduce the impact of malicious clients on the global model [84]. Using methods of aggregation assisted by blockchain can also assist in coping with attacks on the network, although this may result in increased communication load or computational complexity [85].

A communication-efficient FL algorithm based on SignSGD was recently introduced in [86] that exhibits resilience to Byzantine attacks. However, it assumes that all clients consistently participate in every FL iteration, which may not reflect the practical realities of FL. This is especially true in scenarios involving clients with limited resources, such as mobile or IoT devices, in terms of energy or computational capacity. As a result, there is a significant gap in current FL research to address the necessity for communication-efficient FL algorithms that are robust against model-poisoning attacks without imposing additional computational burden on clients. Filling this gap can make FL more feasible and practical for a wider range of devices, particularly those with constrained resources, and can improve the robustness and scalability of FL systems in real-world applications.

This chapter delves into the analysis of the performance of the partial-sharing-based online FL (PSO-Fed) algorithm [63, 87, 88] in the face of Byzantine attacks. Our investigation shows that employing partial sharing in online FL, with the main goal of improving communication efficiency, also provides the additional benefit of reducing the negative impact of model-poisoning attacks. Our rigorous theoretical analysis demonstrates that PSO-Fed converges in both mean and mean-square senses, even when subjected to model-poisoning attacks. Additionally, we calculate the theoretical steady-state mean square error (MSE) of PSO-Fed. Our analysis reveals an important discovery: the identification of an optimal stepsize for PSO-Fed in the presence of model-poisoning attacks. Unlike gradient descent algorithms, where a smaller stepsize often leads to better performance, in PSO-Fed under model-poisoning attacks, a smaller stepsize does not necessarily result in improved estimation accuracy. This contrasts with the typical behavior of gradient-descent-based algorithms, where smaller stepsizes are frequently used to improve performance, albeit at the expense of slower convergence. We present a theoretical determination of the optimal stepsize. To validate our theoretical findings, we carry out extensive numerical experiments using synthetic non-IID data. The experimental results validate our theoretical predictions and demonstrate that PSO-Fed displays greater resilience to Byzantine attacks compared to other algorithms, such as Online-FL, all without incurring additional computational overhead on clients.

3.2 Proposed Method

We consider a federated network consisting of K clients and one server. At every time instance n , each client k has access to a data vector $\mathbf{x}_{k,n} \in \mathbb{R}^L$ and its corresponding response value $y_{k,n} \in \mathbb{R}$, which are related via the model

$$y_{k,n} = \mathbf{w}^\top \mathbf{x}_{k,n} + \nu_{k,n}, \quad (3.1)$$

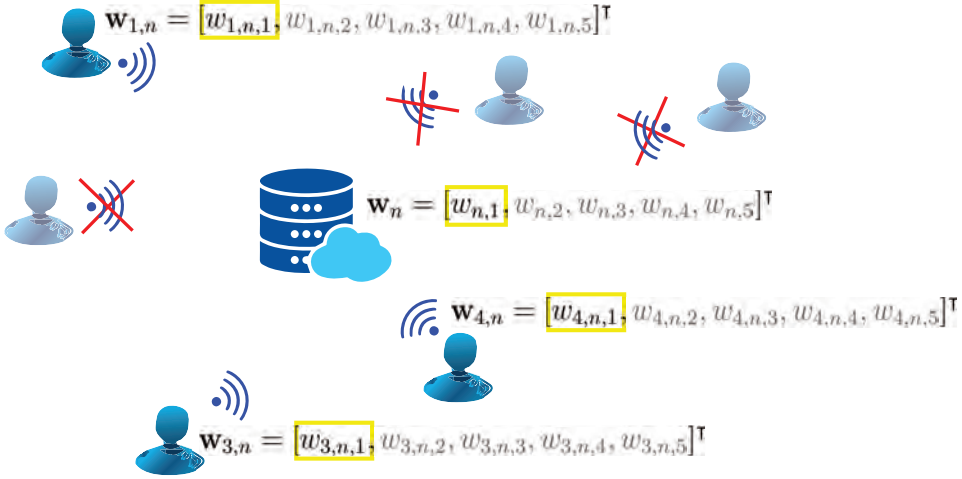


Figure 3.1: PSO-Fed algorithm.

where the model parameter vector $\mathbf{w} \in \mathbb{R}^L$ is collaboratively estimated using the locally stored client data, and $\nu_{k,n}$ is the observation noise. We define the global objective function for estimating \mathbf{w} as

$$\mathcal{J}(\mathbf{w}) = \frac{1}{K} \sum_{k=1}^K \mathcal{J}_k(\mathbf{w}) \quad (3.2)$$

with the local objective function at client k being

$$\mathcal{J}_k(\mathbf{w}) = \mathbb{E} [|y_{k,n} - \mathbf{w}^\top \mathbf{x}_{k,n}|^2]. \quad (3.3)$$

The goal is to find the optimal estimate of \mathbf{w} by minimizing $\mathcal{J}(\mathbf{w})$, i.e., $\mathbf{w}^* = \min_{\mathbf{w}} \mathcal{J}(\mathbf{w})$, albeit, in a distributed fashion via FL.

3.2.1 Partial-Sharing-Based Online FL (PSO-Fed)

In PSO-Fed, the server only sends a portion of the global model estimate to the clients as discussed in chapter 2.5 to reduce the communication load. In addition, the clients also transmit only a part of their local model estimates to the server. The model parameters exchanged between client k and the server at iteration n are specified using a diagonal selection matrix denoted by $\mathbf{S}_{k,n} \in \mathbb{R}^{L \times L}$ with M ones and $L - M$ zeros. The positions of the ones on the diagonal determine which model parameters are shared with the server at each iteration. They can be selected arbitrarily or in a round-robin fashion as in [64, 89] such that the model parameters are exchanged between each client and the server, on average, M times in every L iterations. Hence, the probability of each of the model parameters being shared

Algorithm 3: PSO-Fed.

-
- 1 **Parameters:** stepsize μ , number of clients K , set of clients \mathcal{S}
 - 2 **Initialization:** global model $\mathbf{w}_0 = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \mathbf{0}$
 - 3 **For** $n = 1, \dots$, *Until Convergence*
 - 4 The server randomly selects a subset \mathcal{S}_n of its clients and sends the global model $\mathbf{S}_{k,n} \mathbf{w}_n$ to them.
 - 5 **Client Local Update:**
 - 6 **If** $k \in \mathcal{S}_n$
 - 7 Update $\epsilon_{k,n} = y_{k,n} - [\mathbf{S}_{k,n} \mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n}) \mathbf{w}_{k,n}]^\top \mathbf{x}_{k,n}$.
 - 8 Update $\mathbf{w}_{k,n+1} = \mathbf{S}_{k,n} \mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n}) \mathbf{w}_{k,n} + \mu \mathbf{x}_{k,n} \epsilon_{k,n}$
 - 9 Send $\mathbf{S}_{k,n} \mathbf{w}_{k,n+1}$ to the server.
 - 10 **Else**
 - 11 Update $\epsilon_{k,n} = y_{k,n} - \mathbf{w}_{k,n}^\top \mathbf{x}_{k,n}$.
 - 12 Update $\mathbf{w}_{k,n+1} = \mathbf{w}_{k,n} + \mu \mathbf{x}_{k,n} \epsilon_{k,n}$
 - 13 **EndIf**
 - 14 **Aggregation at the Server:**
 - 15 The server receives $\mathbf{S}_{k,n} \mathbf{w}_{k,n+1}$ from $k \in \mathcal{S}_n$ and aggregates them via
 - 16
$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} [\mathbf{S}_{k,n+1} \mathbf{w}_{k,n+1} + (\mathbf{I}_L - \mathbf{S}_{k,n+1}) \mathbf{w}_n]$$
 - 17 **EndFor**
-

with the server in any iteration is denoted by $p_e = \frac{M}{L}$. Figure 3.1 illustrates how PSO-Fed operates. From Figure 3.1, it can be observed that during iteration n , the set of randomly selected clients is $\mathcal{S}_n = \{1, 3, 4\}$ and the rest of the clients are not communicating with the server. In addition, it is observed that only the first element of the model parameter vector of size $L = 5$ is shared.

Using the selection matrices $\mathbf{S}_{k,n}$, the recursive update equations of PSO-Fed, that iteratively minimize (3.2) through FL, are expressed as

$$\epsilon_{k,n} = y_{k,n} - [\mathbf{S}_{k,n} \mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n}) \mathbf{w}_{k,n}]^\top \mathbf{x}_{k,n} \quad (3.4a)$$

$$\mathbf{w}_{k,n+1} = \mathbf{S}_{k,n} \mathbf{w}_n + (\mathbf{I}_L - \mathbf{S}_{k,n}) \mathbf{w}_{k,n} + \mu \mathbf{x}_{k,n} \epsilon_{k,n} \quad (3.4b)$$

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} [\mathbf{S}_{k,n+1} \mathbf{w}_{k,n+1} + (\mathbf{I}_L - \mathbf{S}_{k,n+1}) \mathbf{w}_n], \quad (3.4c)$$

where $\mathbf{w}_{k,n}$ is the local model estimate at client k and iteration n , \mathbf{w}_n is the global model estimate at iteration n , \mathbf{I}_L is the $L \times L$ identity matrix, μ is the stepsize

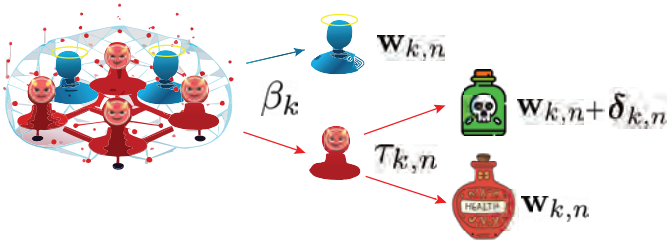


Figure 3.2: Model-poisoning attack model.

controlling the convergence rate and steady-state performance, \mathcal{S}_n denotes the set of client selected at iteration n , and $|\mathcal{S}_n|$ denotes the number of the selected clients in each iteration. Algorithm 3 summarizes PSO-Fed [63].

3.2.2 Model-Poisoning Attack Model

We denote the group of potential Byzantine clients within the network as \mathcal{S}_B . To indicate whether a client exhibits Byzantine behavior, we utilize the indicator variable β_k , where $\beta_k = 1$ signifies that client k is part of the set \mathcal{S}_B (i.e., is a Byzantine client), and $\beta_k = 0$ indicates otherwise. The total count of Byzantine clients in the network is represented by $|\mathcal{S}_B|$. These Byzantine clients aim to compromise the global model (e.g., diminish its accuracy) by intermittently transmitting corrupted local model estimates to the server. We assume that the Byzantine clients have access to accurate local model estimates. More specifically, we consider a situation in which, during each iteration, each Byzantine client intentionally alters its local model estimate by perturbing it before sending it to the server. This alteration is performed with a certain probability of attack, denoted as p_a . We represent the model update shared by each of the Byzantine clients via [12]

$$\mathbf{w}'_{k,n} = \begin{cases} \mathbf{w}_{k,n} + \boldsymbol{\delta}_{k,n} & \text{with probability } p_a \\ \mathbf{w}_{k,n} & \text{with probability } 1 - p_a, \end{cases} \quad (3.5)$$

where $\boldsymbol{\delta}_{k,n} \in \mathbb{R}^L$ denotes the perturbation signal associated with the attack which can be modeled as a zero-mean white Gaussian noise, i.e., $\boldsymbol{\delta}_{k,n} \sim \mathcal{N}(\mathbf{0}, \sigma_B^2 \mathbf{I}_L)$.

A Bernoulli random variable $\tau_{k,n}$ represents the probability of a Byzantine client compromising its local estimate. This variable equals 1 with a probability of p_a , which signifies the occurrence of an attack, and 0 otherwise. This representation is depicted in Figure 3.2. We can observe from Figure 3.2 that clients are categorized into Byzantine and non-Byzantine sets, differentiated by the parameter β_k . In addition, the probability of a Byzantine client executing an attack is represented by the random variable $\tau_{k,n}$ in Figure 3.2.

Therefore, considering that each Byzantine client sends a distorted local model estimate to the server with a probability of p_a when chosen by the server, in situations where Byzantine clients exist in the network, the global model update of PSO-Fed, as shown in (3.4c), can be rewritten as

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{S}_n|} \sum_{k \in \mathcal{S}_n} [\mathbf{S}_{k,n+1} \mathbf{w}'_{k,n+1} + (\mathbf{I}_L - \mathbf{S}_{k,n+1}) \mathbf{w}_n], \quad (3.6)$$

where

$$\mathbf{w}'_{k,n+1} = \mathbf{w}_{k,n+1} + \beta_k \tau_{k,n} \boldsymbol{\delta}_{k,n+1}. \quad (3.7)$$

3.3 Theoretical Results

In this section, we evaluate the robustness of PSO-Fed against Byzantine attacks by analyzing its theoretical mean and mean-square convergence and predicting its steady-state MSE and optimal stepsize.

To facilitate the analysis, we introduce some new quantities. We denote the extended optimal global model as $\mathbf{w}_e^* = \mathbf{1}_{K+1} \otimes \mathbf{w}^*$ and the extended global model estimate as $\mathbf{w}_{e,n} = \text{col}\{\mathbf{w}_n, \mathbf{w}_{1,n}, \dots, \mathbf{w}_{K,n}\}$. We also define the following collective quantities

$$\mathbf{X}_n = \text{bdiag}\{\mathbf{0}, \mathbf{x}_{1,n}, \dots, \mathbf{x}_{k,n}\} \quad (3.8a)$$

$$\boldsymbol{\delta}_{e,n} = \text{col}\{\mathbf{0}, \boldsymbol{\delta}_{1,n}, \dots, \boldsymbol{\delta}_{k,n}\} \quad (3.8b)$$

$$\mathbf{B} = \text{bdiag}\{\mathbf{O}_L, \beta_1 \mathbf{I}_L, \dots, \beta_K \mathbf{I}_L\} \quad (3.8c)$$

$$\mathbf{T}_n = \text{bdiag}\{\mathbf{O}_L, \tau_{1,n} \mathbf{I}_L, \dots, \tau_{K,n} \mathbf{I}_L\} \quad (3.8d)$$

$$\boldsymbol{\nu}_{e,n} = \text{col}\{0, \nu_{1,n}, \dots, \nu_{k,n}\}, \quad (3.8e)$$

where the operators $\text{col}\{\cdot\}$ and $\text{bdiag}\{\cdot\}$ represent column-wise stacking and block diagonalization, respectively. Additionally, $\mathbf{1}_{K+1}$ denotes a column vector of ones with $K + 1$ entries, and \mathbf{O}_L denotes a square zero matrix of size L .

Subsequently, we define

$$\mathbf{y}_{e,n} = \text{col}\{0, y_{1,n}, y_{2,n}, \dots, y_{k,n}\} = \mathbf{X}_n^\top \mathbf{w}_e^* + \boldsymbol{\nu}_{e,n} \quad (3.9a)$$

$$\boldsymbol{\epsilon}_{e,n} = \text{col}\{0, \epsilon_{1,n}, \epsilon_{2,n}, \dots, \epsilon_{k,n}\} = \mathbf{y}_{e,n} - \mathbf{X}_n^\top \mathcal{A}_n \mathbf{w}_{e,n} \quad (3.9b)$$

and

$$\mathcal{A}_n = \begin{bmatrix} \mathbf{I}_L & \mathbf{O}_L & \mathbf{O}_L & \dots & \mathbf{O}_L \\ a_{1,n} \mathbf{S}_{1,n} & \mathbf{I}_L - a_{1,n} \mathbf{S}_{1,n} & \mathbf{O}_L & \dots & \mathbf{O}_L \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k,n} \mathbf{S}_{k,n} & \mathbf{O}_L & \mathbf{O}_L & \dots & \mathbf{I}_L - a_{k,n} \mathbf{S}_{k,n} \end{bmatrix}, \quad (3.10)$$

where $a_{k,n} = 1$ if $k \in \mathcal{S}_n$ and $a_{k,n} = 0$ otherwise. Hence, the global recursion equations of PSO-Fed can be expressed as

$$\mathbf{w}_{e,n+1} = \mathcal{B}_{n+1} (\mathcal{A}_n \mathbf{w}_{e,n} + \mu \mathbf{X}_n \boldsymbol{\epsilon}_{e,n}) + \mathcal{C}_{n+1} \mathbf{B} \mathbf{T}_n \boldsymbol{\delta}_{e,n+1}, \quad (3.11)$$

where

$$\mathcal{B}_{n+1} = \begin{bmatrix} \mathbf{I}_L - \sum_{k=1}^K \frac{a_{k,n}}{|\mathcal{S}_n|} \mathbf{S}_{k,n+1} & \frac{a_{1,n}}{|\mathcal{S}_n|} \mathbf{S}_{1,n+1} & \cdots & \frac{a_{K,n}}{|\mathcal{S}_n|} \mathbf{S}_{K,n+1} \\ \mathbf{O}_L & \mathbf{I}_L & \cdots & \mathbf{O}_L \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O}_L & \mathbf{O}_L & \cdots & \mathbf{I}_L \end{bmatrix} \quad (3.12)$$

$$\mathcal{C}_{n+1} = \begin{bmatrix} \mathbf{O}_L & \frac{a_{1,n}}{|\mathcal{S}_n|} \mathbf{S}_{1,n+1} & \cdots & \frac{a_{K,n}}{|\mathcal{S}_n|} \mathbf{S}_{K,n+1} \\ \mathbf{O}_L & \mathbf{O}_L & \cdots & \mathbf{O}_L \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{O}_L & \mathbf{O}_L & \cdots & \mathbf{O}_L \end{bmatrix}. \quad (3.13)$$

To make the analysis tractable, we adopt the following commonly used assumptions.

A1: The input vectors of each client, $\mathbf{x}_{k,n}$, originate from a wide-sense stationary multivariate random process characterized by the covariance matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{x}_{k,n} \mathbf{x}_{k,n}^\top]$.

A2: The observation noise, $\nu_{k,n}$, and perturbation signal, $\boldsymbol{\delta}_{k,n}$, both follow identical and independent distributions. They are also independent of all other stochastic variables.

A3: The selection matrices, $\mathbf{S}_{k,n}$, of all clients and iterations are independent of each other.

3.3.1 Mean Convergence

We define the deviation (coefficient-error) vector as $\tilde{\mathbf{w}}_{e,n} = \mathbf{w}_e^* - \mathbf{w}_{e,n}$. Since \mathcal{B}_{n+1} and \mathcal{A}_n are block right-stochastic, i.e., their block rows add up to identity matrix, we have $\mathcal{B}_{n+1} \mathbf{w}_e^* = \mathcal{A}_n \mathbf{w}_e^* = \mathbf{w}_e^*$. Therefore, considering (3.11) and the definition of $\tilde{\mathbf{w}}_{e,n}$, we get

$$\begin{aligned} \tilde{\mathbf{w}}_{e,n+1} &= \mathcal{B}_{n+1} (\mathbf{I} - \mu \mathbf{X}_n \mathbf{X}_n^\top) \mathcal{A}_n \tilde{\mathbf{w}}_{e,n} - \mu \mathcal{B}_{n+1} \mathbf{X}_n \boldsymbol{\nu}_{e,n} \\ &\quad - \mathcal{C}_{n+1} \mathbf{B} \mathbf{T}_n \boldsymbol{\delta}_{e,n+1}. \end{aligned} \quad (3.14)$$

Taking the expected value of both sides of (3.14) and under assumptions A1-A3, we obtain

$$\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}] = \mathbb{E}[\mathcal{B}_{n+1}] (\mathbf{I} - \mu \mathcal{R}) \mathbb{E}[\mathcal{A}_n] \mathbb{E}[\tilde{\mathbf{w}}_{e,n}], \quad (3.15)$$

where

$$\mathbf{R} = \text{bdiag}\{\mathbf{O}, \mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_K\}. \quad (3.16)$$

The mean convergence of PSO-Fed is guaranteed if $\|(\mathbf{I} - \mu\mathbf{R})\| < 1$ or equivalently $\|1 - \mu\lambda_i(\mathbf{R}_k)\| < 1 \forall k, i$. Here, $\|\cdot\|$ is any matrix norm and $\lambda_i(\cdot)$ denotes the i th eigenvalue of its matrix argument. As a result, PSO-Fed achieves convergence in the mean sense, even in the presence of Byzantine clients, provided that the stepsize μ is appropriately chosen to satisfy

$$0 < \mu < \frac{2}{\max_{i,k}\{\lambda_i(\mathbf{R}_k)\}}. \quad (3.17)$$

In other terms, PSO-Fed is unbiased, even in the presence of model-poisoning attacks.

3.3.2 Mean-Square Convergence

We denote the weighted norm-square of $\tilde{\mathbf{w}}_{e,n}$ as $\|\tilde{\mathbf{w}}_{e,n}\|_{\Sigma}^2 = \tilde{\mathbf{w}}_{e,n}^{\top} \Sigma \tilde{\mathbf{w}}_{e,n}$, where Σ is a positive semi-definite matrix. Computing the weighted norm-square of both sides of (3.14) results in

$$\begin{aligned} \mathbb{E}[\|\tilde{\mathbf{w}}_{e,n+1}\|_{\Sigma}^2] &= \mathbb{E}[\|\tilde{\mathbf{w}}_{e,n}\|_{\Sigma'}^2] + \mu^2 \mathbb{E}[\boldsymbol{\nu}_{e,n}^{\top} \mathbf{Y}_n \boldsymbol{\nu}_{e,n}] \\ &\quad + \mathbb{E}[\boldsymbol{\delta}_{e,n+1}^{\top} \mathbf{U}_n \boldsymbol{\delta}_{e,n+1}], \end{aligned} \quad (3.18)$$

where the cross terms vanish under assumption A2 and

$$\Sigma' = \mathbb{E}[\mathbf{A}_n^{\top} (\mathbf{I} - \mu \mathbf{X}_n \mathbf{X}_n^{\top}) \mathbf{B}_{n+1}^{\top} \Sigma \mathbf{B}_{n+1} (\mathbf{I} - \mu \mathbf{X}_n \mathbf{X}_n^{\top}) \mathbf{A}_n] \quad (3.19)$$

$$\mathbf{Y}_n = \mathbf{X}_n^{\top} \mathbf{B}_{n+1}^{\top} \Sigma \mathbf{B}_{n+1} \mathbf{X}_n \quad (3.20)$$

$$\mathbf{U}_n = \mathbf{T}_n^{\top} \mathbf{B}^{\top} \mathbf{C}_{n+1}^{\top} \Sigma \mathbf{C}_{n+1} \mathbf{B} \mathbf{T}_n. \quad (3.21)$$

Subsequently, we define

$$\boldsymbol{\sigma}' = \text{bvec}\{\Sigma'\} = \mathcal{F}^{\top} \boldsymbol{\sigma} \quad (3.22)$$

$$\boldsymbol{\sigma} = \text{bvec}\{\Sigma\} \quad (3.23)$$

$$\mathcal{F} = \mathcal{Q}_{\mathbf{B}} \mathcal{Q}_{\mathbf{A}} - \mu \mathcal{Q}_{\mathbf{B}} \mathcal{K} \mathcal{Q}_{\mathbf{A}} + \mu^2 \mathcal{Q}_{\mathbf{B}} \mathcal{H} \mathcal{Q}_{\mathbf{A}} \quad (3.24)$$

$$\mathcal{Q}_{\mathbf{B}} = \mathbb{E}[\mathbf{B}_{n+1} \otimes_b \mathbf{B}_{n+1}] \quad (3.25)$$

$$\mathcal{Q}_{\mathbf{A}} = \mathbb{E}[\mathbf{A}_n \otimes_b \mathbf{A}_n] \quad (3.26)$$

$$\mathcal{K} = (\mathbf{I} \otimes_b \mathcal{R}) + (\mathcal{R} \otimes_b \mathbf{I}) \quad (3.27)$$

$$\mathcal{H} = \mathbb{E}[\mathbf{X}_n \mathbf{X}_n^\top \otimes_b \mathbf{X}_n \mathbf{X}_n^\top] \quad (3.28)$$

$$\mathcal{D} = \begin{bmatrix} \mathcal{K}/2 & -\mathcal{H}/2 \\ \mathbf{I} & \mathbf{O} \end{bmatrix} \quad (3.29)$$

$$\mathcal{Q}_{\mathcal{C}} = \mathbb{E}[\mathbf{C}_{n+1} \otimes_b \mathbf{C}_{n+1}] \quad (3.30)$$

$$\boldsymbol{\omega} = \mathcal{Q}_{\mathcal{C}} \text{bvec}\{\boldsymbol{\Omega}_{\delta}\}, \quad (3.31)$$

$$\boldsymbol{\Omega}_{\delta} = \text{bdiag}\{0, \beta_1 \sigma_B^2 p_a \mathbf{I}_D, \beta_2 \sigma_B^2 p_a \mathbf{I}_D, \dots, \beta_K \sigma_B^2 p_a \mathbf{I}_D\} \quad (3.32)$$

$$\boldsymbol{\phi} = \mathcal{Q}_{\mathcal{B}} \boldsymbol{\phi}_{\nu} \quad (3.33)$$

$$\boldsymbol{\phi}_{\nu} = \text{bvec}\{\mathbb{E}[\mathbf{X}_n \boldsymbol{\Theta}_{\nu} \mathbf{X}_n^\top]\} \quad (3.34)$$

$$\boldsymbol{\Theta}_{\nu} = \mathbb{E}[\boldsymbol{\nu}_{e,n} \boldsymbol{\nu}_{e,n}^\top] = \text{diag}\{0, \sigma_{\nu_1}^2, \dots, \sigma_{\nu_K}^2\}, \quad (3.35)$$

where \otimes_b denotes the block Kronecker product (Tracy–Singh product [90]) and $\text{bvec}\{\cdot\}$ denotes the block vectorization operation [91]. We evaluate $\mathcal{Q}_{\mathcal{A}}$, $\mathcal{Q}_{\mathcal{B}}$, \mathcal{H} , $\mathcal{Q}_{\mathcal{C}}$ and $\boldsymbol{\phi}_{\nu}$ under A1-A3, in Appendices A-D in **P2**, respectively.

Proposition 3.1: PSO-Fed converges in the mean-square sense and has a bounded steady-state MSD, even in the presence of model-poisoning attacks given an appropriate choice of stepsize as

$$0 < \mu < \min\left\{\frac{1}{\lambda_{\max}(\mathcal{K}^{-1}\mathcal{H})}, \frac{1}{\max\{\lambda(\mathcal{D}) \in \mathbb{R}, 0\}}\right\}, \quad (3.36)$$

where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of its matrix argument. Note that we denote the upper bound of (3.36) as μ_{\max} .

Proof. See **P2**.

3.3.3 Steady-State Mean Square Error

Proposition 3.2: The steady-state MSE of PSO-Fed (denoted by \mathcal{E}), while taking into account the impact of model-poisoning attacks can be computed as [92]

$$\begin{aligned} \mathcal{E} &= \frac{1}{K} \left[(\mu^2 \boldsymbol{\phi}^\top + \boldsymbol{\omega}^\top) (\mathbf{I} - \mathcal{F}^\top)^{-1} \mathcal{Q}_{\mathcal{A}}^\top \text{bvec}\{\mathcal{R}\} + \text{tr}(\boldsymbol{\Theta}_{\nu}) \right] \\ &= \underbrace{\frac{\mu^2}{K} \boldsymbol{\phi}^\top (\mathbf{I} - \mathcal{F}^\top)^{-1} \mathcal{Q}_{\mathcal{A}}^\top \text{bvec}\{\mathcal{R}\}}_{\mathcal{E}_{\phi}} \\ &\quad + \underbrace{\frac{1}{K} \boldsymbol{\omega}^\top (\mathbf{I} - \mathcal{F}^\top)^{-1} \mathcal{Q}_{\mathcal{A}}^\top \text{bvec}\{\mathcal{R}\}}_{\mathcal{E}_{\omega}} + \underbrace{\frac{1}{K} \text{tr}(\boldsymbol{\Theta}_{\nu})}_{\mathcal{E}_{\Theta}} \end{aligned} \quad (3.37)$$

Proof. See **P2**.

The partial sharing of model parameters, the scheduling of clients, the statistics of input data, and the value of stepsize all affect the first and second terms on the right-hand side of (3.37), specifically \mathcal{E}_ϕ and \mathcal{E}_ω . The observation noise impacts the first and third terms, \mathcal{E}_ϕ and \mathcal{E}_Θ , with the latter being solely attributable to this noise. Additionally, the influence of model-poisoning attacks from Byzantine clients on the steady-state MSE of PSO-Fed is restricted to the second term, \mathcal{E}_ω , which is induced by these attacks.

Remark 3.1: Reflecting on (3.31) and the computation of \mathcal{Q}_C in Appendix C of **P2**, it is evident that both partial sharing and client scheduling have a reducing impact on ω . As the probability of sharing each element of the model parameter vector decreases, denoted as $p_e = \frac{M}{L}$, the values of ω decrease correspondingly. Consequently, (3.37) implies that partial sharing could provide online FL with increased resilience against model-poisoning attacks. Nonetheless, partial sharing also influences \mathcal{E}_ϕ , causing its value to rise as indicated in [63, Figure 1(a)]. This complicates the ability to easily ascertain the overall influence of partial sharing on the steady-state MSE of PSO-Fed in situations related to model-poisoning attacks, solely by examining (3.37). To achieve a more thorough understanding and substantiate these theoretical predictions, we conduct extensive simulations in section 3.4.

3.3.4 Optimal Stepsize

When there are no model-poisoning attacks and $\mathcal{E}_\omega = 0$, the function \mathcal{E} increases monotonically with the stepsize μ , provided that (3.36) holds true. However, in the presence of model-poisoning attacks, the presence of \mathcal{E}_ω disrupts this monotonic behavior. It amplifies \mathcal{E} for smaller values of μ and moves the point of minimal \mathcal{E} from $\mu = 0$ to a larger value of $\mu > 0$. As a result, this introduces a non-trivial optimal stepsize that achieves the minimum \mathcal{E} when confronted with model-poisoning attacks. We refer to this optimal stepsize as μ^* .

Theorem 3.1. The approximate optimal stepsize as

$$\mu^* \approx \frac{\omega^\top \mathbf{B}_{1,J} \mathbf{Q}_A^\top \text{bvec}\{\mathcal{R}\}}{2(\phi^\top \mathbf{B}_{0,J} + \omega^\top \mathbf{B}_{2,J}) \mathbf{Q}_A^\top \text{bvec}\{\mathcal{R}\}}. \quad (3.38)$$

Note that, when no model-poisoning attack occurs, i.e., $\omega = 0$, the numerator on the RHS of (3.38) becomes zero, leading to $\mu^* = 0$.

Proof. See **P2**.

3.4 Numerical Results

To verify our theoretical findings, we conduct several numerical experiments. We also compare the performance of the PSO-Fed algorithm with that of the Online-Fed, SignSGD, CS-Fed, and QS-Fed algorithms within a federated network comprising $K \in \{50, 100\}$ clients and a model parameter vector of size $L = 5$. Each client possesses non-IID data vectors $\mathbf{x}_{k,n}$ and their associated response values $y_{k,n}$, which are interconnected as per (3.1) with model parameter vector $\mathbf{w} = \frac{1}{\sqrt{L}}[1, \dots, 1]^\top \in \mathbb{R}^L$. Each entry of $\mathbf{x}_{k,n}$ for each client k is drawn from a zero-mean Gaussian distribution with variance ζ_K^2 , where ζ_K^2 itself is sampled from a uniform distribution between 0.2 and 1.2, denoted as $\mathcal{U}(0.2, 1.2)$. In addition, the observation noise $\nu_{k,n}$ is zero-mean IID Gaussian with variance $\sigma_{\nu_k}^2$ drawn from $\mathcal{U}(0.005, 0.025)$. In our experiments, only $M = 1$ entry of the model parameter vector is shared per iteration, unless specified differently.

We evaluate the performance of the considered algorithms on the server's side by employing a test dataset consisting of $N_t = 50$ instances $\{\check{\mathbf{X}}, \check{\mathbf{y}}\}$ and calculating the test MSE at the server as

$$\frac{1}{N_t} \|\check{\mathbf{y}} - \check{\mathbf{X}}^\top \mathbf{w}_n\|_2^2. \quad (3.39)$$

We also evaluate the performance of PSO-Fed on the client side using (3.4a) and calculating the network-wide average steady-state MSE as

$$\frac{1}{K} \sum_{k=1}^K \lim_{n \rightarrow \infty} \epsilon_{k,n}^2. \quad (3.40)$$

In our first experiment, we examine the effect of different numbers of Byzantine clients $|\mathcal{S}_B|$ on the steady-state test MSE for PSO-Fed, Online-Fed, SignSGD, CS-Fed, and QS-Fed. In this experiment, We have $K = 100$ clients. In every iteration n , the server randomly selects a set of $|\mathcal{S}_n| = 5$ clients to participate in FL, with each client having an equal probability of being selected. In addition, all Byzantine clients are characterized by model-poisoning noise variance of $\sigma_B^2 = 0.25$ and attack probability of $p_a = 1$. In each iteration, we allocate the same communication budget to all algorithms. Therefore, to allow all algorithms converge to their lowest steady-state MSD within 3000 iterations, we set $\mu = 0.15$ for PSO-Fed, Online-Fed, CS-Fed and QS-Fed, and $\mu = 0.08$ for SignSGD at all clients. According to (3.36), the maximum stepsize ensuring the mean-square convergence of PSO-Fed in this experiment is $\mu_{max} = 0.245$. The results displayed in Figure 3.3 indicate that PSO-Fed outperforms the existing algorithms across all considered number of Byzantine clients. Notably, Online-Fed and PSO-Fed perform similarly

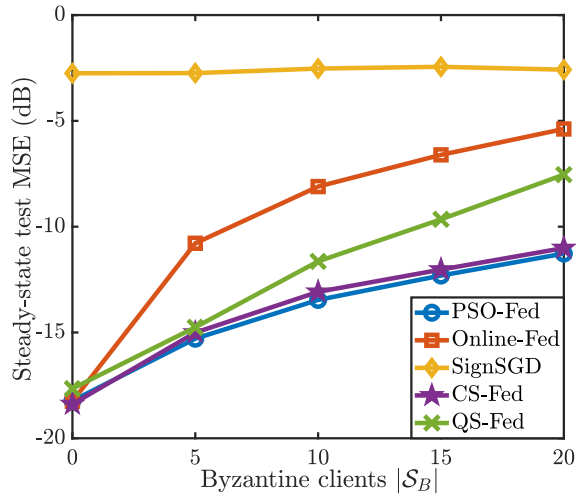


Figure 3.3: Steady-state test MSE for different algorithms with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 1$.

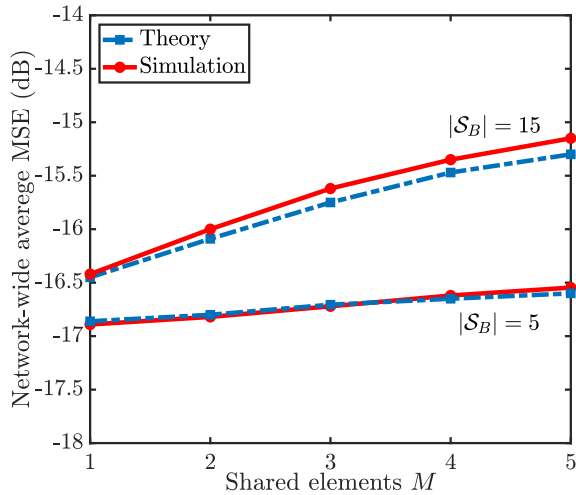


Figure 3.4: Steady-state test MSE of PSO-Fed for different numbers of shared elements M with different numbers of Byzantine clients $|\mathcal{S}_B|$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$.

when there is no Byzantine client or poisoning attack (i.e., $\sigma_B^2 = 0$). However, the performance of Online-Fed deteriorates more than PSO-Fed as the number of Byzantine clients increases. While CS-Fed performs closely to PSO-Fed in our experiment, it is important to note that the computational complexity of CS-Fed exceeds that of PSO-Fed.

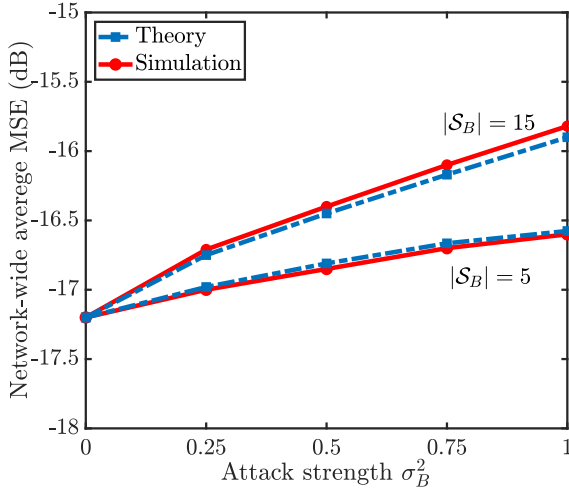


Figure 3.5: Network-wide average MSE of PSO-Fed for different values of attack strengths σ_B^2 and Byzantine clients $|\mathcal{S}_B|$, number of shared elements $M = 1$ and attack probability $p_a = 0.2$.

In our second experiment, we examine the impact of the number of shared entries, M , on the steady-state test MSE of PSO-Fed. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 = 0.5$, and attack probability $p_a = 0.2$. The results are shown in Figure 3.5. We observe that increasing M leads to higher steady-state test MSE. In addition, PSO-Fed exhibits greater resilience to model-poisoning attacks compared to Online-Fed, without incurring any extra computational or communication burden on the clients. This experiment corroborates our theoretical findings discussed in section 3.3.3.

In our third experiment, we investigate the effect of varying the attack strength σ_B^2 on the network-wide average steady state MSE. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 \in \{0, 0.25, 0.5, 0.75, 1\}$, and attack probability $p_a = 0.2$. The server randomly selects 5 clients in each iteration. The results depicted in Figure 3.4 align closely with our theoretical predictions. We notice an upward trend in the network-wide average steady-state MSE as the attack strength or the number of Byzantine clients increases. In addition, the presence of $|\mathcal{S}_B| = \alpha_1$ Byzantine clients with an attack strength of $\sigma_B^2 = \beta_1$ results in the same MSE as having $|\mathcal{S}_B| = \alpha_2$ Byzantine clients with an attack strength $\sigma_B^2 = \beta_2$, provided that the condition $\alpha_1\beta_1 = \alpha_2\beta_2$ is met.

In our fourth experiment, we explore how the attack probability p_a influences the steady-state test MSE of PSO-Fed, considering different numbers of shared entries

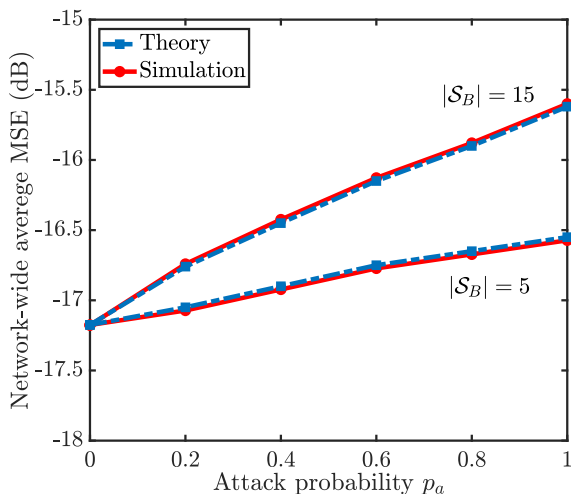


Figure 3.6: Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of Byzantine clients $|\mathcal{S}_B|$, number of shared elements $M = 1$ and attack strength $\sigma_B^2 = 0.25$.

$M \in \{1, 5\}$, with the number of Byzantine clients set to $|\mathcal{S}_B| = 5$ and attack strength to $\sigma_B^2 = 0.25$. The results, presented in Figure 3.7, align with observations from the previous experiment, showing that an increase in the attack probability leads to a higher MSE. Moreover, partial sharing (i.e., when $M < L$) enhances the resilience against model-poisoning attacks as elaborated upon in section 3.3.3.

In our fifth experiment, we investigate the impact of attack probability p_a on the steady-state test MSE of PSO-Fed, considering different numbers of Byzantine clients $|\mathcal{S}_B| \in \{5, 15\}$ and setting the attack strength as $\sigma_B^2 = 0.25$. The results shown in Figure 3.6 indicate that increasing the attack probability increases the steady-state MSE.

In our sixth experiment, we evaluate the performance of PSO-Fed using different stepsize values and Byzantine client numbers in the presence or absence of model-poisoning attacks. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, attack strength $\sigma_B^2 = 0.25$, attack probability $p_a = 0.25$. The server randomly selects 5 clients in each iteration. The results illustrated in Figure 3.8, show that with no model-poisoning attack ($|\mathcal{S}_B| = 0$), the larger the stepsize μ , the worse PSO-Fed performs, a trend consistent with gradient descent optimization methods. However, with model-poisoning attacks ($|\mathcal{S}_B| > 0$), the performance first improves with an increase in μ , then deteriorates when μ grows larger. This observation confirms the existence of an optimal stepsize μ^* that ensures the best performance of PSO-Fed under model-poisoning attacks. By calculating the optimal stepsize using (3.38),

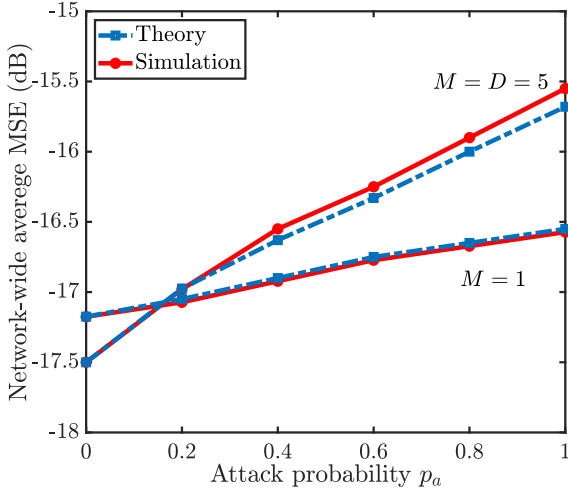


Figure 3.7: Effect of attack probability p_a on steady-state test MSE of PSO-Fed for different numbers of shared elements $M \in \{1, 5\}$, number of Byzantine clients $|\mathcal{S}_B| = 5$ and attack strength $\sigma_B^2 = 0.25$.

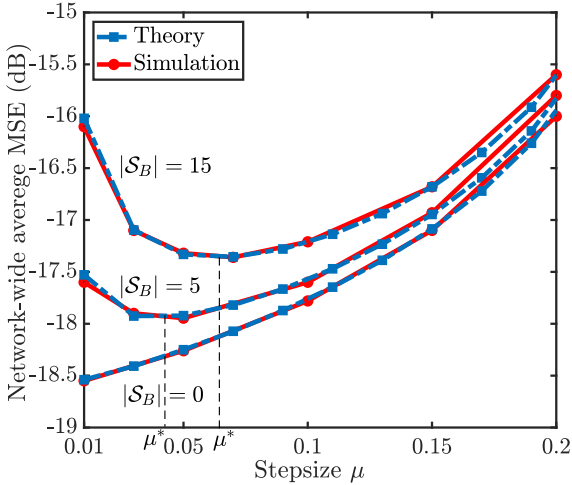


Figure 3.8: Network-wide average MSE of PSO-Fed for different values of stepsize μ , attack strength $\sigma_B^2 = 0.25$ and attack probability $p_a = 0.25$.

we determine $\mu^* \approx 0.03$, which corresponds to the experimental results.

In our seventh experiment, we analyze the performance of PSO-Fed using different stepsizes in the presence or absence of model-poisoning attacks, considering different attack strengths σ_B^2 . We simulate PSO-Fed with $K = 50$ cli-

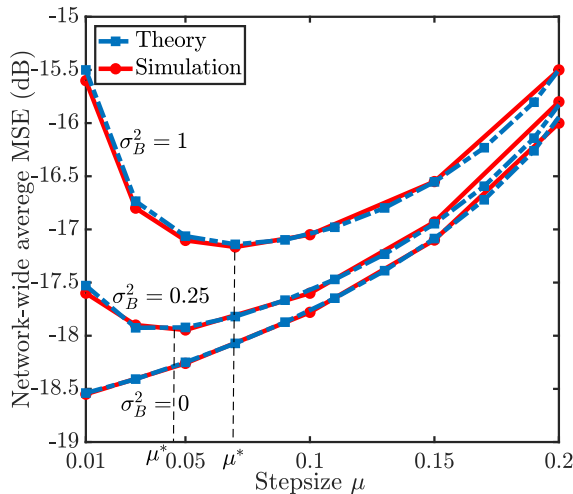


Figure 3.9: Network-wide average MSE of PSO-Fed for different values of stepsize μ , numbers of Byzantine clients $|\mathcal{S}_B| = 5$ and attack probability $p_a = 0.25$.

ents, $L = 5$, $|\mathcal{S}_B| = 5$ Byzantine clients, and attack probability of $p_a = 0.25$. The server randomly selects 5 clients in each iteration. The results, presented in Figure 3.9, echo the observations from the previous experiment: without a poisoning attack ($\sigma_B^2 = 0$), PSO-Fed’s performance decreases as the stepsize μ increases. Conversely, with model-poisoning attacks ($\sigma_B^2 > 0$), performance initially improves with an increase in μ but begins to decline as μ is further increased. This pattern confirms the existence of an optimal stepsize μ^* for scenarios with model-poisoning attacks. Using (3.38) to estimate the optimal stepsize, we find $\mu^* \approx 0.03$, corresponding to the experiment’s findings. This experiment also reinforces our conclusion following the third experiment that identical attack properties result in the same MSE.

In our eighth experiment, we investigate the impact of the commonly-adopted small stepsize approximation (SSA) on predicting the performance of PSO-Fed in the presence or absence of model-poisoning attacks. Specifically, we assume that higher orders of μ are negligible when computing \mathcal{F} via (3.24). We then compare the simulated network-wide average MSE of PSO-Fed with the corresponding theoretical predictions with or without SSA. We simulate PSO-Fed with $K = 50$ clients, $L = 5$, attack probability $p_a = 0.25$, attack strength $\sigma_B^2 = 0.5$, and numbers of Byzantine clients $|\mathcal{S}_B| \in \{0, 10\}$. The server randomly selects 5 clients in each iteration. The results in Figure 3.10 reveal that, when μ is sufficiently small, SSA does not hinder the accurate prediction of steady-state MSE even without the knowledge of the 4th moment of the input vector, \mathcal{H} . However, as the stepsize in-

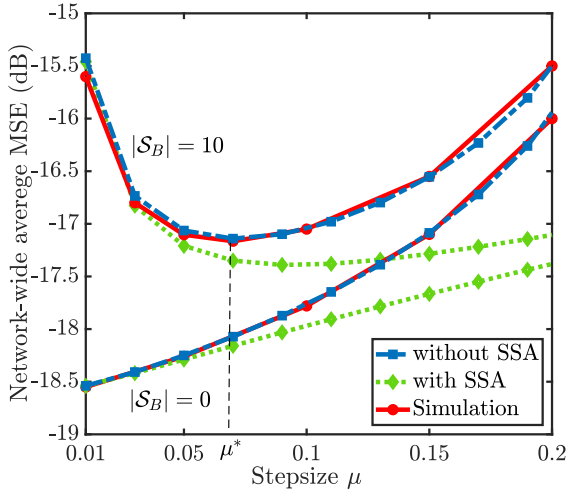


Figure 3.10: Effect of small stepsize approximation (SSA) on network-wide average MSE of PSO-Fed, numbers of Byzantine clients $|\mathcal{S}_B| \in \{0, 10\}$, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.25$.

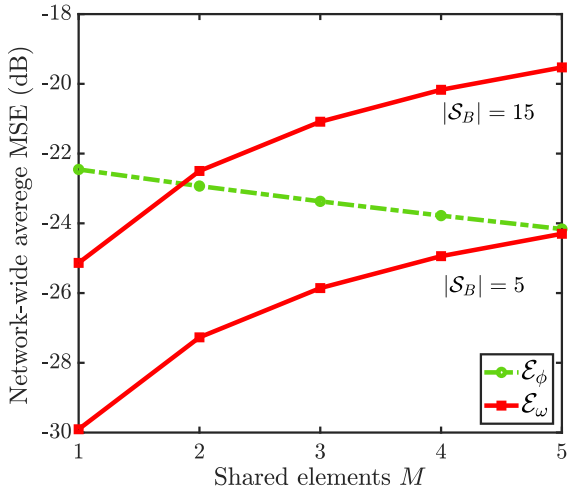


Figure 3.11: Effect of number of shared elements M on \mathcal{E}_ω and \mathcal{E}_ϕ in (3.37) for $|\mathcal{S}_B| \in \{5, 15\}$ Byzantine clients, attack strength $\sigma_B^2 = 0.5$ and attack probability $p_a = 0.2$.

creases, discrepancies arise in both scenarios, with and without model-poisoning attacks, diverging from the ideal case.

In our final experiment, we investigate the effect of partial sharing on different terms of MSE (3.37), namely, \mathcal{E}_ω and \mathcal{E}_ϕ , in the presence of model-poisoning

attacks. We compute both terms using the same setup as in our second experiment. The results depicted in Figure 3.11 show that decreasing the number of shared parameters considerably lowers \mathcal{E}_ω , significantly mitigating the impact of model-poisoning attacks. However, sharing fewer entries increases \mathcal{E}_ϕ . Still, the benefits of partial sharing in reducing \mathcal{E}_ω are more substantial, leading to reduced MSE. Note that \mathcal{E}_ϕ remains unchanged across both considered cases, as it depends solely on scheduling and data noise. Finally, this experiment corroborates our theoretical findings and confirms our assertions regarding the effect of partial sharing on \mathcal{E}_ω and \mathcal{E}_ϕ , as discussed at the end of section 3.3.3.

3.5 Summary

We conducted a theoretical analysis of the recently proposed PSO-Fed algorithm to examine its resilience to model-poisoning (Byzantine) attacks imparted by partial sharing. In our analysis, we considered a linear regression task with the local objective function of each client defined as an empirical risk. We showed that, even under Byzantine attacks, PSO-Fed converges in both mean and mean-square senses, given an appropriate choice of stepsize. More importantly, we showed that, in the presence of Byzantine clients, the steady-state MSE of PSO-Fed is significantly smaller than that of the Online-Fed algorithm, which does not feature partial sharing. Our theoretical analysis also uncovered the existence of a non-trivial optimal stepsize for PSO-Fed in the presence of model-poisoning attacks. The simulation results corroborated our theoretical findings regarding PSO-Fed's effectiveness against Byzantine attacks as well as the accuracy of the theoretically predicted values of its steady-state MSE and optimal stepsize.

Chapter 4

Resource-Efficient FL Robust to Communication Errors

This chapter presents the results of publications **P3**, **P4** and their extension **P5**. It presents an FL algorithm that is robust to communication noise/error while reducing the communication burden of edge devices. We derive the proposed algorithm through solving the weighted least-squares (WLS) regression problem. We first cast the WLS regression problem as distributed convex optimization over a federated network that employs random scheduling for communication efficiency. We then use the alternating direction method of multipliers (ADMM) to solve the problem iteratively. To mitigate the adverse effects of cumulative communication noise/error, we introduce a new local model update at each participating edge device. Our theoretical analysis shows that the convergence of our algorithm is ensured when the server communicates with a random subset of the clients at each iteration. Our numerical results also demonstrate the superiority of our algorithm over the state of the art and corroborate our theoretical findings.

4.1 Motivation

FL depends on effective communication to train a machine learning model using decentralized data, aiming to create a more precise global model. When the communication channels between the server and the edge devices are unreliable, the server receives distorted local updates stemming from uplink noise, while each client gets a noisy version of the aggregated model from the server due to downlink noise [93–95]. Utilizing models that have been affected by communication noise or errors can degrade the accuracy of the learned model. Many studies on FL have mainly concentrated on uplink noise [96, 97]. The work in [98] explores

the effects of downlink noise on FL. These investigations indicate that the effectiveness of gradient-descent-based FL algorithms can decline when noise is present in the communication channels. In [94], a novel loss function is introduced for FL, employing the first-order derivative of the loss function as a regularizer to address the issue of additive noise in communication links. In [98], two strategies are suggested to enhance FL's resilience against downlink noise. The first strategy involves using a quantization technique while transmitting the global model update through digital links and implementing channel coding with a standard rate. The second strategy centers on an analog downlink transmission method, where the server sends an uncoded global model update.

The authors of [99, 100] suggest that by controlling the scale of the communication signal-to-noise ratio, the impact of noise can be mitigated, allowing the convergence rate of FedAvg with reliable communication links to be preserved. Nonetheless, they do not address any strategies to counteract the effects of the noise. In [101], the authors implement precoding and scaling during transmissions to reduce the detrimental impacts of noisy channels and to ensure the convergence of their algorithm. Many of these approaches to handle link noise in FL typically necessitate extra resources on the client side. This can be counterproductive, as clients in FL often function with limited resources regarding power/energy, memory, or computational ability.

FL algorithms that utilize the alternating direction method of multipliers (ADMM) tend to show a degree of resilience against additive communication noise [102]. Nevertheless, these methods necessitate that all clients take part in every round of FL, which can be unrealistic in practical situations where clients are limited in resources or consist of diverse edge devices. Thus, there is a significant need for FL algorithms that can withstand noise in communication channels while requiring minimal additional communication or computational efforts.

This chapter aims to develop a resource-efficient ADMM-based FL algorithm that is robust to communication noise/error while imposing no additional computational burden on the participating clients. We consider the presence of noise in both uplink and downlink communications. Considering the weighted least-squares (WLS) regression problem, we develop our proposed FL algorithm by iteratively solving an appropriately formulated distributed convex optimization problem via ADMM. To achieve communication efficiency, we employ random client scheduling. Furthermore, to prevent error accumulation from degrading the learning, we communicate a linear combination of the last two global model updates, as well as eliminating the dual model parameters at all participating edge devices. Through theoretical analysis, we show that the convergence of the proposed algorithm is ensured when the server chooses a random subset of the clients at each iteration,

even with noisy communication links. Our simulation results also attest to the effectiveness of our algorithm in comparison with the state of the art, as well as corroborating our theoretical findings.

It is noteworthy to mention that the emphasis on the WLS regression problem enables us to address challenges associated with random client participation in noisy FL settings. By concentrating on the WLS problem, we can guarantee mean convergence for both global and local models towards the optimal solution; an assurance that is often lacking in more generalized formulations. Furthermore, this focus allows a novel mean-square convergence analysis, as well as a closed-form expression for the mean-square error.

4.2 Proposed Method

We consider a federated network consisting of K clients and one server, where the clients communicate with the server over wireless channels. Each client k has access to a local dataset denoted by $\mathcal{D}_k = \{\mathbf{X}_k, \mathbf{y}_k\}$, which comprises a column response vector \mathbf{y}_k with d_k entries, and a data matrix \mathbf{X}_k of size $d_k \times L$. For each client k , a linear model is employed to relate the data matrix \mathbf{X}_k to the response vector \mathbf{y}_k as

$$\mathbf{y}_k = \mathbf{X}_k \boldsymbol{\omega} + \boldsymbol{\nu}_k, \quad (4.1)$$

where $\boldsymbol{\omega}$ denotes the global regression model parameter vector of size L , and $\boldsymbol{\nu}_k$ represents the observation noise or perturbation, which is a vector of size d_k with each entry assumed to be zero-mean Gaussian.

4.2.1 Federated Weighted Least-Squares Regression

Weighted least-squares (WLS) regression serves as a logical extension of least-squares regression, providing notable benefits in numerous signal processing applications, including power system state estimation [103], position estimation [104], and image noise reduction [105]. In WLS, various observations are given weights according to their reliability, enabling the model to more accurately represent the data when the quality of observations differs. This method can effectively reduce the influence of less reliable data, leading to more accurate and trustworthy models.

In federated WLS regression, the goal is to collaboratively estimate the global model parameter vector $\boldsymbol{\omega}$ by minimizing a global objective function across a federated network. This is expressed as a global WLS estimation problem in the FL

framework as

$$\begin{aligned} \min_{\{\mathbf{w}_k\}} \sum_{k=1}^K \mathcal{J}_k(\mathbf{w}_k) \\ \text{s.t. } \mathbf{w}_k = \mathbf{w}, k \in \{1, 2, \dots, K\}, \end{aligned} \quad (4.2)$$

where $\mathcal{J}_k(\mathbf{w}_k) = \|\mathbf{y}_k - \mathbf{X}_k \mathbf{w}_k\|_{\mathcal{W}_k}^2$ is the local objective function for estimating ω at client k , \mathcal{W}_k is the weight matrix specific to client k , \mathbf{w}_k is the local model estimate at client k and \mathbf{w} can be viewed as the global model estimate.

The optimal solution to (4.2), denoted by \mathbf{w}^* , can be stated as

$$\mathbf{w}^* = \left(\sum_{k=1}^K \mathbf{X}_k^\top \mathcal{W}_k \mathbf{X}_k \right)^{-1} \left(\sum_{k=1}^K \mathbf{X}_k^\top \mathcal{W}_k \mathbf{y}_k \right). \quad (4.3)$$

To solve (4.2) within an FL framework, we utilize the ADMM algorithm [48]. Therefore, we express the augmented Lagrangian function for (4.2) as

$$\sum_{k=1}^K \mathcal{L}_k(\mathbf{w}_k, \mathbf{w}, \mathbf{z}_k) = \sum_{k=1}^K \mathcal{J}_k(\mathbf{w}_k) + \langle \mathbf{w}_k - \mathbf{w}, \mathbf{z}_k \rangle + \frac{\rho}{2} \|\mathbf{w}_k - \mathbf{w}\|_2^2, \quad (4.4)$$

where \mathbf{z}_k is the Lagrange multiplier vector and $\rho > 0$ is the penalty parameter. Consequently, we derive the recursive update equations at each client k and iteration number n as

$$\mathbf{z}_{k,n} = \mathbf{z}_{k,n-1} + \rho(\mathbf{w}_{k,n} - \mathbf{w}_k) \quad (4.5a)$$

$$\mathbf{w}_{k,n+1} = \hat{\mathbf{w}}_k - \mathbf{N}_k(\mathbf{z}_{k,n} - \rho \mathbf{w}_k), \quad (4.5b)$$

along with the recursive update equation at the server as

$$\mathbf{w}_{n+1} = \frac{1}{K} \sum_{k=1}^K (\mathbf{w}_{k,n+1} + \rho^{-1} \mathbf{z}_{k,n}), \quad (4.6)$$

where we define

$$\mathbf{N}_k = (2\mathbf{X}_k^\top \mathcal{W}_k \mathbf{X}_k + \rho \mathbf{I})^{-1} \quad (4.7)$$

$$\hat{\mathbf{w}}_k = 2\mathbf{N}_k \mathbf{X}_k^\top \mathcal{W}_k \mathbf{y}_k. \quad (4.8)$$

In this solution, after performing local training, i.e., (4.5a) and (4.5b), each client shares its local estimate of $\mathbf{w}_{k,n+1} + \rho^{-1} \mathbf{z}_{k,n}$ with the server. The server then obtains the global estimate as in (4.6) and broadcasts it to all clients while the FL process continues.

4.2.2 Dual Variable Elimination

We posit that, in the recursions (4.5)-(4.6), it is necessary to transmit a combination of the primal and dual model parameter estimates to the server to enable the aggregation that produces the global model estimate. Nevertheless, the dual update information can be incorporated into the primal update by carefully choosing the initial estimates and introducing a new local primal update at the clients.

Accordingly, we can reformulate (4.5)-(4.6) as

$$\mathbf{w}_n = \frac{1}{K} \sum_{k=1}^K \mathbf{w}_{k,n} \quad (4.9a)$$

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho \mathbf{N}_k) \mathbf{w}_{k,n} + \rho \mathbf{N}_k (2\mathbf{w}_n - \mathbf{w}_{n-1}) \quad (4.9b)$$

by initializing with $\mathbf{w}_{-1} = \mathbf{0}$, $\mathbf{z}_{k,-1} = \mathbf{0}$, and $\mathbf{w}_{k,0} = \hat{\mathbf{w}}_k$, hence eliminating the Lagrange multipliers $\mathbf{z}_{k,n}$. The recursion begins with clients sharing their $\hat{\mathbf{w}}_k$ with the server, which then aggregates them and broadcasts the resulting global model estimate to the clients. We further define $\mathbf{s}_n = 2\mathbf{w}_n - \mathbf{w}_{n-1}$ and modify (4.9b) as

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho \mathbf{N}_k) \mathbf{w}_{k,n} + \rho \mathbf{N}_k \mathbf{s}_n. \quad (4.10)$$

One of the main benefits of (4.10) compared to (4.9b) is that the server carries out the linear combination of the latest two global model estimates needed for local model updates, instead of individual clients handling this task. This approach can greatly minimize the effects of communication noise since the combination takes place prior to being sent to the clients.

4.2.3 Communication Noise

Clients and the server often communicate via wireless channels, which are prone to noise in both uplink and downlink communications. In the downlink, clients receive noisy versions of the aggregated model updates from the server. Specifically, at iteration n , client k receives $\tilde{\mathbf{s}}_{k,n} = \mathbf{s}_n + \boldsymbol{\zeta}_{k,n}$ where $\boldsymbol{\zeta}_{k,n}$ represents the downlink noise affecting the transmission. In the uplink, the server receives a noisy version of each client's local model update, i.e., $\tilde{\mathbf{w}}_{k,n+1} = \mathbf{w}_{k,n+1} + \boldsymbol{\eta}_{k,n}$ where $\boldsymbol{\eta}_{k,n}$ denotes the uplink noise for client k at iteration n .

Considering the impact of communication noise and allowing client updates to occur before server aggregation, we obtain

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho \mathbf{N}_k) \mathbf{w}_{k,n} + \rho \mathbf{N}_k \tilde{\mathbf{s}}_{k,n} \quad (4.11a)$$

$$\mathbf{w}_{n+1} = \frac{1}{K} \sum_{k=1}^K \tilde{\mathbf{w}}_{k,n+1} \quad (4.11b)$$

4.3 Resource-efficient FL over Noisy Channels

In this section, we introduce RERCE-Fed, our proposed FL algorithm that is both resource-efficient and robust to communication errors.

4.3.1 Random Scheduling

In (4.11), there is an implicit assumption that all clients are involved in every global model update iteration. However, in FL, clients often have constraints in their communication and computational capabilities. Thus, requiring participation of all clients in every global update round can result in considerable drawbacks, such as extended convergence times or excessive resource consumption. To tackle this issue, we allow the server to randomly select a subset of clients, indicated by \mathcal{S}_n , to take part in model aggregation during each iteration n . We assume that the size of this subset, $\mathcal{C} = |\mathcal{S}_n|$, remains unchanged throughout the FL process.

4.3.2 RERCE-Fed

In our proposed algorithm, during each global iteration n , the selected clients $k \in \mathcal{S}_n$ receive $\tilde{\mathbf{s}}_{k,n}$ from the server and update their models. The server subsequently receives $\tilde{\mathbf{w}}_{k,n+1}$ from these clients and aggregates them. Afterward, it broadcasts the most recent global update to a new set of selected clients in the next iteration. Clients that are not selected by the server in a specific round keep their most recent local update until their next selection. Hence, the recursions of the proposed RERCE-Fed algorithm are expressed as [106, 107]

$$\mathbf{w}_{k,n+1} = (\mathbf{I} - a_{k,n}\rho\mathbf{N}_k) \mathbf{w}_{k,n} + a_{k,n}\rho\mathbf{N}_k\tilde{\mathbf{s}}_{k,n} \quad (4.12a)$$

$$\mathbf{w}_{n+1} = \frac{1}{\mathcal{C}} \sum_{k=1}^K a_{k,n} \tilde{\mathbf{w}}_{k,n+1}, \quad (4.12b)$$

where $a_{k,n}$ is the indicator variable for random scheduling, with $a_{k,n} = 1$ when client k is selected by the server in iteration n (i.e., $k \in \mathcal{S}_n$) and $a_{k,n} = 0$ otherwise. We summarize RERCE-Fed in Algorithm 4.

4.3.3 RERCE-Fed with Continual Local Updates

In contrast to the traditional random scheduling method outlined in section 4.3.1, where clients that are not selected refrain from making local updates and their most recent model estimates are excluded from the global aggregation, we introduce a novel approach in which all clients, irrespective of their selection status, consistently revise their local model estimates during each iteration. This innovative strategy can improve overall performance without adding any extra communication overhead or significantly increasing the computational burden for either the clients or the server, as will be demonstrated later.

Algorithm 4: RERCE-Fed.

- 1 **Parameters:** penalty parameter ρ , number of clients K , set of clients \mathcal{S}
 - 2 **Initialization:** global model $\mathbf{w}_0 = \mathbf{w}_{-1} = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \tilde{\mathbf{w}}_k$
 - 3 **For** $n = 1, \dots$, *Until Convergence*
 - 4 The server randomly selects a subset \mathcal{S}_n of its clients and sends the aggregated global model \mathbf{s}_n to them.
 - 5 **Client Local Update:**
 - 6 **If** $k \in \mathcal{S}_n$
 - 7 Receive $\tilde{\mathbf{s}}_{k,n}$, a noisy version of \mathbf{s}_n , from the server.
 - 8 Update the local model as $\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho \mathbf{N}_k) \mathbf{w}_{k,n} + \rho \mathbf{N}_k \tilde{\mathbf{s}}_{k,n}$
 - 9 Send $\mathbf{w}_{k,n+1}$ to the server.
 - 10 **EndIf**
 - 11 **Aggregation at the Server:**
 - 12 The server receives $\tilde{\mathbf{w}}_{k,n+1}$, noisy versions of the locally updated models from the selected clients $k \in \mathcal{S}_n$ and aggregates them via
 - 13
$$\mathbf{w}_{n+1} = \frac{1}{C} \sum_{k \in \mathcal{S}_n} \tilde{\mathbf{w}}_{k,n+1}$$
 - 14
$$\mathbf{s}_{n+1} = 2\mathbf{w}_{n+1} - \mathbf{w}_n$$
 - 15 **EndFor**
-

To implement this method, clients retain the latest global model estimate they receive from the server, while the server maintains the most up-to-date local model estimates from all clients. As a result, clients are able to continually refine their local models using the most current global model estimate, and the server updates the global model by incorporating the latest local updates from every client, regardless of the random scheduling. When a client is chosen at iteration n , its most recent local model estimate is synchronized with the server, and the global model estimate provided by the server replaces the previous version held by the client.

Therefore, the recursions of the RERCE-Fed algorithm with continual local updates are given by [106, 108]

$$\begin{aligned} \mathbf{w}_{k,n+1} &= (\mathbf{I} - \rho \mathbf{N}_k) \mathbf{w}_{k,n} \\ &\quad + \rho \mathbf{N}_k [a_{k,n} \tilde{\mathbf{s}}_{k,n} + (1 - a_{k,n}) \tilde{\mathbf{s}}_{k,m}] \end{aligned} \quad (4.13a)$$

$$\mathbf{w}_{n+1} = \frac{1}{K} \sum_{k=1}^K [a_{k,n} \tilde{\mathbf{w}}_{k,n+1} + (1 - a_{k,n}) \tilde{\mathbf{w}}_{k,m}], \quad (4.13b)$$

Algorithm 5: RERCE-Fed with continual local updates.

-
- 1 **Parameters:** penalty parameters ρ , number of clients K , set of clients \mathcal{S}
 - 2 **Initialization:** global model $\mathbf{w}_0 = \mathbf{w}_{-1} = \mathbf{0}$, local models $\mathbf{w}_{k,0} = \tilde{\mathbf{w}}_k$
 - 3 **For** $n = 1, \dots$, *Until Convergence*
 - 4 The server randomly selects a subset \mathcal{S}_n of its clients and sends the aggregated global model \mathbf{s}_n to them.
 - 5 **Client Local Update:**
 - 6 **If** $k \in \mathcal{S}_n$
 - 7 Receive $\tilde{\mathbf{s}}_{k,n}$, a noisy version of \mathbf{s}_n , from the server.
 - 8 Store the latest global model $\tilde{\mathbf{s}}_{k,m} = \tilde{\mathbf{s}}_{k,n}$.
 - 9 Update the local model as $\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho \mathbf{N}_k) \mathbf{w}_{k,n} + \rho \mathbf{N}_k \tilde{\mathbf{s}}_{k,n}$
 - 10 Send $\mathbf{t}_{k,n+1} = 2\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}$ to the server.
 - 11 **Else**
 - 12 Update the local model as $\mathbf{w}_{k,n+1} = (\mathbf{I} - \rho \mathbf{N}_k) \mathbf{w}_{k,n} + \rho \mathbf{N}_k \tilde{\mathbf{s}}_{k,m}$
 - 13 **EndIf**
 - 14 **Aggregation at the Server:**
 - 15 The server receives $\tilde{\mathbf{t}}_{k,n+1}$, noisy versions of the locally updated models from the selected clients $k \in \mathcal{S}_n$ and aggregates them with $\tilde{\mathbf{t}}_{k,m}$, the stored local model estimates of the non-selected clients via
 - 16
$$\mathbf{s}_{n+1} = \frac{1}{K} \sum_{k=1}^K [a_{k,n} \tilde{\mathbf{t}}_{k,n+1} + (1 - a_{k,n}) \tilde{\mathbf{t}}_{k,m}]$$
 - 17 **EndFor**
-

where $\tilde{\mathbf{s}}_{k,m}$ denotes the most recent global model estimate received from the server and stored in client k . This estimate is utilized when the client is not selected by the server. Additionally, $\tilde{\mathbf{w}}_{k,m}$ represents the most recent local model estimate associated with client k , which is stored at the server and utilized during iterations when the client is not chosen through random scheduling. Defining $\mathbf{t}_{k,n+1} = 2\mathbf{w}_{k,n+1} - \mathbf{w}_{k,n}$, we can restate (4.13b) as

$$\mathbf{s}_{n+1} = \frac{1}{K} \sum_{k=1}^K [a_{k,n} \tilde{\mathbf{t}}_{k,n+1} + (1 - a_{k,n}) \tilde{\mathbf{t}}_{k,m}]. \quad (4.14)$$

We summarize RERCE-Fed with continual local updates in Algorithm 5.

4.4 Theoretical Results

In this section, we examine the theoretical performance of RERCE-Fed. We demonstrate that the iterates $\mathbf{w}_{k,n}$ converge in both mean and mean-square senses as $n \rightarrow \infty$, even when there are noisy communication links. Since \mathbf{w}_n denotes the average of client estimates $\mathbf{w}_{k,n}$, its convergence follows accordingly.

We define the extended optimal global model as $\mathbf{w}_e^* = \mathbf{1}_{2K} \otimes \mathbf{w}^*$ and the vector containing the client local model estimates as

$$\mathbf{w}_{e,n} = \text{col}\{\mathbf{w}_{1,n}, \dots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n-1}, \dots, \mathbf{w}_{K,n-1}\},$$

where $\mathbf{1}_{2K}$ is the $2K \times 1$ vector of all ones, \otimes is the Kronecker product, and $\text{col}\{\cdot\}$ denotes column-wise stacking.

Substituting (4.12b) into (4.12a), the global recursion of the proposed algorithm can be stated as

$$\mathbf{w}_{e,n+1} = \mathcal{A}_n \mathbf{w}_{e,n} + \boldsymbol{\zeta}_n + \boldsymbol{\eta}_n, \quad (4.15)$$

where

$$\mathcal{A}_n = \begin{bmatrix} \mathcal{A}_{n,1} & \mathcal{A}_{n,2} \\ \mathcal{A}_{n,3} & \mathcal{A}_{n,4} \end{bmatrix}, \quad (4.16)$$

and the extended noise vectors $\boldsymbol{\zeta}_n$ and $\boldsymbol{\eta}_n$ stack the vectors

$$a_{k,n} \rho \mathbf{N}_k \boldsymbol{\zeta}_{k,n} \quad (4.17a)$$

$$a_{k,n} \frac{\rho}{\mathcal{C}} \mathbf{N}_k \sum_{j=1}^K (2a_{j,n-1} \boldsymbol{\eta}_{j,n-1} - a_{j,n-2} \boldsymbol{\eta}_{j,n-2}) \quad (4.17b)$$

at their top halves, respectively, and zeros at their bottom halves. The value of $\mathcal{A}_n \in \mathbb{R}^{2LK \times 2LK}$ depends on the iteration number n as the server selects a random number of clients at each iteration. Its sub-matrices of size $LK \times LK$ are block matrices whose $L \times L$ blocks are calculated as

$$[\mathcal{A}_{n,1}]_{ii} = \mathbf{I} - a_{i,n} \rho \mathbf{N}_i + 2a_{i,n} a_{i,n-1} \frac{\rho}{\mathcal{C}} \mathbf{N}_i \quad (4.18a)$$

$$[\mathcal{A}_{n,1}]_{ij} = 2a_{i,n} a_{j,n-1} \frac{\rho}{\mathcal{C}} \mathbf{N}_i, \quad i \neq j \quad (4.18b)$$

$$[\mathcal{A}_{n,2}]_{ij} = -a_{i,n} a_{j,n-2} \frac{\rho}{\mathcal{C}} \mathbf{N}_i. \quad (4.18c)$$

To make the analysis tractable, we adopt the following assumptions:

A1: The communication noise vectors of both uplink and downlink, $\boldsymbol{\eta}_{k,n}$ and $\boldsymbol{\zeta}_{k,n} \forall k, n$, are independently and identically distributed. In addition, they are independent of each other and all other stochastic variables.

A2: The random scheduling variables, $a_{k,n} \forall k, n$, are independent and follow the same Bernoulli distribution with parameter $\bar{a} = \frac{C}{K}$, i.e., $a_{k,n} = 1$ with probability \bar{a} and $a_{k,n} = 0$ with probability $1 - \bar{a}$.

4.4.1 Mean Convergence

Proposition 4.1: RERCE-Fed algorithm is unbiased and converges in the mean sense such that

$$\lim_{n \rightarrow \infty} \mathbb{E}[\mathbf{w}_{e,n}] = \mathbf{w}_e^*. \quad (4.19)$$

Proof. See **P5**.

4.4.2 Mean-Square Convergence

We define the deviation vector as $\tilde{\mathbf{w}}_{e,n} = \mathbf{w}_{e,n} - \mathbf{w}_e^*$. Since \mathcal{A}_n is block right-stochastic, i.e., its block rows add up to the identity matrix, we have $\mathcal{A}_n \mathbf{w}_e^* = \mathbf{w}_e^*$. Therefore, by defining the weighted norm-square of $\tilde{\mathbf{w}}_{e,n}$ as $\|\tilde{\mathbf{w}}_{e,n}\|_{\Sigma}^2 = \tilde{\mathbf{w}}_{e,n}^T \Sigma \tilde{\mathbf{w}}_{e,n}$, where Σ is an arbitrary positive semi-definite matrix, we obtain the variance relation as

$$\mathbb{E}[\|\tilde{\mathbf{w}}_{e,n+1}\|_{\Sigma}^2] = \mathbb{E}[\|\tilde{\mathbf{w}}_{e,n}\|_{\Sigma'}^2] + \mathbb{E}[\zeta_n^T \Sigma \zeta_n] + \mathbb{E}[\boldsymbol{\eta}_n^T \Sigma \boldsymbol{\eta}_n], \quad (4.20)$$

where the cross terms vanish under assumption A1.

Subsequently, we define

$$\Sigma' = \mathbb{E}[\mathcal{A}_n^T \Sigma \mathcal{A}_n] \quad (4.21)$$

$$\boldsymbol{\sigma}' = \text{bvec}\{\Sigma'\} = \mathcal{Q} \boldsymbol{\sigma} \quad (4.22)$$

$$\boldsymbol{\sigma} = \text{bvec}\{\Sigma\} \quad (4.23)$$

$$\mathcal{Q} = \mathbb{E}[\mathcal{A}_n^T \otimes_b \mathcal{A}_n^T] \quad (4.24)$$

$$\mathbb{E}[\zeta_n^T \Sigma \zeta_n] = \boldsymbol{\phi}^T \boldsymbol{\sigma} \quad (4.25)$$

$$\mathbb{E}[\boldsymbol{\eta}_n^T \Sigma \boldsymbol{\eta}_n] = \boldsymbol{\varphi}^T \boldsymbol{\sigma} \quad (4.26)$$

$$\boldsymbol{\phi} = \text{bvec}\{\mathbb{E}[\zeta_n \zeta_n^T]\} \quad (4.27)$$

$$\boldsymbol{\varphi} = \text{bvec}\{\mathbb{E}[\boldsymbol{\eta}_n \boldsymbol{\eta}_n^T]\} \quad (4.28)$$

$$\mathbb{E}[\zeta_n \zeta_n^T] = \bar{a} \rho^2 \text{bdiag}\{\sigma_{\zeta_1}^2 \mathbf{N}_1^2, \dots, \sigma_{\zeta_K}^2 \mathbf{N}_K^2, \underbrace{\mathbf{O}, \dots, \mathbf{O}}_K\} \quad (4.29)$$

$$\mathbb{E}[\boldsymbol{\eta}_n \boldsymbol{\eta}_n^T] = \frac{5\rho^2}{K^2} \sum_{k=1}^K \sigma_{\eta_k}^2 \text{bdiag}\{\mathbf{N}_1^2, \dots, \mathbf{N}_K^2, \underbrace{\mathbf{O}, \dots, \mathbf{O}}_K\}, \quad (4.30)$$

where \otimes_b denotes the block Kronecker product, $\text{bvec}\{\cdot\}$ denotes the block vectorization operation, $\text{bdiag}\{\cdot\}$ denotes the block diagonalization, $\sigma_{\zeta_k}^2$ denotes the

variance of the downlink noise for client k and $\sigma_{\eta_k}^2$ denotes the variance of the uplink noise for client k . Note that in (4.30), $\mathbb{E}[a_{i,n}a_{j,n}] \ll 1 \forall i \neq j$, and can be neglected. Note that we evaluate \mathcal{Q} in Appendix A in **P5**.

Utilizing the above results, we can write the global recursion for the weighted mean-squared error (MSE) of RERCE-Fed as

$$\mathbb{E} \left[\|\tilde{\mathbf{w}}_{e,n+1}\|_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}^2 \right] = \mathbb{E} \left[\|\tilde{\mathbf{w}}_{e,1}\|_{\text{bvec}^{-1}\{\mathcal{Q}^n \boldsymbol{\sigma}\}}^2 \right] + \boldsymbol{\psi}^\top \sum_{i=0}^{n-1} \mathcal{Q}^i \boldsymbol{\sigma}, \quad (4.31)$$

where $\boldsymbol{\psi} = \boldsymbol{\phi} + \boldsymbol{\varphi}$ and $\text{bvec}^{-1}\{\cdot\}$ denotes the reverse operation of block vectorization.

Using the Jordan canonical form of \mathcal{Q} , we have

$$\mathcal{Q}^i = \mathbf{U} \mathcal{J}^i \mathbf{U}^{-1} = \sum_{\ell=1}^{4L^2K^2} \mu_\ell^i \mathbf{u}_\ell \mathbf{v}_\ell^\top, \quad (4.32)$$

where μ_ℓ , \mathbf{u}_ℓ and \mathbf{v}_ℓ denote the ℓ th eigenvalue of \mathcal{Q} and its corresponding right and left eigenvectors, respectively. In Appendix B in **P5**, we show that the spectral radius of \mathcal{Q} is one with the geometric and algebraic multiplicity of L^2 , i.e., $\mu_\ell = 1 \forall \ell \in \{1, \dots, L^2\}$.

Proposition 4.2: $\forall \ell \in \{1, \dots, L^2\}$, we have

$$\boldsymbol{\psi}^\top \mathbf{u}_\ell \mathbf{v}_\ell^\top \boldsymbol{\sigma} = 0. \quad (4.33)$$

Proof: See Appendix C in **P5**.

4.4.3 Steady-State Mean-Square Error

Setting $\boldsymbol{\sigma} = \text{bvec}\{\mathbf{I}_{2KL}\}$, letting $n \rightarrow \infty$ on both sides of (4.31), and using Proposition 4.2, we compute the steady-state MSE of RERCE-Fed, denoted by \mathcal{E} , as

$$\begin{aligned} \mathcal{E} &= \lim_{n \rightarrow \infty} \mathbb{E} \left[\tilde{\mathbf{w}}_{e,n}^\top \tilde{\mathbf{w}}_{e,n} \right] = \underbrace{\tilde{\mathbf{w}}_{e,1}^\top \boldsymbol{\Sigma}_\infty \tilde{\mathbf{w}}_{e,1}}_{\mathcal{E}_\nu} + \underbrace{\sum_{\ell=L^2+1}^{4L^2K^2} (1 - \mu_\ell)^{-1} \boldsymbol{\psi}^\top \mathbf{u}_\ell \mathbf{v}_\ell^\top \boldsymbol{\sigma}}_{\mathcal{E}_\psi}, \\ \boldsymbol{\Sigma}_\infty &= \text{bvec}^{-1} \left\{ \mathcal{Q}^\infty \boldsymbol{\sigma} \right\} = \text{bvec}^{-1} \left\{ \sum_{\ell=1}^{L^2} \mathbf{u}_\ell \mathbf{v}_\ell^\top \text{bvec}\{\mathbf{I}_{2KL}\} \right\}. \end{aligned} \quad (4.34)$$

Remark 4.1: The observation noise ν_k in (4.1) and the initial client estimates $\mathbf{w}_{k,0}$ influence the first component on the right-hand side of (4.34), denoted as \mathcal{E}_ν , leading to a noise floor. Furthermore, scheduling parameters such as the probability of client participation and the number of clients involved in each iteration, along with the penalty parameter, the total number of clients, the local dataset at each client, and the noise variance in uplink and downlink communications affect the second component on the right-hand side of (4.34), represented as \mathcal{E}_ψ . We will delve deeper into these factors through simulations in section 4.5.3.

4.5 Numerical Results

In this section, we conduct several numerical experiments to evaluate the performance of the proposed algorithm and validate our theoretical findings. We consider a federated network consisting of K clients. Each client has non-IID data $\{\mathbf{X}_k, \mathbf{y}_k\}$, where the entries of \mathbf{X}_k are drawn from a normal distribution $\mathcal{N}(\mu_k, \sigma_k^2)$ with $\mu_k \in \mathcal{U}(-0.5, 0.5)$ and $\sigma_k^2 \in \mathcal{U}(0.5, 1.5)$. The local data size for each client is randomly selected from a uniform distribution, i.e., $d_k \in \mathcal{U}(50, 90)$. We set the weight matrices at each client k to the inverse covariance matrix of \mathbf{y}_k , i.e., $\mathbf{W}_k = \mathbb{E}[(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])(\mathbf{y}_k - \mathbb{E}[\mathbf{y}_k])^\top]^{-1}$.

The local data is related as per (4.1) given the model parameter vector ω with its entries drawn from a normal distribution $\mathcal{N}(0, 1)$. The observation noise ν_k for each client is zero-mean IID Gaussian with variance $\sigma_{\nu_k}^2$. The additive noise in both the uplink and downlink channels is zero-mean IID white Gaussian with variances $\sigma_{\eta_k}^2$ and $\sigma_{\zeta_k}^2$, respectively. In our experiments, we set the penalty parameter as $\rho = 1$. In addition, the server randomly selects a subset of \mathcal{C} clients with uniform probability in each iteration.

We evaluate the algorithm performance on the client side via the normalized MSE (NMSE) defined at iteration n as

$$\frac{1}{K} \sum_{k=1}^K \frac{\|\mathbf{w}_{k,n} - \mathbf{w}^*\|_2^2}{\|\mathbf{w}^*\|_2^2}. \quad (4.35)$$

We average the NMSE learning curves over 100 independent trials to obtain our simulation results.

We outline the findings of our numerical experiments in three subsections. In the first subsection, we assess the effectiveness of the RERCE-Fed algorithm in comparison to its predecessors. The second subsection focuses on evaluating RERCE-Fed with continual local updates and comparing it to RERCE-Fed without continual local updates. Finally, in the last subsection, we validate our theoretical results by illustrating mean convergence and comparing theoretical predictions with

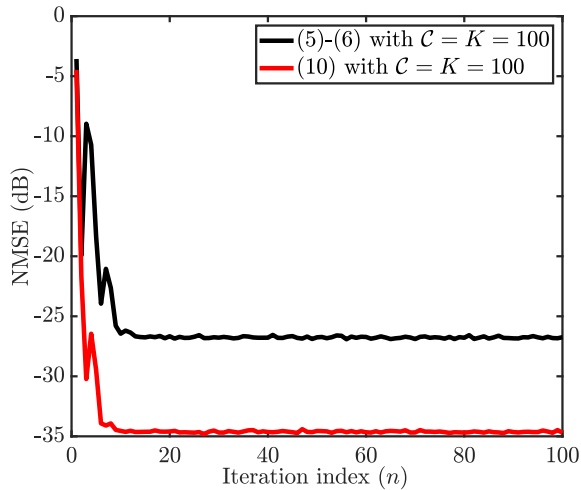


Figure 4.1: NMSE of (4.5)-(4.6) and (4.10) for $\mathcal{C} = K = 100$.

simulation results, confirming the accuracy of our theoretical expression for MSE of RERCE-Fed.

4.5.1 Performance of RERCE-Fed

In our first experiment, we simulate the algorithms described by (4.5)-(4.6) and (4.10) to solve the considered WLS problem. We run these simulations with $K = 100$ clients, model parameter vector size of $L = 128$, and link noise variance of $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 = 6.25 \times 10^{-4}$. All clients participate in the FL process, i.e., $\mathcal{C} = K = 100$. The corresponding learning curves are shown in Figure 4.1. We observe that (4.10) exhibits a 7dB improvement over (4.5)-(4.6) in the presence of noisy communication links when all clients are involved in each iteration of the FL process.

In our second experiment, we examine the performance of (4.5)-(4.6) and (4.10) under similar conditions as the first experiment, but with the server randomly selecting a subset of clients to participate in each iteration. We simulate (4.5)-(4.6) with $\mathcal{C} = 4$ and (4.10) with $\mathcal{C} \in \{4, 75, 90\}$. The corresponding learning curves are shown in Figure 4.2. Unlike the first experiment, Figure 4.2 illustrates that (4.10) fails to converge due to error accumulation, even when a majority of clients participate in every FL round, as observed for $\mathcal{C} \in \{75, 90\}$. Additionally, the performance of (4.5)-(4.6) degrades significantly when only a small subset of clients participate in each FL round. Consequently, both (4.5)-(4.6) and (4.10) exhibit an inability to cope with additive noise in communication links when the server selects only a subset of clients in each iteration.

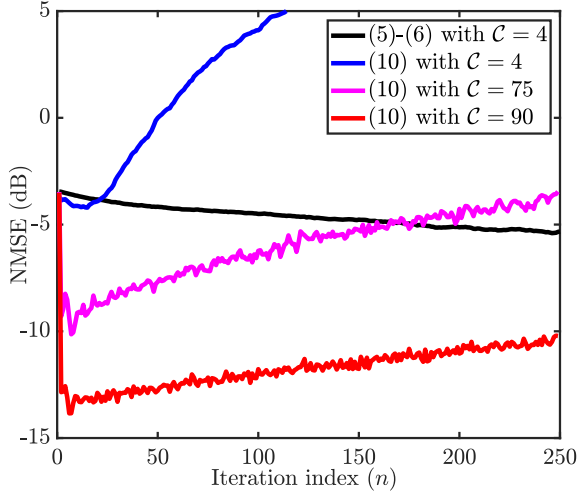


Figure 4.2: NMSE of (4.5)-(4.6) with $C = 4$ and (4.10) with $C \in \{4, 75, 90\}$.

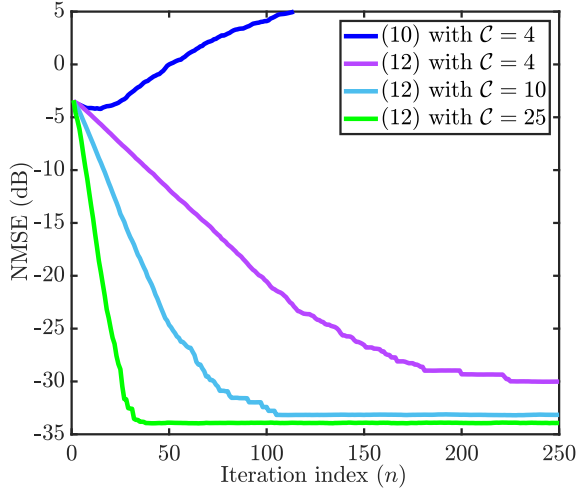


Figure 4.3: NMSE of (4.10) with $C = 4$ and RERCE-Fed (4.12) for different numbers of participating clients $C \in \{4, 10, 25\}$.

In our third experiment, we assess the performance of the proposed RERCE-Fed algorithm in the presence of link noise and random client scheduling by the server to enhance communication efficiency. We simulate RERCE-Fed with $K = 100$ clients, $L = 128$, link noise variance of $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 = 6.25 \times 10^{-4}$, and different numbers of participating clients $C \in \{4, 10, 25\}$. The corresponding learning curves are depicted in Figure 4.3. As shown in Figure 4.3, RERCE-Fed exhibits robustness against communication noise, even when only a subset of clients par-

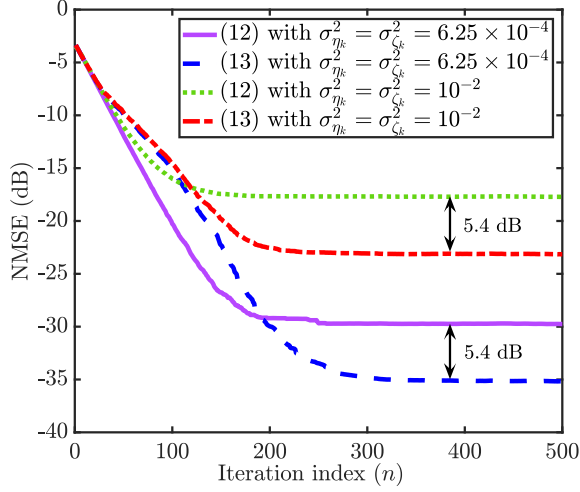


Figure 4.4: NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 4$ and different uplink and downlink noise variances $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

ticipate in each FL round, a contrast to the results observed with (4.5)-(4.6) and (4.10) in the second experiment. Another important observation from Figure 4.3 is the trade-off between the number of participating clients \mathcal{C} and the performance and convergence rate of RERCE-Fed. Specifically, increasing the number of participating clients leads to faster convergence and lower NMSE. However, this benefit diminishes once $\mathcal{C} \geq 10$, as the performance of RERCE-Fed with $\mathcal{C} \geq 10$ approaches that of the scenario where all clients participate in each FL round. This implies that RERCE-Fed can achieve accurate model parameter estimation while making more efficient use of available communication resources, even in the presence of noisy communication links.

4.5.2 Performance of RERCE-Fed with Continual Local Updates

In our fourth experiment, we compare the performance of RERCE-Fed with and without continual local updates by plotting their corresponding learning curves given different numbers of participating clients and link noise variances. We present the results in Figures 4.4-4.6. The number of clients selected at each iteration is $\mathcal{C} \in \{4, 10, 25\}$, and the link noise variances are $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$. From these figures, we observe that RERCE-Fed with continual local updates consistently exhibits robustness against communication noise, even when only a small subset of clients participate in every FL round. In addition, RERCE-Fed with continual local updates significantly outperforms its counterpart

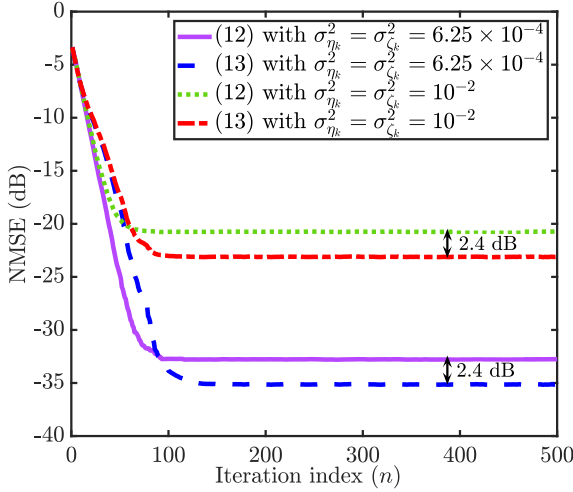


Figure 4.5: NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 10$ and different uplink and downlink noise variances $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

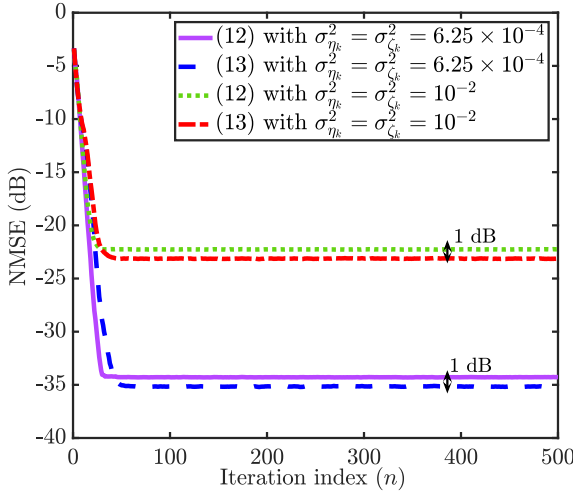


Figure 4.6: NMSE of RERCE-Fed (4.12) and RERCE-Fed with continual local updates (4.14) for $\mathcal{C} = 25$ and different uplink and downlink noise variances $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$.

without continual local updates in all considered scenarios. These results indicate that allowing clients to continually update their local models, even when not selected by the server, leads to a substantial improvement in steady-state NMSE without adversely impacting the convergence rate. Additionally, as anticipated, increasing

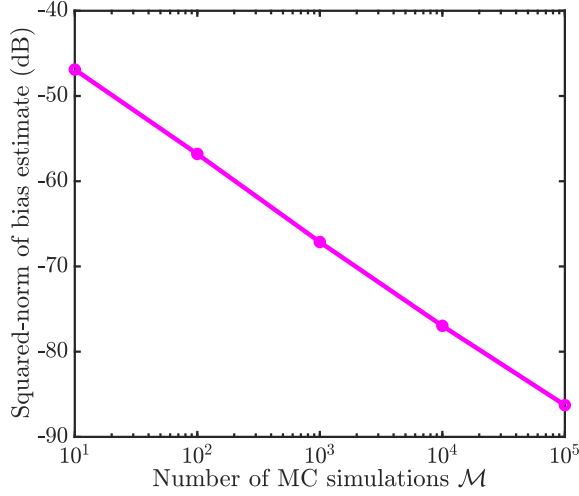


Figure 4.7: Squared-norm of bias estimate of RERCE-Fed (4.12) with $K = 6$, $L = 6$, $\mathcal{C} = 3$, and $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 = 10^{-4}$ for different numbers of MC runs $\mathcal{M} \in \{10, 10^2, 10^3, 10^4, 10^5\}$.

the link noise variance results in performance degradation.

4.5.3 Comparisons of Theory and Experiment

In our fifth experiment, we demonstrate the mean convergence of RERCE-Fed. We simulate RERCE-Fed with $K = 6$ clients, $L = 6$, and link noise variances $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 = 10^{-4}$. In addition, the server randomly selects $\mathcal{C} = 3$ clients in each iteration. In Figure 4.7, we plot the squared ℓ_2 -norm of the global model parameter bias estimated for different numbers of Monte Carlo (MC) runs, denoted by \mathcal{M} , specifically $\frac{1}{L} \left\| \frac{1}{\mathcal{M}} \sum_{i=1}^{\mathcal{M}} \mathbf{w}_n^{(i)} - \mathbf{w}^* \right\|_2^2$, where $\mathbf{w}_n^{(i)}$ is the global model parameter estimate of the i th MC run. We can observe from Figure 4.7 that the squared-norm of the multivariate bias estimate decreases log-linearly with the number of MC runs, indicating unbiasedness.

In our sixth experiment, we validate the accuracy of our theoretical expression for the steady-state MSE of RERCE-Fed in (4.34) and explore the impact of varying uplink and downlink noise variances, $\sigma_{\eta_k}^2$ and $\sigma_{\zeta_k}^2$, on performance. We simulate RERCE-Fed with $K = 6$ clients, $L = 6$ parameters, and different values for uplink and downlink noise variances. The server randomly selects $\mathcal{C} = 3$ clients in each iteration. We present the theoretical predictions of steady-state MSE using (4.34) alongside the corresponding experimental values in Figures 4.8-4.9 as functions of $\sigma_{\eta_k}^2$, $\sigma_{\zeta_k}^2$, and \mathcal{C} . The results show a close alignment between theory and experiment. Furthermore, we observe an upward trend in the steady-state NMSE

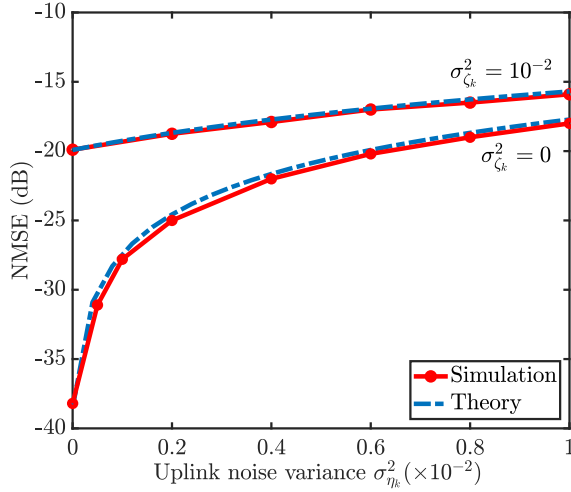


Figure 4.8: NMSE of RERCE-Fed (4.12) with $\mathcal{C} = 3$ for different uplink noise variances.

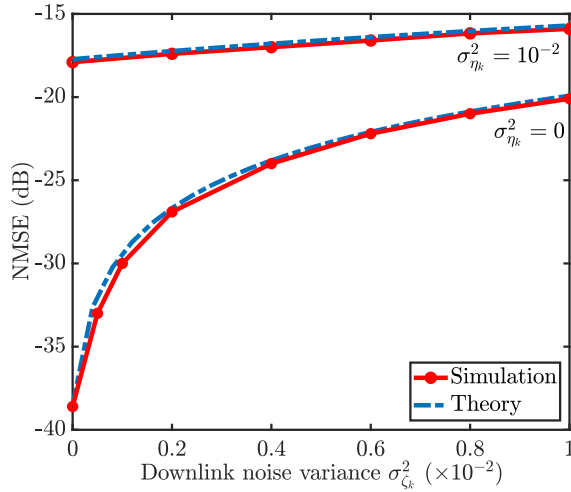


Figure 4.9: NMSE of RERCE-Fed (4.12) with $\mathcal{C} = 3$ for different downlink noise variances.

as either uplink or downlink noise variance increases.

The uplink and downlink noise variances exhibit distinct effects depending on the number of participating clients \mathcal{C} . While the error induced by uplink noise remains constant, the impact of downlink noise intensifies with an increasing number of participating clients. This observation is consistent with the intuition that averaging the model parameter estimates at the server can mitigate the adverse effect

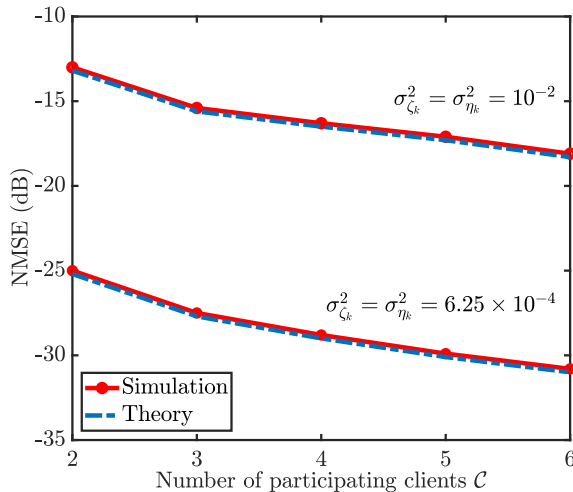


Figure 4.10: NMSE of RERCE-Fed (4.12) for different numbers of participating clients $C \in \{2, 3, 4, 5, 6\}$ and different link noise variances.

of uplink noise on the performance of RERCE-Fed. However, changing C also affects \mathcal{Q} . To obtain a better understanding, we investigate the impact of C on the performance of RERCE-Fed in our final experiment, where we consider $K = 6$, $L = 6$, $\sigma_{\eta_k}^2 = \sigma_{\zeta_k}^2 \in \{6.25 \times 10^{-4}, 10^{-2}\}$, and $C \in \{2, 3, 4, 5, 6\}$. The results presented in Figure 4.10 illustrate that increasing the number of participating clients improves the performance of RERCE-Fed. However, this improvement comes at the cost of higher resource utilization on the client side, which can be limiting in real-world FL scenarios.

4.6 Summary

We proposed RERCE-Fed, an FL algorithm designed to effectively reduce communication load while maintaining robustness against additive communication noise or errors. By employing ADMM to solve the WLS problem, we introduced a new local model update at the clients. This innovative solution mitigates the effects of communication noise without imposing additional computational burden on clients. Furthermore, we improved the communication efficiency by randomly selecting a subset of clients to participate in each learning round. To further optimize performance, we enabled non-selected clients to continue with their local updates, resulting in a modified version of RERCE-Fed. Our theoretical analysis confirmed the convergence of RERCE-Fed in both mean and mean-square sense, even with random client scheduling and communication over noisy communication links. In addition, we derived a closed-form expression for the steady-state

MSE of the RERCE-Fed algorithm. Comprehensive numerical analysis substantiated our findings and confirmed the accuracy of our theoretical predictions.

Chapter 5

Privacy-Preserving Nonnegative Matrix Factorization

This chapter, which outlines the findings of publication **P6**, introduces a privacy-preserving algorithm designed for fully-distributed nonnegative matrix factorization (NMF). This algorithm aims to decompose a large data matrix into left and right matrix factors in a distributed manner while ensuring the privacy of each agent's local data. It enables collaborative estimation of the left matrix factor among agents and allows them to estimate their respective right factors without compromising raw data privacy. To achieve data privacy, the algorithm employs the Paillier cryptosystem, a probabilistic asymmetric algorithm for public-key cryptography that supports computations on encrypted data without decryption, to secure information exchanges between neighboring agents. Simulation results based on synthetic and real-world datasets highlight the efficacy of the proposed algorithm in achieving privacy-preserving distributed NMF over ad-hoc networks.

5.1 Motivation

Nonnegative Matrix Factorization (NMF) [109–113] is a powerful technique in linear algebra and machine learning. NMF is a specific case of constrained low-rank matrix approximation [114] and a linear dimensionality reduction (LDR) technique aimed at representing nonnegative data more compactly through nonnegative factors. Formally, given a matrix \mathbf{Z} with dimensions $N \times M$, NMF aims to find two matrices \mathbf{X} with dimensions $N \times K$ and \mathbf{Y} with dimensions $K \times M$ such that $\mathbf{Z} \approx \mathbf{XY}$, where all elements in \mathbf{Z} , \mathbf{X} , and \mathbf{Y} are nonnegative. The dimension K , typically chosen to be smaller than both N and M , represents the number of latent features or components in the factorization. This decomposition allows for a

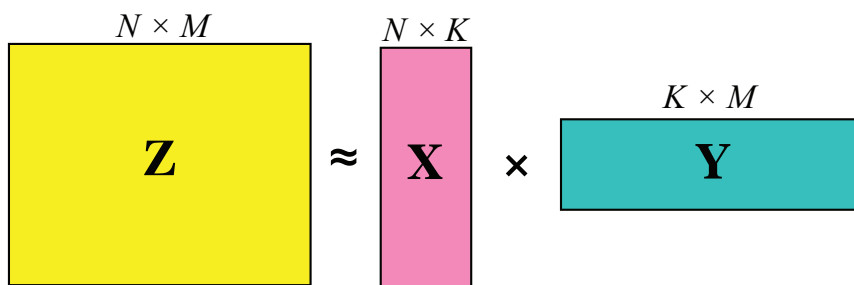


Figure 5.1: Illustration of nonnegative matrix factorization.

parts-based representation of the data, where each column of X can be interpreted as a basis vector and each column of Y as the corresponding encoding. The interpretation of NMF results is often intuitive and domain-specific. For instance, in image processing, the columns of X might represent distinct facial features such as eyes, nose, and mouth when applied to a dataset of face images, while the corresponding columns in Y would indicate the strength of each feature's presence in a given image. In text analysis, the matrix X might represent topics or themes, with the matrix Y indicating the relevance of each topic to individual documents. Figure 5.1 provides an illustration of NMF.

NMF was originally introduced as positive matrix factorization in [115] and has gained significant research interest, particularly after being popularized by [116]. It has found widespread applications in various fields such as signal and image processing, data mining and analytics, machine learning, and federated learning. Examples include air emission control [115], visual object recognition [117], video background-foreground separation [118], spectral unmixing [119], text mining [120], blind source separation [121], clustering [122], computational biology [123], music analysis [124], molecular pattern discovery [111], efficient implementation of deep neural networks [125], and detecting malware activities [126]. Its popularity stems from its utility in identifying and extracting meaningful features from data in addition to serving as a powerful LDR technique.

NMF has also proven to be a valuable tool in recommender systems and matrix completion tasks in recent years, e.g., the Netflix challenge [127]. Recommender systems often work with user-item interaction matrices, where rows represent users, columns represent items, and entries indicate preferences or ratings. NMF can decompose this matrix into user and item factor matrices, effectively capturing latent features that explain user preferences and item characteristics. This factorization allows for the prediction of missing entries, enabling person-

alized recommendations. For example, in a movie recommendation system, the matrix \mathbf{X} might represent user preferences for different movie genres or styles, while the matrix \mathbf{Y} could represent how strongly each movie aligns with these latent features. The product of these matrices can then be used to predict a user’s potential rating for a movie they haven’t seen yet [127]. In matrix completion tasks, where the goal is to fill in missing entries in a partially observed matrix, NMF can leverage the low-rank structure and nonnegativity constraints to provide meaningful estimates for the unknown values. This approach has been successfully applied in various domains, including collaborative filtering [128] and image inpainting [129].

This chapter presents a privacy-preserving distributed NMF (PPDNMF) algorithm for scenarios where data is spread among agents in an ad-hoc network, with each agent holding part of the data matrix’s columns. Our goal is to perform NMF on the entire dataset in a secure, decentralized manner. Agents collaborate to estimate the left and right factors, sharing information only with immediate neighbors over secure links, while keeping their local data private. We use the block coordinate-descent (BCD) method and alternating direction method of multipliers (ADMM) to develop the algorithm and integrate the Paillier cryptosystem for privacy preservation. Performance evaluations through simulations with synthetic and real data show that our algorithm achieves results comparable to centralized methods.

5.2 Proposed Method

As mentioned earlier, the objective of NMF is to approximate a data matrix $\mathbf{Z} \in \mathbb{R}^{N \times M}$ consisting of nonnegative entries using the product of left and right factor matrices, both with nonnegative entries. That is, $\mathbf{Z} = \mathbf{X}\mathbf{Y}$ where $\mathbf{X} \in \mathbb{R}^{N \times L}$ and $\mathbf{Y} \in \mathbb{R}^{L \times M}$, typically with $L \leq \min(N, M)$. This approximation represents the N -dimensional datapoints (columns of the data matrix) within a L -dimensional linear subspace spanned by the columns of the left factor, whose coordinates are given by the columns of the right factor. The nonnegativity constraint on the factors induces sparsity, further enhancing the compactness of the representation. Moreover, in many applications, the factors’ nonnegativity is essential to their physical plausibility and intuitive interpretability.

We utilize the least-squares criterion, which is appropriate when the perturbation in the data matrix \mathbf{Z} can be modeled as a Gaussian process. Therefore, the NMF problem can be formulated as

$$\begin{aligned} \min_{\mathbf{X}, \mathbf{Y}} \quad & \frac{1}{2} \|\mathbf{Z} - \mathbf{X}\mathbf{Y}\|_{\text{F}}^2 \\ \text{s. t.} \quad & \mathbf{X} \geq 0, \mathbf{Y} \geq 0, \end{aligned} \tag{5.1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

We consider the scenario where \mathbf{Z} is distributed over a network with K agents such that we have $\mathbf{Z} = [\mathbf{Z}_1, \dots, \mathbf{Z}_K]$ and consequently $\mathbf{Y} = [\mathbf{Y}_1, \dots, \mathbf{Y}_K]$ with $\mathbf{Y}_k \in \mathbb{R}^{L \times M_k}$ and we have $\sum_{k=1}^K M_k = M$. Therefore, we rewrite (5.1) as

$$\begin{aligned} \min_{\mathbf{X}, \{\mathbf{Y}_k\}} \quad & \frac{1}{2} \sum_{k=1}^K \|\mathbf{Z}_k - \mathbf{X}\mathbf{Y}_k\|_F^2 \\ \text{s. t.} \quad & \mathbf{X} \geq 0, \mathbf{Y}_k \geq 0. \end{aligned} \quad (5.2)$$

In a fully distributed approach, every agent, indexed by k , aims to estimate \mathbf{X} and its own \mathbf{Y}_k using its local data \mathbf{Z}_k and by exchanging information solely with its immediate neighbors through single-hop communication. To this end, we utilize the BCD algorithm and iteratively solve two optimization subproblems for \mathbf{X} and \mathbf{Y} . That is, we repeat the following alternating minimizations until convergence is achieved:

$$\begin{aligned} \mathbf{X}_n = \min_{\mathbf{X}} \quad & \frac{1}{2} \sum_{k=1}^K \|\mathbf{Z}_k - \mathbf{X}\mathbf{Y}_{k,n-1}\|_F^2 \\ \text{s. t.} \quad & \mathbf{X} \geq 0, \end{aligned} \quad (5.3)$$

$$\begin{aligned} \mathbf{Y}_{k,n} = \min_{\mathbf{Y}_k} \quad & \frac{1}{2} \|\mathbf{Z}_k - \mathbf{X}_n \mathbf{Y}_k\|_F^2, \forall k \in \{1, \dots, K\} \\ \text{s. t.} \quad & \mathbf{Y}_k \geq 0. \end{aligned} \quad (5.4)$$

The subscript n denotes the estimate of its respective parameter at the n th BCD iteration.

The solution of (5.4) can be localized straightforwardly, provided that each agent has access to the estimate \mathbf{X}_n . To solve (5.3) in a fully distributed manner, we introduce the variable \mathbf{X}_k at each agent k as a local copy of \mathbf{X} and enforce it to be equal to those of the agents within the immediate neighborhood of agent k , thereby achieving consensus across the network. Thus, we reformulate (5.3) into the following equivalent form

$$\begin{aligned} \mathbf{X}_{k,n} = \min_{\mathbf{X}_k} \quad & \frac{1}{2} \|\mathbf{Z}_k - \mathbf{X}_k \mathbf{Y}_{k,n-1}\|_F^2 + \iota(\mathbf{X}_k) \\ \text{s. t.} \quad & \mathbf{X}_k = \mathbf{X}_\ell \quad \forall \ell \in \mathcal{N}_k, \forall k \in \{1, \dots, K\} \end{aligned} \quad (5.5)$$

where $\iota(\cdot)$ denotes the indicator function accounting for the nonnegativity constraint and \mathcal{N}_k denotes the set of neighbors of agent k with cardinality $d_k = |\mathcal{N}_k|$.

Subsequently, we decompose and decouple the optimization problems at the agents by introducing the auxiliary variables $\mathbf{U}_k, \mathbf{S}_{k,\ell} \in \mathbb{R}^{N \times L}$ and rewriting the optimization in (5.5) as

$$\begin{aligned}
\min_{\mathbf{X}_k, \mathbf{U}_k, \mathbf{S}_{k,\ell}} \quad & \frac{1}{2} \|\mathbf{Z}_k - \mathbf{U}_k \mathbf{Y}_{k,n-1}\|_{\mathbb{F}}^2 + \iota(\mathbf{X}_k) \\
\text{s. t.} \quad & \mathbf{U}_k = \mathbf{X}_k \\
& \mathbf{S}_{k,\ell} = \mathbf{U}_k \quad \forall \ell \in \mathcal{N}_k, \forall k \in \{1, \dots, K\} \\
& \mathbf{S}_{\ell,k} = \mathbf{S}_{k,\ell}.
\end{aligned} \tag{5.6}$$

We can express the corresponding aggregate augmented Lagrangian function as

$$\begin{aligned}
& \mathcal{L}(\{\mathbf{X}_k\}, \{\mathbf{U}_k\}, \{\mathbf{S}_{k,\ell}\}, \{\mathbf{P}_k\}, \{\mathbf{Q}_{k,\ell}\}) \\
& = \frac{1}{2} \sum_{k=1}^K \|\mathbf{Z}_k - \mathbf{U}_k \mathbf{Y}_{k,n-1}\|_{\mathbb{F}}^2 + \sum_{k=1}^K \iota(\mathbf{X}_k) \\
& + \frac{\mu}{2} \sum_{k=1}^K \|\mathbf{X}_k - \mathbf{U}_k - \mathbf{P}_k\|_{\mathbb{F}}^2 \\
& + \frac{1}{2} \sum_{k=1}^K \sum_{\ell \in \mathcal{N}_k} \rho_{k,\ell} \|\mathbf{U}_k - \mathbf{S}_{k,\ell} - \mathbf{Q}_{k,\ell}\|_{\mathbb{F}}^2,
\end{aligned} \tag{5.7}$$

where μ and $\rho_{k,\ell}$ are penalty parameters and $\mathbf{P}_k, \mathbf{Q}_{k,\ell} \in \mathbb{R}^{N \times L}$ are scaled Lagrange multipliers. We maintain μ consistent across all agents and iterations. However, we allow each $\rho_{k,\ell}$, unique to the edge connecting agents k and ℓ , to vary over iterations [130]. Additionally, we consider $\rho_{k,\ell} = \rho_{\ell,k} \forall k, \ell$.

5.2.1 Estimating the Left Factor

Minimizing (5.7) using ADMM, which leads to the elimination of the auxiliary variables $\{\mathbf{S}_{k,\ell}\}$, yields the following iterations at each agent i as [131]

$$\mathbf{X}_{k,n,m} = \Pi_{\geq 0} \{\mathbf{U}_{k,m-1} + \mathbf{P}_{k,m-1}\} \quad (5.8)$$

$$\begin{aligned} \mathbf{U}_{k,m} = & \left[\mathbf{Z}_k \mathbf{Y}_{k,n-1}^\top + \mu (\mathbf{X}_{k,n,m} - \mathbf{P}_{k,m-1}) \right. \\ & \left. + \rho_{k,m} (\mathbf{U}_{k,m-1} + 2\mathbf{Q}_{k,m-1} - \mathbf{Q}_{k,m-2}) \right] \\ & \times \left[\mathbf{Y}_{k,n-1} \mathbf{Y}_{k,n-1}^\top + (\mu + \rho_{k,m}) \mathbf{I} \right]^{-1} \end{aligned} \quad (5.9)$$

$$\mathbf{P}_{k,m} = \mathbf{P}_{k,m-1} - (\mathbf{X}_{k,n,m} - \mathbf{U}_{k,m}) \quad (5.10)$$

$$\mathbf{Q}_{k,m} = \mathbf{Q}_{k,m-1} + \sum_{\ell \in \mathcal{N}_k} \rho_{k,\ell,m} (\mathbf{U}_{\ell,m} - \mathbf{U}_{k,m}). \quad (5.11)$$

Here, the subscript m denotes the m th ADMM iteration index, $\Pi_{\geq 0}$ represents the projection onto the nonnegative orthant, and $(\cdot)^\top$ stands for matrix transpose. In addition, we define $\rho_{k,m} = \sum_{\ell \in \mathcal{N}_k} \rho_{k,\ell,m}$ and $\mathbf{Q}_{k,m} = \sum_{\ell \in \mathcal{N}_k} \rho_{k,\ell,m} \mathbf{Q}_{k,\ell,m}$. These ADMM iterations can be executed in a fully distributed manner, relying solely on locally available information and single-hop communications. Upon convergences of the algorithm, we utilize the latest estimates $\mathbf{X}_{k,n,m}$ for optimizing \mathbf{Y}_k in the subsequent BCD iteration, i.e., $\mathbf{X}_{k,n} \leftarrow \mathbf{X}_{k,n,m}$. Note that we enforce the nonnegativity constraint and consensus simultaneously.

5.2.2 Estimating the Right Factor

Similarly, we can employ ADMM to iteratively solve (5.4) via [131]

$$\mathbf{Y}_{k,n,r} = \Pi_{\geq 0} \{\mathbf{V}_{k,r-1} + \mathbf{R}_{k,r-1}\} \quad (5.12)$$

$$\begin{aligned} \mathbf{V}_{k,r} = & \left(\mathbf{X}_{k,n}^\top \mathbf{X}_{k,n} + \eta \mathbf{I} \right)^{-1} \\ & \times \left[\mathbf{X}_{k,n}^\top \mathbf{Z}_k + \eta (\mathbf{Y}_{k,n,r} - \mathbf{R}_{k,r-1}) \right] \end{aligned} \quad (5.13)$$

$$\mathbf{R}_{k,r} = \mathbf{R}_{k,r-1} - (\mathbf{Y}_{k,n,r} - \mathbf{V}_{k,r}). \quad (5.14)$$

Here, the superscript r represents the r th ADMM iteration index and η is the penalty parameter. Once convergence is attained, we utilize the latest estimates $\mathbf{Y}_{k,n,r}$ to update \mathbf{X}_k estimates in the subsequent BCD iteration, i.e., $\mathbf{Y}_{k,n} \leftarrow \mathbf{Y}_{k,n,r}$.

Note that, we employ warm start in both ADMM algorithms for estimating the left and right factors. At the onset of each BCD iteration, we initialize both ADMM inner iterations using the most recent estimates from the preceding iterations.

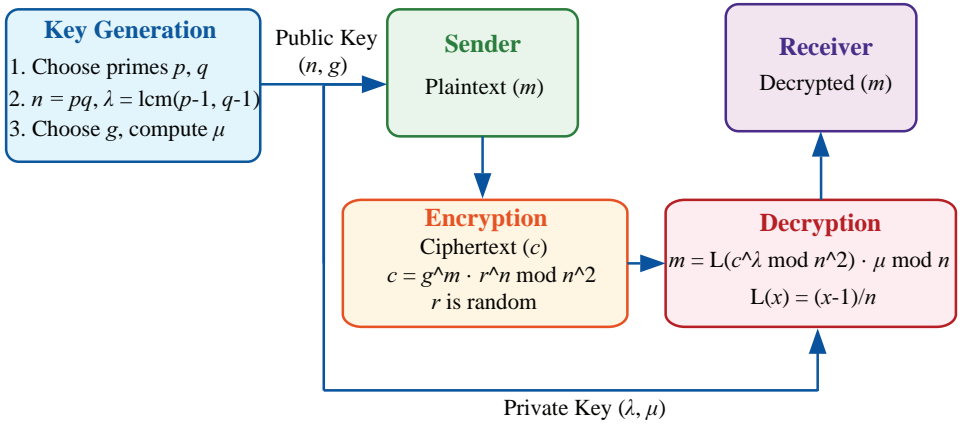


Figure 5.2: The Paillier cryptosystem flow diagram.

5.2.3 Convergence Analysis

Proposition 5.1: The ADMM iterations (5.8)-(5.11) and (5.12)-(5.14) converge using the approach proposed in [48, 132, 133].

Proof. See **P6**.

5.3 Privacy-Preserving Distributed NMF

In this section, we provide an overview of the Paillier cryptosystem, which we employ to enhance the privacy of the distributed NMF algorithm developed in section 5.2. Subsequently, we introduce our proposed privacy-preserving distributed NMF (PPDNMF) algorithm.

5.3.1 Paillier Cryptosystem

The Paillier cryptosystem, introduced by Pascal Paillier in 1999 [134], is a probabilistic asymmetric algorithm for public key cryptography. It is based on the mathematical problem of computing n th residue classes, which is believed to be computationally difficult. The system is notable for its homomorphic properties, particularly its additive homomorphism [130, 135, 136], which allows for certain operations to be performed on encrypted data without decrypting it first, i.e., $\mathcal{E}(m_3(m_1 + m_2)) = (\mathcal{E}(m_1) \mathcal{E}(m_2))^{m_3}$. The randomness in encryption ensures semantic security and makes it resistant to attacks. The Paillier cryptosystem's unique properties make it particularly useful in various applications, including e-voting systems and privacy-preserving data mining.

The Paillier cryptosystem operates as illustrated in Figure 5.2 as follows:

- Key Generation:
 1. Choose two large prime numbers p and q randomly.
 2. Confirm that $\gcd(pq, (p-1)(q-1)) = 1$, where $\gcd(\cdot, \cdot)$ is the greatest common divisor of its arguments.
 3. Compute $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$, where $\text{lcm}(\cdot, \cdot)$ is the least common multiple of its arguments.
 4. Select a random integer g , where $g \in \mathbb{Z}_{n^2}^*$, i.e., integers between 1 and n^2 .
 5. Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$, where $L(x) = \frac{x-1}{n}$.
 6. The public key is (n, g) and the private key is (λ, μ) .
- Encrypt a message m where $m < n$:
 1. Select a random r where $r < n$.
 2. Compute the ciphertext as $c = g^m \cdot r^n \bmod n^2$.
- Decrypt a ciphertext c as $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$.

5.3.2 Privacy Preservation

The only update equation among (5.8)-(5.14) that relies on information received from neighboring agents is (5.11). To protect the privacy of agents at this step, we adopt a similar approach to [130] and enable the agents to encrypt all messages communicated with their neighbors. To this end, we decompose each edge-specific penalty parameter as $\rho_{k,\ell,m} = g_m^{k \rightarrow \ell} g_m^{\ell \rightarrow k}$ where $g_m^{k \rightarrow \ell}$ and $g_m^{\ell \rightarrow k}$ are exclusively known to agents k and ℓ , respectively. In addition, we implement a secure data exchange procedure as outlined in lines 9-14 of Algorithm 6, which provides a summary of the proposed PPDNMF algorithm. In Figure 5.3, we summarize the information exchange part of the PPDNMF algorithm.

Consequently, note the following:

- Data exchanged between agents k and ℓ is encrypted, rendering it inaccessible to other agents or eavesdroppers, even if intercepted.
- The parameter $g_m^{k \rightarrow \ell}$ is unique to each edge and iteration. Therefore, an agent cannot infer the private information $U_{\ell,m}$ of any of its neighbors by decrypting the messages it receives from them, as each neighbor ℓ uses its unique $g_m^{\ell \rightarrow k}$ in its encrypted message to agent k .

Algorithm 6: PPDNMF.

```

1 Initialization:  $\mathbf{X}_{k,0} = \mathbf{U}_{k,0} = \mathbf{1}_{N \times L}$ ,  $\mathbf{P}_{k,0} = \mathbf{Q}_{k,0} = \mathbf{Q}_{k,-1} = \mathbf{O}_{N \times L}$ ,
    $\mathbf{Y}_{k,0} = \mathbf{V}_{k,0} = \mathbf{R}_{k,0} = \mathbf{O}_{L \times M_k}$ 
2 For  $n = 1, \dots$ , Until Convergence
3   For  $m = 1, \dots$ , Until Convergence
4      $\mathbf{X}_{k,n,m} = \Pi_{\geq 0} \{ \mathbf{U}_{k,m-1} + \mathbf{P}_{k,m-1} \}$ 
5      $\mathbf{U}_{k,m} = \left[ \mathbf{Z}_k \mathbf{Y}_{k,n-1}^\top + \mu (\mathbf{X}_{k,n,m} - \mathbf{P}_{k,m-1}) \right.$ 
6        $\left. + \rho_{k,m} (\mathbf{U}_{k,m-1} + 2\mathbf{Q}_{k,m-1} - \mathbf{Q}_{k,m-2}) \right]$ 
7        $\times \left[ \mathbf{Y}_{k,n-1} \mathbf{Y}_{k,n-1}^\top + (\mu + \rho_{k,m}) \mathbf{I} \right]^{-1}$ 
8      $\mathbf{P}_{k,m} = \mathbf{P}_{k,m-1} - (\mathbf{X}_{k,n,m} - \mathbf{U}_{k,m})$ 
9     Encrypt  $-\mathbf{U}_{k,m}$  using the public key  $\kappa_{pk}$  as  $\mathcal{E}_k(-\mathbf{U}_{k,m})$ 
10    Send  $\mathcal{E}_k(-\mathbf{U}_{k,m})$  and  $\kappa_{pk}$  to all neighbors in  $\mathcal{N}_k$ 
11    For  $\ell \in \mathcal{N}_k$ 
12      Encrypt  $\mathbf{U}_{\ell,m}$  using  $\kappa_{pk}$  as  $\mathcal{E}_k(\mathbf{U}_{\ell,m})$ 
13      Send  $[\mathcal{E}_k(\mathbf{U}_{\ell,m}) \mathcal{E}_k(-\mathbf{U}_{k,m})]^{g_m^{\ell \rightarrow k}}$  to agent  $k$ 
14    EndFor
15    Decrypt messages received from neighbors using the private key  $\kappa_{sk}$ 
16    Multiply the received messages by  $g_m^{k \rightarrow \ell}$ 
17     $\mathbf{Q}_{k,m} = \mathbf{Q}_{k,m-1} + \sum_{\ell \in \mathcal{N}_k} g_m^{k \rightarrow \ell} g_m^{\ell \rightarrow k} (\mathbf{U}_{\ell,m} - \mathbf{U}_{k,m})$ .
18  EndFor
19   $\mathbf{X}_{k,n} \leftarrow \mathbf{X}_{k,n,m}$ ,  $\mathbf{U}_{k,0} \leftarrow \mathbf{U}_{k,m}$ ,  $\mathbf{P}_{k,0} \leftarrow \mathbf{P}_{k,m}$ ,
20   $\mathbf{Q}_{k,0} \leftarrow \mathbf{Q}_{k,m}$ ,  $\mathbf{Q}_{k,-1} \leftarrow \mathbf{Q}_{k,m-1}$ 
21  For  $r = 1, \dots$ , Until Convergence
22     $\mathbf{Y}_{k,n,r} = \Pi_{\geq 0} \{ \mathbf{V}_{k,r-1} + \mathbf{R}_{k,r-1} \}$ 
23     $\mathbf{V}_{k,r} = \left( \mathbf{X}_{k,n}^\top \mathbf{X}_{k,n} + \eta \mathbf{I} \right)^{-1} \left[ \mathbf{X}_{k,n}^\top \mathbf{Z}_k + \eta (\mathbf{Y}_{k,n,r} - \mathbf{R}_{k,r-1}) \right]$ 
24     $\mathbf{R}_{k,r} = \mathbf{R}_{k,r-1} - (\mathbf{Y}_{k,n,r} - \mathbf{V}_{k,r})$ 
25  EndFor
26   $\mathbf{Y}_{k,n} \leftarrow \mathbf{Y}_{k,n,r}$ ,  $\mathbf{V}_{k,0} \leftarrow \mathbf{V}_{k,r}$ ,  $\mathbf{R}_{k,0} \leftarrow \mathbf{R}_{k,r}$ 
27 EndFor

```

- The Paillier cryptosystem is intended for encrypting scalar unsigned integers. To encrypt the entries of $\mathbf{U}_{k,m}$, which are typically floating-point values, we ini-

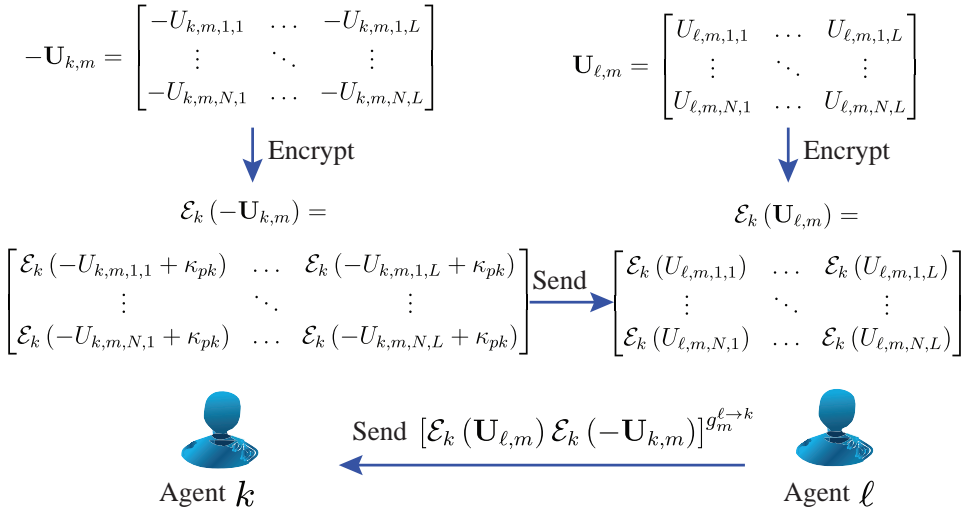


Figure 5.3: The information exchange between agents k and $\ell \in \mathcal{N}_k$ in PPDNMF.

tially quantize them. This involves multiplying each entry by a positive integer N_{\max} , which determines the quantization resolution, and then rounding the result to the nearest integer. To undo the quantization, we divide the decrypted values by N_{\max} .

- To handle the encryption of negative quantized values (note line 10 in Algorithm 6), we convert them to positive integers by adding the public key κ_{pk} to them [135].
- To guarantee convergence, we ensure that the parameters $g_m^{k \rightarrow \ell}$ increase monotonically over iterations without becoming unbounded [130]. Thus, we select each parameter uniformly from the interval $(g_{m-1}^{k \rightarrow \ell}, g_k]$, where g_k is a predefined positive constant, known only to agent k , and $g_0^{k \rightarrow \ell} = 0$.

5.4 Numerical Results

In this section, we conduct a series of numerical experiments to evaluate the performance of our PPDNMF algorithm. We consider a network consisting of $K = 10$ agents, interconnected arbitrarily, with each agent having three neighbors on average. We test our algorithm on two datasets, namely, a synthetic dataset and the MIT Center for Biological and Computational Learning (MIT-CBCL) face database [137], a widely utilized benchmark in the field of computer vision, comprises a diverse set of images capturing various poses, expressions, and lighting conditions, providing a robust foundation for evaluating the performance of face recognition algorithms.

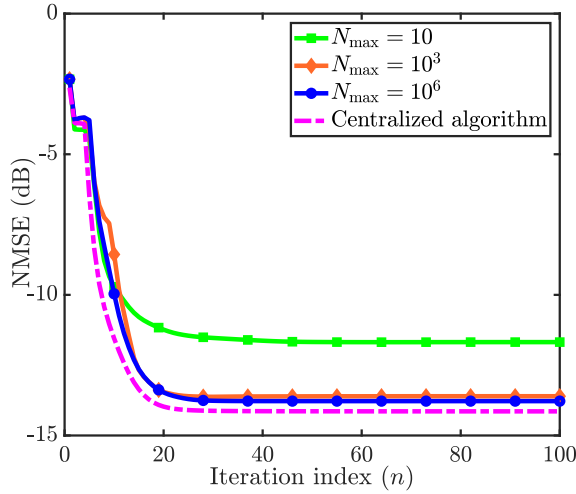


Figure 5.4: NMSE (5.15) of PPDNMF and centralized algorithm versus the BCD iteration index for different values of N_{\max} on synthetic data.

The agents collaboratively factorize a data matrix $\mathbf{Z} \in \mathbb{R}^{N \times M}$ to left and right factor matrices $\mathbf{X} \in \mathbb{R}^{N \times L}$ and $\mathbf{Y} \in \mathbb{R}^{L \times M}$, where $N \in \{30, 361\}$, $M \in \{200, 2429\}$, and $L \in \{5, 49\}$ in our two experiments. We set the number of BCD iterations to 100 and the number of ADMM iterations to 30. In our implementation of the Paillier cryptosystem, we use 128-bit public and private keys.

We evaluate the performance of PPDNMF in comparison with the centralized algorithm, i.e., where all data is available at a central hub. To quantify the performance, we utilize the normalized mean-square error (NMSE) at each BCD iteration, defined as

$$\frac{1}{K} \sum_{k=1}^K \frac{\|\mathbf{Z}_k - \mathbf{X}_{k,n} \mathbf{Y}_{k,n}\|_F}{\|\mathbf{Z}_k\|_F}. \quad (5.15)$$

In addition, we average the presented results over 100 independent trials.

In our first experiment utilizing synthetic data, we draw the entries of the nonnegative factor matrices $\mathbf{X} \in \mathbb{R}^{30 \times 5}$ and $\mathbf{Y} \in \mathbb{R}^{5 \times 200}$ independently from exponential distributions with parameter values 0.033 and 0.8, respectively. We calculate the data matrix as $\mathbf{Z} = \mathbf{X}\mathbf{Y} + \mathbf{\Gamma}$, where we draw the entries of $\mathbf{\Gamma}$ independently from a Gaussian distribution with zero mean and variance 3.6×10^{-4} , resulting in an SNR of approximately 20dB. We set $\mu = 0.1$, $\eta = 1$, and $g_i = 0.033$ for all agents. We consider \mathbf{Z} to be distributed among the agents such that each agent has a varying number of columns between four and 40. We present the NMSE learning curves of

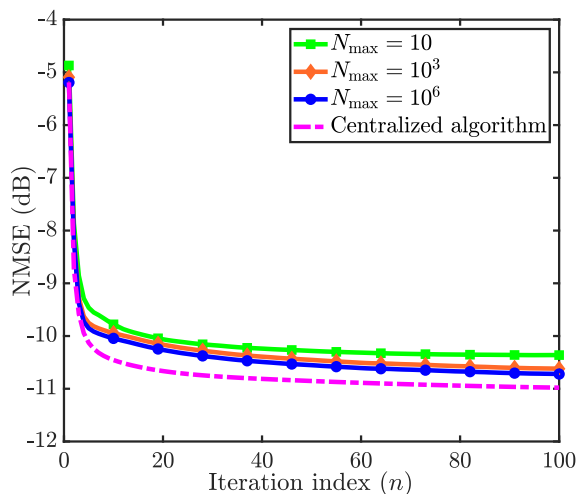


Figure 5.5: NMSE (5.15) of PPDNMF and centralized algorithm versus the BCD iteration index for different values of N_{\max} on MIT-CBCL database.

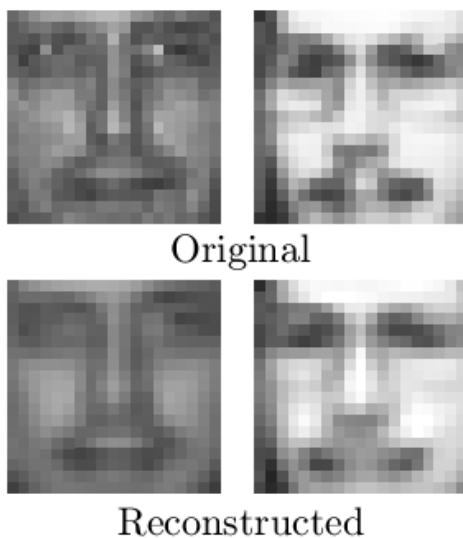


Figure 5.6: The original and reconstructed faces #1 and #2429 from the MIT-CBCL database.

PPDNMF for different values of N_{\max} alongside that of the corresponding centralized algorithm in Figure 5.4. From Figure 5.4, it is evident that the proposed PPDNMF algorithm closely matches the performance of the centralized algorithm regarding both convergence rate and steady-state NMSE. Furthermore, it is clear

that increasing the value of N_{\max} results in a lower steady-state NMSE.

Our second experiment utilizes the MIT-CBCL face database, which contains 2429 monochromatic face images in its training dataset. We allocate the corresponding data matrix among the agents so that each agent has between 224 and 245 columns. For this experiment, we configured $\mu = 2$, $\eta = 2$, and set $g_i = 0.05$ for all agents. The results presented in Figure 5.5 highlight the effectiveness of PPDNMF. Notably, PPDNMF exhibits robust performance even with $N_{\max} = 10$. Additionally, we compare the original faces #1 and #2429 with their reconstructed versions by PPDNMF, using $N_{\max} = 10^6$ in Figure 5.6. The reconstructed images closely match their original versions.

5.5 Summary

In this chapter, we introduced a novel privacy-preserving distributed nonnegative matrix factorization (PPDNMF) algorithm that employs the Paillier cryptosystem to enable secure collaboration among agents, thereby safeguarding their privacy and mitigating the risk of sensitive data leakage over ad-hoc networks. Our simulation results, based on both synthetic data and a real-world database, confirmed the efficacy of our proposed PPDNMF algorithm.

Chapter 6

Distributed Maximum Consensus over Noisy Links

This chapter presents the results of publication **P7**, which delves into the problem of estimating the maximum value within a multi-agent ad-hoc network in the presence of noise in the communication links. Our approach involves redefining the maximum consensus problem as a distributed optimization problem, allowing for a solution using ADMM. Unlike current algorithms that fail to converge when they work with noise-corrupted estimates, our noise-robust distributed maximum consensus (RD-MC) algorithm converges in a noisy setup. To achieve this robustness, we employ only a single set of noise-corrupted estimates and a moving averaging on the local estimates to mitigate the impact of link noise. Our extensive simulations demonstrate that RD-MC displays significantly improved resilience to communication link noise compared to existing maximum consensus algorithms.

6.1 Motivation

Consensus algorithms play a pivotal role in facilitating coordination and consensus formation among multiple agents within a distributed system [25, 138–145]. Numerous studies have delved into the challenge of attaining network-wide consensus on various values, such as average, minimum, and median, in a distributed manner. Achieving consensus in a multi-agent network requires local computations by agents and data exchange among neighboring agents [146–155]. The presence of noise in the communication links between agents can significantly impact the performance of consensus algorithms, introducing challenges in achieving accurate and reliable convergence. Navigating the effects of communication noise is a critical consideration in the design and implementation of consensus-based distrib-

uted systems, as it can lead to higher steady-state mean-square error and potential instability in the consensus process [156].

The distributed maximum consensus problem involves determining the largest value within a network in a decentralized manner. Extensive research has been dedicated to exploring this problem across various contexts [157–163]. The work in [157] presents a distributed algorithm for achieving maximum consensus, although it assumes noise-free communication links. Furthermore, the study by [158] establishes limits on the anticipated convergence time for achieving maximum consensus in asynchronous networks, without taking into account communication noise. The method employed in [159] tackles the maximum consensus problem by using the soft-max function as an approximation of the maximum function. Nevertheless, its effectiveness is constrained by a trade-off between estimation error and convergence speed. Although [160] introduces a distributed maximum consensus algorithm resilient to noise, the variance of its error increases proportionally with the network size.

This chapter presents a fully-distributed algorithm termed noise-robust distributed maximum consensus (RD-MC), aimed at effectively determining the maximum value within a multi-agent network, especially in situations where communication links are affected by noise. In RD-MC, we utilize the advantages of strategically planned parameter exchanges. We demonstrate the efficacy of RD-MC through comprehensive simulations and by contrasting its performance with that of existing algorithms.

6.2 Background

We consider a distributed network with K agents, modeled by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The set of vertices \mathcal{V} corresponds to the agents and the edge set \mathcal{E} represents the communication links between the agents. The adjacency matrix \mathbf{A} describes the structure of the graph \mathcal{G} with entries $a_{k\ell}$, where $a_{k\ell} = 1$ if and only if the agents k and ℓ are neighbors and $a_{k\ell} = 0$ otherwise. The set of neighbors for agent k is denoted by \mathcal{N}_k with its cardinality denoted by $d_k = |\mathcal{N}_k|$. We assume that the set \mathcal{N}_k does not include the agent k itself. We also assume only simple graphs, i.e., there are no self-loops or multiple edges.

The conventional maximum consensus algorithm relies on the agents communicating only with their immediate neighbors and can be written as

$$x_{k,n+1} = \max(x_{k,n}, \{x_{\ell,n}\}_{\ell \in \mathcal{N}_k}), \quad \forall k \in \mathcal{V}, \quad (6.1)$$

where $x_{k,n}$ is the estimate of the k th agent at time instant n , and the initial value is $x_{k,0} = a_k \quad \forall k \in \mathcal{V}$. The solution to (6.1) is denoted by $\max(\{a_k\}_{k \in \mathcal{V}}) =$

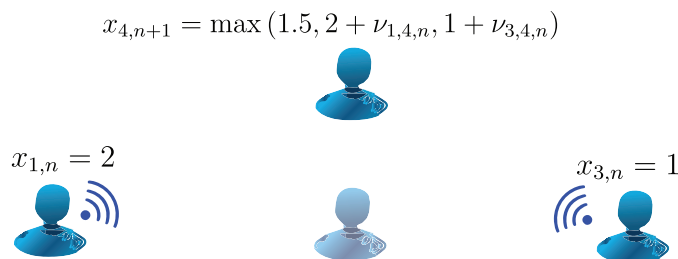


Figure 6.1: An illustration of the naive maximum consensus (naive-MC) algorithm.

a^* . Given that the communications between agents are noiseless, (6.1) reaches consensus after a finite number of iterations denoted by N , i.e., $x_{k,n} = a^* \forall n \geq N$ and $k \in \mathcal{V}$ [161].

We want to investigate the convergence of (6.1) when the communication links are noisy. The noise in the received message by agent k from agent ℓ at time instant n is denoted by $\nu_{\ell,k,n}$ and modeled as zero-mean additive white Gaussian noise with variance σ^2 . The noise is assumed to be uncorrelated across different agents and time instants. Hence, upon accounting for additive link noise, (6.1) can be written as

$$x_{k,n+1} = \max(x_{k,n}, \{x_{\ell,n} + \nu_{\ell,k,n}\}_{\ell \in \mathcal{N}_k}), \quad \forall k \in \mathcal{V}. \quad (6.2)$$

We refer to (6.2) as naive maximum consensus (naive-MC). We argue that even though (6.1) reaches the maximum value when we have noiseless communication links, (6.2) might not reach convergence because of potential deviation caused by noise in estimating the maximum value in each iteration. This potential deviation is illustrated in Figure 6.1. From Figure 6.1, we can observe that agent 4 has neighbors in the set $\mathcal{N}_4 \in \{1, 3\}$. It is evident that the states received from agents 1 and 3 are affected by link noise. One possible scenario is that this noise raises the true maximum state of 2 to a higher value. This deviation, accumulating over time, could result in substantial inaccuracies and even lead to divergence of (6.2).

To tackle this challenge, we reformulate the maximum consensus problem (6.1) as a distributed optimization problem that can be solved via ADMM. We show that our proposed RD-MC algorithm achieves convergence even when there is additive noise in communication links.

6.3 Proposed Method

6.3.1 Distributed Maximum Consensus Algorithm

A consensus-based reformulation of the maximum consensus problem (6.1) has been proposed in [161] as

$$\begin{aligned} a^* &= \arg \min_x x \\ \text{s.t. } & x \geq a_k, \forall k \in \mathcal{V}. \end{aligned} \quad (6.3)$$

In (6.3), we reformulate the problem (6.1) in epigraph form as a linear program (LP). To solve (6.3) in a distributed manner, we assign each agent its own variable $\{x_k\}_{k=1}^K$. This approach imposes local epigraph constraints that enforce consensus among neighboring agents. Additionally, we utilize a separable objective that is equivalent to the original when all local variables are in agreement, i.e., we have consensus.

Hence, we can rewrite (6.3) as

$$\begin{aligned} \min_{\{x_k\}} & \frac{1}{K} \sum_{k=1}^K x_k \\ \text{s.t. } & x_k \geq a_k \quad \forall k \in \mathcal{V} \\ & x_k = x_\ell \quad \forall k \in \mathcal{V}, \ell \in \mathcal{N}_k. \end{aligned} \quad (6.4)$$

In (6.4), we enforce local consensus across each agent's neighborhood through equality constraints. It can be shown that (6.1) and (6.4) have an identical solution through the Karush-Kuhn-Tucker (KKT) optimality conditions [161], i.e. the solution of (6.4) is $x_k^* = a^* \forall k \in \mathcal{V}$.

In addition to employing a mathematical approach, such as KKT conditions, to establish this equivalence, it is beneficial to utilize our intuition regarding (6.4). In (6.4), the goal is to minimize the objective function while simultaneously ensuring that each local variable is greater than its initial state. Consequently, this reformulation is expected to converge to the maximum initial value.

To facilitate the reformulation of (6.4) for a distributed solution via utilizing ADMM, we incorporate the constraints into the objective function through the use of indicator functions. This approach transforms the problem (6.4) to a fully separable form of two distinct blocks, as elaborated in section 2.6.

Therefore, we can further express (6.4) as

$$\begin{aligned} \min_{\{x_k, y_k, q_k^\ell\}} \quad & \frac{1}{K} \sum_{k=1}^K x_k + \frac{1}{K} \sum_{k=1}^K \mathcal{I}_{a_k}(y_k) \\ \text{s.t.} \quad & x_k = y_k \quad \forall k \in \mathcal{V} \\ & x_k = q_k^\ell, x_\ell = q_k^\ell \quad \forall k \in \mathcal{V}, \ell \in \mathcal{N}_k, \end{aligned} \quad (6.5)$$

where the indicator function $\mathcal{I}_{a_k}(y_k)$, defined as $\mathcal{I}_a(y) = 0$, if $y \geq a$ and ∞ otherwise, imposes the inequality constraint to seek the maximum value, and the auxiliary variables $\mathcal{Q} = \{q_k^\ell\}_{k \in \mathcal{V}, \ell \in \mathcal{N}_k}$ facilitate consensus within each agent's neighborhood and, consequently, across the network. The optimization problem (6.5) can be tackled using various methods, including those based on subgradients or ADMM. However, distributed subgradient methods applied to affine objective functions are known to converge slowly [164]. Therefore, we opt for ADMM to solve (6.5).

Let $\mathcal{L}(\{x_k, y_k\}_{k \in \mathcal{V}}, \mathcal{Q}, \mathcal{M})$ denote the augmented Lagrangian function associated with (6.5) as

$$\begin{aligned} \mathcal{L}(\{x_k, y_k\}_{k \in \mathcal{V}}, \mathcal{Q}, \mathcal{M}) = & \sum_{k=1}^K \left(\frac{x_k + \mathcal{I}_{a_k}(y_k)}{K} + u_k(x_k - y_k) + \frac{\rho_y}{2}(x_k - y_k)^2 \right) \\ & + \sum_{k=1}^K \sum_{\ell \in \mathcal{N}_k} \left(\mu_{k,\ell}(x_k - q_k^\ell) + \pi_{k,\ell}(x_\ell - q_k^\ell) \right) \\ & + \frac{\rho_z}{2} \sum_{k=1}^K \sum_{\ell \in \mathcal{N}_k} \left((x_k - q_k^\ell)^2 + (x_\ell - q_k^\ell)^2 \right), \end{aligned} \quad (6.6)$$

where $\mathcal{M} = \{u_k, \mu_{k,\ell}, \pi_{k,\ell}\}_{k \in \mathcal{V}, \ell \in \mathcal{N}_k}$ represents the respective Lagrange multipliers. Minimizing \mathcal{L} , while applying the KKT conditions to (6.5) and defining $v_{k,n} = 2 \sum_{\ell \in \mathcal{N}_k} \mu_{k,\ell,n}$, leads to the following iterative updates at the k th agent along with the elimination of $\{\pi_{k,\ell}\}_{k \in \mathcal{V}, \ell \in \mathcal{N}_k}$ and \mathcal{Q} [15, 161] as

$$x_{k,n+1} = n_k \left(\frac{-1}{K} + \rho_y [y_{k,n} - \bar{u}_{k,n}] - v_{k,n} + \rho_z \sum_{\ell \in \mathcal{N}_k} [x_{k,n} + \tilde{x}_{\ell,n}] \right), \quad (6.7a)$$

$$y_{k,n+1} = \max(x_{k,n+1} + \bar{u}_{k,n}, a_k), \quad (6.7b)$$

$$\bar{u}_{k,n+1} = \bar{u}_{k,n} + x_{k,n+1} - y_{k,n+1}, \quad (6.7c)$$

$$v_{k,n+1} = v_{k,n} + \rho_z \sum_{\ell \in \mathcal{N}_k} [x_{k,n+1} - \tilde{x}_{\ell,n+1}]. \quad (6.7d)$$

Here, n is the iteration index, $n_k = (\rho_y + 2\rho_z d_k)^{-1}$, $\rho_y > 0$ and $\rho_z > 0$ are the penalty parameters, and $\bar{u}_k = u_k/\rho_y$. In addition, all initial values $\{x_{k,0}, y_{k,0}, \bar{u}_{k,0}, v_{k,0}\}_{k \in \mathcal{V}}$ are set to zero. Note that, in (6.7a), agent k has access to $\tilde{x}_{\ell,n} = x_{\ell,n} + \nu_{\ell,k,n}$ rather than $x_{\ell,n}$. The iterations (6.7) can be implemented locally at each agent in a fully-distributed fashion, as the required information is available within each agent's neighborhood. We refer to this algorithm as distributed maximum consensus (D-MC).

6.3.2 Noise-Robust Distributed Maximum Consensus Algorithm

Our goal is to investigate the effect of noisy links in (6.4). Therefore, we describe two modifications to the D-MC algorithm (6.7). These modifications are aimed at enhancing the robustness of distributed maximum consensus to communication noise and lead to the proposed RD-MC algorithm.

Using the initial values $v_{k,0} = 0 \forall k \in \mathcal{V}$, we obtain

$$v_{k,n} = \rho_z \sum_{m=1}^n \sum_{\ell \in \mathcal{N}_k} [x_{k,m} - \tilde{x}_{\ell,m}] \quad (6.8)$$

from (6.7d). Substituting (6.8) into (6.7a) while using the initial values $x_{k,0} = 0$ and $x_{k,1} = -K^{-1}n_k \forall k \in \mathcal{V}$, we can eliminate $v_{k,n}$ and modify (6.7a) as [165]

$$x_{k,n+1} = (1 - \rho_y n_k)x_{k,n} - \rho_z d_k n_k x_{k,n-1} + n_k \left(\rho_y z_{k,n} + \rho_z \sum_{\ell \in \mathcal{N}_k} \tilde{s}_{\ell,n} \right), \quad (6.9a)$$

$$\bar{x}_{k,n+1} = \sum_{m=0}^{\mathcal{C}-1} \alpha_m x_{k,n-m+1}, \quad (6.9b)$$

$$z_{k,n+1} = 2y_{k,n+1} - y_{k,n}, \quad (6.9c)$$

$$s_{k,n+1} = 2\bar{x}_{k,n+1} - x_{k,n}. \quad (6.9d)$$

Note that, in (6.9b), to enhance robustness against spurious noise, we compute the convex combination of \mathcal{C} past local estimates, utilizing the weights α_ℓ that sum to one.

In this alternative formulation, instead of $x_{k,n}$, agents exchange $s_{k,n}$, which is a smoothed version of $x_{k,n}$. However, due to communication noise, they receive noisy versions from their neighbors, i.e., agents $\ell \in \mathcal{N}_k$ receive $\tilde{s}_{k,n} = s_{k,n} + \nu_{k,\ell,n}$ from agent k . The recursions (6.9) alongside (6.7b) and (6.7c) constitute the proposed noise-robust distributed maximum consensus (RD-MC) algorithm, summarized in Algorithm 7.

Algorithm 7: RD-MC.

-
- 1 **Parameters:** penalty parameters ρ_z and ρ_y
 - 2 **Initialization:** $x_{k,0} = 0$, $x_{k,1} = -K^{-1}n_k$, $\bar{u}_{k,1} = 0$,
 - 3 $z_{k,1} = 0$, $s_{k,1} = -2K^{-1}n_k$, $\forall i \in \mathcal{V}$
 - 4 **For** $n = 1, \dots, N$ *until convergence*
 - 5 Receive $\tilde{s}_{\ell,n}$ from neighbors $\ell \in \mathcal{N}_k$
 - 6 $x_{k,n+1} = (1 - \rho_y n_k)x_{k,n} - \rho_z d_k n_k x_{k,n-1} + n_k \left(\rho_y z_{k,n} + \rho_z \sum_{\ell \in \mathcal{N}_k} \tilde{s}_{\ell,n} \right)$
 - 7 $\bar{x}_{k,n+1} = \sum_{m=0}^{\mathcal{C}-1} \alpha_m x_{k,n-m+1}$, $\sum_m \alpha_m = 1$
 - 8 $y_{k,n+1} = \max(x_{k,n+1} + \bar{u}_{k,n}, a_k)$
 - 9 $\bar{u}_{k,n+1} = \bar{u}_{k,n} + x_{k,n+1} - y_{k,n+1}$
 - 10 $z_{k,n+1} = 2y_{k,n+1} - y_{k,n}$
 - 11 Send $s_{k,n+1} = 2\bar{x}_{k,n+1} - x_{k,n}$ to neighbors $\ell \in \mathcal{N}_k$.
 - 12 **EndFor**
-

We mitigate the effect of noisy links in RD-MC through two key modifications to D-MC. First, the introduction of $s_{k,n}$, a linear combination of $\bar{x}_{k,n}$ and $x_{k,n-1}$, offers a strategic advantage in alleviating the adverse effects of communication noise. By exchanging $s_{k,n}$ instead of $x_{k,n}$ over noisy links, we enhance robustness. Notably, while the aggregation of two sets of noisy estimates received from neighbors at consecutive iterations [i.e., $\tilde{x}_{\ell,n}$ in (6.7a) and $\tilde{x}_{\ell,n+1}$ in (6.7d)] renders D-MC vulnerable to noise accumulation, RD-MC's reliance on a single set of noisy estimates [i.e., $\tilde{s}_{k,n}$ in (6.9a)] enhances its resilience to link noise. Second, we further enhance robustness to link noise by applying a weighted averaging of $x_{k,n+1}$ over a sliding window of size \mathcal{C} as in (6.9b).

6.4 Numerical Results

We conduct a series of experiments to assess the performance of the proposed RD-MC algorithm. We examine a network of $K = 20$ agents, as illustrated in Figure 6.2. The initial values (estimates) for the agents are drawn independently from a standard normal distribution, specifically, $a_k \sim \mathcal{N}(0, 1) \forall k \in \mathcal{V}$, and we define $a^* = \max(\{a_k\}_{k \in \mathcal{V}})$. Furthermore, we set the penalty parameters to $\rho_z = \rho_y = 1$ and assign the weights in (6.9b) to $\alpha_\ell = 1/\mathcal{C}$ across all our experiments. The results are obtained by averaging over 1000 independent instances of communication noise. To represent the noise in the communication links, we utilize a truncated zero-mean normal distribution, limiting the noise to within $\pm 3\sigma$ to keep it within a reasonable range.

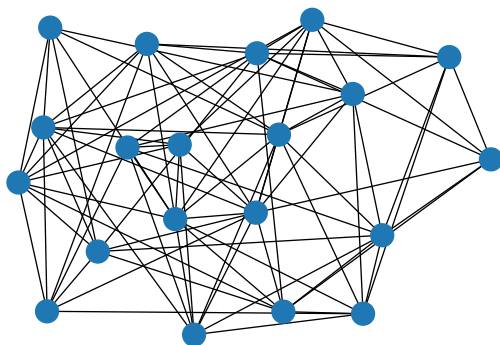


Figure 6.2: The considered network with an arbitrary topology and $K = 20$ agents.

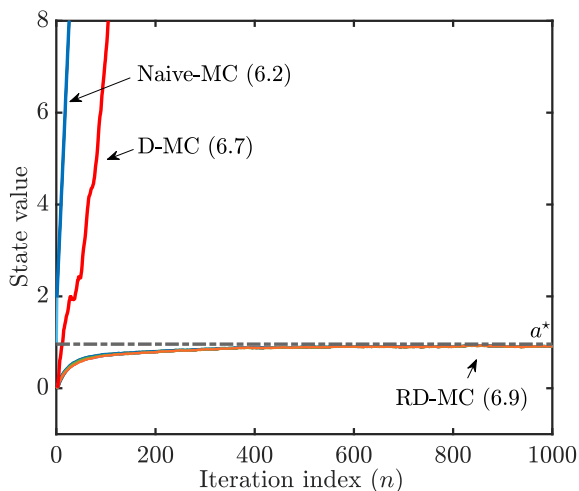


Figure 6.3: The impact of noise on the performance of Naive-MC (6.2), D-MC algorithm (6.7) and RD-MC algorithm (6.9) with window size $\mathcal{C} = 3$ and noise variance $\sigma^2 = 0.1$.

In our first experiment, we investigate the effects of noise on the performance of RD-MC, D-MC, and the naive solution (6.2), referred to as naive-MC, using a noise variance of $\sigma^2 = 0.1$ and a window size of $\mathcal{C} = 3$. Figure 6.3 depicts the evolution of the estimates of all agents using the considered algorithms over 1000 iterations. It is evident that RD-MC converges to the maximum value with a bounded error, while the other two algorithms diverge.

In our second experiment, we study the effect of noise variance on the network-

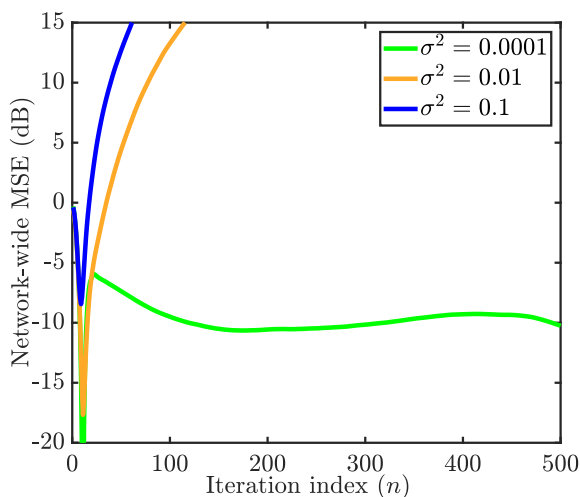


Figure 6.4: The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 1$ and different noise variances $\sigma^2 \in \{0.0001, 0.01, 0.1\}$.

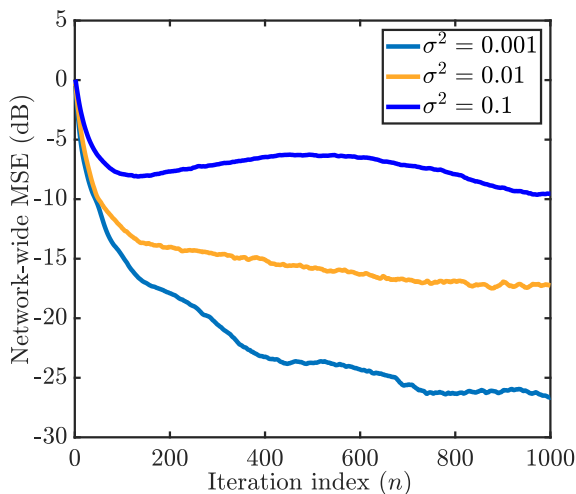


Figure 6.5: The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 2$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$.

wide mean square error (MSE) of RD-MC calculated as

$$\frac{1}{K} \sum_{k=1}^K \mathbb{E} [(x_{k,n} - a^*)^2]. \quad (6.10)$$

We conduct simulations of RD-MC using different window sizes $\mathcal{C} \in \{1, 2, 3\}$

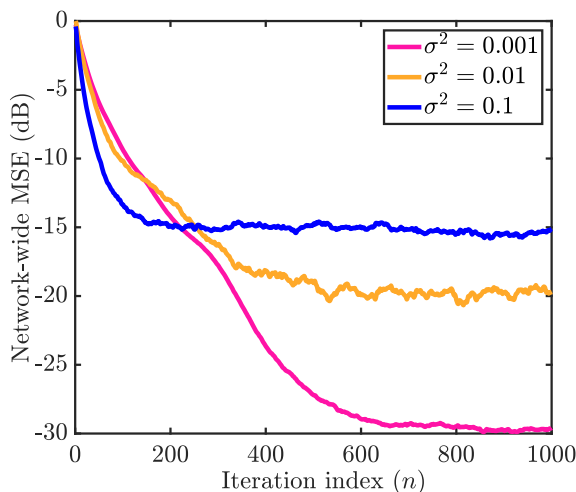


Figure 6.6: The effect of noise variance on the steady-state network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ and different noise variances $\sigma^2 \in \{0.001, 0.01, 0.1\}$.

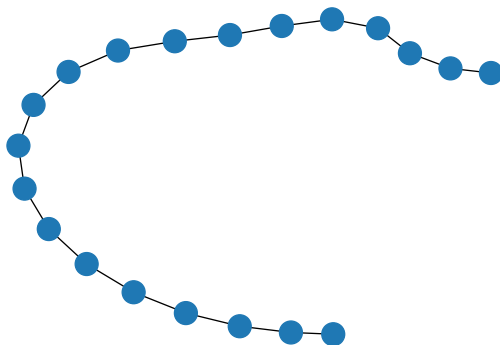


Figure 6.7: The considered network with linear topology and $K = 20$ agents.

and noise variances σ^2 , and present the results in Figs. 6.4-6.6. We observe that increasing σ^2 results in higher steady-state network-wide error across all experiments. Nevertheless, the selection of window size has a significant impact on RD-MC's ability to reduce communication noise. While RD-MC struggles to converge with $\mathcal{C} = 1$, it maintains convergence with $\mathcal{C} \geq 2$ and increasing \mathcal{C} enhances its robustness to noise. Figs. 6.3 and 6.6 demonstrate that RD-MC with $\mathcal{C} = 3$ shows a markedly improved resistance to link noise in comparison to D-MC, all while not adding any extra computational or communication burden.

In our final experiment, we evaluate how sensitive RD-MC's performance is to net-

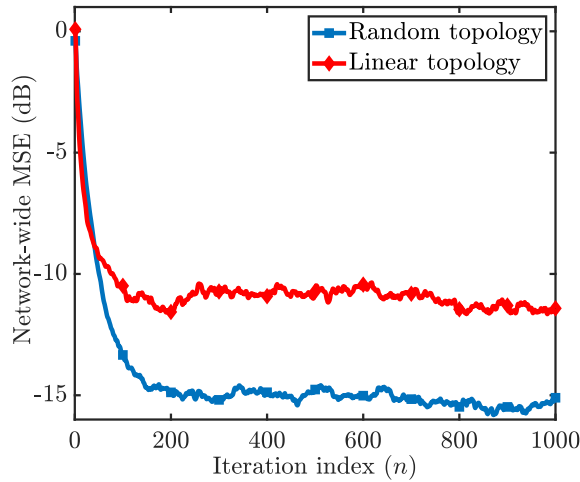


Figure 6.8: The impact of network connectivity on the network-wide MSE of RD-MC with window size $\mathcal{C} = 3$ in the presence of link noise with variance $\sigma^2 = 0.1$.

work topology in the presence of link noise. We simulate RD-MC with a window size of $\mathcal{C} = 3$ across two networks: the network shown in 6.2 and another with a linear topology illustrated in Figure 6.7. The noise variance is set to $\sigma^2 = 0.1$. We present the results in Figure 6.8. It is evident that, in the case of a linear network topology, the network-wide steady-state MSE of RD-MC is greater compared to that of a network with a more complex topology and higher average degree. Nevertheless, RD-MC continues to exhibit good performance in the linear network topology, despite its lower connectivity.

6.5 Summary

In this chapter, a noise-robust distributed algorithm termed RD-MC has been proposed to mitigate the adverse effects of communication noise on the convergence of conventional distributed maximum consensus algorithms. This algorithm incorporates two modifications compared to the conventional distributed maximum consensus algorithm. Unlike existing algorithms tailored for ad-hoc networks, the proposed algorithm demonstrates robustness against additive communication noise. Extensive simulation results have underscored the effectiveness of RD-MC in diverse scenarios.

Chapter 7

Conclusion and Future Work

This thesis developed solutions for various application scenarios within distributed and federated learning settings. We can summarize the thesis and provide future directions for continuing this work as follows:

7.1 Summary

In chapter 2, we provided an overview of distributed and federated learning and their abilities and challenges. In addition, we introduced online federated learning, partial diffusion of information, and the alternating direction method of multipliers.

In chapter 3, we conducted a theoretical analysis of PSO-Fed to examine its resilience to model-poisoning (Byzantine) attack. We showed that PSO-Fed converges in both mean and mean-square senses, given an appropriate choice of stepsize even under Byzantine attacks. More importantly, we showed that, in the presence of Byzantine clients, the steady-state MSE of PSO-Fed is significantly smaller than that of the Online-Fed algorithm, which does not feature partial sharing. Our theoretical analysis also uncovered the existence of a non-trivial optimal stepsize for PSO-Fed in the presence of model-poisoning attacks. The simulation results corroborated our theoretical findings regarding PSO-Fed's effectiveness against Byzantine attacks as well as the accuracy of the theoretically predicted values of its steady-state MSE and optimal stepsize.

In chapter 4, we developed a resource-efficient FL algorithm that has improved robustness against noise/error in communication links. To motivate the developed algorithm, we considered a weighted least-squares regression problem. To achieve robustness, we proposed to combine the last two global model updates and send

them together alongside eliminating the dual model update performed at each participating edge device. The proposed algorithm, called RERCE-Fed, ensures that clients receive a less corrupted global model update from the server even when the server uses random scheduling to achieve communication efficiency. We proved the convergence of RERCE-Fed in the mean sense at the presence of link noise. We also verified the desirable performance of RERCE-Fed via simulations, particularly, its robustness against additive communication link noise in comparison to existing related algorithms.

In chapter 5, we introduced a novel privacy-preserving distributed nonnegative matrix factorization (PPDNMF) algorithm that employs the Paillier cryptosystem to enable secure collaboration among agents, thereby safeguarding their privacy and mitigating the risk of sensitive data leakage over ad-hoc networks. Our simulation results, based on both synthetic and real data, confirmed the efficacy of the proposed algorithm. In future work, we plan to conduct a comprehensive theoretical privacy analysis of the proposed algorithm, exploring its resilience across various attack scenarios.

In chapter 6, we developed a distributed algorithm, called noise-robust distributed maximum consensus (RD-MC), to tackle the challenge of identifying the maximum value within an ad-hoc multi-agent network utilizing noisy communication channels. Unlike existing algorithms designed for ad-hoc networks, RD-MC exhibits robustness against additive communication noise. Our extensive simulation results demonstrated the effectiveness of RD-MC in different scenarios.

7.2 Future Directions

Building upon the proposed algorithms and findings of this thesis, several promising research directions emerge. These directions are motivated by the need for more robust, scalable, and privacy-preserving FL frameworks in dynamic and adversarial environments. For future research, a list of works can be done, such as:

1. Resilience Analysis from a Bayesian Perspective: A promising extension of the PSO-Fed algorithm involves analyzing its resilience to adversarial model-poisoning attacks using a Bayesian framework. This probabilistic approach enables modeling the local and global model updates as well as the attack signals as independent random processes. Such a formulation would allow the study of PSO-Fed's robustness under distributional shifts and stochastic perturbations, thereby offering more in-depth resilience analysis. However, Bayesian methods often come with computational overhead and convergence challenges, especially in a decentralized setting, which must be addressed to make this practical.

2. **Privacy, Fairness, and Personalization in Distributed and Federated Learning:** Another vital direction is to deepen the privacy guarantees of algorithms like PPDNMF through a rigorous theoretical privacy analysis, possibly incorporating differential privacy or local privacy mechanisms. Moreover, as FL moves toward real-world adoption, fairness and personalization are increasingly important. Ensuring that models do not disproportionately benefit certain user groups over others, while allowing tailored performance to individual user preferences or data distributions, is a key challenge. This requires balancing global model utility with local user adaptation, which could be addressed through personalized models.
3. **Towards Privacy-Preserving Distributed Recommender Systems: Leveraging PPDNMF for building distributed recommender systems that maintain data locality** presents a viable alternative to traditional centralized systems. Such a system would inherently protect user privacy while potentially preserving or even enhancing recommendation quality. The main challenge here lies in maintaining global performance with only local and incomplete data.
4. **Application of PSO-Fed and RERCE-Fed to Emerging Domains:** The implementation of PSO-Fed and RERCE-Fed could involve the utilization of advanced deep neural networks with real-world datasets and applications that feature a substantial number of complex features and parameters. Exploring the concept of partial sharing may prove effective in training large distributed models while enhancing their resilience against potential model-poisoning attacks.
5. **Hardware Implementation and Empirical Validation:** Finally, the proposed algorithms can be implemented on hardware to validate their effectiveness and gain deeper insights into their advantages and potential limitations.

Bibliography

- [1] P. Sethi and S. R. Sarangi, “Internet of things: Architectures, protocols, and applications,” *J. elec. comput. eng.*, vol. 2017, no. 1, p. 9 324 035, 2017.
- [2] P. Bellini, P. Nesi and G. Pantaleo, “IoT-enabled smart cities: A review of concepts, frameworks and key technologies,” *Appl. Sci.*, vol. 12, no. 3, p. 1607, 2022.
- [3] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan and S. K. Das, “Edge-computing-driven internet of things: A survey,” *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–41, 2022.
- [4] J. Sheth and B. Dezfouli, “Enhancing the energy-efficiency and timeliness of IoT communication in wifi networks,” *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9085–9097, 2019.
- [5] L. Tawalbeh, F. Muheidat, M. Tawalbeh and M. Quwaider, “IoT privacy and security: Challenges and solutions,” *Appl. Sci.*, vol. 10, no. 12, p. 4102, 2020.
- [6] B. V. Philip, T. Alpcan, J. Jin and M. Palaniswami, “Distributed real-time IoT for autonomous vehicles,” *IEEE Trans. Ind. Inform.*, vol. 15, no. 2, pp. 1131–1140, 2018.
- [7] J. Park, S. Samarakoon, A. Elgabli, J. Kim, M. Bennis, S.-L. Kim and M. Debbah, “Communication-efficient and distributed learning over wireless networks: Principles and applications,” *Proc. IEEE*, vol. 109, no. 5, pp. 796–819, 2021.
- [8] Z. Zhao, Y. Mao, Y. Liu, L. Song, Y. Ouyang, X. Chen and W. Ding, “Towards efficient communications in federated learning: A contemporary survey,” *J. Frankl. Inst.*, vol. 360, no. 12, pp. 8669–8703, 2023.

- [9] H. Wang and Y. Chi, “Communication-efficient federated optimization over semi-decentralized networks,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 11, pp. 147–160, 2025.
- [10] M. Rapp, R. Khalili, K. Pfeiffer and J. Henkel, “Distreal: Distributed resource-aware learning in heterogeneous systems,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, 2022, pp. 8062–8071.
- [11] A. Moradi, N. K. Venkategowda, S. P. Talebi and S. Werner, “Privacy-preserving distributed kalman filtering,” *IEEE Trans. Signal Process.*, vol. 70, pp. 3074–3089, 2022.
- [12] B. Kailkhura, S. Brahma and P. K. Varshney, “Data falsification attacks on consensus-based detection systems,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 1, pp. 145–158, Mar. 2017. DOI: 10.1109/TSIPN.2016.2607119.
- [13] Z. Wang, Q. Kang, X. Zhang and Q. Hu, “Defense strategies toward model poisoning attacks in federated learning: A survey,” in *Proc. IEEE Wireless Comm. Net. Conf.*, 2022, pp. 548–553. DOI: 10.1109/WCNC51071.2022.9771619.
- [14] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi and M. Guizani, “A survey on federated learning: The journey from centralized to distributed on-site learning and beyond,” *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, 2021. DOI: 10.1109/JIOT.2020.3030072.
- [15] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas and H. Zhu, *Decentralized learning for wireless communications and networking*. Springer, 2017, pp. 461–497.
- [16] V. Borkar and P. Varaiya, “Asymptotic agreement in distributed estimation,” *IEEE Trans. Automatic Control*, vol. 27, no. 3, pp. 650–655, 1982. DOI: 10.1109/TAC.1982.1102982.
- [17] J. Tsitsiklis, “Problems in decentralized decision making and computation,” Ph.D. dissertation, Massachusetts Institute of Technology, 1984.
- [18] J. Tsitsiklis, D. Bertsekas and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE Trans. Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986. DOI: 10.1109/TAC.1986.1104412.
- [19] D. Bertsekas and J. Tsitsiklis, “A survey of some aspects of parallel and distributed iterative algorithms,” 1989.
- [20] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation*. Upper Saddle River: Prentice-Hall, 1989.

-
- [21] J. A. Fax, *Optimal and cooperative control of vehicle formations*. California Institute of Technology, 2002.
- [22] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.
- [23] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proc. Am. Control Conf.*, IEEE, vol. 2, 2003, pp. 951–956.
- [24] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.
- [25] R. Olfati-Saber, J. A. Fax and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007. DOI: 10.1109/JPROC.2006.887293.
- [26] A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003. DOI: 10.1109/TAC.2003.812781.
- [27] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, 2014. DOI: 10.1109/JPROC.2014.2306253.
- [28] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014.
- [29] A. H. Sayed, F. Cattivelli, S. Haykin, K. R. Liu *et al.*, "Distributed adaptive learning mechanisms," in *Handbook on array processing and sensor networks*, Wiley Online Library, 2009, pp. 695–722.
- [30] D. P. Bertsekas, "A new class of incremental gradient methods for least squares problems," *SIAM J. Optim.*, vol. 7, no. 4, pp. 913–926, 1997.
- [31] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.
- [32] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4064–4077, 2007. DOI: 10.1109/TSP.2007.896034.
- [33] F. S. Cattivelli and A. H. Sayed, "Analysis of spatial and incremental lms processing for distributed estimation," *IEEE Trans. Signal Process.*, vol. 59, no. 4, pp. 1465–1480, 2011. DOI: 10.1109/TSP.2010.2100386.
- [34] M. H. DeGroot, "Reaching a consensus," *J. Am. Stat. Assoc.*, vol. 69, no. 345, pp. 118–121, 1974.

- [35] W. Ren and R. Beard, “Consensus seeking in multiagent systems under dynamically changing interaction topologies,” *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, 2005. DOI: 10.1109/TAC.2005.846556.
- [36] F. S. Cattivelli, C. G. Lopes and A. H. Sayed, “Diffusion recursive least-squares for distributed estimation over adaptive networks,” *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1865–1877, 2008. DOI: 10.1109/TSP.2007.913164.
- [37] J. Chen and A. H. Sayed, “Diffusion adaptation strategies for distributed optimization and learning over networks,” *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, 2012. DOI: 10.1109/TSP.2012.2198470.
- [38] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao and Z. J. Towfic, “Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior,” *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 155–171, 2013. DOI: 10.1109/MSP.2012.2231991.
- [39] C. G. Lopes and A. H. Sayed, “Diffusion least-mean squares over adaptive networks: Formulation and performance analysis,” *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, 2008. DOI: 10.1109/TSP.2008.917383.
- [40] J. Chen and A. H. Sayed, “On the learning behavior of adaptive networks—part i: Transient analysis,” *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3487–3517, 2015. DOI: 10.1109/TIT.2015.2427360.
- [41] J. Chen and A. H. Sayed, “On the learning behavior of adaptive networks—part ii: Performance analysis,” *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3518–3548, 2015. DOI: 10.1109/TIT.2015.2427352.
- [42] S. Haykin, *Adaptive filter theory*. Prentice-Hall, NJ, 1996.
- [43] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Prentice-Hall, NJ, 1985.
- [44] P. S. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*. Springer Nature, 2019.
- [45] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2008.
- [46] V. H. Nascimento and M. T. M. Silva, “Adaptive filters,” in *Academic Press Library in Signal Processing: Volume 1*, Elsevier, 2014, pp. 619–761.
- [47] A. H. Sayed, T. Y. Al-Naffouri and V. H. Nascimento, “Energy conservation in adaptive filtering,” in *Nonlinear Signal and Image Processing*, CRC Press, 2003, pp. 23–58.

-
- [48] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, 2011.
- [49] Y. Chen, L. Su and J. Xu, “Distributed statistical machine learning in adversarial settings: Byzantine gradient descent,” *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 2, pp. 1–25, 2017.
- [50] M. Chen, D. Gündüz, K. Huang, W. Saad, M. Bennis, A. V. Feljan and H. V. Poor, “Distributed learning in wireless networks: Recent progress and future challenges,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3579–3605, 2021. DOI: 10.1109/JSAC.2021.3118346.
- [51] C. Dwork, F. McSherry, K. Nissim and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Proc. Theory Cryptography*, Springer, 2006, pp. 265–284.
- [52] Q. Lou and L. Jiang, “She: A fast and accurate deep neural network for encrypted data,” *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
- [53] J. Park, D.-J. Han, M. Choi and J. Moon, “Sageflow: Robust federated learning against both stragglers and adversaries,” *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 840–851, 2021.
- [54] A. Reisizadeh, I. Tziotis, H. Hassani, A. Mokhtari and R. Pedarsani, “Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity,” *IEEE J. Sel. Areas Inf. Theory*, vol. 3, no. 2, pp. 197–205, 2022. DOI: 10.1109/JSAIT.2022.3205475.
- [55] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. Y. Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. Int. Conf. Artif. Intell. Stat.*, Apr. 2017, pp. 1273–1282.
- [56] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon and D. Ramage, “Federated learning for mobile keyboard prediction,” *arXiv preprint arXiv:1811.03604*, 2018.
- [57] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [58] C. Huang, J. Huang and X. Liu, “Cross-silo federated learning: Challenges and opportunities,” *arXiv preprint arXiv:2206.12949*, 2022.

- [59] C. Ren, H. Yu, H. Peng, X. Tang, B. Zhao, L. Yi, A. Z. Tan, Y. Gao, A. Li, X. Li, Z. Li and Q. Yang, “Advances and open challenges in federated foundation models,” *IEEE Commun. Surv. Tutor.*, pp. 1–41, 2025. DOI: 10.1109/COMST.2025.3552524.
- [60] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari and A. S. Avestimehr, “Federated learning for the internet of things: Applications, challenges, and opportunities,” *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 24–29, 2022. DOI: 10.1109/IOTM.004.2100182.
- [61] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020. DOI: 10.1109/MSP.2020.2975749.
- [62] A. Kuh, “Real time kernel learning for sensor networks using principles of federated learning,” in *Proc. IEEE Int. Conf. Asia-Pacific Signal Info. Process. Assoc.*, Dec. 2021, pp. 2089–2093.
- [63] V. C. Gogineni, S. Werner, Y.-F. Huang and A. Kuh, “Communication-efficient online federated learning strategies for kernel regression,” *IEEE Internet Things J.*, vol. 10, no. 5, pp. 4531–4544, Mar. 2023. DOI: 10.1109/JIOT.2022.3218484.
- [64] R. Arablouei, S. Werner, Y.-F. Huang and K. Doğançay, “Distributed least mean-square estimation with partial diffusion,” *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 472–484, Jan. 2014. DOI: 10.1109/TSP.2013.2292035.
- [65] D. Gabay and B. Mercier, “A dual algorithm for the solution of nonlinear variational problems via finite element approximation,” *Comput. Math. Appl.*, vol. 2, no. 1, pp. 17–40, 1976.
- [66] R. Glowinski and A. Marroco, “Sur l’approximation, par éléments finis d’ordre un, et la résolution, par pénalisation-dualité d’une classe de problèmes de dirichlet non linéaires,” *Revue française d’automatique, informatique, recherche opérationnelle. Analyse numérique*, vol. 9, no. R2, pp. 41–76, 1975.
- [67] E. M. E. Mhamdi, R. Guerraoui and S. Rouault, “The hidden vulnerability of distributed learning in byzantium,” in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 3521–3530.
- [68] X. Cao and N. Gong, “MPAF: Model poisoning attacks to federated learning based on fake clients,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2022, pp. 3395–3403.

- [69] J. Bernstein, J. Zhao, K. Azizzadenesheli and A. Anandkumar, “SignSGD with majority vote is communication efficient and fault tolerant,” in *Proc. Int. Conf. Learn. Represent.*, 2019.
- [70] M. Fang, X. Cao, J. Jia and N. Gong, “Local model poisoning attacks to Byzantine-Robust federated learning,” in *USENIX Security Symp.*, 2020, pp. 1605–1622.
- [71] X. Cao, M. Fang, J. Liu and N. Z. Gong, “FLTrust: Byzantine-robust federated learning via trust bootstrapping,” *arXiv preprint arXiv:2012.13995*, 2020.
- [72] X. Cao, J. Jia, Z. Zhang and N. Z. Gong, “FedRecover: Recovering from poisoning attacks in federated learning using historical information,” in *Proc. IEEE Symp. Security Privacy*, Jul. 2023, pp. 326–343.
- [73] P. Blanchard, E. M. El Mhamdi, R. Guerraoui and J. Stainer, “Machine learning with adversaries: Byzantine tolerant gradient descent,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
- [74] K. Pillutla, S. M. Kakade and Z. Harchaoui, “Robust aggregation for federated learning,” *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022. DOI: 10.1109/TSP.2022.3153135.
- [75] G. Xia, J. Chen, C. Yu and J. Ma, “Poisoning attacks in federated learning: A survey,” *IEEE Access*, vol. 11, pp. 10 708–10 722, 2023. DOI: 10.1109/ACCESS.2023.3238823.
- [76] F. Hu, W. Zhou, K. Liao, H. Li and D. Tong, “Toward federated learning models resistant to adversarial attacks,” *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16 917–16 930, 2023. DOI: 10.1109/JIOT.2023.3272334.
- [77] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu and J. Liu, “Data poisoning attacks on federated machine learning,” *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11 365–11 375, 2022. DOI: 10.1109/JIOT.2021.3128646.
- [78] Y. Chen, X. Zhu, X. Gong, X. Yi and S. Li, “Data poisoning attacks in internet-of-vehicle networks: Taxonomy, state-of-the-art, and future directions,” *IEEE Trans. Industr. Inform.*, vol. 19, no. 1, pp. 20–28, 2023. DOI: 10.1109/TII.2022.3198481.
- [79] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto and F. Roli, “Evasion attacks against machine learning at test time,” in *Proc. Mach. Learn. Knowl. Discovery Databases*, Springer, 2013, pp. 387–402.

- [80] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang and X. Zhang, “Trojaning attack on neural networks,” in *Proc. Net. Dist. Syst. Security Symp.*, Internet Soc, 2018.
- [81] R. Shokri, M. Stronati, C. Song and V. Shmatikov, “Membership inference attacks against machine learning models,” in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 3–18. DOI: 10.1109/SP.2017.41.
- [82] L. Liu, Y. Wang, G. Liu, K. Peng and C. Wang, “Membership inference attacks against machine learning models via prediction sensitivity,” *IEEE Trans. Dependable Secure Computing*, vol. 20, no. 3, pp. 2341–2347, 2023. DOI: 10.1109/TDSC.2022.3180828.
- [83] C. Douligeris and D. N. Serpanos, *Network security: current status and future directions*. John Wiley & Sons, 2007.
- [84] D. Yin, Y. Chen, R. Kannan and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.
- [85] Z. Liu, K. Zheng, L. Hou, H. Yang and K. Yang, “A novel blockchain-assisted aggregation scheme for federated learning in IoT networks,” *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17 544–17 556, 2023. DOI: 10.1109/JIOT.2023.3277463.
- [86] R. Jin, Y. Liu, Y. Huang, X. He, T. Wu and H. Dai, “Sign-based gradient descent with heterogeneous data: Convergence and byzantine resilience,” *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, 2024. DOI: 10.1109/TNNLS.2023.3345367.
- [87] V. C. Gogineni, S. Werner, Y.-F. Huang and A. Kuh, “Communication-efficient online federated learning framework for nonlinear regression,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2022, pp. 5228–5232. DOI: 10.1109/ICASSP43922.2022.9746228.
- [88] E. Lari, V. C. Gogineni, R. Arablouei and S. Werner, “On the resilience of online federated learning to model poisoning attacks through partial sharing,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Apr. 2024, pp. 9201–9205.
- [89] R. Arablouei, K. Doğançay, S. Werner and Y.-F. Huang, “Adaptive distributed estimation based on recursive least-squares and partial diffusion,” *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3510–3522, Jul. 2014. DOI: 10.1109/TSP.2014.2327005.
- [90] D. S. Tracy and R. P. Singh, “A new matrix product and its applications in partitioned matrix differentiation,” *Statistica Neerlandica*, vol. 26, no. 4, pp. 143–157, 1972.

- [91] R. H. Koning, H. Neudecker and T. Wansbeek, “Block kronecker products and the vecb operator,” *Linear algebra and its applications*, vol. 149, pp. 165–184, 1991.
- [92] E. Lari, R. Arablouei, V. C. Gogineni and S. Werner, “Resilience in online federated learning: Mitigating model-poisoning attacks via partial sharing,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 11, pp. 388–400, 2025. DOI: 10.1109/TSIPN.2025.3559444.
- [93] S. Zhou and G. Y. Li, “Communication-efficient ADMM-based federated learning,” *arXiv preprint arXiv:2110.15318*, Jan. 2021.
- [94] F. Ang, L. Chen, N. Zhao, Y. Chen, W. Wang and F. R. Yu, “Robust federated learning with noisy communication,” *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3452–3464, Mar. 2020. DOI: 10.1109/TCOMM.2020.2979149.
- [95] S. Zheng, C. Shen and X. Chen, “Design and analysis of uplink and downlink communications for federated learning,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2150–2167, Jul. 2021. DOI: 10.1109/JSAC.2020.3041388.
- [96] M. M. Amiri and D. Gündüz, “Federated learning over wireless fading channels,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3546–3557, May 2020. DOI: 10.1109/TWC.2020.2974748.
- [97] H. Guo, A. Liu and V. K. N. Lau, “Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis,” *IEEE Internet Things J.*, vol. 8, no. 1, pp. 197–210, Jan. 2021. DOI: 10.1109/JIOT.2020.3002925.
- [98] M. M. Amiri, D. Gündüz, S. R. Kulkarni and H. V. Poor, “Convergence of federated learning over a noisy downlink,” *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1422–1437, Mar. 2022. DOI: 10.1109/TWC.2021.3103874.
- [99] X. Wei and C. Shen, “Federated learning over noisy channels,” in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500833.
- [100] X. Wei and C. Shen, “Federated learning over noisy channels: Convergence analysis and design examples,” *IEEE Trans. Cognitive Commun. and Networking*, vol. 8, no. 2, pp. 1253–1268, Jun. 2022. DOI: 10.1109/TCCN.2022.3140788.
- [101] T. Sery, N. Shlezinger, K. Cohen and Y. C. Eldar, “COTAF: Convergent over-the-air federated learning,” in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322580.

- [102] I. D. Schizas, A. Ribeiro and G. B. Giannakis, “Consensus in ad hoc wsns with noisy links—part I: Distributed estimation of deterministic signals,” *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 350–364, Jan. 2008. DOI: 10.1109/TSP.2007.906734.
- [103] A. Abur and A. G. Exposito, *Power system state estimation: theory and implementation*. CRC press, 2004.
- [104] L. Li, J. Zhong and M. Zhao, “Doppler-aided gnss position estimation with weighted least squares,” *IEEE Trans. Veh. Technol.*, vol. 60, no. 8, pp. 3615–3624, 2011. DOI: 10.1109/TVT.2011.2163738.
- [105] Z. Farbman, R. Fattal, D. Lischinski and R. Szeliski, “Edge-preserving decompositions for multi-scale tone and detail manipulation,” *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–10, 2008.
- [106] E. Lari, R. Arablouei, V. C. Gogineni and S. Werner, *Noise-robust and resource-efficient admm-based federated learning for wls regression*, 2024. [Online]. Available: <https://arxiv.org/abs/2409.13451>.
- [107] E. Lari, V. C. Gogineni, R. Arablouei and S. Werner, “Resource-efficient federated learning robust to communication errors,” in *Proc. IEEE Stat. Signal Process. Workshop*, 2023, pp. 265–269.
- [108] E. Lari, V. C. Gogineni, R. Arablouei and S. Werner, “Continual local updates for federated learning with enhanced robustness to link noise,” in *Proc. Asia-Pacific Signal Inf. Process. Assoc.*, 2023, pp. 1199–1203.
- [109] N. Gillis, “The why and how of nonnegative matrix factorization,” *Connections*, vol. 12, no. 2, 2014.
- [110] A. Cichocki, R. Zdunek, A. H. Phan and S.-i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.
- [111] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca and R. J. Plemmons, “Algorithms and applications for approximate nonnegative matrix factorization,” *Comput. Stat. Data Anal.*, vol. 52, no. 1, pp. 155–173, 2007.
- [112] Y.-X. Wang and Y.-J. Zhang, “Nonnegative matrix factorization: A comprehensive review,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1336–1353, 2013. DOI: 10.1109/TKDE.2012.51.
- [113] D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Proc. Adv. Neural Inf. Process. Syst.*, T. Leen, T. Dietterich and V. Tresp, Eds., vol. 13, MIT Press, 2000.
- [114] M. Udell, C. Horn, R. Zadeh, S. Boyd *et al.*, “Generalized low rank models,” *Found. Trends Mach. Learn.*, vol. 9, no. 1, pp. 1–118, 2016.

-
- [115] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.
- [116] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
- [117] M. W. Spratling and P. Dayan, “Learning image components for object recognition,” *J. Mach. Learn. Res.*, vol. 7, no. 5, 2006.
- [118] A. Kumar and V. Sindhwani, “Near-separable non-negative matrix factorization with ℓ_1 and Bregman loss functions,” in *Proc. SIAM Int. Conf. Data Min.*, 2015, pp. 343–351.
- [119] W.-K. Ma, J. M. Bioucas-Dias, T.-H. Chan, N. Gillis, P. Gader, A. J. Plaza, A. Ambikapathi and C.-Y. Chi, “A signal processing perspective on hyperspectral unmixing: Insights from remote sensing,” *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 67–81, 2013.
- [120] D. Godfrey, C. Johns, C. Meyer, S. Race and C. Sadek, “A case study in text mining: Interpreting twitter data from world cup tweets,” *arXiv preprint arXiv:1408.5427*, 2014.
- [121] T.-H. Chan, W.-K. Ma, C.-Y. Chi and Y. Wang, “A convex analysis framework for blind separation of non-negative sources,” *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 5120–5134, 2008.
- [122] A. C. Türkmen, “A review of nonnegative matrix factorization methods for clustering,” *arXiv preprint arXiv:1507.03194*, 2015.
- [123] K. Devarajan, “Nonnegative matrix factorization: An analytical and interpretive tool in computational biology,” *PLoS Comput. Biol.*, vol. 4, no. 7, pp. 1–12, 2008.
- [124] C. Févotte, N. Bertin and J.-L. Durrieu, “Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis,” *Neural Comput.*, vol. 21, no. 3, pp. 793–830, 2009.
- [125] S. Bhattacharya and N. D. Lane, “Sparsification and separation of deep learning layers for constrained resource inference on wearables,” in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2016, pp. 176–189.
- [126] Y.-W. Chang, H.-Y. Chen, C. Han, T. Morikawa, T. Takahashi and T.-N. Lin, “FINISH: Efficient and scalable NMF-based federated learning for detecting malware activities,” *IEEE Trans. Emerg. Top. Comput.*, vol. 11, no. 4, pp. 934–949, 2023. DOI: 10.1109/TETC.2023.3292924.

- [127] Y. Koren, R. Bell and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Comput.*, vol. 42, no. 8, pp. 30–37, 2009. DOI: 10.1109/MC.2009.263.
- [128] P. Melville and V. Sindhvani, “Recommender systems,” *Encycl. Mach. Learn.*, vol. 1, pp. 829–838, 2010.
- [129] C. Guillemot and O. Le Meur, “Image inpainting: Overview and recent advances,” *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 127–144, 2014. DOI: 10.1109/MSP.2013.2273004.
- [130] C. Zhang, M. Ahmad and Y. Wang, “ADMM based privacy-preserving decentralized optimization,” *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 3, pp. 565–580, 2019.
- [131] E. Lari, R. Arablouei and S. Werner, “Privacy-preserving distributed non-negative matrix factorization,” in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 1022–1026.
- [132] W. Deng and W. Yin, “On the global and linear convergence of the generalized alternating direction method of multipliers,” *J. Sci. Comput.*, vol. 66, pp. 889–916, 2016.
- [133] Y. Wang, W. Yin and J. Zeng, “Global convergence of admm in nonconvex nonsmooth optimization,” *J. Sci. Comput.*, vol. 78, pp. 29–63, 2019.
- [134] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, Springer, 1999, pp. 223–238.
- [135] K. Kogiso and T. Fujita, “Cyber-security enhancement of networked control systems using homomorphic encryption,” in *Proc. IEEE Conf. Decis. Control*, 2015, pp. 6836–6843. DOI: 10.1109/CDC.2015.7403296.
- [136] M. Ruan, H. Gao and Y. Wang, “Secure and privacy-preserving consensus,” *IEEE Trans. Automat. Control*, vol. 64, pp. 4035–4049, 2019. DOI: 10.1109/TAC.2019.2890887.
- [137] R. Fischer, J. Skelley and B. Heisele, *The MIT-CBCL facial expression database*. [Online]. Available: <http://cbcl.mit.edu/software-datasets/FaceData2.html>.
- [138] Y. Zhang, Z. Peng, G. Wen, J. Wang and T. Huang, “Privacy preserving-based resilient consensus for multiagent systems via state decomposition,” *IEEE Trans. Control. Netw. Syst.*, vol. 10, no. 3, pp. 1172–1183, 2023. DOI: 10.1109/TCNS.2022.3182234.

- [139] L. Xiao, S. Boyd and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, 2007.
- [140] J. Zhang, J. Lu, J. Liang and K. Shi, "Privacy-preserving average consensus in multiagent systems via partial information transmission," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 53, pp. 2781–2791, 2023. DOI: 10.1109/TSMC.2022.3220578.
- [141] X. Chen, L. Huang, K. Ding, S. Dey and L. Shi, "Privacy-preserving push-sum average consensus via state decomposition," *IEEE Trans. Automat. Contr.*, vol. 68, no. 12, pp. 7974–7981, 2023. DOI: 10.1109/TAC.2023.3256479.
- [142] A.-R. Lagos, H. E. Psillakis and A. K. Gkesoulis, "Almost-sure finite-time stochastic min-max consensus," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 9, pp. 3509–3513, 2023. DOI: 10.1109/TCSII.2023.3265567.
- [143] Y. Zhang and S. Li, "Distributed biased min-consensus with applications to shortest path planning," *IEEE Trans. Automat. Contr.*, vol. 62, no. 10, pp. 5429–5436, Oct. 2017. DOI: 10.1109/TAC.2017.2694547.
- [144] J. Hu, Q. Sun, M. Zhai and B. Wang, "Privacy-preserving consensus strategy for secondary control in microgrids against multilink false data injection attacks," *IEEE Trans. Ind. Inform.*, vol. 19, no. 10, pp. 10 334–10 343, 2023. DOI: 10.1109/TII.2023.3240878.
- [145] B. C. Tedeschini, S. Savazzi and M. Nicoli, "Weighted average consensus algorithms in distributed and federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 12, no. 2, pp. 1369–1382, 2025. DOI: 10.1109/TNSE.2025.3528982.
- [146] H. Rezaee and F. Abdollahi, "Average consensus over high-order multiagent systems," *IEEE Trans. Automat. Contr.*, vol. 60, no. 11, pp. 3047–3052, Nov. 2015. DOI: 10.1109/TAC.2015.2408576.
- [147] V. Khatana and M. V. Salapaka, "Noise resilient distributed average consensus over directed graphs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 9, pp. 770–785, 2023. DOI: 10.1109/TSIPN.2023.3324583.
- [148] G. Oliva, R. Setola and C. N. Hadjicostis, "Distributed finite-time average-consensus with limited computational and storage capability," *IEEE Trans. Control. Netw. Syst.*, vol. 4, no. 2, pp. 380–391, Jun. 2017. DOI: 10.1109/TCNS.2016.2524983.

- [149] W. Chen, L. Liu and G.-P. Liu, “Privacy-preserving distributed economic dispatch of microgrids: A dynamic quantization-based consensus scheme with homomorphic encryption,” *IEEE Trans. Smart Grid*, vol. 14, no. 1, pp. 701–713, 2023. DOI: 10.1109/TSG.2022.3189665.
- [150] D. Deplano, N. Bastianello, M. Franceschelli and K. H. Johansson, “A unified approach to solve the dynamic consensus on the average, maximum, and median values with linear convergence,” in *Proc. IEEE Conf. Decis. Control*, 2023, pp. 6442–6448.
- [151] L. Rong, Y. Kan, X. Xie, G.-P. Jiang and S. Xu, “Edge-preserving consensus via non-recursive filters: A parallel system design,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 1, pp. 181–185, 2023. DOI: 10.1109/TCSII.2022.3204942.
- [152] L. Gao, Y. Zhou, X. Chen, R. Cai, G. Chen and C. Li, “Privacy-preserving dynamic average consensus via random number perturbation,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 70, no. 4, pp. 1490–1494, 2023. DOI: 10.1109/TCSII.2022.3219929.
- [153] E. Montijano, J. I. Montijano, C. Sagüés and S. Martínez, “Robust discrete time dynamic average consensus,” *Automatica*, vol. 50, no. 12, pp. 3131–3138, 2014.
- [154] M. Franceschelli, A. Giua and A. Pisano, “Finite-time consensus on the median value with robustness properties,” *IEEE Trans. Automat. Contr.*, vol. 62, no. 4, pp. 1652–1667, 2017. DOI: 10.1109/TAC.2016.2590602.
- [155] S. Yu, Y. Chen and S. Kar, “Dynamic median consensus over random networks,” in *Proc. IEEE Conf. Decis. Control*, 2021, pp. 5695–5702. DOI: 10.1109/CDC45484.2021.9683179.
- [156] S. Kar and J. M. F. Moura, “Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise,” *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 355–369, Jan. 2009. DOI: 10.1109/TSP.2008.2007111.
- [157] M. Abdelrahim, J. M. Hendrickx and W. Heemels, “Max-consensus in open multi-agent systems with gossip interactions,” in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 4753–4758. DOI: 10.1109/CDC.2017.8264362.
- [158] A. Nowzari and M. G. Rabbat, “Improved bounds for max consensus in wireless networks,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 2, pp. 305–319, Jun. 2019.

-
- [159] S. Zhang, C. Tepedelenlioglu, M. K. Banavar and A. Spanias, “Max consensus in sensor networks: Non-linear bounded transmission and additive noise,” *IEEE Sens. J.*, vol. 16, no. 24, pp. 9089–9098, Dec. 2016. DOI: 10.1109/JSEN.2016.2612642.
- [160] G. Muniraju, C. Tepedelenlioglu and A. Spanias, “Analysis and design of robust max consensus for wireless sensor networks,” *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 779–791, Dec. 2019.
- [161] N. K. D. Venkategowda and S. Werner, “Privacy-preserving distributed maximum consensus,” *IEEE Signal Process. Lett.*, vol. 27, pp. 1839–1843, 2020. DOI: 10.1109/LSP.2020.3029706.
- [162] M. Lippi, A. Furchi, A. Marino and A. Gasparri, “An adaptive distributed protocol for finite-time infimum or supremum dynamic consensus,” *IEEE Control Syst. Lett.*, vol. 7, pp. 401–406, 2023. DOI: 10.1109/LCSYS.2022.3188941.
- [163] D. Deplano, M. Franceschelli and A. Giua, “Dynamic min and max consensus and size estimation of anonymous multiagent networks,” *IEEE Trans. Automat. Contr.*, vol. 68, no. 1, pp. 202–213, 2023. DOI: 10.1109/TAC.2021.3135452.
- [164] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, pp. 48–61, Jan. 2009. DOI: 10.1109/TAC.2008.2009515.
- [165] E. Lari, R. Arablouei, N. K. Venkategowda and S. Werner, “Distributed maximum consensus over noisy links,” in *Proc. Eur. Signal Process. Conf.*, 2024, pp. 2247–2251.

