Doctoral theses at NTNU, 2024:92

François Gauthier

On Enabling Scalable, Personalized, and Private Federated Learning

Doctoral thesis

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

NTNU
Norwegian University of
Science and Technology

François Gauthier

On Enabling Scalable, Personalized, and Private Federated Learning

Thesis for the Degree of Philosophiae Doctor

Trondheim, March 2024

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

NTNU

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

ISBN 978-82-326-7778-8 (printed ver.)
ISBN 978-82-326-7777-1 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2024:92

Printed by NTNU Grafisk senter

Abstract

This thesis investigates distributed machine learning for emerging Internet of Things (IoT) and Cyber-Physical Systems (CPS) applications. These applications involve large-scale data collection from distributed, often privately owned devices, which raises many logistical, moral, and legal issues if centralized processing is used. Therefore, this thesis explores how to take advantage of the local processing power of the devices and enable them to collaborate via inter-device communication to achieve a common learning goal without explicit disclosure of local data.

Federated learning is a framework for distributed machine learning that can address these issues. This thesis aims to develop and analyze new federated learning algorithms that overcome some of the practical challenges of distributed machine learning encountered in IoT and CPS settings. In particular, it tackles the following challenges.

- The scalability of the distributed machine learning architecture with respect to the increasing number of participating devices.
- The privacy preservation of the data owners from both external and internal adversaries.
- The robustness and efficiency of the distributed machine learning process under resource constraints and device failures.
- The personalization of the machine learning models for different devices and tasks within the same network.

The main contributions of this thesis are as follows. The first contribution of this thesis is to propose a peer-to-peer federated learning algorithm in which the par-

ticipating devices collaborate without the need for a coordinator while preserving their confidentiality. Secondly, we devise a robust online federated learning algorithm that can handle resource-constrained devices with sporadic participation and failures, as well as delays in a single-server architecture. Third, we develop a multi-server federated learning algorithm that supports personalized model training for device-specific tasks while preserving the data privacy of the participants. Finally, we advance peer-to-peer personalized federated learning with reinforcement learning techniques to enhance localized personalized learning.

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 August 2019 at the Department of Electronic Systems, NTNU, Trondheim, Norway. The work has been supervised by Professor Stefan Werner and co-supervised by Professor Pierluigi Salvo Rossi.

The members of the assessment committee are: Professor Zhi Tian, George Mason University, Associate Professor Paolo Di Lorenzo, Sapienza University of Rome, and Professor Kimmo Kansanen, Norwegian University of Science and Technology.

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

Acknowledgements

I express my sincere gratitude to my supervisor, Professor Stefan Werner, for giving me the opportunity to join his team, for always being available when I needed help, and for his unwavering support, guidance, and belief in my abilities throughout my Ph.D. journey. I would also like to thank my co-supervisor, Professor Pierluigi Salvo-Rossi, for his support and availability during these formative years. I am deeply thankful to the members of the assessment committee, Professor Zhi Tian, Associate Professor Paolo Di Lorenzo, and Professor Kimmo Kansanen, for taking the time to appraise this thesis.

I wish to thank all my co-authors for their valuable contributions to the papers included in this thesis, and for the stimulating discussions that emerged from our collaboration. Special thanks to Dr. Vinay Chakravarthi Gogineni for generously sharing with me his expertise and insights, which have greatly influenced the work presented herein.

I extend my appreciation to all the members of the Department of Electronic Systems, especially the Signal Processing Group, and the faculty, for fostering a supportive and intellectually stimulating environment conducive to research.

My deepest appreciation goes to my family, especially my fiancée, parents, and sister, for their unwavering support and encouragement. To my colleagues and friends, thank you for infusing these years with laughter and light-heartedness, providing a refreshing contrast to the seriousness of academia.

Contents

List of Figures

Abbreviations and Symbols

Abbreviations

ADMM Alternating direction method of multipliers

BRL Batch Reinforcement learning

CDP Concentrated differential privacy

DP Differential privacy

DPG Deterministic policy gradient

FL Federated learning

GFL Graph federated learning

MSD Mean squared deviation

MSE Mean squared error

NFL Networked federated learning

NPFL Networked personalized federated learning

OFL Online federated learning

PFL Personalized federated learning

PGFL Personalized graph federated learning

PS Partial-sharing-based

RFF	Random Fourier feature
RL	Reinforcement learning
SAC	Stochastic actor critic
zCDP	Zero-concentrated differential privacy

Symbols

$\mathcal{D}_k$	Dataset of client k
$\mathcal{C}$	Set of clients
$\mathcal{C}_s$	Set of clients associated with server s
$\mathcal{C}_{(q)}$	Set of clients of cluster q
$\mathcal{E}$	Set of edges
$\mathcal{G}$	Graph
$\mathcal{N}_k$	Neighborhood of client k (does not contain k)
$\mathcal{N}_s$	Neighborhood of server s (contains s)
$\mathcal{N}_s^-$	Neighborhood of server s (does not contain s)
$\mathcal{S}$	Set of servers
$\mathcal{Q}$	Set of clusters
$\mathbf{w}^{(n)}$	Global shared model at iteration n
$\mathbf{w}_k$	Model of client k
$\mathbf{w}_s$	Model of server s
$\widetilde{\mathbf{w}}_k$	Model of client k after noise perturbation.
$\boldsymbol{\xi}_k$	Client k's perturbation noise
d	Number of model parameters
D_k	Number of samples in the dataset of client k
e	Euler's number
K	Number of clients

Chapter 1

Introduction

As more and more devices are given sensing and communication capabilities, the Internet of Things (IoT) is unveiling new opportunities to improve our surroundings and lives. It is a centerpiece of many current advancements, such as Cyber-Physical Systems (CPS), Industry 4.0, smart healthcare, smart city, smart home, etc., and is yet to be utilized to its full potential. More importantly, the number of connected devices in our environment and the average device data-gathering capacity are increasing exponentially, leading to the availability of much more data than what our current infrastructure can handle and utilize [1]. An IoT network creates added value by extracting information from the data collected by distributed devices, a process known as learning. This information can be used for inference and decision-making, either manual or automated. Many systems rely upon such data-driven mechanisms, such as anomaly detection, autonomous vehicles, public transport optimization, and weather prediction.

However, making the original device output, also called raw data, available to the end user is neither efficient nor desirable [2]. The sheer quantity of data gathered by modern connected devices is a strain on existing infrastructure; and for devices equipped with cameras and microphones, for instance, the recordings can contain private information subject to regulations such as GDPR. Given that most existing devices possess local computational power, extracting the desired information from the data locally and sharing only its compact mathematical representation, called a model, is preferable. This reduces the communication load and, therefore, the strain on the digital infrastructure, and diminishes the privacy risks associated with the information transfer. Moreover, this distributed approach presents the advantage of reducing the computational load of the device receiving the models, as those are easier to analyze than raw data. In particular, existing servers and cloud

architecture, usually serving as aggregators, could not handle the quantity of data generated by modern distributed devices [3].

Another challenge associated with the IoT is the diversity of participants in modern applications. While traditionally identical ad-hoc sensors would be deployed for a specific task, recent applications rely on the voluntary participation of various devices. Those devices may be positioned at the edge, meaning they might be intermittently available, suffer from resource constraints, be concurrently solicited, or stop functioning. For instance, smartphones with varying amounts of relevant information and battery may participate in a variety of learning tasks through applications. Alternatively, surveillance cameras with limited communication and computation capacity may be employed to supplement the existing infrastructure in smart city planning. In those and many other applications, device diversity and imperfect communication channels lead to uneven participation and delays in the exchanged messages. Statistical heterogeneity refers to the disparity of devices' data quantity and quality, while system heterogeneity refers to the devices' various computational and communication capacities. Both of those pose challenges to the effective utilization of IoT networks [4–7].

Considering these concerns, it is imperative that large-scale networks establish a trustworthy and reliable distributed learning framework capable of learning from heterogeneous data arising over a myriad of devices with varying capabilities while complying with the individual privacy preferences of data holders.

Federated Learning (FL), illustrated in Fig. 1.1 on the right, is a distributed learning framework that addresses many of the abovementioned challenges and can be extended to address many more. It has been proposed in [3] as an alternative to centralized learning, illustrated in Fig. 1.1 on the left. In FL, numerous and varied distributed devices, called clients, perform local learning on their own data. These clients then transfer their models to the server or cloud for aggregation into a global model. At this stage, a weighting mechanism may be implemented to emphasize specific clients' data. Subsequently, the server communicates the global model back to the clients to replace their local model. This iterative procedure continues for a predefined number of iteration rounds or until a pre-specified convergence criterion is met, allowing a network of devices to learn from each other. FL can seamlessly handle unbalanced data across large-scale networks with varying statistical properties.

Unfortunately, the original definition of FL fails to scale with the exponentially growing number of IoT devices and the ever-increasing needs of learning tasks. In particular, the single-server architecture is strained in applications where many devices exchange high-dimensional models. This has led to alternative architec-

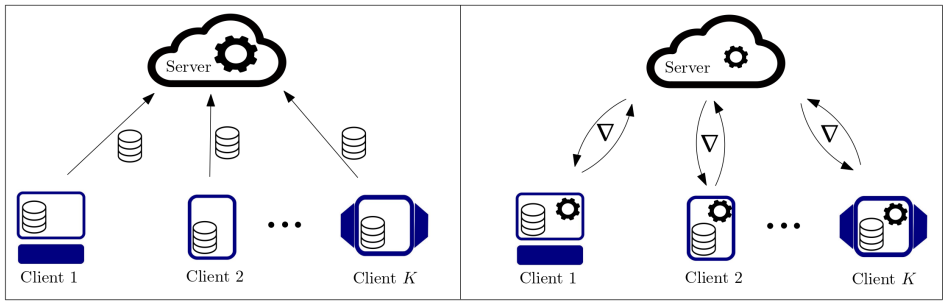


Figure 1.1: Conventional learning over distributed devices where data is moved (left) and a typical FL implementation where models are exchanged (right).

tures such as networked FL [8] and graph FL [9] that scale efficiently with the growing number of clients. As well as alternative FL implementations that further reduce the communication and computation strain on the distributed devices and server [10–12]. This latter alternative presents the advantage of making participation in the learning task more accessible.

As the size of the FL network increases, and more geographically dispersed and diverse clients take part in the learning process, the growing diversity among the participating devices becomes inconsistent with the single global model. Instead of splitting the network into smaller networks and missing out on potential collaboration between distant devices, it is preferable to allow the network to learn several models [13]. Personalized FL enables FL networks to learn as many models as there are clients or groups of similar clients, called clusters. To do so, it utilizes intra-cluster and inter-cluster learning, referring to learning from clients with similar and different learning tasks, respectively [14–16]. Careful weightage of intra- and inter-cluster learning enables personalized FL to achieve great performance in applications where device-specific behaviors are expected [17].

Many modern devices, such as autonomous vehicles [18], smart healthcare devices [19], and augmented reality glasses, can benefit from participating in an FL process. However, these devices are used for time-sensitive applications and cannot repeatedly re-train models using all the available data. More so, for some of them, such as autonomous vehicles, reacting quickly to changes in the environment is paramount. Online FL has been developed for such applications. In online FL, clients share models learned from the last available section of their data stream, and the server rapidly updates and redistributes the global model on-the-fly [20].

Although raw data is not exchanged between devices in FL, the exchanged models are learned from this data and, therefore, contain a part of its information. Ex-

ternal eavesdroppers or honest-but-curious participants may take advantage of this privacy hazard. Such adversaries can infer private information by reconstructing the data from the models, so-called reconstruction attacks. As in most applications, not all devices can be trusted; one must ensure that the clients do not jeopardize their data privacy by participating in the learning process. To do so, it is common to implement local privacy-preserving mechanisms that limit the clients' information leakage, such as differential privacy [21].

This thesis responds to the growing academic and industrial interest in FL. It takes part in the scientific community's endeavor to design, analyze, and evaluate FL algorithms that overcome the real-world challenges of distributed machine learning. Since FL involves trade-offs between various constraints and performance, no single algorithm can be optimal across a wide range of applications. This thesis advances the field of FL by developing new solutions that address limitations in the existing literature and meet practical needs.

1.1 Objectives

The purpose of this thesis is to design, analyze, and implement FL algorithms to solve learning tasks that arise in the real world. In particular, we concentrate on four key aspects: scalability, data privacy, personalization, and performance. The objectives can be summarized as follows.

O1 To provide a learning architecture that scales with the number of participating devices and accommodates their potential resource constraints.

O2 To safeguard the data integrity and privacy of the participating devices from external and internal adversaries.

O3 To enable the learning of personalized models while taking advantage of the global learning process.

1.2 List of Publications

The results contained in this thesis are published in eight papers. They include four conferences, three journals, and a magazine. Below is the list of the publications.

P1 F. Gauthier, C. Gratton, N. K. D. Venkategowda and S. Werner, "Privacy-Preserving Distributed Learning with Nonsmooth Objective Functions", in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, November, 2020.

P2 F. Gauthier, C. Gratton, N. K. D. Venkategowda and S. Werner, "Private Networked Federated Learning for Nonsmooth Objectives", submitted to *Elsevier Signal Processing*.

P3 F. Gauthier, V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Resource-aware asynchronous online federated learning for nonlinear regression", in *Proceedings of IEEE International Conference on Communications*, May, 2022.

P4 F. Gauthier, V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Asynchronous Online Federated Learning with Reduced Communication Requirements", in *IEEE Internet of Things Journal*.

P5 F. Gauthier, V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Clustered Graph Federated Personalized Learning", in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, October, 2022.

P6 F. Gauthier, V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Personalized Graph Federated Learning with Differential Privacy", in *IEEE Transactions on Signal and Information Processing over Networks*.

P7 F. Gauthier, V. C. Gogineni, and S. Werner, "Networked Personalized Federated Learning Using Reinforcement Learning", in *Proceedings of IEEE International Conference on Communications*, June, 2023.

P8 V. C. Gogineni, S. Werner, F. Gauthier, Y. Huang, and A. Kuh, "Personalized online federated learning for IoT/CPS: challenges and future directions", in *IEEE Internet of Things Magazine*, December, 2022.

1.3 Structure and Contributions

The rest of this thesis is organized as follows. Chapter 2 provides an overview of the various FL architectures and variations relevant to this thesis, as well as an introduction to differential privacy in the context of FL. The contribution of this thesis is divided into four chapters, each addressing the abovementioned research objectives in a different manner.

- Chapter 3 proposes a peer-to-peer FL algorithm that does not need a central coordinator and protects data privacy with differential privacy. It also ensures fast convergence and handles nonsmooth objectives.

- Chapter 4 develops an online FL algorithm dealing with resource-constrained clients in real-time. It handles poor, unpredictable, and unreliable client participation in an agile manner and reduces their load.
- Chapter 5 designs a multi-server FL algorithm that enables cluster-specific model training for local tasks. It alleviates data scarcity within clusters by using inter-cluster learning.
- Chapter 6 extends the inter-cluster learning in personalized FL to peer-to-peer settings by using reinforcement learning to control the amount of inter-cluster learning at each client, adapting to network topology and data distribution.

Finally, Chapter 7 concludes the thesis and proposes future research directions.

Chapter 2

Federated Learning and Differential Privacy

This chapter provides the background information required for the rest of the thesis. Section 2.1 introduces federated learning in its conventional form, as well as two alternative scalable architectures. Section 2.2 presents online federated learning, an alternative federated learning framework that operates in real-time over data streams. Section 2.3 presents personalized federated learning, an adaptation of the federated learning framework in which client-specific models are learned. Section 2.4 introduces differential privacy, its application to FL, and its most common variations, including zero-concentrated differential privacy.

2.1 Federated Learning

Many machine learning applications involve multiple devices, each with access to a portion of the training data. Unlike conventional centralized machine learning approaches that consolidate all data at a single node, or processing center, distributed machine learning methods keep the training data decentralized and collaboratively learn a solution. The two main advantages of such approaches are to prevent raw data transfer, which may lead to privacy and communication challenges, and to reduce the computational load at the processing center.

Federated learning (FL), introduced in [3], is a distributed machine learning approach that uses a server to orchestrate the collaborative learning process. It was designed to handle systems where data distribution across devices the data is imbalanced and not identically and independently distributed.

In FL, clients train models on their local data and share these models with the

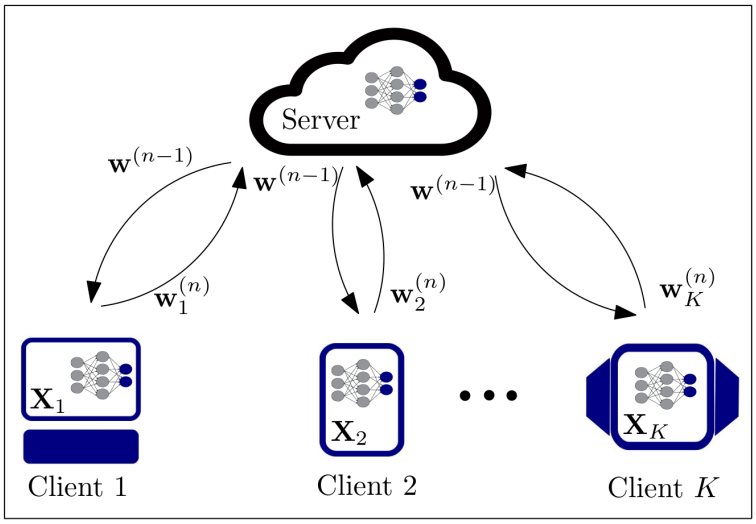


Figure 2.1: Conventional federated learning.

server. The server aggregates all received models into a global model, usually using a weighted averaging mechanism. This global model is then shared with all clients, allowing them to benefit from information across the network. The clients then train their local copy of the global model on their data to form new local models. This iterative process is repeated until a convergence criterion is met or after a predetermined number of iterations.

Throughout the thesis, we denote the set of clients as $\mathcal{C}$, where each client represents a device or node in our distributed learning system. Each client $k \in \mathcal{C}$ has access to its own dataset $\mathcal{D}_k = (\mathbf{X}_k, \mathbf{y}_k)$ consisting of a data matrix $\mathbf{X}_k$ and a response vector $\mathbf{y}_k$. At iteration n, a client k trains the model $\mathbf{w}_k^{(n)}$ using its data $\mathcal{D}_k$ and the global shared model $\mathbf{w}^{(n-1)}$. The FL behavior is illustrated in Fig. 2.1, and a conventional federated learning algorithm is provided in Algorithm. 1, where μ is the learning rate which controls how much the model changes in response to the estimated error each time the model weights are updated. The function f represents the specific objective used to train the models on their respective datasets.

2.1.1 Graph Federated Learning

Although the FL framework offers flexibility over centralized learning, its reliance on a single server to receive the local models trained by all the clients is a significant limitation. If a given system is poorly scaled or suddenly receives increased participation, the large number of clients and the high dimension of the models to be shared would strain the communication channels and can lead to

Algorithm 1 Conventional federated learning algorithm

Initialization: $\mathbf{w}^{(0)}$ and $\mathbf{w}_k^{(0)}, k \in \mathcal{C}$ are set to $\mathbf{0}$
Procedure at client k
for iteration $n = 1, 2, \ldots$ **do**
  Receive $\mathbf{w}^{(n)}$ from the server.
  $\mathbf{w}_k^{(n+1)} = \mathbf{w}^{(n)} + \mu f(\mathcal{D}_k)$.
  Share $\mathbf{w}_k^{(n+1)}$ with the server.
end for
Procedure at the server
for iteration $n = 1, 2, \ldots$ **do**
  Receive $\mathbf{w}_k^{(n)}$ from all clients $k \in \mathcal{C}$
  $\mathbf{w}^{(n+1)} = \frac{1}{|\mathcal{C}|} \sum_{k \in \mathcal{C}} \mathbf{w}_k^{(n)}$.
  Share $\mathbf{w}^{(n+1)}$ with the clients.
end for

blockages. Furthermore, the potentially complex aggregation mechanisms handling many local models at the server might create computational bottlenecks, thus limiting the learning speed.

Several works have studied alternatives to the single-server FL architecture to address this limitation. Among them, [22] proposes a graph federated learning architecture comprising several interconnected servers, each associated with its own set of clients. In graph FL, learning is done in two stages. First, each server performs learning with its respective clients, referred to as intra-server learning. Thereafter, the servers, amongst themselves, perform distributed learning, referred to as inter-server learning.

Figure 2.2 illustrates a graph federated architecture. A server s is associated with the clients in the set $\mathcal{C}_s$, maintains a local model $\mathbf{w}_s$, and can communicate with the servers in its neighborhood $\mathcal{N}_s$. By convention throughout the thesis, s is included in $\mathcal{N}_s$ and we denote $\mathcal{N}_s^- = \mathcal{N}_s \backslash s$. At iteration n, a client $k \in \mathcal{C}_s$ uses its local data and the model $\mathbf{w}_s^{(n-1)}$ to update its local model $\mathbf{w}_k$, which is then shared with server s. A server s aggregates the models received from the clients in $\mathcal{C}_s$ and share this aggregate with its neighbors in $\mathcal{N}_s$. It then uses the aggregates received from its neighbors alongside its own aggregate to form the model $\mathbf{w}_s^{(n)}$, which is then shared with the clients in $\mathcal{C}_s$.

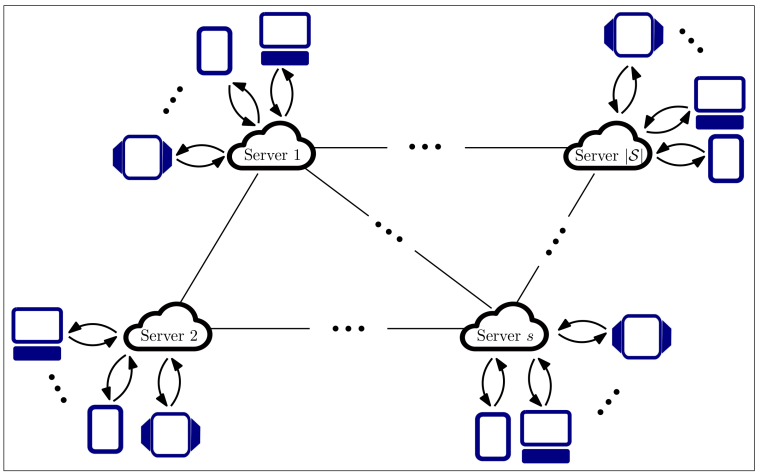


Figure 2.2: Graph federated learning.

2.1.2 Networked Federated Learning

Networked federated learning is an alternative architecture for FL that also addresses the limitations associated with a single-server architecture. It relies on the fully distributed learning architecture, where participating devices or nodes communicate in a peer-to-peer manner to collaboratively achieve consensus over a global shared model. Given the limitations of distributed devices, it is assumed that a given client can only communicate with a limited number of other devices, its neighbors. The resulting network forms a graph where the clients constitute the node set, and the presence of a communication channel between two clients forms an edge. Assuming that the graph is connected, information is transmitted throughout the network until the clients reach a consensus.

Networked FL typically exhibits slower convergence speed than conventional and graph FL because information dissemination throughout the network takes longer. However, its inherent adaptability to dynamic network conditions and robustness to device failure make it a compelling option for specific applications. In addition, it is the de-facto architecture in scenarios where the deployment of servers is not feasible, whether due to logistical constraints or privacy considerations.

Figure 2.3 illustrates a networked architecture. A client $k \in \mathcal{C}$ can communicate with the clients in $\mathcal{N}_k$, its neighborhood. By convention throughout the thesis, k is not included in $\mathcal{N}_k$. For instance, in Fig. 2.3, the neighborhood of client 2 is composed of clients 1 and 4. At iteration n, a client $k \in \mathcal{C}$ receives the models $\mathbf{w}_l^{(n-1)}, l \in \mathcal{N}_k$ from its neighbors, and uses them alongside its local data $\mathcal{D}_k$ to

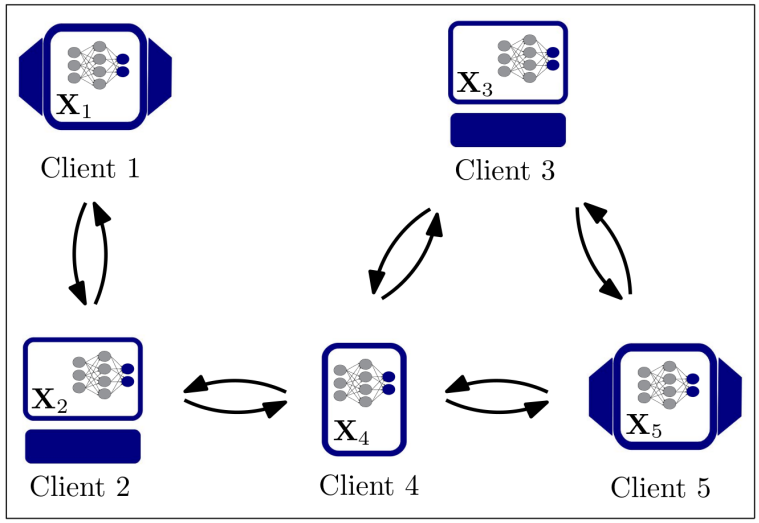


Figure 2.3: Networked federated learning.

form its new local model, $\mathbf{w}_k^{(n)}$.

2.2 Online Federated Learning

An IoT network is a typical application where participating clients continuously receive data. Such data streams collected by edge devices in IoT networks may be used to perform real-time local learning and update global models. Importantly, the underlying model is likely to evolve with time. In this regard, more emphasis should be given to recent data during the learning process. Furthermore, in some applications, e.g., wireless communications and networked vehicles, the real-time responsiveness of the model is paramount. Because the traditional FL learns from fixed data batches that are not usually timely enough, it is not appropriate for these scenarios. Online federated learning (Online FL) [20] is a viable solution for such applications.

In online FL, the clients receive continuous streams of data. At each iteration, they perform local training of their models using the most recent data only, then share their updated local models with the server. The server aggregates the local models received from the clients to build the new global model, which is then shared with the clients and replaces their local models. This behavior makes online FL learn from streaming data in a computationally efficient manner without requiring periodic retraining of the model. The learning rate is especially critical in online FL as it dictates how fast the network can adapt to underlying model change. It must be fixed according to the expected relative relevance of older and newer data

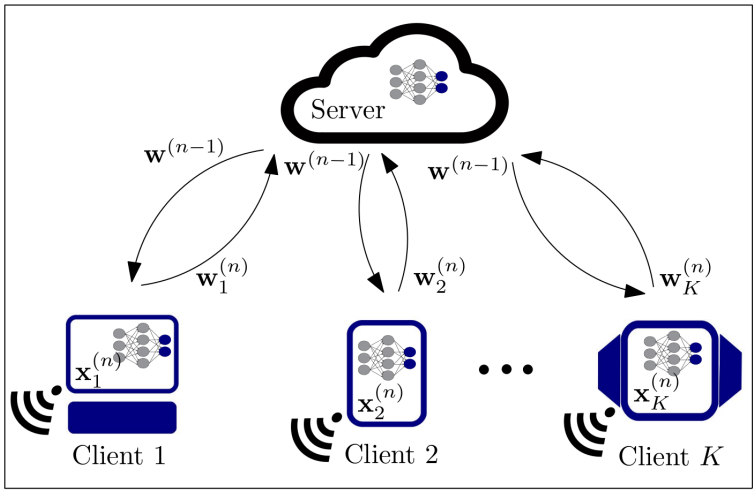


Figure 2.4: Online federated learning.

to compromise between accuracy and reactivity of the network.

Figure 2.4 illustrates the behavior of online FL. At iteration n, a client $k \in \mathcal{C}$ receives a new stream of data, $\mathbf{x}_k^{(n)}, y_k^{(n)}$, and uses it alongside the previous server model $\mathbf{w}^{(n-1)}$ to train its local model $\mathbf{w}_k^{(n)}$, which is then shared with the server. The server uses the received clients' models to update its model $\mathbf{w}^{(n-1)}$ into $\mathbf{w}^{(n)}$, with a carefully chosen learning rate.

2.3 Personalized Federated Learning

In many IoT and CPS applications, such as networked vehicles [18, 23] and personalized healthcare [19, 24], data from devices exhibit varying statistical properties due to their geographical dispersion or the intrinsic characteristics of the underlying processes. The more critical aspect of these edge devices is that they behave semi-independently and collaborate among themselves mainly to improve their decision-making capability. Therefore, learning a single universal model for device-specific tasks is neither reasonable nor realistic. To adequately address these problems, it is necessary to allow each device to learn and use a local, personalized model [13, 25–27]. This model corresponds, for instance, to a networked vehicle's close neighborhood and a patient's health status. In some other applications, it is convenient for groups of clients, called clusters, to share a single model, i.e., a personalized model for a cluster instead of an individual client.

The intruiging aspect of these device-specific tasks is that they differ but may be related [28], i.e., some similarities may exist between the tasks performed by in-

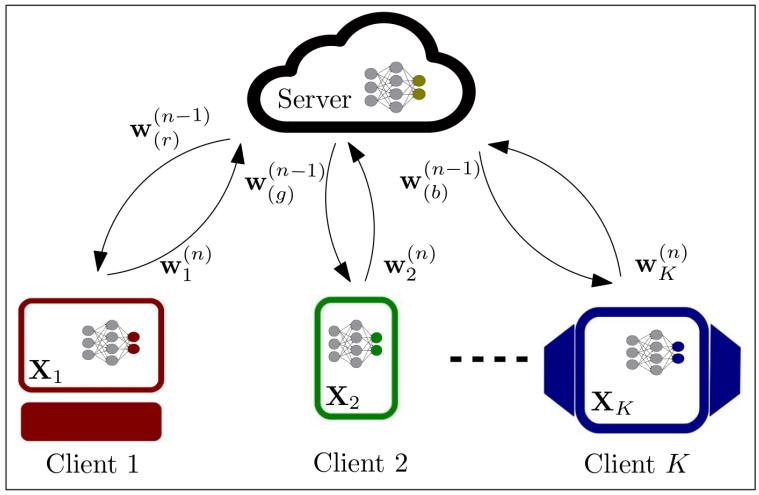


Figure 2.5: Personalized federated learning.

dividual clients or clusters (e.g., $\mathcal{F}$-similar tasks [29]). It is often necessary to take advantage of these similarities when a limited amount of data is available at a single client or cluster to build satisfactory personalized models [17, 28]. Promoting similarities in device-specific tasks during the learning process enables every client or cluster to build their best-customized models tailored to their local needs. Striking the right balance between leveraging similarities and maintaining cluster independence is complex and significantly impacts performance. For this reason, careful control of inter-cluster learning is at the heart of many works on personalized FL.

Figure 2.5 illustrates the behavior of personalized FL. The clients of a cluster $q \in \mathcal{Q}$ are grouped into the set $\mathcal{C}_{(q)}$, $\mathcal{Q}$ being the set of clusters. The server's model for the clients in $\mathcal{C}_{(q)}$ is denoted $\mathbf{w}_{(q)}$. At iteration n, a client $k \in \mathcal{C}_{(q)}$ uses its local data and the server model $\mathbf{w}_{(q)}^{(n-1)}$ to update its local model $\mathbf{w}_k^{(n)}$, which is then shared with the server. The server aggregates for each cluster q the models received from the clients in $\mathcal{C}_{(q)}$. It then refines those aggregates with inter-cluster learning to form the new models $\mathbf{w}_{(q)}^{(n)}, \forall q \in \mathcal{Q}$.

2.4 Differential Privacy

One of the major concerns in IoT and CPS is that many emerging distributed systems, such as smart homes, smart cities, and autonomous vehicles, rely heavily on data collected from private individuals. Those individuals wish to preserve the confidentiality of their own data [30]. Most existing works assume the trustwor-

thiness of all involved devices or of the devices having direct access to shared sensitive data, which may not be realistic. Encrypting data before communication, for instance, ensures privacy with respect to external agents but does not offer protection from compromised or ill-intentioned participants. Moreover, centralizing private data can lead to potential leaks, for instance, via poor anonymization, as was the case with Netflix's user data [31]. A viable approach to ensure data confidentiality with respect to all participants and external devices is to add noise to the shared data.

In FL, raw data is never shared; only models trained on this data by the participants are exchanged. Those models, however, contain information about the initial data. Their confidentiality must, therefore, be preserved as well. Given the distributed nature of FL, it is challenging to prevent eavesdroppers, i.e., external adversaries listening to the exchanged messages, and honest-but-curious clients, i.e., participating clients gathering information to send to a third party, having access to the models shared by the clients. Those adversaries may attempt to reconstruct the initial data using the exchanged models in what is called a reconstruction attack or model inversion attacks [32]. Therefore, mitigating information leakage by adding noise to the exchanged models is imperative. Naturally, adding noise to the models diminishes the learning speed and accuracy. For this reason, it is preferable to perturb the models only as much as necessary to ensure satisfactory privacy protection.

Differential privacy (DP), introduced in [21], is a mathematical definition that enables the quantification of privacy protection. It quantifies how precisely an adversary may reconstruct data. Adding noise protects clients from reconstruction attacks by ensuring minimal detectable changes in the output of a random process, regardless of whether an individual data sample is present during the computation [21, 33, 34]. Precisely, it corresponds to a bound on the ratio of probabilities of the output of a random process computed on two neighboring datasets, i.e., datasets that differ in only one data sample. This can be expressed as

$$\frac{P(\mathcal{K}(\mathcal{D}_1) \in \mathcal{S})}{P(\mathcal{K}(\mathcal{D}_2) \in \mathcal{S})} \leqslant e^{\epsilon}, \tag{2.1}$$

where $\mathcal{K}$ is a random process, $\mathcal{S} \subseteq \text{range}(\mathcal{K})$, and $\mathcal{D}_1$ and $\mathcal{D}_2$ are two neighboring datasets. Note that the exact definition is $P(\mathcal{K}(\mathcal{D}_1) \in \mathcal{S}) \leqslant e^{\epsilon} P(\mathcal{K}(\mathcal{D}_2) \in \mathcal{S})$, so that the case where both probabilities are null satisfies the property. For simplicity, we will omit this case and use the expression in (2.1) for the rest of the thesis.

2.4.1 Local Differential Privacy for FL

In this thesis, we use DP specifically to ensure participants' privacy in FL algorithms. This can be achieved by either perturbing the local model prior to communication or perturbing the local objective [35]. We chose to implement differential privacy in FL by perturbing the exchanged models. That is, if $\mathbf{w}_k$ is the model for client k, its perturbed model that is shared with the server or other clients is given by

$$\widetilde{\mathbf{w}}_k = \mathbf{w}_k + \boldsymbol{\xi}_k. \tag{2.2}$$

We then control the noise perturbation $\boldsymbol{\xi}_k$ to implement local DP providing satisfactory protection.

To implement local DP, we see the learning process at the client as a random process, where (2.2) ensures randomness even if the training process is deterministic. The implementation comprises two steps. The first step is to quantify the impact of an individual data sample on the model. The second step is to control the noise perturbation to ensure satisfactory protection.

For any variation of DP, the computation of the impact of an individual data sample on the model is identical. We consider a client k that trains a model $\widetilde{\mathbf{w}}_{k,\mathcal{D}_k}$ on its dataset $\mathcal{D}_k$. We consider a neighboring dataset $\mathcal{D}'_k$ and compute the model $\widetilde{\mathbf{w}}_{k,\mathcal{D}'_k}$ trained on this dataset. Further, we define the l_2-norm sensitivity, which quantifies the impact of an individual data sample on the model, as

$$\Delta_{k,2} = \max_{\mathcal{D}_k, \mathcal{D}'_k} ||\widetilde{\mathbf{w}}_{k,\mathcal{D}_k} - \widetilde{\mathbf{w}}_{k,\mathcal{D}'_k}||. \tag{2.3}$$

The second step in the implementation of DP is to decide how much the model needs to be perturbed to cripple the adversaries' ability to infer private information. Two aspects need to be taken into account for this purpose. The first is the subject of the next section: multiple messages are exchanged in FL, which may increase an adversary's ability to infer private information. The second is the subject of the following sections: different variations of DP exist, each with its own purpose and privacy-accuracy trade-off.

2.4.2 Privacy Composition for Iterative Processes

An FL algorithm is an iterative process in which server and clients exchange models until a certain convergence criterion is met. From a privacy standpoint, each client repeatedly shares models containing information on its local data. In the worst-case scenario, an adversary will aggregate all the messages a given client sends to infer information. It is crucial to consider this fact when evaluating the privacy leakage of a given algorithm.

At a given iteration n, the effective privacy leakage of client k in an FL algorithm is equal to the privacy leakage of the set $\{\widetilde{\mathbf{w}}_k^{(i)}, 0 < i \leqslant n\}$ of the models shared by this client until the current iteration. This privacy leakage is greater than the privacy leakage of the last shared model, $\widetilde{\mathbf{w}}_k^{(n)}$, and increases with the number of iterations.

It is possible for the privacy budget to evolve dynamically throughout the computation, leading to iteration-dependent noise perturbation. This is called dynamic differential privacy [36], and can be illustrated as

$$\widetilde{\mathbf{w}}_k^{(n)} = \mathbf{w}_k^{(n)} + \boldsymbol{\xi}_k^{(n)}, \tag{2.4}$$

where the noise sequence $\boldsymbol{\xi}_k^{(n)}$ for client k is varying. One of the most common uses of dynamic DP in FL is to increase the privacy budget throughout the computation to increase accuracy.

Different notions of DP exist and are controlled by different parameters. For each variation, composition theorems exist, quantifying the privacy guarantees of a random process composed of several steps that may have different privacy budgets. We utilize those theorems to evaluate the privacy leakage throughout the entire computation. The following sections present the notions most relevant to this thesis and their composition theorems.

2.4.3 (ϵ) - Differential Privacy

The (ϵ) - DP variation is the original DP definition, illustrated in (2.1) in its mathematical form. For a client k participating in an FL algorithm, the (ϵ) - DP of its training process can be expressed as

$$\frac{P(\widetilde{\mathbf{w}}_{k,\mathcal{D}_k} \in \mathcal{W})}{P(\widetilde{\mathbf{w}}_{k,\mathcal{D}'_k} \in \mathcal{W})} \leqslant e^{\epsilon}, \tag{2.5}$$

where $\mathcal{W}$ is a subset of the ensemble of possible values for the model $\widetilde{\mathbf{w}}_k$, that is, $\mathcal{W} \subseteq \text{Range}(\widetilde{\mathbf{w}}_k)$, and $\widetilde{\mathbf{w}}_{k,\mathcal{D}_k}$ and $\widetilde{\mathbf{w}}_{k,\mathcal{D}'_k}$ are models trained on neighboring datasets $\mathcal{D}_k$ and $\mathcal{D}'_k$.

Given (2.5) and the l_2-norm sensitivity, one can quantify the noise perturbation required to achieve the desired level of privacy protection at a given iteration. Further, the composition theorem for (ϵ) - DP is available in [33, Theorem 3.15] and can be expressed as follows.

Theorem 2.1. Let $\mathcal{M}_i$ be an ϵ_i-DP random process for $i \in \{1, \ldots, n\}$. Then the random process $\mathcal{M} = \big(\mathcal{M}_1, \ldots, \mathcal{M}_n\big)$ is $\sum_{i=1}^{n} \epsilon_i$-DP.

This result illustrates that in the event of the composition of several differentially private random processes, the privacy loss increases in an additive manner.

2.4.4 (ϵ, δ) - Differential Privacy

The (ϵ, δ)-DP variant is the most commonly used. In this implementation, the two parameters ϵ and δ dictate the privacy leakage at a given iteration. In practice, this implementation of DP ensures that at any given iteration, the difference between two models computed from neighboring datasets is at most ϵ with probability $1-\delta$. Precisely, given that the models are perturbed to implement (ϵ, δ)-DP, we have

$$P\left(\frac{P(\widetilde{\mathbf{w}}_{k,\mathcal{D}_k} \in \mathcal{W})}{P(\widetilde{\mathbf{w}}_{k,\mathcal{D}'_k} \in \mathcal{W})} \leqslant e^{\epsilon}\right) > 1 - \delta, \tag{2.6}$$

where $\mathcal{W} \subseteq \text{Range}(\mathbf{w})$, and $\widetilde{\mathbf{w}}_{k,\mathcal{D}_k}$ and $\widetilde{\mathbf{w}}_{k,\mathcal{D}'_k}$ are models trained on neigbhoring datasets $\mathcal{D}_k$ and $\mathcal{D}'_k$.

The effective privacy leakage increases with the number of exchanged messages. The composition theorem for (ϵ, δ)-DP is available in [33, Theorem 3.16] and can be expressed as follows.

Theorem 2.2. Let $\mathcal{M}_i$ be an (ϵ_i, δ_i)-DP random process for $i \in \{1, \ldots, n\}$. Then the random process $\mathcal{M} = \big(\mathcal{M}_1, \ldots, \mathcal{M}_n\big)$ is $\Big(\sum_{i=1}^{n} \epsilon_i, \sum_{i=1}^{n} \delta_i\Big)$-DP.

2.4.5 Concentrated and Zero-Concentrated Differential Privacy

Concentrated differential privacy (CDP) was introduced in [37] as a relaxation of $(\boldsymbol{\epsilon}, \boldsymbol{\delta})$-DP. Its purpose is to enable better accuracy under an identical privacy budget when several messages are exchanged, making it ideal for iterative algorithms and the implementation of dynamic DP.

Zero-concentrated differential privacy (zCDP) was introduced in [38] as a relaxation of CDP. It builds upon a special case of CDP, and offers a notion easier to analyze while offering the same benefits. The privacy protection in zCDP is controlled by a single parameter, ϕ, and its composition theorem is given in [38, Lemma 1.7] and can be expressed as follows.

Theorem 2.3. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be ϕ_1- and ϕ_2-zCDP random processes, respectively. Then, the random process $\mathcal{M} = \big(\mathcal{M}_1, \mathcal{M}_2\big)$ is $(\phi_1 + \phi_2)$-zCDP.

This theorem is extended in [38, Lemma 2.3] to include the case where the result of the previous iteration is an input of the following iteration. This theorem can be expressed as follows.

Theorem 2.4. Let $\mathcal{M}_1$ be a ϕ_1-zCDP random process, and $\mathcal{M}_2$ be a ϕ_2-zCDP random process with respect to its first argument. Then, the random process $\mathcal{M} = \mathcal{M}_1(x, \mathcal{M}_2)$ is $(\phi_1 + \phi_2)$-zCDP.

We note that Theorem 2.4 is a particular case of [38, Lemma 2.3] as we focus on ϕ-zCDP which is equivalent to $(0, \phi)$-zCDP, the latter being introduced as an alternative definition of zCDP in [38]. Furthermore, both Theorem 2.3 and Theorem 2.4 can naturally be extended to encompass n iterations. Finally, we note that Theorem 2.4 is necessary for FL as the output of a client depends on the previous global shared model, which depends on this client's previous estimate.

2.5 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. In Chapter 3, we will develop a networked FL algorithm for nonsmooth objective functions that utilize dynamic zCDP. Chapter 4 will focus on online FL in the presence of unreliable and resource-constrained clients. In Chapter 5, we implement personalized FL in a graph federated architecture and utilize dynamic zCDP. Finally, chapter 6 studies personalized FL in a networked architecture where network topology and data distribution affect the local needs for inter-cluster learning.

Chapter 3

Networked Federated Learning with Differential Privacy for Nonsmooth Objectives

This chapter summarizes the results of publication **P1** and its extension **P2** which propose a novel differentially private federated learning algorithm for solving optimization problems with nonsmooth objective functions. The proposed algorithm is designed for a networked architecture, where a large number of clients collaborate by training local models on their own data and sharing them with their neighbors. Furthermore, the algorithm is privacy-preserving and protects the local data of clients from leaking to potential eavesdroppers and honest-but-curious clients. To protect clients from data leakage, we use the zero-concentrated differential privacy notion (zCDP), adjusted dynamically throughout the computation. This privacy notion is well-suited for iterative processes. The algorithm is based on the Alternating Direction Method of Multipliers (ADMM), a popular method for solving distributed optimization problems, which is modified to comply with the networked setting. In addition, we employ an approximation of the augmented Lagrangian so that convergence can be guaranteed for nonsmooth objective functions. The performance of this algorithm is thoroughly studied, with theoretical results guaranteeing its privacy protection and convergence to the optimal point in $O(1/n)$ iterations. Finally, numerical simulations illustrate its performance on both nonsmooth and smooth objectives.

3.1 Motivation

Networked federated learning, presented in the previous chapter, offers many advantages over alternative architectures, such as robustness to device failure as well as communication and computation bottlenecks. In addition, it scales easily with the number of participants. For these reasons, it is receiving a lot of interest, and many works utilize this architecture. However, most of those works focus on smooth objective functions or do not consider the need for data privacy. This chapter develops a privacy-preserving networked federated learning algorithm capable of handling nonsmooth objectives. Closely related to this work are the two following ADMM-based networked algorithms: the algorithm in [39] converges in $O(1/n)$ iterations but is limited to smooth objectives; and the algorithm in [40] accommodates a nonsmooth regularizer but requires the loss to be smooth and converges in $O(1/\sqrt{n})$ iterations. In contrast, the proposed algorithm accommodates nonsmooth loss and regularizer and converges in $O(1/n)$ iterations. To do so, we distribute the ADMM and utilize the first-order approximation of the objective function. The proposed algorithm outperforms existing methods on both smooth and nonsmooth objectives while providing identical privacy protection.

In a privacy-preserving algorithm, a compromise must be struck between convergence speed and privacy protection. This is because the privacy-related noise perturbation degrades performance. However, different privacy notions are better suited to different assumptions and may lead to different trade-offs and convergence rates. This is the main motivation for using the zero-concentrated differential privacy (zCDP) notion instead of more conventional notions, such as (ϵ, δ)-DP. It has been shown that assuming that an adversary will aggregate all the available messages, zCDP offers a better privacy-accuracy trade-off than (ϵ, δ)-DP [37, 38]. Moreover, we implement zCDP dynamically throughout the computation with decreasing noise perturbation variance, this is required to guarantee convergence to the exact optimal solution. The decreasing mechanism is chosen so that the algorithm converges in $O(1/n)$ iterations. In contrast, typical implementations of dynamic (ϵ, δ)-DP chose a decreasing mechanism that only enables convergence in $O(1/\sqrt{n})$, as can be seen, for example, in [41]. Given a fixed number of iterations, the faster decrease rate can be compensated by a higher starting value for the perturbation noise variance. In the simulation section, we observe how dynamic zCDP enables better accuracy than dynamic (ϵ, δ)-DP while providing identical privacy protection throughout the computation.

3.2 Proposed Method

3.2.1 Distributed Empirical Risk Minimization

We consider a networked architecture; it is modeled as an undirected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C} = \{1, \dots, K\}$ is the set of clients and $\mathcal{E}$ is the set of edges. That is, $(k, l) \in \mathcal{E}$ if and only if the clients k and l are connected. The set $\mathcal{N}_k$ contains the indexes of the neighbors of client k.

Each client $k \in \mathcal{C}$ has a private data set $\mathcal{D}_k := \{(\mathbf{X}_k, \mathbf{y}_k) : \mathbf{X}_k = [\mathbf{x}_{k,1}, \dots, \mathbf{x}_{k,D_k}]^\mathsf{T} \in \mathbb{R}^{D_k \times P},\ \mathbf{y}_k = [y_{k,1}, \dots, y_{k,D_k}]^\mathsf{T} \in \mathbb{R}^{D_k}\}$, where D_k is the number of data samples and P the number of features in the data.

The centralized version of the empirical risk minimization problem is given by

$$\min_{\mathbf{w}} \sum_{k=1}^{K} \left(\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}) + \frac{\lambda}{K} R(\mathbf{w}) \right), \tag{3.1}$$

where $\ell(\cdot)$ is the loss function, $R(\cdot)$ is the regularize function, $\lambda > 0$ is the regularization parameter, and $\mathbf{w}$ is a global optimization variable.

In a networked setting, relying on a global optimization variable is infeasible. Instead, each client maintains a local estimate, and consensus is enforced with auxiliary variables. Therefore, we recast the above optimization problem with local primal variables $\mathcal{V} := \{\mathbf{w}_k\}_{k=1}^K$ into:

$$\begin{aligned} \min_{\{\mathbf{w}_k\}} \quad & \sum_{k=1}^{K} \Big(\frac{1}{D_k} \sum_{j=1}^{D_k} \ell(\mathbf{x}_{k,j}, \mathbf{y}_{k,j}; \mathbf{w}_k) + \frac{\lambda}{K} R(\mathbf{w}_k) \Big) \\ \text{s.t.} \quad & \mathbf{w}_k = \mathbf{z}_k^l,\ \mathbf{w}_l = \mathbf{z}_k^l, \quad l \in \mathcal{N}_k, \quad \forall k \in \mathcal{C}, \end{aligned} \tag{3.2}$$

where the equality constraints enforce consensus. The auxiliary variables $\mathcal{Z} := \{\mathbf{z}_k^l\}_{l \in \mathcal{N}_k}$ are only used to derive the local recursions and are eventually eliminated. In the following, we consider the learning problem where $\ell(\cdot)$ and $R(\cdot)$ are convex, but not necessarily strongly convex or smooth.

3.2.2 Approximate Augmented Lagrangian

To solve the distributed empirical risk minimization problem with the ADMM, we employ the augmented Lagrangian associated with (3.2). It is given by

$$\begin{aligned}\mathcal{L}_\rho(\mathcal{V},\mathcal{M},\mathcal{Z}) &= \sum_{k=1}^{K}\Big(\frac{\ell(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k)}{D_k}+\frac{\lambda R(\mathbf{w}_k)}{K}\Big)\\ &+\sum_{k=1}^{K}\sum_{l\in\mathcal{N}_k}\Big[\boldsymbol{\mu}_k^{l\mathsf{T}}(\mathbf{w}_k-\mathbf{z}_k^l)+\boldsymbol{\gamma}_k^{l\mathsf{T}}(\mathbf{w}_l-\mathbf{z}_k^l)\Big]\\ &+\frac{\rho}{2}\sum_{k=1}^{K}\sum_{l\in\mathcal{N}_k}\Big(||\mathbf{w}_k-\mathbf{z}_k^l||^2+||\mathbf{w}_l-\mathbf{z}_k^l||^2\Big)\end{aligned} \tag{3.3}$$

where $\rho>0$ is a penalty parameter and $\mathcal{M}\coloneqq\{\{\boldsymbol{\mu}_k^l\}_{l\in\mathcal{N}_k},\{\boldsymbol{\gamma}_k^l\}_{l\in\mathcal{N}_k}\}_{k=1}^K$ are the Lagrange multipliers associated with the constraints in (3.2).

Given that the Lagrange multipliers $\mathcal{M}$ are initialized to zero, by using the Karush-Kuhn-Tucker conditions of optimality for (3.2) and setting $\boldsymbol{\gamma}_k^{(n)}=2\sum_{l\in\mathcal{N}_k}(\boldsymbol{\gamma}_k^l)^{(n)}$, it can be shown that the Lagrange multipliers $\{\boldsymbol{\mu}_k^l\}_{l\in\mathcal{N}_k}$ and the auxiliary variables $\mathcal{Z}$ are eliminated [42, 43].

After simplification, we obtain the following iterative steps to be computed by each client k:

$$\begin{aligned}\mathbf{w}_k^{(n)} &= \arg\min_{\mathbf{w}_k}\left[f_k(\mathbf{w}_k)+\mathbf{w}_k^{\mathsf{T}}\boldsymbol{\gamma}_k^{(n-1)}+\rho\sum_{l\in\mathcal{N}_k}\left\|\mathbf{w}_k-\frac{\mathbf{w}_k^{(n-1)}+\mathbf{w}_l^{(n-1)}}{2}\right\|^2\right]\\ \boldsymbol{\gamma}_k^{(n)} &= \boldsymbol{\gamma}_k^{(n-1)}+\rho\sum_{l\in\mathcal{N}_k}\Big(\mathbf{w}_k^{(n)}-\mathbf{w}_l^{(n)}\Big)\end{aligned} \tag{3.4}$$

where n is the iteration index and

$$f_k(\mathbf{w}_k)=\frac{\ell(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k)}{D_k}+\frac{\lambda R(\mathbf{w}_k)}{K}. \tag{3.5}$$

To handle nonsmooth $\ell(\cdot)$ and $R(\cdot)$ functions, we take the first-order approximation of f_k with an l_2-norm prox function, denoted as $\hat{f}_k$. Similarly as in [41, 44], such an approximation is given by

$$\begin{aligned}\hat{f}_k(\mathbf{w}_k;\mathcal{V}^{(n)}) &= \frac{\ell(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k^{(n)})}{D_k}+\frac{\lambda R(\mathbf{w}_k^{(n)})}{K}+\frac{\left\|\mathbf{w}_k-\mathbf{w}_k^{(n)}\right\|^2}{2\eta_k^{(n)}}\\ &+\Big(\mathbf{w}_k-\mathbf{w}_k^{(n)}\Big)^{\mathsf{T}}\Big(\frac{\ell'(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k^{(n)})}{D_k}+\frac{\lambda R'(\mathbf{w}_k^{(n)})}{K}\Big)\end{aligned} \tag{3.6}$$

where $\mathcal{V}^{(n)} = \{\mathbf{w}_k^{(n)}, k \in \mathcal{C}\}$ is the iteration-specific primal variable set, and $\eta_k^{(n)}$ is a time-varying step size.

Taking the first-order approximation of f_k leads to an inexact update at a given iteration; however, the algorithm does not need to solve the problem with high precision at each iteration to guarantee overall accuracy [41]. In the end, considering $\hat{f}_k$ instead of f_k in the primal update makes the algorithm capable of solving nonsmooth objectives with a minimal impact on overall accuracy.

3.2.3 Privacy Preservation

To prevent leakage of private information, we introduce local differential privacy to the algorithm via message perturbation. For this purpose, each client k shares at iteration n with its neighbors the perturbed estimate

$$\widetilde{\mathbf{w}}_k^{(n)} = \mathbf{w}_k^{(n)} + \boldsymbol{\xi}_k^{(n)} \tag{3.7}$$

with $\boldsymbol{\xi}_k^{(n)} \sim \mathcal{N}(\mathbf{0}, \sigma_k^{2(n)} \mathbf{I}_P)$. We denote $\widetilde{\mathcal{V}}^{(n)} = \{\widetilde{\mathbf{w}}_k^{(n)}, k \in \mathcal{C}\}$.

The value of the noise perturbation variance, $\sigma_k^{2(n)}$, in (3.7) dictates the privacy protection of the algorithm. To guarantee convergence to the optimal solution, as opposed to a neighborhood of it, the variance must decrease with the iterations [45]. This is made possible by using dynamic zCDP, in which the privacy protection is iteration-dependent.

The proposed zero-Concentrated Differentially Private Networked Federated Learning (zCDP-NFL) algorithm is detailed in algorithm 2. This algorithm is a networked federated learning algorithm that safeguards client privacy through local dynamic differential privacy and is capable of handling nonsmooth objective functions. In the following sections, we conduct mathematical analysis to quantify the level of privacy protection offered by zCDP-NFL and establish its convergence guarantees..

3.3 Theoretical Results

In this section, we present the theoretical results established for the proposed zCDP-NFL algorithm. While the detailed mathematical proofs can be found in the manuscript at the end of this thesis, our focus here is on the various theorems and their implications on the algorithm's performance.

3.3.1 Privacy analysis

As developed in the previous chapter, it is first necessary to define the l_2-norm sensitivity to implement differential privacy. The l_2-norm sensitivity is defined

Algorithm 2 zCDP-NFL

Initialization: $\mathbf{w}_k^{(0)} = \mathbf{0}, \boldsymbol{\gamma}_k^{(0)} = \mathbf{0}, \forall k \in \mathcal{K}$
– Procedure at client k –
For iteration $n = 1, 2, \ldots$:

$$\mathbf{w}_k^{(n)} = \arg\min_{\mathbf{w}_k} \hat{f}_k(\mathbf{w}_k; \widetilde{\mathcal{V}}^{(n-1)}) + \mathbf{w}_k^{\mathsf{T}} \boldsymbol{\gamma}_k^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k} \left\| \mathbf{w}_k - \frac{\widetilde{\mathbf{w}}_k^{(n-1)} + \widetilde{\mathbf{w}}_l^{(n-1)}}{2} \right\|^2 \tag{3.8}$$

$$\widetilde{\mathbf{w}}_k^{(n)} = \mathbf{w}_k^{(n)} + \boldsymbol{\xi}_k^{(n)} \tag{3.9}$$

$$\boldsymbol{\gamma}_k^{(n)} = \boldsymbol{\gamma}_k^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k} \left(\widetilde{\mathbf{w}}_k^{(n)} - \widetilde{\mathbf{w}}_l^{(n)} \right) \tag{3.10}$$

End For

as the maximal difference between primal variables computed using neighboring datasets.

Definition I. *The l_2-norm sensitivity is given by*

$$\Delta_{k,2} = \max_{\mathcal{D}_k, \mathcal{D}_k'} \left\| \mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}_k'}^{(n)} \right\| \tag{3.11}$$

where $\mathbf{w}_{k,\mathcal{D}_k}^{(n)}$ and $\mathbf{w}_{k,\mathcal{D}_k'}^{(n)}$ denote the local primal variable updates from two neighboring data sets $\mathcal{D}_k$ and $\mathcal{D}_k'$ differing in only one data sample $(\mathbf{x}_{k,D_k}', y_{k,D_k}')$, i.e., $\mathcal{D}_k' \coloneqq \{(\mathbf{X}_k', \mathbf{y}_k') : \mathbf{X}_k' = [\mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \ldots, \mathbf{x}_{k,D_k-1}, \mathbf{x}_{k,D_k}']^{\mathsf{T}} \in \mathbb{R}^{D_k \times P},\ \mathbf{x}_{k,j} \in \mathbb{R}^P,\ j = 1, \ldots, D_k, \mathbf{y'}_k = [y_{k,1}, y_{k,2}, \ldots, y_{k,D_k-1}, y_{k,D_k}']^{\mathsf{T}} \in \mathbb{R}^{D_k}\}$.

Two parameters govern privacy protection in dynamic zCDP, namely, the initial privacy value, $\phi_k^{(0)}$, and the variance decrease rate, ζ. To establish a relation between the privacy value at a given iteration, $\phi_k^{(n)}$, and the noise perturbation, it is necessary to take the following assumption.

Assumption 1. *The functions $\ell_k(\cdot)$ have bounded gradient, that is, there exists a constant c_1 such that $||\ell_k'(\cdot)|| \leqslant c_1, \forall k \in \mathcal{C}$.*

This assumption is common in the literature and helps to establish bounds in the privacy and convergence analysis. We now quantify the l_2-norm sensitivity in the following result.

Lemma I. *Under Assumption 1, the l_2-norm sensitivity is given by*

$$\Delta_{k,2}(n) = \max_{\mathcal{D},\mathcal{D}'} ||\mathbf{w}_{k,\mathcal{D}}^{(n)} - \mathbf{w}_{k,\mathcal{D}'}^{(n)}|| \leqslant \frac{2c_1}{D_k(2\rho|\mathcal{N}_k| + \frac{1}{\eta^{(n)}})}. \tag{3.12}$$

Proof. See **P2**, Appendix A. □

The l_2-norm sensitivity quantifies the maximum impact a single difference in the data set can have on the trained model, which connects to how easy it might be for an adversary to reconstruct this element of the data. With it, we can establish the relation between the noise perturbation in (3.9) and the privacy value $\phi_k^{(n)}$, quantifying the local privacy guarantee of the algorithm in terms of zCDP.

Theorem I. *Under Assumption 1, zCDP-NFL satisfies dynamic $\phi_k^{(n)}$-zCDP with the relation between $\phi_k^{(n)}$ and $\sigma_k^{2(n)}$ given by*

$$\sigma_k^{2(n)} = \frac{\Delta_{k,2}^{2(n)}}{2\phi_k^{(n)}}. \tag{3.13}$$

Proof. See **P2**, Appendix B. □

This theorem quantifies the privacy protection of the proposed zCDP-NFL algorithm in terms of zCDP. That is, given the initial client-specific privacy parameter $\phi_k^{(0)}$, as well as the decrease rate of the variance of the noise perturbation ζ, the privacy protection guaranteed for a given client is known for any given iteration number. Using the result above, it is possible to obtain the total privacy guarantee throughout the computation in terms of (ϵ, δ)-DP using [38, Proposition 1.3]. We establish the following.

Corollary. *For any $\zeta \in (0,1)$ and $\delta \in (0,1)$, zCDP-NFL guarantees (ϵ, δ)-DP throughout the computation with $\epsilon = \max_{k\in\mathcal{C}} \epsilon_k$, where $\epsilon_k = \phi_k^{(1)} \frac{1-\zeta^T}{\zeta^{T-1}-\zeta^T} + 2\sqrt{\phi_k^{(1)} \frac{1-\zeta^T}{\zeta^{T-1}-\zeta^T} \log \frac{1}{\delta}}$, and T is the final iteration index.*

Proof. See **P2**, Appendix C. □

This corollary is especially important as it allows us to compare the convergence properties of algorithms using the zCDP and (ϵ, δ)-DP notions for identical privacy

protection. We note that since the result of the corollary is valid $\forall \delta \in (0,1)$, any algorithm using (ϵ, δ)-DP with $\delta < 1$ is at an advantage in the comparison.

3.3.2 Convergence analysis

This section proves that the zCDP-NFL algorithm converges to the optimal value in $O(1/n)$ iterations under the assumption that the objective function $f(\cdot)$ is convex. Additionally, we derive the privacy-accuracy trade-off bound of the algorithm.

3.3.3 Alternative Representation

We begin by transforming the minimization problem (3.2) into (3.14) by reformulating the conditions. We denote by $\boldsymbol{\mathfrak{w}} = [\mathbf{w}_1^\mathsf{T}, \mathbf{w}_2^\mathsf{T}, ..., \mathbf{w}_K^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{KP}$, and $\boldsymbol{\mathfrak{z}} = [(\boldsymbol{z}_k^l)^\mathsf{T}, (\boldsymbol{z}_l^k)^\mathsf{T}; \forall (k,l) \in \mathcal{E}]^\mathsf{T} \in \mathbb{R}^{2EP}$ the vectors of the concatenated vectors $\mathbf{w}_k$ and $\boldsymbol{z}_k^l$ respectively. We also introduce the matrices $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathbb{R}^{2EP \times KP}$ composed of $P \times P$-sized blocks. Given a couple of connected clients $(k,l) \in \mathcal{E}$, their associated auxiliary variable $\boldsymbol{z}_{k,l}$, and its corresponding index in $\boldsymbol{\mathfrak{z}}$, q; the blocks $\big(\mathbf{A}_1\big)_{q,k}$ and $\big(\mathbf{A}_2\big)_{q,l}$ are equal to the identity matrix $\mathbf{I}_d$, all other blocks are null. Finally, we set $\boldsymbol{A} = [\boldsymbol{A}_1; \boldsymbol{A}_2] \in \mathbb{R}^{4EP \times KP}$ and $\boldsymbol{B} = [-\boldsymbol{I}_{2EP}; -\boldsymbol{I}_{2EP}] \in \mathbb{R}^{4EP \times 2EP}$. Hence, we can reformulate (3.2) as

$$\begin{aligned} \min_{\boldsymbol{\mathfrak{w}}} \quad & \sum_{k=1}^{K} \Big(\frac{1}{D_k} \sum_{j=1}^{D_k} \ell(\mathbf{x}_{k,j}, \mathbf{y}_{k,j}; \mathbf{w}_k) + \frac{\lambda}{K} R(\mathbf{w}_k) \Big). \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}} = 0 \end{aligned} \tag{3.14}$$

The newly introduced matrices can be used to reformulate the Lagrangian, the objective function, and the ADMM steps. The conventional augmented Lagrangian in (3.3) can be expressed as

$$\mathcal{L}_\rho = f(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)}) + \langle \boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}, \boldsymbol{\lambda} \rangle + \frac{\rho}{2} ||\boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}||^2$$

where $f(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)}) = \sum_{k=1}^{K} f(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$. Similarly, the augmented Lagrangian, corresponding to the use of the first-order approximation of the objective function in (3.6), can be expressed as

$$\hat{\mathcal{L}}_\rho = \hat{f}(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)}) + \langle \boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}, \boldsymbol{\lambda} \rangle + \frac{\rho}{2} ||\boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}||^2$$

where $\hat{f}(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)}) = \sum_{k=1}^{K} \hat{f}_k(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$ with $\hat{f}_k(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$, as defined in (3.6).

From now on, we will denote $\hat{f}(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)})$ and $\hat{f}_k(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$ by $\hat{f}(\boldsymbol{\mathfrak{w}})$ and $\hat{f}_k(\mathbf{w}_k)$, respectively. Further, we let $\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}$, $\boldsymbol{\mathfrak{w}}^{(n)}$, and $\boldsymbol{\xi}^{(n)}$ denote the concatenation of $\widetilde{\mathbf{w}}_k^{(n)}$, $\mathbf{w}_k^{(n)}$, and $\boldsymbol{\xi}_k^{(n)}$, respectively, such that $\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} = \boldsymbol{\mathfrak{w}}^{(n)} + \boldsymbol{\xi}^{(n)}$.

We introduce the diagonal matrix $\boldsymbol{D}^{(n+1)} \in \mathbb{R}^{K\times K}$ comprising the time-varying step sizes, i.e., $[\boldsymbol{D}^{(n+1)}]_{k,k} = \frac{1}{\sqrt{2\eta_k^{(n+1)}}}$, and reformulate $\hat{f}(\boldsymbol{\mathfrak{w}}^{(n+1)})$ in matrix form:

$$\begin{aligned}\hat{f}(\boldsymbol{\mathfrak{w}}^{(n+1)}) =& f(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}) + ||\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\boldsymbol{\mathfrak{w}}^{(n+1)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})||^2 \\ &+ (\boldsymbol{\mathfrak{w}}^{(n+1)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})^\mathsf{T} f'(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})\end{aligned} \tag{3.15}$$

The resulting function $\hat{f}$ is convex with respect to $\boldsymbol{\mathfrak{w}}$. That is, it satisfies $\hat{f}(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}) - \hat{f}(\boldsymbol{\mathfrak{w}}) \leqslant \langle \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} - \boldsymbol{\mathfrak{w}}, \hat{f}'(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})\rangle$, where the subgradient $\hat{f}'(\boldsymbol{\mathfrak{w}}^{(n+1)}) \in \mathbb{R}^{KP}$ is given by $\hat{f}'(\boldsymbol{\mathfrak{w}}^{(n+1)}) = 2\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\boldsymbol{\mathfrak{w}}^{(n+1)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}) + f'(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})$.

The steps of the ADMM, consisting of the minimization of $\hat{\mathcal{L}}_\rho$ with respect to $\boldsymbol{\mathfrak{w}}$, $\boldsymbol{\mathfrak{z}}$ and $\boldsymbol{\lambda}$ alternatively, can now be reformulated with the newly introduced variables as follows:

$$\begin{aligned}\hat{f}'(\boldsymbol{\mathfrak{w}}^{(n+1)}) + \boldsymbol{A}^\mathsf{T}\boldsymbol{\lambda}^{(n)} + \rho\boldsymbol{A}^\mathsf{T}(\boldsymbol{A}\boldsymbol{\mathfrak{w}}^{(n+1)} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}^{(n)}) &= 0 \\ \boldsymbol{B}^\mathsf{T}\boldsymbol{\lambda}^{(n)} + \rho\boldsymbol{B}^\mathsf{T}(\boldsymbol{A}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}^{(n+1)}) &= 0 \\ \boldsymbol{\lambda}^{(n+1)} - \boldsymbol{\lambda}^{(n)} + \rho(\boldsymbol{A}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}^{(n+1)}) &= 0\end{aligned} \tag{3.16}$$

We introduce the following auxiliary matrices in order to reduce (3.16) to two steps, similarly as in [46]: $\boldsymbol{H}_+ = \boldsymbol{A}_1^\mathsf{T} + \boldsymbol{A}_2^\mathsf{T}$, $\boldsymbol{H}_- = \boldsymbol{A}_1^\mathsf{T} - \boldsymbol{A}_2^\mathsf{T}$, $\boldsymbol{\alpha} = \boldsymbol{H}_-^\mathsf{T}\boldsymbol{\mathfrak{w}}$, $\boldsymbol{L}_+ = \frac{1}{2}\boldsymbol{H}_+\boldsymbol{H}_+^\mathsf{T}$, $\boldsymbol{L}_- = \frac{1}{2}\boldsymbol{H}_-\boldsymbol{H}_-^\mathsf{T}$ and $\boldsymbol{M} = \frac{1}{2}(\boldsymbol{L}_+ + \boldsymbol{L}_-)$. We note that $\boldsymbol{L}_+$ and $\boldsymbol{L}_-$ correspond to the signless Laplacian and signed Laplacian matrices of the network, respectively. Hence, $\boldsymbol{L}_-$ is positive semi-definite with the nullspace given by $Null(\boldsymbol{L}_-) = span\{1\}$. Then, as derived in [46, Section II.B], (3.16) becomes

$$\begin{aligned}\hat{f}'(\boldsymbol{\mathfrak{w}}^{(n+1)}) + \boldsymbol{\alpha}^{(n)} + 2\rho\boldsymbol{M}\boldsymbol{\mathfrak{w}}^{(n+1)} - \rho\boldsymbol{L}_+\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} &= 0 \\ \boldsymbol{\alpha}^{(n+1)} - \boldsymbol{\alpha}^{(n)} - \rho\boldsymbol{L}_-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)} &= 0\end{aligned} \tag{3.17}$$

The last reformulation step is based on the work in [47]. We introduce the matrix $\boldsymbol{Q} = \sqrt{\boldsymbol{L}_-/2}$, note that by construction $Null(\boldsymbol{Q}) = span\{1\}$, the auxiliary sequence $\boldsymbol{r}^{(n)} = \sum_{s=0}^{n}\boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(s)}$, vector $\boldsymbol{q}^{(n)} = \begin{pmatrix}\boldsymbol{r}^{(n)} \\ \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}\end{pmatrix}$, and matrix $\boldsymbol{G} = \begin{pmatrix}\rho\boldsymbol{I} & 0 \\ 0 & \rho\boldsymbol{L}_+/2\end{pmatrix}$. Combining both equations in (3.17), as in [47, Lemma 1], and reformulating the result, see see [47, Lemma 2], we obtain

$$\frac{\hat{f}'(\boldsymbol{\mathfrak{w}}^{(n+1)})}{\rho} + 2\boldsymbol{Q}\boldsymbol{r}^{(n+1)} + \boldsymbol{L}_+(\boldsymbol{\mathfrak{w}}^{(n+1)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}) = 2\boldsymbol{M}\boldsymbol{\xi}^{(n+1)}. \tag{3.18}$$

3.3.4 Convergence Theorem

We start by establishing a bound for the distance to the optimal solution, denoted $\mathfrak{w}^*$, at a given iteration.

Lemma II. *For any $\boldsymbol{r} \in \mathbb{R}^{KP}$ and at any iteration n, we have*

$$\begin{aligned}
&\frac{f(\widetilde{\mathfrak{w}}^{(n)}) - f(\mathfrak{w}^*)}{\rho} + \langle \widetilde{\mathfrak{w}}^{(n)}, 2\boldsymbol{Q}\boldsymbol{r}\rangle \qquad (3.19)\\
&\leqslant \frac{1}{\rho}(||\boldsymbol{q}^{(n-1)} - \boldsymbol{q}^*||^2_{\boldsymbol{G}} - ||\boldsymbol{q}^{(n)} - \boldsymbol{q}^*||^2_{\boldsymbol{G}}) - 2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}}\\
&\quad + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle + \frac{4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)}{\Phi_{\min}(\boldsymbol{L}_-)}||\boldsymbol{\xi}^{(n+1)}||_2^2\\
&\quad + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle
\end{aligned}$$

where $\boldsymbol{q}^* = [\boldsymbol{r}^T, (\mathfrak{w}^*)^T]$.

Proof. See **P2**, Appendix D. □

Following the result of Lemma II, we can establish the following theorem from which we will derive the converge results.

Theorem II. *Under Assumption 2, and given the final iteration $T > 0$, we can bound the expected error of the zCDP-NFL algorithm as*

$$\begin{aligned}
\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] &\leqslant \frac{\rho}{T}\sum_{n=1}^{T}\Big(-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}}\\
&\quad + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle\\
&\quad - \langle \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle + ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+}\Big)\\
&\quad + \frac{1}{T}\frac{\rho P 4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)\sum_{k=1}^{K}\sigma_k^{2(0)}}{\Phi_{\min}(\boldsymbol{L}_-)(1-\zeta)}\\
&\quad + \frac{\langle \widetilde{\mathfrak{w}}^{(1)}, \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(1)} - \widetilde{\mathfrak{w}}^{(0)})\rangle}{T} + \frac{\rho||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(0)}||_2^2}{T} + \frac{\rho||\widetilde{\mathfrak{w}}^{(0)} - \mathfrak{w}^*||^2_{\frac{\boldsymbol{L}_-}{2}}}{T}. \qquad (3.20)
\end{aligned}$$

where $\hat{\mathfrak{w}}^{(T)} = \frac{1}{T}\sum_{n=1}^{T}\widetilde{\mathfrak{w}}^{(n)}$, and the expectation is taken with respect to the noise. Since $\mathfrak{w}^*$ is the optimal solution, $\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)]$ is positive.

Proof. See **P2**, Appendix E. □

Where Lemma II provides a bound of the difference in output for a given iteration, Theorem II provides a bound for the expectation of the error at the end of the computation. Although the implications of the existence of this bound are not evident, further analysis of the result of this theorem is possible and provides us with all the desired results.

3.3.5 Convergence Properties

We can derive three important results from Theorem II. The first is that the zCDP-NFL algorithm converges to the exact solution of (3.2). The second is the rate of this convergence. The third result is the privacy accuracy trade-off bound of the algorithm. First, we define the required assumptions for convergence.

Assumption 3. *We require that* $\lim_{n\to+\infty} \eta_k^{(n)} = 0, \forall k \in \mathcal{C}$. *This will enforce the asymptotic stability of the local estimates.*

Theorem III. *Under Assumptions 2 and 3, the zCDP-NFL algorithm defined by the steps* (3.8)-(6.6)*, converges to the exact solution.*

Proof. See **P2**, Theorem III. □

We now introduce the required assumption to establish the convergence rate of the algorithm.

Assumption 4. *The* $\eta_k^{(n)}, k \in \mathcal{C}$ *are chosen such that* $||\boldsymbol{D}^{(n+1)}||_2^2$ *is a convergent series. This assumption, stronger than Assumption 3, is necessary to guarantee the exponential stability of the local estimates.*

Theorem IV. *Under Assumptions 2 and 4, the zCDP-NFL algorithm converges with a rate of* $\mathcal{O}(1/n)$ *iterations.*

Proof. See **P2**, Theorem IV. □

Remark: In practice, Assumption 4 can be relaxed in most cases.

3.3.6 Privacy Accuracy Trade-off

The last result established by Theorem II is the privacy accuracy trade-off bound. The privacy accuracy trade-off quantifies how ensuring more privacy deteriorates

the accuracy of the algorithm and is one of the most important parameters of a privacy-preserving algorithm. Under Assumption 4, we can reformulate (3.20) as

$$\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] \leqslant \frac{\alpha}{T} + \frac{\alpha_{\boldsymbol{\xi}}}{T} \frac{\sum_{k=1}^{K} \sigma_k^2(0)}{1-\zeta} \tag{3.21}$$

where α is a constant with respect to T and the noise perturbation and $\alpha_{\boldsymbol{\xi}} = \frac{\rho P 4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)}{\Phi_{\min}(\boldsymbol{L}_-)}$.

By combining this result with Theorem I, we obtain

$$\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] \leqslant \frac{\alpha}{T} + \frac{\alpha_{\boldsymbol{\xi}}}{T} \frac{\sum_{k=1}^{K} \frac{\Delta_{k,2}^2(0)}{2\phi_k^{(1)}}}{1-\zeta} \tag{3.22}$$

In the common case where the privacy parameter $\phi_k^{(1)}$ is identical for all clients, i.e., $\phi_k^{(1)} = \phi^{(1)}, \forall k \in \mathcal{C}$, we have

$$\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] \leqslant \frac{\alpha}{T} + \frac{\alpha_{\boldsymbol{\xi}}}{T} \frac{\sum_{k=1}^{K} \Delta_{k,2}^2(0)}{2K\phi^{(1)}(1-\zeta)} \tag{3.23}$$

With this result, we see that ensuring more privacy, which can be done by decreasing $\phi^{(1)}$ or having ζ closer to 1, would result in a less restrictive convergence bound for the algorithm.

3.4 Numerical Results

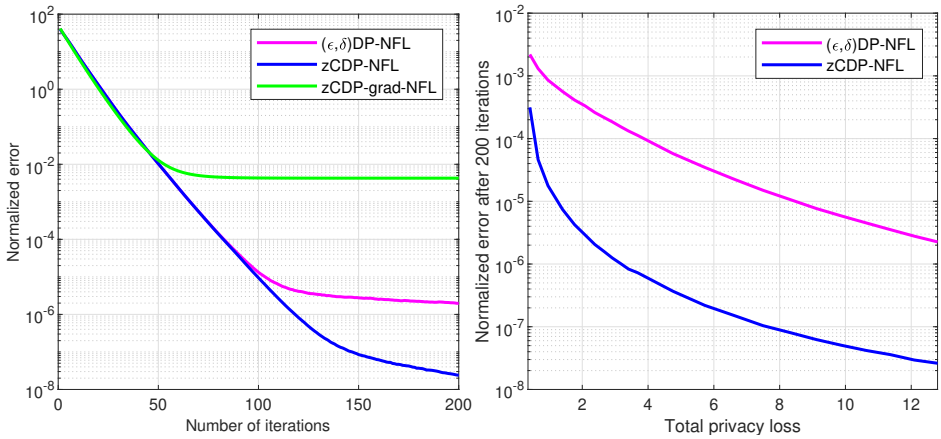
This section presents simulation results to evaluate the performance and privacy accuracy trade-off of the proposed zCDP-NFL. To compare the different DP implementations, we introduce the (ϵ, δ)DP-NFL, identical to zCDP-NFL except for the fact that it uses conventional (ϵ, δ)-DP rules to control the local noise perturbation. On nonsmooth objective functions, we benchmark the proposed algorithm against conventional subgradient-based networked FL, as presented in [48], modified to use zCDP and denoted zCDP-grad-NFL. On smooth objective functions, we benchmark the proposed algorithm against the existing distributed learning algorithm P-ADMM, presented in [39] that uses a networked FL architecture, zCDP, and the ADMM, but is constrained to smooth objective functions. In the following, we consider the elastic net, least absolute deviation, and ridge regression problems, all presented in [49].

For a fair comparison, the algorithms are tuned to provide the same total privacy guarantees throughout the computation - this is made possible by the corollary of Theorem I. This corollary provides (ϵ, δ)-DP guarantees for an algorithm using

zCDP with both $\phi^{(1)}$ and ζ as parameters. Furthermore, the algorithms are tuned to observe identical initial convergence speeds when possible.

In the following, we consider a network with a random topology comprising $K = 50$ nodes, where each node connects to 3 other nodes on average. Each node k possesses $D_k = 50$ local noisy observations of the unknown parameter $\mathbf{w}$ of dimension $P = 8$. The proposed simulations are performed on synthetic data and performance is evaluated by computing the normalized error defined as $\sum_{k=1}^{K} ||\mathbf{w}_k^{(n)} - \mathbf{w}_c||^2 / ||\mathbf{w}_c||^2$, $\mathbf{w}_c$ being the centralized solution obtained by the CVX toolbox [50].

3.4.1 Simulations on the elastic net problem



(a) **(b)**

Figure 3.1: Learning curves (left) and privacy-accuracy trade-off (right) for the elastic net problem.

Figure 3.1 presents results obtained on the elastic net problem, defined by a smooth loss function $\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{X}_k\mathbf{w}_k - \mathbf{y}_k||^2$ and a nonsmooth regularizer function $R(\mathbf{w}_k) = \lambda_1||\mathbf{w}_k||_1 + \lambda_2||\mathbf{w}_k||^2$ with $\lambda_1 = 0.001||\mathbf{X}^\mathsf{T}\mathbf{y}||_\infty$, as in [49], and $\lambda_2 = 1$. Figure 3.1 (a) shows the learning curve, i.e., normalized error versus iteration index, and Fig. 3.1 (b) shows the privacy-accuracy trade-off, i.e., normalized error after 200 iterations versus total privacy loss. We observe that the proposed ADMM-based algorithm significantly outperforms the subgradient-based algorithm. Furthermore, we can see that the use of the dynamic zCDP notion as opposed to the dynamic (ϵ, δ)-DP notion allows for better accuracy given the same privacy budget throughout the computation. This same fact is illustrated for various total privacy losses on the privacy-accuracy trade-off curve.

3.4.2 Simulations on the least absolute deviation problem

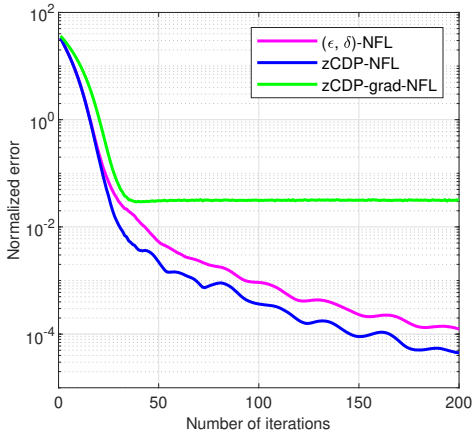


Figure 3.2: Learning curves for the least absolute deviation problem.

Figure 3.2 shows the learning curves for the algorithms under consideration for the least absolute deviation objective, i.e., a nonsmooth loss function $\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{X}_k\mathbf{w}_k - \mathbf{y}_k||_1$ and no regularizer function. We observe once again that the proposed algorithm significantly outperforms its subgradient-based counterpart, as the learning rate required for the zCDP-grad-NFL algorithm to attain a similar initial convergence speed to the ADMM-based algorithms does not allow it to reach high accuracy. In addition, we observe that the use of zCDP allows for better accuracy than (ϵ, δ)-DP. In the end, the proposed zCDP-NFL algorithm significantly improves over existing methods on nonsmooth objectives.

3.4.3 Simulations on the ridge regression problem

The following simulations study the performance of the algorithms on the ridge regression objective, which is a smooth and convex objective defined by

$$\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{X}_k\mathbf{w}_k - \mathbf{y}_k||^2, R(\mathbf{w}_k) = ||\mathbf{w}_k||^2. \tag{3.24}$$

Those simulations are conducted specifically to compare the proposed CDP-ADMM algorithm with the P-ADMM algorithm proposed in [39], which was developed for smooth objectives.

Figure 3.3 presents results obtained on the ridge regression problem, defined by a smooth loss function $\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{X}_k\mathbf{w}_k - \mathbf{y}_k||^2$ and a smooth regularizer function $R(\mathbf{w}_k) = ||\mathbf{w}_k||^2$. Figure 3.3 (a) shows the learning curves, and Fig. 3.3 (b) shows the privacy-accuracy trade-off. We observe that the proposed zCDP-NFL algorithm slightly outperforms the P-ADMM algorithm on smooth objectives, despite the inexact ADMM update required to handle nonsmooth objectives.

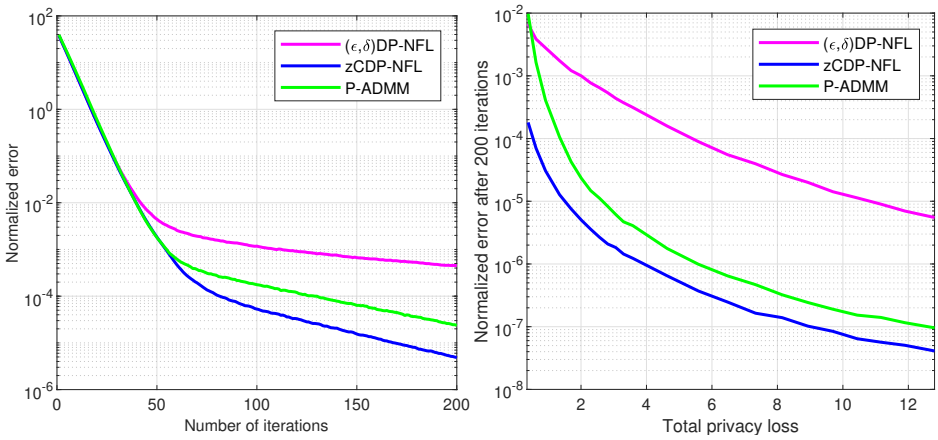


Figure 3.3: Learning curves (left) and privacy-accuracy trade-off (right) for the ridge regression problem.

As can be observed on the privacy-accuracy trade-off curve, this is the case for all total privacy losses. This better performance is due to the use of the time-varying step size $\eta^{(n)}$ in the proposed algorithm. Even though the zCPD-NFL algorithm is designed for nonsmooth objective functions, it offers very good performances on smooth objective functions.

3.5 Summary

This chapter investigated key challenges of networked federated learning, such as privacy protection and objective functions properties. Its contribution was to develop and analyze both theoretically and experimentally a privacy-preserving, networked federated learning algorithm capable of handling nonsmooth objective functions. This algorithm adapts the distributed ADMM to use a first-order approximation of the objective function and share perturbed messages to protect client data privacy through dynamic zCDP. This algorithm is proven to converge to the optimal point in $O(1/n)$ iterations and, thanks to its use of a time-varying time-step, outperforms existing privacy-preserving networked federated learning algorithms on both nonsmooth and smooth objectives. While the proposed algorithm inherently exhibits resilience to practical communication links due to its built-in tolerance to message perturbations, it operates under the assumption of synchronous communication and complete client availability. In the next chapter, we will explore additional practical scenarios where these assumptions are relaxed.

Chapter 4

Asynchronous Online Federated Learning with Reduced Communication Requirements

This chapter presents the results of publication **P3** and its extension **P4**. It proposes an online FL solution designed to accommodate asynchronous settings. These settings are characterized by heterogeneous client participation which can be unpredictable, and potential message delays due to straggler clients and imperfect communication channels. This chapter proposes an alternative implementation of online federated learning that is tailored to these asynchronous settings, aiming to mitigate the impact of asynchronous behaviors on accuracy, and drastically reduce the communication load associated with the learning task, thereby making it more accessible and efficient. Making the learning task more efficient can alleviate strain on resource-constrained clients, which may in turn reduce asynchronous behaviors in practical applications. In this chapter, a server is tasked to solve a nonlinear optimization problem using a set of unpredictable clients exhibiting high statistical and system heterogeneity. Statistical heterogeneity refers to data being imbalanced and non-i.i.d. [5] across devices, while system heterogeneity pertains to their variations in computational and communication capacities. To reduce the strain on clients and communication channels, we implement partial-sharing-based (PS) communications, which reduce the communication load without discarding potentially valuable client participation. The nonlinear model is projected on a fixed-dimensional random Fourier feature space (RFF), which is well-suited for decentralized learning as it does not require the exchange of dictionary elements. The proposed Partial-sharing-based Asynchronous Online Federated learn-

ing (PAO-Fed) handles asynchronous settings, diminishes the communication load with PS communications, and mitigates the impact of delayed updates on accuracy with a weight-decreasing mechanism. We conduct first- and second-order convergence analysis for the proposed PAO-Fed algorithm and obtain an expression for its steady-state mean square deviation. Numerical simulations on synthetic and real data show that the PAO-Fed algorithm achieves performance comparable to or better than conventional online FL in an asynchronous environment while reducing the communication load by 98 percent.

4.1 Motivation

Online federated learning, as presented in Chapter 2, allows clients with local access to data streams to train a global model in real-time without sharing their data. Most of the available literature on online FL assumes homogeneous client computational power, as well as availability and perfect communication channels [11, 12, 51–59]. These assumptions are not often met in practical applications. In a realistic setting, clients have heterogeneous computational power and availability, e.g., due to varied device characteristics, resource constraints or channel availability [6, 7, 60, 61]. Further, clients may become unresponsive for extended periods due to factors such as malfunctioning devices or loss of server range [6, 7]. In addition, the presence of straggler clients, i.e., clients with significantly below-average performance, and imperfect communication channels can also introduce delays in the exchanged messages [7, 60–62]. Although each of those challenges is tackled by existing solution, simplifying assumptions are often taken, such as the absence of delayed updates [63], or predetermined and predictable patterns of client availability [62]. In this chapter, we consider unpredictable and random client participation, as well as delayed communications.

Several methods to reduce communication load are available in the literature. Client scheduling, introduced in the classical federated averaging (FedAVG) [12], reduces the communication load by selecting a subset of clients among the available clients to participate in each iteration. While convenient in a perfect setting, discarding client participation in a setting where this participation might be rare is unappealing. Compressed client updates, used in many works such as [62, 64], reduce the communication load by allowing the clients to compress their updates prior to communication. However, this is associated with an accuracy penalty caused by the projection into a lower dimensional space, a potential computation bottleneck at the server during the simultaneous unpacking of all received updates, and an additional computational burden on the clients for performing the compression. The latter is especially unappealing when the participating clients have limited resources. PS communication, introduced in [65], reduces the communic-

ation load by allowing the clients to send only a subset of their learning parameters at each iteration. With appropriate tuning of the server's aggregation mechanism, this comes at nearly no accuracy cost. Furthermore, allowing the clients to perform several parameter updates on their remaining parameters can potentially increase learning speed. One particularity of PS communications that makes it especially appealing for asynchronous settings is its inherent resilience to adversarial input [66], which dampens the effect of delayed updates in asynchronous settings.

4.2 Proposed Method

4.2.1 Nonlinear Learning

We consider a federated network where a server is connected to a set $\mathcal{C}$ of K clients. In online FL [67], data is progressively made available to the clients throughout the learning process. The data stream available to client $k \in \mathcal{C}$ at time (or iteration) n is of the form $(y_k^{(n)}, \mathbf{x}_k^{(n)})$. The output $y_k^{(n)}$ is linked to $\mathbf{x}_k^{(n)}$ by

$$y_k^{(n)} = f(\mathbf{x}_k^{(n)}) + n_k^{(n)}, \tag{4.1}$$

where the function $f(\cdot)$ is nonlinear and $n_k^{(n)}$ is the observation noise.

The objective is for the server and clients to learn the nonlinear function $f(\cdot)$ from the streaming data without sharing this data among clients or with the server. For this purpose, clients train local models on their local streaming data and share these with the server, which then aggregates them. We rely on the kernel trick to learn the nonlinear function $f(\cdot)$, that is, we project the models and data onto a higher-dimensional space where the function appears linear.

Several adaptive methods can handle nonlinear model estimation problems, e.g., [68–71]. The conventional kernel least-mean-square (KLMS) algorithm [68] is one of the most popular choices but suffers from a growing dimensionality problem leading to prohibitive computation and communication requirements. Coherence-check-based methods [69] sparsify the original dictionary by selecting the regressors using a coherence measure. Although feasible, this method is not attractive for online FL, especially in asynchronous settings, since each new dictionary element must be made available throughout the network, inducing a significant communication overhead, especially if the underlying model changes. The random Fourier feature (RFF) space method [70, 71] approximates the kernel function evaluation by projecting the model into a pre-selected fixed-dimensional space. The selected RFF-space does not change throughout the computation, and, given that the chosen dimension is large enough, the obtained linearizations can be as precise as desired. Therefore, we use RFF-based KLMS for the nonlinear regression task, as it is data-independent, resilient to model change, and does not

require extra communication overhead, unlike conventional or coherence-check-based KLMS.

In the following, we approximate the nonlinear model by projecting it on a d-dimensional RFF-space, in which the function $f(\cdot)$ is approximated by the linear model $\mathbf{w}^*$. To estimate the global shared model using the local streaming data, we solve the following problem:

$$\min_{\mathbf{w}} \mathcal{J}(\mathbf{w}), \tag{4.2}$$

where $\mathcal{J}(\mathbf{w})$ is given by:

$$\begin{aligned} \mathcal{J}(\mathbf{w}) &= \frac{1}{K} \sum_{k \in \mathcal{C}} \mathcal{J}_k(\mathbf{w}) \\ \mathcal{J}_k(\mathbf{w}) &= \mathbb{E}[|y_k^{(n)} - \mathbf{w}^\intercal \mathbf{z}_k^{(n)}|^2], \end{aligned} \tag{4.3}$$

and $\mathbf{z}_k^{(n)}$ is the mapping of $\mathbf{x}_k^{(n)}$ into the d-dimensional RFF-space.

4.2.2 Asynchronous Behavior Simulation

To propose an algorithm that is able to operate in practical applications successfully, it is necessary to model the asynchronous system behaviors in such a way that it accounts for all potential shortcomings. We group the potential shortcomings of a system into the following categories:

- Statistical heterogeneity,
- System heterogeneity,
- Straggler clients,
- Unpredictable participation,
- Delayed updates.

While statistical heterogeneity is modeled with the data, the other aspects of asynchronous settings need to be modeled in the clients' behavior. We model those using two probabilistic mechanisms. First, we model system heterogeneity and unpredictable participation by giving the clients participation probabilities. A client $k \in \mathcal{C}$ has a probability $p_k^{(n)}$ to participate in the learning process at iteration n. A client's probability dictates its quality, and assuming that $p_k^{(n)} < 1, \forall k \in \mathcal{C}, n \in \mathbb{N}$, all clients may suffer downtimes by failing multiple Bernoulli trials. Furthermore,

this makes client participation inherently unpredictable. Second, we model straggler behavior and delayed updates in the same manner, as their impact on the learning process is identical. Each model shared with the server has a probability δ^l to be delayed for l iterations or more. We denote $\mathcal{C}^{(n)}$ as the set of all clients whose updates reach the server at iteration n. This set can be decomposed as

$$\mathcal{C}^{(n)} = \bigcup_{l=0}^{\infty} \mathcal{C}_l^{(n)}, \tag{4.4}$$

where $\mathcal{C}_l^{(n)}$ is the set of clients who sent an update to the server at iteration $n-l$, which arrive at iteration n. The subscript l corresponds to the length of the delay.

4.2.3 Partial-Sharing-Based Communications

Many of the aforementioned asynchronous behaviors result from devices being overburdened and communication channels being strained. Therefore, when performing a learning task in asynchronous settings, it is desirable to reduce the burden of the learning task on the communication channels and resource-constrained devices [60].

To this aim, we implement partial-sharing-based (PS) communications, where clients and server share only a portion of their model. Unlike compressed updates [56–59, 64] and client scheduling [12, 61], PS communications do not increase the computation load or exclude any client participation.

In partial sharing, the portion to be shared is extracted prior to communication. This operation is computationally trivial and does not delay communication. Mathematically, we model this extraction using a diagonal selection matrix with diagonal elements being 0 or 1; the locations of the latter specify the parameters to share. We denote $\mathbf{M}_k^{(n)}$ the selection matrix for server-to-client communication; it is used by the server to select a portion of the global model to share with client k at iteration n. We denote $\mathbf{S}_k^{(n)}$ the selection matrix for client-to-server communication; it is used by client k to select a portion of its local model to share with the server at iteration $n+1$. The number of shared parameters, denoted m, is identical in all selection matrices. PS communications are illustrated in Fig. 4.1 for the simple case where $m = \frac{d}{3}$.

Two variants of partial sharing communications exist, coordinated and uncoordinated. In coordinated partial sharing, used in Fig. 4.1, all clients share the same portion of the model with the server. Doing so, this portion of the global model is aggregated from a large number of clients, making it highly accurate. While ideal in perfect settings, this type of coordination may degrade performances in asynchronous settings as delayed updates, arriving later and updating the same portion

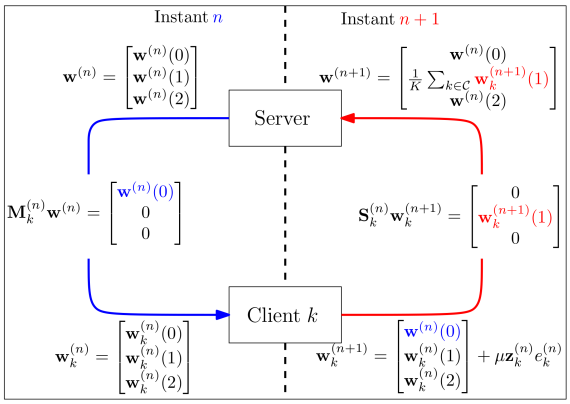


Figure 4.1: Partial sharing in a simple scenario. Where μ is the learning rate, and $e_k^{(n)}$ is the error of the local model and the local data at iteration n.

of the model, partially overwrite the aggregated model. To tackle this issue, one can use a weight-decreasing mechanism on the delayed updates, as presented in the next subsection, or use uncoordinated partial sharing. Uncoordinated partial sharing refers to any setting where different clients may share different model portions. For instance, one can choose to spread the portions shared uniformly, leading to an algorithm that quickly reacts to underlying model changes, but may suffer from low accuracy, as each model parameter is learned from a small number of clients.

For every client to participate in the learning of the entire model, it is necessary for the portion shared to change throughout the computation. For this purpose, we set the selection matrices to evolve as follows.

$$\text{diag}(\mathbf{M}_k^{(n+1)}) = \text{circshift}(\text{diag}(\mathbf{M}_k^{(n)}), m), \tag{4.5}$$

$$\mathbf{S}_k^{(n)} = \mathbf{M}_k^{(n+1)}, \tag{4.6}$$

where circshift denotes a circular shift operation. Alternatively, a pseudo-random selection process can be used in place of the circshift operator. It is crucial to set $\mathbf{S}_k^{(n)} = \mathbf{M}_k^{(n+1)}$ as opposed to $\mathbf{S}_k^{(n)} = \mathbf{M}_k^{(n)}$, as the former enable the clients to share a portion of their model refined several times by the learning process while the latter require them to share the last received server's model portion refined only once by the local learning process. For instance, in Fig. 4.1, we observe that the client shares

$$\begin{aligned} \mathbf{S}_k^{(n)}\mathbf{w}_k^{(n+1)} &= \mathbf{S}_k^{(n)}(\mathbf{w}_k^{(n)} + \mu\mathbf{z}_k^{(n)}e_k^{(n)}), \\ &= \mathbf{S}_k^{(n)}(\mathbf{w}^{(n-2)} + \mu\mathbf{z}_k^{(n-2)}e_k^{(n-2)} + \mu\mathbf{z}_k^{(n-1)}e_k^{(n-1)} + \mu\mathbf{z}_k^{(n)}e_k^{(n)}), \end{aligned} \tag{4.7}$$

if $\mathbf{S}_k^{(n)} = \mathbf{M}_k^{(n+1)}$, which corresponds to a previous server's model portion refined thrice, as opposed to

$$\mathbf{w}_k^{(n+1)}(0) = \mathbf{w}^{(n)}(0) + \mathbf{S}_k^{(n)}(\mu \mathbf{z}_k^{(n)} e_k^{(n)}), \tag{4.8}$$

if $\mathbf{S}_k^{(n)} = \mathbf{M}_k^{(n)}$, which corresponds to the last server's model portion refined once. Doing so ensures that more client-specific information is gathered by the learning process, more so as the value of m decreases. This, however, comes with the drawback that the original server's model portion is older.

4.2.4 Proposed Algorithm

The proposed PAO-Fed algorithm performs nonlinear learning in asynchronous settings while reducing communication load and mitigating the impact of delayed updates on accuracy.

At a given iteration n, the server shares a portion of its model, $\mathbf{M}_k^{(n)}\mathbf{w}^{(n)}$, with all the participating clients. Only available clients receive and utilize this information. An available client k uses the received model portion and its local streaming data to update its local model as follows,

$$\mathbf{w}_k^{(n+1)} = \mathbf{M}_k^{(n)}\mathbf{w}^{(n)} + (\mathbf{I}_d - \mathbf{M}_k^{(n)})\mathbf{w}_k^{(n)} + \mu \mathbf{z}_k^{(n)} e_k^{(n)}, \tag{4.9}$$

where μ is the learning rate and $e_k^{(n)}$ is the error of the local model on the local data given by:

$$e_k^{(n)} = y_k^{(n)} - (\mathbf{M}_k^{(n)}\mathbf{w}^{(n)} + (\mathbf{I}_d - \mathbf{M}_k^{(n)})\mathbf{w}_k^{(n)})^\intercal \mathbf{z}_k^{(n)}. \tag{4.10}$$

A portion of this local model is then extracted using the selection matrix $\mathbf{S}_k^{(n)}$ and shared with the server. Note that it may arrive at the present iteration or be delayed.

If a client k is unavailable at iteration n, but is not malfunctioning and has access to streaming data at this iteration, it may update its local model independently from the server. We call this an autonomous update. That is, an unavailable client that has access to streaming data at iteration n updates its local model autonomously as

$$\mathbf{w}_k^{(n+1)} = \mathbf{w}_k^{(n)} + \mu \mathbf{z}_k^{(n)} e_k^{(n)}, \tag{4.11}$$

where $e_k^{(n)}$ is, in that case, given by:

$$e_k^{(n)} = y_k^{(n)} - \mathbf{w}_k^{(n)\intercal} \mathbf{z}_k^{(n)}. \tag{4.12}$$

The purpose of the autonomous update is to share better-trained model parameters in the next communication with the server. The selection process illustrated in (4.5) enables the algorithm to utilize autonomous updates seamlessly.

During iteration n, the server has received updates from the clients in $\mathcal{C}^{(n)}$, defined in (4.4). For each non-empty set $\mathcal{C}_l^{(n)}, 0 \leqslant l < \infty$, we define the deviation from the current global model $\Delta_l^{(n)}$. It illustrates the direction suggested by the updates received from the clients in $\mathcal{C}_l^{(n)}$ and is given by

$$\Delta_l^{(n)} = \frac{1}{|\mathcal{C}_l^{(n)}|} \sum_{k \in \mathcal{C}_l^{(n)}} \mathbf{S}_k^{(n-l)} (\mathbf{w}_k^{(n+1-l)} - \mathbf{w}^{(n)}), \tag{4.13}$$

with $\Delta_l^{(n)} = 0$ if the set $\mathcal{C}_l^{(n)}$ is empty.

Delayed updates degrade accuracy by providing outdated information to the learning process. To mitigate their negative impact, we propose the following weight-decreasing mechanism. We introduce the weights $\alpha_l \in [0, 1], 0 \leqslant l < \infty$ to be applied to the above deviations, and set by convention the weight $\alpha_0 = 1$ for updates that are not delayed. The resulting aggregation mechanism is given by

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \sum_{l=0}^{\infty} \alpha_l \Delta_l^{(n)}. \tag{4.14}$$

We introduce a maximum effective delay $l_{\max}$; any update delayed for more than $l_{\max}$ iterations is discarded. That is, $\alpha_l = 0, \forall l > l_{\max}$. By doing so, we can replace ∞ by $l_{\max}$ in (4.4) and (4.14) without changing the meaning of the equations. The resulting algorithm is presented in Algorithm 3.

4.3 Theoretical Results

In this section, we present the theoretical results established for the PAO-Fed algorithm. This includes its first- and second-order convergence analysis and the expression of its steady-state mean square deviation. The mathematical proofs of the various theorem can be found in the manuscript **P4**.

4.3.1 First-Order Analysis

The first step of this analysis is to express the network-wide update recursions in matrix form. To do so, we introduce the following vector,

$$\begin{aligned} \mathbf{w}_e^{(n)} = \Big[& \mathbf{w}^{(n)\intercal}, \mathbf{w}_1^{(n)\intercal}, \dots, \mathbf{w}_K^{(n)\intercal}, \mathbf{w}_1^{(n)\intercal} \dots, \mathbf{w}_K^{(n)\intercal}, \mathbf{w}_1^{(n-1)\intercal}, \\ & \dots, \mathbf{w}_K^{(n-1)\intercal}, \dots, \mathbf{w}_1^{(n-l_{\max})\intercal}, \dots, \mathbf{w}_K^{(n-l_{\max})\intercal} \Big]^{\intercal}, \end{aligned} \tag{4.15}$$

Algorithm 3 PAO-Fed

Initialization: $\mathbf{w}_0$ and $\mathbf{w}_{k,0}, k \in \mathcal{C}$ set to $\mathbf{0}$
– Procedure at the server –
For iteration $n = 1, 2, \ldots$:
  Receive updates from the clients in $\mathcal{C}^{(n)}$.
  Compute $\mathbf{w}^{(n+1)}$ as in (4.14).
  Share $\mathbf{M}_k^{(n+1)}\mathbf{w}^{(n+1)}$ with the available clients.
End For
– Procedure at client k –
For iteration $n = 1, 2, \ldots$:
  if client k receives new data:
    if client k is available:
      Receive $\mathbf{M}_k^{(n)}\mathbf{w}^{(n)}$ from the server.
      Compute $\mathbf{w}_k^{(n+1)}$ as in (4.9).
      Share $\mathbf{S}_k^{(n)}\mathbf{w}_k^{(n+1)}$ with the server.
    else
      Update $\mathbf{w}_k$ as in (4.11).
    end if
  end if
End For

and matrices

$$\begin{aligned}\mathbf{A}_e^{(n)} &= \text{blockdiag}\{\mathbf{A}^{(n)}, \mathbf{I}_{dK}, \ldots, \mathbf{I}_{dK}\},\\ \mathbf{Z}_e^{(n)} &= \text{blockdiag}\{\mathbf{Z}^{(n)}, \mathbf{0}_{dK\times K}, \ldots, \mathbf{0}_{dK\times K}\},\end{aligned} \tag{4.16}$$

with

$$\mathbf{A}^{(n)} = \begin{bmatrix} \mathbf{I} & \mathbf{0}_d & \cdots & \mathbf{0}_d \\ a_1^{(n)}\mathbf{M}_1^{(n)} & \mathbf{I} - a_1^{(n)}\mathbf{M}_1^{(n)} & & \vdots \\ \vdots & \mathbf{0}_d & \ddots & \mathbf{0}_d \\ a_K^{(n)}\mathbf{M}_K^{(n)} & \vdots & & \mathbf{I} - a_K^{(n)}\mathbf{M}_K^{(n)} \end{bmatrix},$$

$$\mathbf{Z}^{(n)} = \text{blockdiag}\{\mathbf{0}_d, \mathbf{z}_1^{(n)}, \ldots, \mathbf{z}_K^{(n)}\}, \tag{4.17}$$

where $a_k^{(n)} = 1$ if the client k is available at iteration n and 0 otherwise. The operator blockdiag$\{\cdot\}$ represents block diagonalization. These enable us to express the extended observation vector $\mathbf{y}_e^{(n)} = [0, y_1^{(n)}, y_2^{(n)}, \ldots, y_K^{(n)}, \mathbf{0}_K^\intercal, \ldots, \mathbf{0}_K^\intercal]^\intercal$ as

$$\mathbf{y}_e^{(n)} = \mathbf{Z}_e^{(n)\intercal}\mathbf{w}_e^* + \boldsymbol{\eta}_e^{(n)}, \tag{4.18}$$

where $\mathbf{w}_e^* = \mathbf{1}_{(K+1)l_{\max}+1} \otimes \mathbf{w}^*$ and the extended observation noise $\boldsymbol{\eta}_e^{(n)} = [0, \eta_1^{(n)}, \eta_2^{(n)}, \ldots, \eta_K^{(n)}, \mathbf{0}_K^\intercal, \ldots, \mathbf{0}_K^\intercal]^\intercal$. We can then express the extended estimation error vector as

$$\mathbf{e}_e^{(n)} = \mathbf{y}_e^{(n)} - \mathbf{Z}_e^{(n)\intercal} \mathbf{A}_e^{(n)} \mathbf{w}_e^{(n)}. \tag{4.19}$$

Using the introduced notations, we can express the recursion of the extended model vector $\mathbf{w}_e^{(n)}$. This recursion represents an iteration of the PAO-Fed algorithm in the matrix form.

$$\mathbf{w}_e^{(n+1)} = \mathbf{B}_e^{(n)} (\mathbf{A}_e^{(n)} \mathbf{w}_e^{(n)} + \mu \mathbf{Z}_e^{(n)} \mathbf{e}_e^{(n)}), \tag{4.20}$$

where

$$\begin{aligned}
\mathbf{B}_e^{(n)} &= \begin{bmatrix}
\mathbf{B}^{(n)} & \mathbf{B}_0^{(n)} & \mathbf{0}_{d\times dK} & \mathbf{B}_1^{(n)} & \cdots & \mathbf{B}_{l_{\max}}^{(n)} \\
\mathbf{0}_{d\times 1} & \mathbf{I}_{dK} & \mathbf{0}_{dK} & \cdots & \cdots & \mathbf{0}_{dK} \\
\vdots & \mathbf{I}_{dK} & \mathbf{0}_{dK} & \cdots & \cdots & \mathbf{0}_{dK} \\
\vdots & \mathbf{0}_{dK} & \mathbf{I}_{dK} & \mathbf{0}_{dK} & \cdots & \mathbf{0}_{dK} \\
\vdots & \vdots & \ddots & \ddots & \ddots & \mathbf{0}_{dK} \\
\mathbf{0}_{d\times 1} & \mathbf{0}_{dK} & \cdots & \mathbf{0}_{dK} & \mathbf{I}_{dK} & \mathbf{0}_{dK}
\end{bmatrix} \\
\mathbf{B}^{(n)} &= \mathbf{I}_d - \sum_{l=0}^{l_{\max}} \alpha_l \sum_{k\in\mathcal{C}_l^{(n)}} \frac{b_{k,l}^{(n)}}{|\mathcal{C}_l^{(n)}|} \mathbf{S}_k^{(n-l)} \\
\mathbf{B}_l^{(n)} &= \left[\frac{\alpha_l b_{1,l}^{(n)}}{|\mathcal{C}_l^{(n)}|} \mathbf{S}_1^{(n-l)}, \cdots, \frac{\alpha_l b_{K,l}^{(n)}}{|\mathcal{C}_l^{(n)}|} \mathbf{S}_K^{(n-l)} \right].
\end{aligned} \tag{4.21}$$

where $b_{k,l}^{(n)} = 1$ if $k \in \mathcal{C}_l^{(n)}$ and 0 otherwise.

We introduce the following assumptions to facilitate the convergence analysis. We note that the variables $\alpha_l \in [0, 1]$ are not subject to any additional assumption, as the convergence analysis relies on the existence of $l_{\max}$ to bound the impact of delayed updates.

Assumption 1: The data vectors projected on the d-dimensional RFF space, $\mathbf{z}_k^{(n)}$, are drawn from a WSS multivariate random sequence with correlation matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{z}_k^{(n)} \mathbf{z}_k^{(n)\intercal}]$.

Assumption 2: The observation noise $n_k^{(n)}$ is assumed to be zero mean white Gaussian, and independent of all input and output data.

Assumption 3: At each client, the model parameter vector is assumed to be independent of the input data.

Assumption 4: The selection matrices are assumed independent from each other, and of any other data.

Assumption 5: The learning rate μ is small enough for terms involving higher-order powers of μ to be neglected.

Using the above assumptions, we establish the following result on the convergence of the PAO-Fed algorithm to $\mathbf{w}^*$, the projection of the nonlinear function $f(\cdot)$ in the RFF space.

Theorem 4.1. Let Assumptions 1–4 hold true. Then, the proposed PAO-Fed algorithm converges in mean if and only if

$$0 < \mu < \frac{2}{\max\limits_{\forall k,i} \lambda_i(\mathbf{R}_k)}. \tag{4.22}$$

Proof. See **P4**, page 6. □

4.3.2 Second-Order Analysis

To analyze the mean square deviation of the proposed algorithm, we introduce $\widetilde{\mathbf{w}}_e^{(n)}$, the model error vector, and the following weighted norm square. For a given positive semidefinite matrix $\mathbf{\Sigma}$, the weighted norm-square of $\widetilde{\mathbf{w}}_e^{(n)}$ is given by

$$||\widetilde{\mathbf{w}}_e^{(n)}||_{\mathbf{\Sigma}}^2 = \widetilde{\mathbf{w}}_e^{(n)\intercal}\mathbf{\Sigma}\widetilde{\mathbf{w}}_e^{(n)}. \tag{4.23}$$

Using this formulation and the expression of the model error recursion derived from (4.20), we have

$$\mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n+1)}||_{\mathbf{\Sigma}}^2] = \mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n)}||_{\mathbf{\Sigma}'}^2] + \mu^2\mathbb{E}[\boldsymbol{\eta}_e^{(n)\intercal}\mathbf{Y}_{\mathbf{\Sigma}}^{(n)}\boldsymbol{\eta}_e^{(n)}], \tag{4.24}$$

where the cross terms are null under Assumption 2 and the matrices $\mathbf{\Sigma}'$ and $\mathbf{Y}_{\mathbf{\Sigma}}^{(n)}$ are given by

$$\begin{aligned}\mathbf{\Sigma}' &= \mathbb{E}[\mathbf{A}_e^{(n)\intercal}(\mathbf{I} - \mu\mathbf{Z}_e^{(n)}\mathbf{Z}_e^{(n)\intercal})\mathbf{B}_e^{(n)\intercal}\,\mathbf{\Sigma}\mathbf{B}_e^{(n)}(\mathbf{I} - \mu\mathbf{Z}_e^{(n)}\mathbf{Z}_e^{(n)\intercal})\mathbf{A}_e^{(n)}],\\ \mathbf{Y}_{\mathbf{\Sigma}}^{(n)} &= \mathbf{Z}_e^{(n)\intercal}\mathbf{B}_e^{(n)\intercal}\mathbf{\Sigma}\mathbf{B}_e^{(n)}\mathbf{Z}_e^{(n)}.\end{aligned} \tag{4.25}$$

Using the properties of the block Kronecker product and the block vectorization operator bvec$\{\cdot\}$ [72], and under Assumption 3, we can establish a relationship between $\boldsymbol{\sigma} = \text{bvec}\{\mathbf{\Sigma}\}$ and $\boldsymbol{\sigma}' = \text{bvec}\{\mathbf{\Sigma}'\}$ as

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}^\intercal\boldsymbol{\sigma}, \tag{4.26}$$

where

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{Q}}_{\mathbf{B}}\boldsymbol{\mathcal{Q}}_{\mathbf{A}} - \mu\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{I} \otimes_b \mathbf{R}_e)\boldsymbol{\mathcal{Q}}_{\mathbf{A}} - \mu\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{R}_e \otimes_b \mathbf{I})\boldsymbol{\mathcal{Q}}_{\mathbf{A}},$$

where, under Assumption 5, the higher-order powers of μ are neglected and

$$\begin{aligned}\boldsymbol{\mathcal{Q}}_{\mathbf{A}} &= \mathbb{E}[\mathbf{A}_e^{(n)} \otimes_b \mathbf{A}_e^{(n)}],\\ \boldsymbol{\mathcal{Q}}_{\mathbf{B}} &= \mathbb{E}[\mathbf{B}_e^{(n)} \otimes_b \mathbf{B}_e^{(n)}].\end{aligned} \tag{4.27}$$

Appendix B in **P4** evaluates the matrices $\boldsymbol{\mathcal{Q}}_{\mathbf{A}}$ and $\boldsymbol{\mathcal{Q}}_{\mathbf{B}}$, and shows that their entries are real, non-negative, and add up to unity row-wise. Therefore, both matrices are right-stochastic, and their spectral radius is equal to one.

We evaluate the term $\mathbb{E}[\boldsymbol{\eta}_e^{(n)\mathsf{T}}\mathbf{Y}_{\boldsymbol{\Sigma}}^{(n)}\boldsymbol{\eta}_e^{(n)}]$ under Assumption 2 and using the properties of block Kronecker product on the last line:

$$\begin{aligned}\mathbb{E}[\boldsymbol{\eta}_e^{(n)\mathsf{T}}\mathbf{Y}_{\boldsymbol{\Sigma}}^{(n)}\boldsymbol{\eta}_e^{(n)}] &= \mathbb{E}[\boldsymbol{\eta}_e^{(n)\mathsf{T}}\mathbf{Z}_e^{(n)\mathsf{T}}\mathbf{B}_e^{(n)\mathsf{T}}\boldsymbol{\Sigma}\mathbf{B}_e^{(n)}\mathbf{Z}_e^{(n)}\boldsymbol{\eta}_e^{(n)}]\\ &= \mathbb{E}[\text{trace}(\boldsymbol{\eta}_e^{(n)\mathsf{T}}\mathbf{Z}_e^{(n)\mathsf{T}}\mathbf{B}_e^{(n)\mathsf{T}}\boldsymbol{\Sigma}\mathbf{B}_e^{(n)}\mathbf{Z}_e^{(n)}\boldsymbol{\eta}_e^{(n)})]\\ &= \text{trace}(\mathbb{E}[\mathbf{B}_e^{(n)}\mathbf{Z}_e^{(n)}\mathbb{E}[\boldsymbol{\eta}_e^{(n)\mathsf{T}}\boldsymbol{\eta}_e^{(n)}]\mathbf{Z}_e^{(n)\mathsf{T}}\mathbf{B}_e^{(n)\mathsf{T}}]\boldsymbol{\Sigma})\\ &= \text{trace}(\mathbb{E}[\mathbf{B}_e^{(n)}\boldsymbol{\Phi}^{(n)}\mathbf{B}_e^{(n)\mathsf{T}}]\boldsymbol{\Sigma})\\ &= \mathbf{h}^{\mathsf{T}}\boldsymbol{\sigma},\end{aligned} \tag{4.28}$$

where

$$\begin{aligned}\mathbf{h} &= \text{bvec}\{\mathbb{E}[\mathbf{B}_e^{(n)}\boldsymbol{\Phi}^{(n)}\mathbf{B}_e^{(n)\mathsf{T}}]\}\\ &= \boldsymbol{\mathcal{Q}}_{\mathbf{B}}\text{bvec}\{\mathbb{E}[\boldsymbol{\Phi}^{(n)}]\}.\end{aligned} \tag{4.29}$$

and $\boldsymbol{\Phi}^{(n)} = \mathbf{Z}_e^{(n)}\boldsymbol{\Lambda}_\eta\mathbf{Z}_e^{(n)\mathsf{T}}$ with $\boldsymbol{\Lambda}_\eta = \text{diag}\{0, \sigma_{\eta,1}^2, \ldots, \sigma_{\eta,K}^2\}$, a diagonal matrix whose diagonal contains the noise variances of all clients.

By combining (4.24), (4.26), and (4.28), we obtain the recursion of the weighted mean square deviation of the PAO-Fed algorithm:

$$\mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n+1)}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] = \mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n)}||^2_{\text{bvec}^{-1}\{\boldsymbol{\mathcal{F}}^{\mathsf{T}}\boldsymbol{\sigma}\}}] + \mu^2\mathbf{h}^{\mathsf{T}}\boldsymbol{\sigma}, \tag{4.30}$$

where $\text{bvec}^{-1}\{\cdot\}$ represents the reverse operation of block vectorization. By doing so, we can establish the following theorem.

Theorem 4.2. Let Assumptions 1–5 hold true. Then, the PAO-Fed algorithm exhibits stable mean square deviation if and only if:

$$0 < \mu < \frac{1}{\max\limits_{\forall k,i}\lambda_i(\mathbf{R}_k)}. \tag{4.31}$$

Proof. See **P4**, page 7. □

Another result that can be derived from the expression of the recursion of the weighted mean square deviation of the algorithm is the steady-state mean square deviation. For this purpose, we express the relation between $\mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n+1)}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ and $\mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n)}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ as follows

$$\begin{aligned}\mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n+1)}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] &= \mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n)}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] \\ &\quad + \mathbb{E}[||\widetilde{\mathbf{w}}_e^{(0)}||^2_{\text{bvec}^{-1}\{(\boldsymbol{\mathcal{F}}^\mathsf{T}-\mathbf{I})(\boldsymbol{\mathcal{F}}^\mathsf{T})^n\boldsymbol{\sigma}\}}] \\ &\quad + \mu^2\mathbf{h}^\mathsf{T}(\boldsymbol{\mathcal{F}}^\mathsf{T})^n\boldsymbol{\sigma}. \end{aligned} \tag{4.32}$$

Using (4.32) with $\boldsymbol{\sigma} = \text{bvec}\{\text{blockdiag}\{\mathbf{I_d}, \mathbf{0}, \ldots, \mathbf{0}\}\}$, we obtain the transient expression for the mean square deviation of the global model at iteration n:

$$\mathbb{E}[||\widetilde{\mathbf{w}}^{(n)}||^2] = \mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n)}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] \tag{4.33}$$

.

Under Assumptions 1–5 and the conditions of Theorem 4.2, and by letting $n \to \infty$ in (4.30), we obtain the expression of the steady-state mean square deviation for the PAO-Fed algorithm:

$$\lim_{n\to\infty} \mathbb{E}[||\widetilde{\mathbf{w}}_e^{(n)}||^2_{\text{bvec}^{-1}\{(\mathbf{I}-\boldsymbol{\mathcal{F}}^\mathsf{T})\boldsymbol{\sigma}\}}] = \mu^2\mathbf{h}^\mathsf{T}\boldsymbol{\sigma}. \tag{4.34}$$

Setting $\boldsymbol{\sigma} = (\mathbf{I} - \boldsymbol{\mathcal{F}}^\mathsf{T})^{-1}\text{bvec}\{\text{blockdiag}\{\mathbf{I_d}, \mathbf{0}, \ldots, \mathbf{0}\}\}$ isolates the steady-state mean square deviation of the server's model.

4.4 Numerical Results

This section presents numerical experiments conducted on synthetic and real data to illustrate the performance of the proposed PAO-Fed algorithm and compare it with existing methods. In addition to the PAO-Fed algorithm, we have simulated Online-FedSGD, Online-Fed [12], and PSO-Fed [66].

4.4.1 Simulation Setup

We considered a federated network comprising $K = 256$ clients connected to a server. Synthetic data is progressively made available to the clients in an imbalanced and non-IID manner. For this purpose, the clients are separated into 4 data groups for which training sets are composed of 500, 1000, 1500, and 2000

samples, respectively. A single data sample is of the form $\{\mathbf{x}_k^{(n)}, y_k^{(n)}\}$, and related by the following nonlinear relation:

$$y_k^{(n)} = \sqrt{\mathbf{x}_k^{2(n)}[1] + \sin^2(\pi \mathbf{x}_k^{(n)}[4])} + (0.8 - 0.5\exp(-\mathbf{x}_k^{2(n)}[2])\mathbf{x}_k^{(n)}[3]) + n_k^{(n)}, \tag{4.35}$$

where $\mathbf{x}_k^{(n)}[i]$ denotes the ith element of vector $\mathbf{x}_k^{(n)} = [x_k^{(n)}, x_k^{(n-1)}, x_k^{(n-4)}, x_k^{(n-3)}]$. A first-order autoregressive model is used to produce the non-IID input signal $x_k^{(n)} = \theta_k\, x_k^{(n-1)} + \sqrt{1-\theta_k^2}\, u_k^{(n)}$, with $u_k^{(n)} \in \mathcal{N}(\mu_k, \sigma_{u_k}^2)$, and, for a given client k, $\theta_k \in \mathcal{U}(0.2, 0.9)$, $\mu_k \in \mathcal{U}(-0.2, 0.2)$, and $\sigma_{u_k}^2 \in \mathcal{U}(0.2, 1.2)$. The observation noise $\nu_k^{(n)}$ is assumed to be white Gaussian with variance $\sigma_{\nu_k}^2 \in \mathcal{U}(0.005, 0.03)$. Further, the cosine feature function is used to map $\mathbf{x}_k^{(n)}$ from dimension $L = 4$ into the RFF-space of dimension $d = 200$.

The performance of the algorithms is evaluated on a test dataset with the mean squared error (MSE) given at iteration n by:

$$\text{MSE-test} = \frac{1}{MT}\sum_{e=1}^{M} ||\mathbf{y}_{\text{test},e} - (\mathbf{Z}_{\text{test},e})^{\intercal}\mathbf{w}_e^{(n)}||_2^2, \tag{4.36}$$

where M is the number of Monte Carlo iterations, T is the size of the test dataset, $\mathbf{y}_{\text{test},c}$ and $\mathbf{Z}_{\text{test},c}$ are the realization of the data for a given Monte Carlo iteration, and $\mathbf{w}_c^{(n)}$ is the server model for the considered method. When comparing the PAO-Fed algorithm with other methods, the learning rates were set to yield identical initial convergence rates so that steady-state values may be compared. Some algorithms were not able to reach this common convergence rate, but since their steady-state error is also higher, comparison is still possible. All the learning rates satisfy the convergence conditions obtained in Section IV for PAO-Fed, and given in [12, 66] for Online-Fed, Online-FedSGD, and PSO-Fed. For instance, in Fig. 4.2, 4.3, and 4.4, the step-size for the PAO-Fed algorithm is set to $\mu = 0.4$ with $\max_{\forall k,i} \lambda_i(\mathbf{R}_k) = 1.02$.

Asynchronous settings are modeled using the client participation probabilities, $p_k^{(n)}, k \in \mathcal{C}$, and delay probabilities, $\delta^l, 0 < l < l_{\text{max}}$. The Bernoulli trial on $p_k^{(n)}$ dictates if the client k is available or not at iteration n. This trial is performed only if a client receives streaming data at this iteration, as a client cannot participate without new data. Further, the clients are split into four availability groups, dictating their participation probabilities $p_k^{(n)}$, with associated probabilities 0.25, 0.1, 0.025, and 0.005, unless stated otherwise. This distribution is independent of

the distribution among data groups. Finally, each communication to the server has a probability $\delta^l, 0 < l < l_{\text{max}}$ to be delayed by l iterations or more, with $\delta = 0.2$ and $l_{\text{max}} = 10$, unless stated otherwise. The asynchronous behaviors are simulated once per Monte-Carlo iteration and are identical for all the algorithms.

In the simulations, we implement uncoordinated partial-sharing-based communications from the server to the clients with $\text{diag}(\mathbf{M}_k^{(n)}) = \text{circshift}(\text{diag}(\mathbf{M}_1^{(n)}), mk)$ and $\text{diag}(\mathbf{M}_1^{(n)}) = \text{circshift}(\text{diag}(\mathbf{M}_{1,0}), mn)$. This, in turn, dictates the portion of the model sent by the clients to the server, as illustrated in (4.6). Doing so, all portions of the model are equally represented in the aggregation on average. We recall that m is the number of model parameters shared at each iteration by both the server and the clients, and dictates the communication savings in partial-sharing-based communications.

4.4.2 Hyper Parameters Selection

In this section, we conduct numerical simulations to optimize the proposed PAO-Fed algorithm. For this purpose, we simulated the following versions of PAO-Fed:

- **PAO-Fed-C0** and **PAO-Fed-U0** utilize coordinated and uncoordinated partial-sharing, respectively, without employing the weight-decreasing mechanism in (4.14), that is, $\alpha_l = 1, 0 \leqslant l \leqslant l_{\text{max}}$. Further, the clients share the last received server model portion, refined once by the local update process.
- **PAO-Fed-C1** and **PAO-Fed-U1** utilize coordinated and uncoordinated partial-sharing, respectively, without employing the weight-decreasing mechanism in (4.14). Their selection matrices evolve as described in (4.5) and (4.6).
- **PAO-Fed-C2** and **PAO-Fed-U2** utilize coordinated and uncoordinated partial-sharing, respectively, and employ the weight-decreasing mechanism in (4.14) with $\alpha_l = 0.2^l, 0 \leqslant l \leqslant l_{\text{max}}$. Their selection matrices evolve as described in (4.5) and (4.6).

Unless explicitly specified, each PAO-Fed implementation shares $m = 4$ model parameters per communication round, resulting in a 98% reduction in communication.

In the first experiments, we study the impact of the hyperparameters on the convergence properties of the PAO-Fed algorithm. Specifically, we investigate the impact of the choice of the selection matrices, the number of model parameters shared, and the scale of the weight-decreasing mechanism for delayed updates. The corresponding learning curves in Fig. 4.2 shows the MSE-test in dB versus the iteration index.

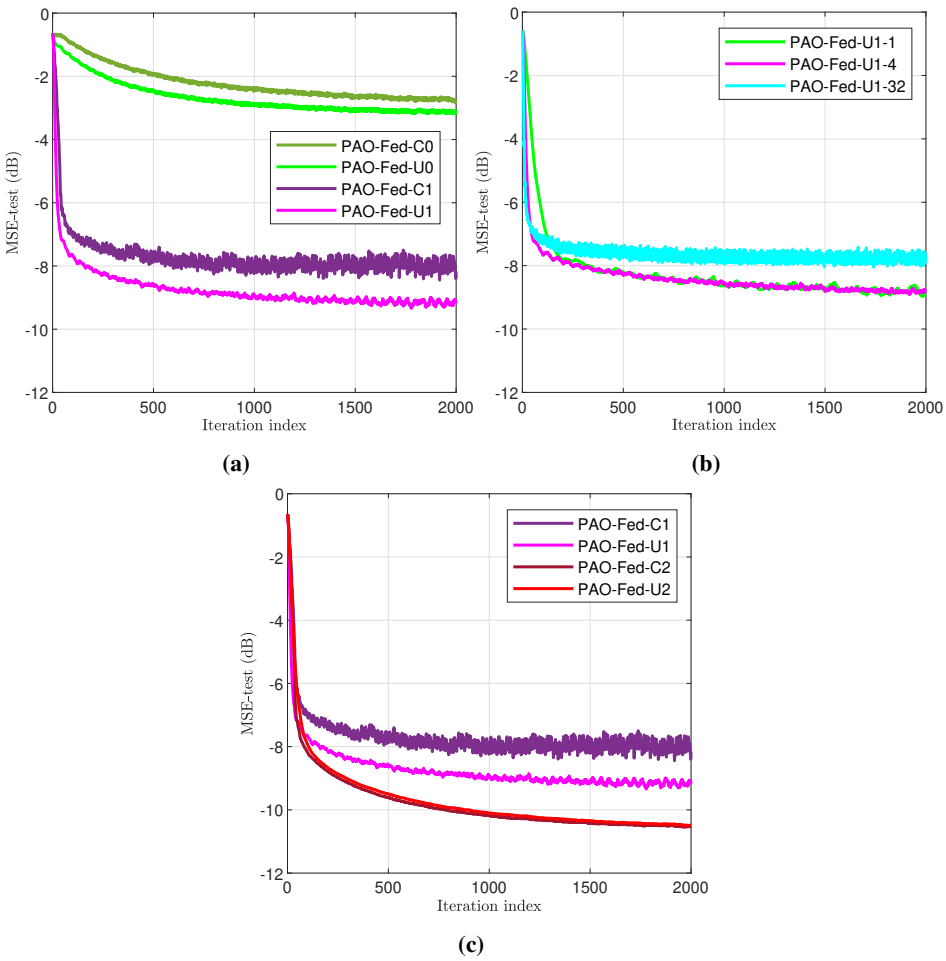


Figure 4.2: Optimization of the PAO-Fed method. (a) Utilizing local updates and coordinated/uncoordinated partial-sharing, (b) Communication savings, (c) Utilizing weight-decreasing mechanism for delayed updates.

First, we examined how the choice of the selection matrices $\mathbf{M}_k^{(n)}$ and $\mathbf{S}_k^{(n)}$ impact the convergence properties of the PAO-Fed algorithm. The versions PAO-Fed-C0 and PAO-Fed-U0 are set with $\mathbf{S}_k^{(n)} = \mathbf{M}_k^{(n)}$; that is, the last received portion from the server is updated once by the local learning process at the clients before being sent back to the server. On the contrary, the versions PAO-Fed-C1 and PAO-Fed-U1 are set as in (4.5) and (4.6); that is, the received portions from the server will be updated several times by the local learning process to accumulate information, in a manner similar to batch learning, before being sent back to the server. We observe

in Fig. 4.2 (a) that the versions PAO-Fed-(C/U)1 outperform the versions PAO-Fed-(C/U)0. For this reason, we will only consider the versions of the PAO-Fed algorithm making full use of the local updates in the following. We also notice in this experiment that it is best to use uncoordinated partial-sharing in asynchronous settings, this contradicts the behavior of partial-sharing-based communications in ideal settings, where coordinated partial-sharing performs slightly better than uncoordinated, as explained in [66].

Second, we studied the impact of the number of model parameters m shared by participating clients and the server during the learning process. Fig. 4.2 (b) shows the performance of the PAO-Fed-U1 algorithm (uncoordinated, making use of local updates) for different values of m, namely $m = 1$, $m = 4$, and $m = 32$. Although sharing more model parameters increases the initial convergence speed, we observed that it decreases the final accuracy for larger m values. This contradicts previous results in the literature about the behavior of partial-sharing in ideal settings [66]. In fact, sharing more model parameters increases the potential negative impact of one single delayed update carrying outdated information, decreasing the overall accuracy. Sharing a small number of model parameters limits the impact of a given update, providing some level of protection against outdated information, and ensuring better model fitting [73]. We chose to set $m = 4$ as a baseline, as it presents a good compromise between initial convergence speed, steady-state accuracy, and communication reduction.

Finally, to reduce the harmful effect of delayed updates on the convergence properties of the algorithm, we introduce the weight-decreasing mechanism for delayed updates proposed in (4.14) in the versions PAO-Fed-C2 and PAO-Fed-U2. We set $\alpha_l = 0.2^l, 0 \leqslant l \leqslant l_{\text{max}}$. In Fig. 4.2 (c), we display the performance of these methods alongside PAO-Fed-C1 and PAO-Fed-U1. We observe that decreasing the weight of the delayed updates significantly improves the performance of the PAO-Fed algorithm on the considered asynchronous settings. The proposed mechanism considers the relevance of delayed and potentially outdated updates by effectively reducing their impact on the server model, especially for substantial delays. By doing so, the negative effect of delayed updates is mitigated; in particular, when using the aforementioned weight-decreasing mechanism, PAO-Fed-C2 using coordinated partial sharing and PAO-Fed-U2 using uncoordinated partial sharing exhibit the same performance.

4.4.3 Comparison of PAO-Fed with Existing Methods

In the following experiments, we compare the performance of the PAO-Fed algorithm with existing online FL methods in the literature. Figs. 4.3 (a) and (c) display the MSE-test in dB versus the iteration index, and Fig. 4.3 (b) displays

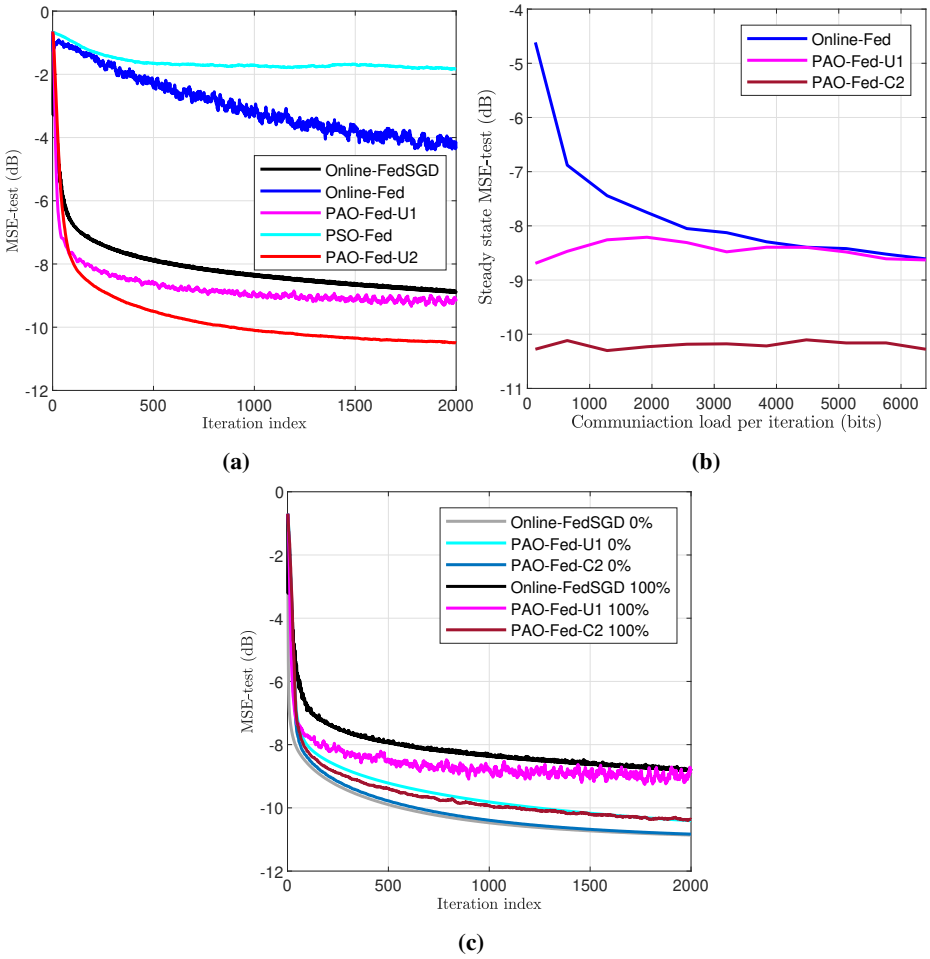


Figure 4.3: Comparison of PAO-Fed with existing methods. (a) Learning curves, (b) Steady state MSE vs. communication load, (c) Impact of straggler clients.

accuracy variation versus communication savings.

First, we compared PAO-Fed-U1 and PAO-Fed-U2 with PSO-Fed [66], Online-Fed [12], and Online-FedSGD. The corresponding learning curves are displayed in Fig. 4.3 (a). First, we observe that Online-Fed and PSO-Fed both perform poorly; sub-sampling the already reduced pool of available clients is not a viable solution to reduce communication in asynchronous settings. Then, we observe that both PAO-Fed-U1 and PAO-Fed-U2 outperform Online-FedSGD while using 98% less communication. The reason for this very good performance is twofold. First, using the local and autonomous local updates in the PAO-Fed algorithm al-

lows it to extract more information from the sparsely participating clients. Second, partial-sharing-based communication provides the PAO-Fed algorithm with an innate resilience to the negative impact of delayed updates; this resilience is further increased in the PAO-Fed-U2 algorithm with the weight-decreasing mechanism, hence its better performance.

Second, we study the relationship between communication load and accuracy. Figure 4.3 (b) shows the steady-state mean squared error on the test dataset versus the average communication load per iteration when the clients employ either PAO-Fed-U1, PAO-Fed-C2, or Online-Fed algorithms. The communication load is obtained by multiplying the average number of model parameters shared by a client during a given iteration, corresponding to m for the PAO-Fed algorithms, by 32, which is the number of bits on which a model parameter is stored. We find the MSE reached after 2000 iterations in the previous figure by the three algorithms in this figure for a communication load of 128 bits. Similarly, we find the MSE reached after 2000 iterations in the previous figure by Online-FedSGD in this figure for the Online-Fed algorithm with a communication load of 6400 bits. Further, we observe that the higher the communication load is, the better the performance of Online-Fed is. However, the performances of the algorithms using partial-sharing-based communication vary very little with the communication load, as the lower amount of communication is compensated by the use of local updates and the resilience to delayed communications.

Finally, to observe the impact of the straggler clients on the convergence properties of the algorithms, we compare the performance of the algorithms in the proposed settings (100% of clients are potential stragglers) to their performance in an ideal setting where clients are always available when they receive new data and their communication channels do not suffer from delays (0% of clients are potential stragglers). The learning curves are shown in Fig. 4.3 (c). We observe that, in the absence of straggler clients, the methods using coordinated partial-sharing achieve greater accuracy, almost identical to methods with no communication reduction, while the methods using uncoordinated partial-sharing have slightly worse performance, this corresponds to the results obtained in [66]. Furthermore, we see that the PAO-Fed-C2 algorithm used on straggler clients has convergence properties almost similar to the ones of algorithms in a perfect setting.

4.4.4 Performance on a Real-world Dataset

Fig. 4.4 shows the performance of the proposed PAO-Fed algorithm on the real-world California Cooperative Oceanic Fisheries Investigations (CalCOFI) dataset [74]. This dataset comprises oceanographic data from seawater samples collected at various stations and contains more than 800,000 samples. Each sample contains

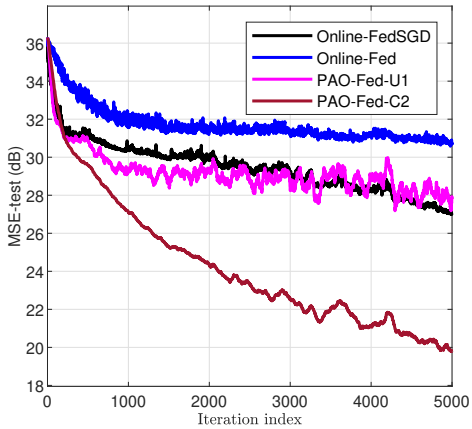


Figure 4.4: Learning curves on the CalCOFI dataset.

parameters such as temperature, salinity, O_2 saturation, etc. The salinity of the water is linked in a nonlinear manner to the other available parameters, and we employed the proposed method to learn this nonlinear model relating the salinity level in a decentralized manner. For the purpose of the experiment, we consider only 80,000 samples that we distribute progressively and unevenly to the 256 clients throughout the learning process (to ensure non-IID and imbalanced data settings). Further, we simulated the straggler-like behavior of the clients as mentioned above (availability groups are 0.25, 0.1, 0.025, and 0.005; each communication to the server will be delayed by more than l iterations with probability $\delta^l, 0 < l < l_{\max}$, with $\delta = 0.2$ and $l_{\max} = 10$). We observe similar performance for the PAO-Fed, Online-Fed, and Online-FedSGD algorithms to the experiments on synthetic datasets. The PAO-Fed-U1 algorithm is able to achieve the same accuracy as Online-FedSGD while using 98% less communications, and the PAO-Fed-C2 algorithm, also using 98% less communications, is able to outperform all other methods.

4.4.5 Comparison of Various Communication Reduction Methods in Asynchronous Settings

In this simulation, we compare the performance of the proposed method with the PSO-Fed [66], Online-Fed [12], and SignSGD [75] algorithms. The PSO-Fed algorithm combines client scheduling and partial-sharing-based communications. For a fair comparison, it has been tailored to reduce the overall communication load by 98%, similar to the proposed PAO-Fed-C2 algorithms. By design, the SignSGD drastically reduces the communication load from clients to server but does not reduce the communication load from server to clients. Its communication load reduction is, therefore, less than 50%. For this reason, the Online-Fed

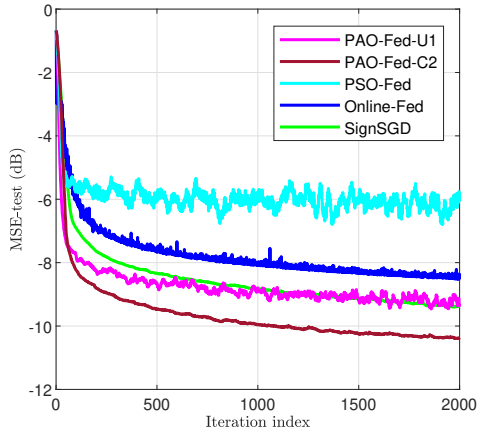


Figure 4.5: Learning curves of PAO-Fed, PSO-Fed, Online-Fed, and SignSGD.

algorithm has been tailored to reduce the communication load by only 50%. The learning curves are displayed in Fig. 4.5. We observe that reducing the communication load via a combination of client scheduling and partial-sharing-based communication, as in PSO-Fed, is not desirable in asynchronous settings. Furthermore, we see that the SignSGD achieves significantly better performance than Online-Fed for a similar communication load reduction, making it a viable alternative to partial-sharing-based communication in asynchronous settings. However, it would need to be complemented by server-to-client communication reduction and a weight-decreasing mechanism to achieve the same accuracy and communication load reduction as the proposed PAO-Fed-C2.

4.4.6 Impact of the Environment on Convergence Properties

In these last experiments, we study the impact that a change in the external environment can have on the convergence properties of the proposed algorithms and existing methods. The corresponding learning curves are shown in Fig. 4.6.

First, we studied in Fig. 4.6 (a) the importance of using partial-sharing-based communications both at the server and at the clients. The algorithms using partial-sharing-based communications have been altered in this simulation with $\mathbf{M}_k^{(n)} = \mathbf{I}, \forall k, n$; that is, the server sends its entire model to the participating clients at each iteration. This modification can be appealing if the server is not subject to power constraints. The clients behave normally and only send a portion of their local model; however, unlike in the other simulations, the received global model replaces the local model at each participant, see (4.9). In such a case, we observe that the performance of the partial-sharing-based methods is drastically reduced.

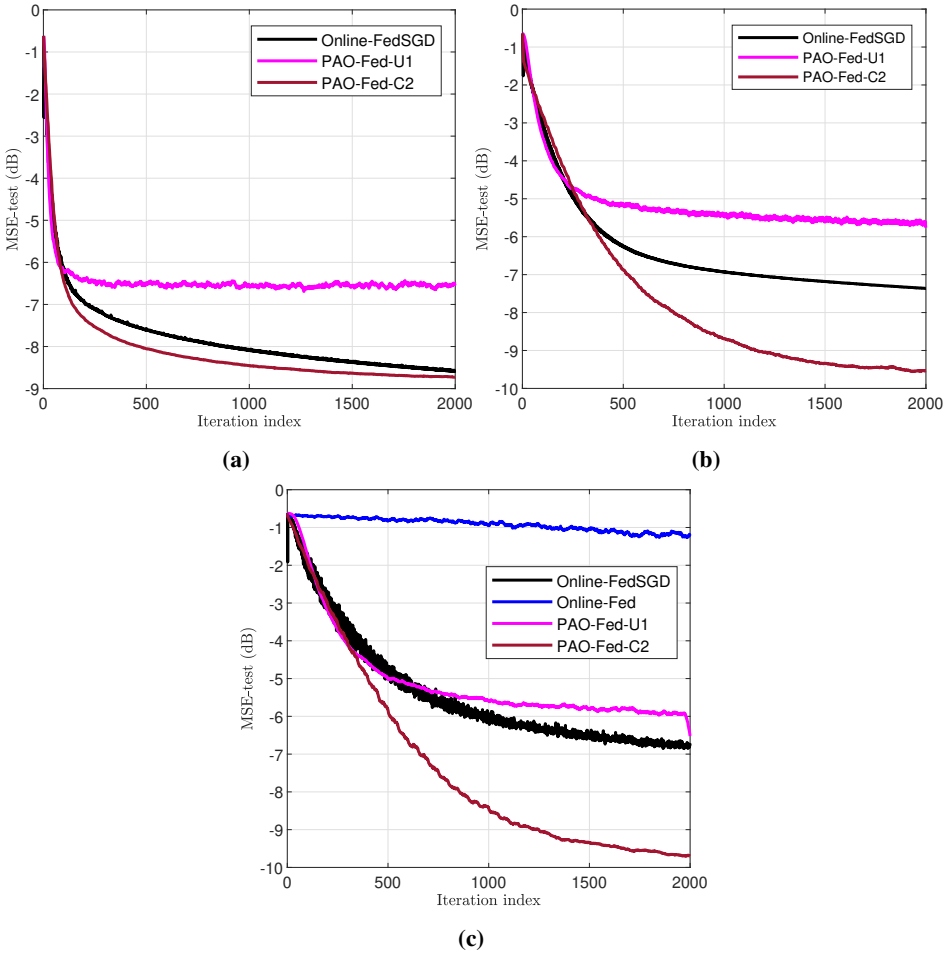


Figure 4.6: Learning curves in different environments. (a) Full server communication, (b) Common delays, (c) Increased straggler behavior.

It is the information kept by the clients in the not-yet-shared portions of their local models that allow partial-sharing-based methods to outperform Online-FedSGD. We note that clients may choose to ignore part of the received model to avoid this downfall.

Second, we studied the algorithm behaviors in an environment where most communications are delayed, but delays cannot be too lengthy. To this aim, the delay probability has been significantly increased and the maximum possible delay reduced ($\delta = 0.8$ and $l_{\max} = 5$). We observe in Fig. 4.6 (b) that the limited maximum delay allows Online-FedSGD to outperform PAO-Fed-U1, as the benefit of

partial-sharing against data of poor quality does not out-weight the smaller amount of communication available to PAO-Fed-U1. To compensate for the fact that most incoming information is weighted down by the weight-decreasing mechanism of PAO-Fed-C2, its learning rate has been increased to near its maximum value obtained in Theorem 4.2. Despite this, the PAO-Fed-C2 algorithm reaches very low steady-state error and significantly outperforms Online-FedSGD.

Finally, we modeled an environment where availability groups are given the probabilities 0.025, 0.01, 0.0025, and 0.0005; communications to the server have a probability $\delta = 0.4$ to be delayed. Further, delays last for more than l iterations, l taking the values $10i, 0 \leqslant i \leqslant 6$, with probability $\delta^{\frac{l}{10}}$; $l_{\max}$ is set to 60. This notably implies that, in this environment, delayed updates have a greater probability of arriving after a non-delayed update coming from the same client. Such an environment where clients are less likely to be available to participate, communications are more likely to be delayed, and delays last for more iterations, is less favorable to learning. An application relying on edge devices that are poorly available and unreliable would evolve in an environment similar to this. Fig. 4.6 (c) presents the learning curves of Online-Fed, Online-FedSGD, and the PAO-Fed algorithm in this new environment to see how it may impact the convergence properties of the algorithms. We observe that, in this environment, reducing the weight given to the delayed updates gains importance as the accuracy difference between PAO-Fed-C2 and PAO-Fed-U1 increases. In fact, delayed updates may carry information that is significantly outdated and, therefore, prevent the algorithms not using a weight-decreasing mechanism for delayed updates to reach satisfactory steady-state error. For this reason, the PAO-Fed-C2 algorithm achieves significantly better accuracy than Online-FedSGD in this environment.

4.5 Summary

This chapter investigated the obstacles faced by federated learning in the presence of resource-constrained devices, straggler clients, and imperfect communication channels. It developed an online FL algorithm adapted to a realistic environment that not only reduces the negative impact of asynchronous settings on the convergence properties but also alleviates the communication load of the clients and the communication channels to reduce the strain on the network. In practical applications, reducing the strain on the system may reduce the asynchronous behaviors by making the learning task more accessible, but this was not simulated to ensure fairness in the comparisons with existing methods. The mathematical analysis and numerical results confirm that the proposed approach is ideal for extracting information in real-time from diverse geographically dispersed devices without overloading the system, making it highly desirable in IoT applications in particular. While

this chapter has focused on learning scenarios where clients have different capacities, the following chapter will focus on learning scenarios where clients have different needs.

Chapter 5

Graph Personalized Federated Learning

This chapter presents the results of publications **P5** and **P6**, proposing a framework that utilizes a graph federated architecture, enabling the learning of personalized models, and preserving the client confidentiality and privacy. This proposed personalized graph federated learning (PGFL) framework allows distributedly connected servers and their respective clients to learn client- or cluster-specific models tailored for various learning tasks present in the network. The proposed approach exploits similarities among different learning tasks to speed up learning and alleviate data scarcity. To ensure a secure approach to collaborative learning, we modify the PGFL framework to utilize local zCDP. We implement a privacy-preserving PGFL algorithm as proof of concept and study its theoretical and experimental performance. This algorithm uses the ADMM as a local learning mechanism and is tested in a setting with diverse data distributions and disproportionate datasets. Our mathematical analysis shows that the proposed privacy-preserving PGFL algorithm converges to the optimal cluster-specific solution for each cluster in linear time. It also shows that exploiting similarities among clusters leads to an alternative solution whose distance to the optimal cluster-specific solution is bounded. The distance can be made arbitrarily small at the cost of limited exploitation of inter-cluster learning. Further, privacy analysis shows that the algorithm ensures local dynamic zCDP for all clients, paving the way for striking a trade-off between privacy and accuracy. Finally, the effectiveness of the proposed PGFL algorithm is showcased through numerical experiments conducted in the context of regression and classification tasks using some of the National Institute of Standards and Technology's (NIST's) synthetic datasets, namely, MNIST, and MedMNIST.

5.1 Motivation

One of the main challenges in recent distributed machine learning applications is their reliance on a variety of distributed devices that may exhibit significant statistical heterogeneity. In many cases, the substantial differences in the underlying statistical distributions among clients warrant the use of device-specific models instead of a single global shared model [18, 19, 76]. Personalized FL, presented in Chapter 2, is a variation of the FL framework that enables learning such device- or cluster-specific models [25–27, 77]. Achieving satisfactory accuracy for several models across a network requires more data than when learning a single model. Hence, it is necessary for the successful implementation of personalized FL to aggregate a large number of participants in a single network. This raises the need for an architecture that scales easily with the number of participants; however, most works on personalized FL restrict their study to the single-server case [28, 78–82], in which the maximum number of clients, hence the maximum number of satisfyingly learned models, is bounded. For this reason, we propose to use the graph federated architecture for personalized FL. This architecture is highly scalable with the number of clients, making it possible to aggregate enough data to learn different models within a single connected network, and enables faster convergence than the networked architecture [9, 83].

The client- or cluster-specific models learned in personalized FL typically share some similarities [28]. Leveraging those similarities can improve performance [17, 28], a process known as inter-cluster learning, which is particularly important when some clients or clusters have insufficient data [81, 82]. Using a graph federated architecture and inter-cluster learning, the proposed personalized graph FL (PGFL) framework aims to enable efficient learning of personalized models for distributed clients. Furthermore, the proposed framework aims to be generic; therefore, we do not make any assumption about the cluster-specific distribution of clients, as clients within a cluster may be geographically grouped in a specific application and dispersed in another. We propose to implement this framework using the ADMM, which is well-suited for distributed applications [84–86] and demonstrates fast, often linear [46, 87], convergence. Task similarity is leveraged using the Euclidean norm, providing a simple and intuitive mechanism for inter-cluster learning that may be replaced with custom-made solutions in practical applications. Finally, to ensure that the proposed algorithm can be used in a wide variety of applications, we protect client data from eavesdroppers and honest-but-curious clients with local differential privacy. Specifically, we use dynamic zero-concentrated differential privacy (zCDP), introduced in Chapter 2.

5.2 Proposed Method

The proposed PGFL framework solves a personalized optimization problem in a graph federated architecture and utilizes the similarities among clusters to enhance learning performance. For this purpose, we consider a distributed network of S servers associated with a total of K clients. The server network is modeled as an undirected graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, where $\mathcal{S}$ is the set of servers and $\mathcal{E}$ is the set of edges so that two servers s and p can communicate if and only if $(s, p) \in \mathcal{E}$. The set of neighbors to a server s is denoted $\mathcal{N}_s$, and contains s, we denote $\mathcal{N}_s^- = \mathcal{N}_s \backslash s$. Each server s is associated with a set of clients, denoted $\mathcal{C}_s$, with $\bigcup_{s\in\mathcal{S}} \mathcal{C}_s = \mathcal{C}$ and $\mathcal{C}_s \bigcap \mathcal{C}_p = \emptyset, \forall s \neq p$. Every client $k \in \mathcal{C}$ has access to a local dataset $\mathcal{D}_k$ of cardinality $|\mathcal{D}_k| = D_k$, which is composed of a data matrix $\mathbf{X}_k = [\mathbf{x}_{k,1} \dots \mathbf{x}_{k,D_k}]^\intercal$, where $\mathbf{x}_{k,i}, i \in \{1, \dots, D_k\}$ is a vector of size d, and a response vector $\mathbf{y}_k = [y_{k,1}, \dots, y_{k,D_k}]^\intercal$ that is subject to white observation noise. Each client $k \in \mathcal{C}$ aims to learn a personalized, client-specific model $\mathbf{w}_k$.

The learning task for each client is defined by the set $\{\mathcal{D}_k, \ell_k\}$, which represents its local data and loss function. All clients connected to distributed servers, regardless of their associated servers, are grouped into Q clusters. These clusters are formed by clients with similar learning tasks, such as $\mathcal{F}$-similar tasks [29], with the aim of collectively learning a shared model. It is assumed that there is a degree of relationship among the learning tasks across clusters, which can manifest in various ways. For example, clusters may share the same loss and regularizer functions while having different data distributions, or they may have the same data distribution but distinct objective functions. For instance, in healthcare, clusters can represent various patient diagnostics, independent of their respective associated hospitals, with a hospital functioning akin to a server. We denote the set of clusters as $\mathcal{Q} = \{1, \dots, Q\}$. The clients belonging to a specific cluster $q \in \mathcal{Q}$ form the set $\mathcal{C}_{(q)}$ aiming to learn the model $\mathbf{w}^*_{(q)}$. Additionally, the set of clients associated with server s within cluster q is denoted as $\mathcal{C}_{s,(q)}$, with $\mathcal{C}_{s,(q)} = \mathcal{C}_s \bigcap \mathcal{C}_{(q)}$.

5.2.1 Personalized Graph Federated Learning

To address task variations, personalized (cluster) models are preferable. However, despite these differences, the underlying relationship among tasks, or equivalently, clusters, can still be exploited in decentralized learning. Here, we consider a modified regularized empirical risk minimization problem to leverage similarities among the clusters. For this purpose, we introduce an additional regularizer function that enforces similarity among the cluster-specific personalized models. This additional regularizer function corresponds to inter-cluster learning and is controlled by the inter-cluster learning parameter $\tau \in (0, 1)$. The resulting optim-

ization problem for a cluster q is formulated as:

$$\min_{\mathbf{w}_{(q)}} \sum_{k\in\mathcal{C}_{(q)}} \frac{1}{D_k}\sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{(q)}) + \lambda R(\mathbf{w}_{(q)}) + \tau \sum_{r\in\mathcal{Q}\setminus q} ||\mathbf{w}_{(r)} - \mathbf{w}_{(q)}||_2^2, \tag{5.1}$$

where $\ell_k(\cdot)$, $R(\cdot)$, and λ denote the client loss function, the global regularizer function, and the regularization parameter, respectively. The larger the τ value is, the more the similarities among cluster-specific personalized models are exploited.

The centralized optimization problem above relies on the global variable $\mathbf{w}_{(q)}$. In a multi-server architecture, the servers maintain local cluster-specific models and communicate among neighbors to reach a consensus for each cluster. The equivalent distributed optimization problem for cluster q, is given by

$$\begin{aligned} \min_{\{\mathbf{w}_{s,(q)}\}} \quad & \sum_{q\in\mathcal{Q}}\Bigg(\sum_{k\in\mathcal{C}_{s,(q)}} \frac{1}{D_k}\sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{s,(q)}) \\ & + \lambda R(\mathbf{w}_{s,(q)}) + \tau \sum_{r\in\mathcal{Q}\setminus q}\sum_{p\in\mathcal{N}_s} ||\mathbf{w}_{p,(r)} - \mathbf{w}_{s,(q)}||_2^2\Bigg), \\ \text{s.t.} \quad & \mathbf{w}_{s,(q)} = \mathbf{z}_{s,p,(q)}, \mathbf{w}_{p,(q)} = \mathbf{z}_{s,p,(q)}; \forall (s,p)\in\mathcal{E}, \forall q\in\mathcal{Q}, \end{aligned} \tag{5.2}$$

where $\mathbf{w}_{s,(q)}$ denotes the model for the cluster q connected to server s and consensus is enforced by the cluster-specific auxiliary variables $\{\mathbf{z}_{s,p,(q)}; \forall (s,p) \in \mathcal{E}, \forall q \in \mathcal{Q}\}$. From (5.2), the augmented Lagrangian with penalty parameter ρ can be derived as

$$\begin{aligned} \mathcal{L}_{\rho,q}(\mathcal{V}_q, \mathcal{M}, \mathcal{Z}) = \sum_{s\in\mathcal{S}}\Bigg[& \sum_{k\in\mathcal{C}_{s,(q)}} \frac{\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_{s,(q)})}{D_k} + \lambda R(\mathbf{w}_{s,(q)}) \\ & + \tau \sum_{r\in\mathcal{Q}\setminus q}\sum_{p\in\mathcal{N}_s} ||\mathbf{w}_{p,(r)} - \mathbf{w}_{s,(q)}||_2^2 \\ & + \sum_{p\in\mathcal{N}_s^-}\Big(\boldsymbol{\mu}_{s,p}^\intercal(\mathbf{w}_{s,(q)} - \mathbf{z}_{s,p,(q)}) + \boldsymbol{\psi}_{s,p}^\intercal(\mathbf{w}_{p,(q)} - \mathbf{z}_{s,p,(q)})\Big) \\ & + \frac{\rho}{2}\sum_{p\in\mathcal{N}_s^-}\Big(||\mathbf{w}_{s,(q)} - \mathbf{z}_{s,p,(q)}||_2^2 + ||\mathbf{w}_{p,(q)} - \mathbf{z}_{s,p,(q)}||_2^2\Big)\Bigg], \end{aligned} \tag{5.3}$$

with the set of primal variables $\mathcal{V}_q = \{\mathbf{w}_{s,(q)}; s \in \mathcal{S}\}$, Lagrange multipliers $\mathcal{M} = (\{\boldsymbol{\mu}_{s,p}\}, \{\boldsymbol{\psi}_{s,p}\})$, and auxiliary variables $\mathcal{Z} = \{\mathbf{z}_{s,p,(q)}\}$. Given that the Lagrange

multipliers are initialized to zero, using the Karush-Kuhn-Tucker conditions of optimality and setting $\boldsymbol{\psi}_s = 2\sum_{p\in\mathcal{N}_s^-}\boldsymbol{\psi}_{s,p}$, it can be shown that the Lagrange multipliers $\boldsymbol{\mu}_{s,p}$ and the auxiliary variables $\bar{\mathcal{Z}}$ are eliminated [42]. From (5.3), it is possible to derive the local update steps of the ADMM for clients and servers. For client $k \in \mathcal{C}_{s,(q)}$, the primal and dual updates are given by

Client primal update:

$$\begin{aligned}\mathbf{w}_k^{(n)} =& \arg\min_{\mathbf{w}} \frac{1}{D_k}\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}) + \frac{\lambda}{|\mathcal{C}_s|}R(\mathbf{w}) \\ &- \left\langle \boldsymbol{\varphi}_k^{(n-1)}, \mathbf{w} - \mathbf{w}_{s,(q)}^{(n-1)} \right\rangle + \frac{\rho}{2}||\mathbf{w} - \mathbf{w}_{s,(q)}^{(n-1)}||_2^2,\end{aligned} \tag{5.4}$$

Client dual update:

$$\boldsymbol{\varphi}_k^{(n)} = \boldsymbol{\varphi}_k^{(n-1)} + \rho(\mathbf{w}_{s,(q)}^{(n)} - \mathbf{w}_k^{(n)}), \tag{5.5}$$

where the superscript n denotes the iteration number. Further, the primal and dual updates for a server $s \in \mathcal{S}$ are given by:

- **Server primal update:**

$$\begin{aligned}\mathbf{w}_{s,(q)}^{(n)} =& \frac{1}{1+\tau^{(n)}+\rho|\mathcal{N}_s^-|}\Bigg[\frac{1}{|\mathcal{C}_{s,(q)}|}\sum_{k\in\mathcal{C}_{s,(q)}}\mathbf{w}_k^{(n)} \\ &- \frac{1}{\rho|\mathcal{C}_{s,(q)}|}\sum_{k\in\mathcal{C}_{s,(q)}}\boldsymbol{\varphi}_k^{(n-1)} \\ &- \frac{1}{2}\boldsymbol{\psi}_{q,s}^{(n-1)} + \frac{\rho}{2}\sum_{p\in\mathcal{N}_s^-}(\mathbf{w}_{s,(q)}^{(n-1)} - \mathbf{w}_{p,(q)}^{(n-1)}) \\ &+ \tau^{(n)}\frac{1}{Q-1}\frac{1}{|\mathcal{N}_s|}\sum_{r\in\mathcal{Q}\setminus q}\sum_{p\in\mathcal{N}_s}\mathbf{w}_{p,(r)}^{(n-1)}\Bigg],\end{aligned} \tag{5.6}$$

- **Server dual update**

$$\boldsymbol{\psi}_{q,s}^{(n)} = \boldsymbol{\psi}_{q,s}^{(n-1)} + \rho\sum_{p\in\mathcal{N}_s^-}\Big(\mathbf{w}_{p,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}\Big), \tag{5.7}$$

where $\tau^{(n)}$, the inter-cluster learning parameter, is iteration-dependent. Since inter-cluster learning may degrade performance toward the end of the computation, it may be necessary for $\tau^{(n)}$ to follow a decreasing sequence.

The computation in (5.6) performs local aggregation (first two lines), inter-server aggregation (third line), and inter-cluster learning (fourth line) in a single step. This presents the major drawback of using the models of the previous iteration for inter-server aggregation, i.e., $\mathbf{w}_{p,(q)}^{(n-1)}$, and inter-cluster learning, i.e., $\mathbf{w}_{p,(r)}^{(n-1)}$ [14, 15]. A multi-step mechanism addresses this issue by replacing the primal and dual updates of the server as follows:

- **Server aggregation**

$$\widetilde{\mathbf{w}}_{s,(q)}^{(n)} = \frac{1}{|\mathcal{C}_{s,(q)}|} \sum_{k \in \mathcal{C}_{s,(q)}} \mathbf{w}_k^{(n)} - \frac{1}{\rho |\mathcal{C}_{s,(q)}|} \sum_{k \in \mathcal{C}_{s,(q)}} \boldsymbol{\varphi}_k^{(n-1)}. \tag{5.8}$$

- **Inter-server aggregation**

$$\widehat{\mathbf{w}}_{s,(q)}^{(n)} = \frac{1}{|\mathcal{N}_s|} \sum_{p \in \mathcal{N}_s} \widetilde{\mathbf{w}}_{p,(q)}^{(n)}. \tag{5.9}$$

- **Inter-cluster learning**

$$\mathbf{w}_{s,(q)}^{(n)} = \left(1 - \tau^{(n)}\right) \widehat{\mathbf{w}}_{s,(q)}^{(n)} + \frac{\tau^{(n)}}{Q-1} \sum_{r \in \mathcal{Q} \setminus q} \widehat{\mathbf{w}}_{s,(r)}^{(n)}. \tag{5.10}$$

The above multi-step mechanism has two main advantages. First, performing server aggregation prior to inter-server aggregation enables the servers to maintain models composed of the last available client estimates. Second, the fact that inter-cluster learning is performed at the end of the multi-step mechanism ensures that model similarities are leveraged evenly; that is, the same weight is given to any two clients' estimates within the server neighborhood. The resulting PGFL algorithm is summarized in Algorithm 4.

5.2.2 Privacy Preservation in PGFL

This section presents a privacy-preserving variant of the PGFL algorithm that uses dynamic zero-concentrated differential privacy to protect the participants' data.

The motivation for choosing zCDP over conventional (ϵ, δ)-DP is that, like CDP, it offers improved accuracy for identical privacy loss in the worst-case scenario, where an eavesdropper aggregates all the exchanged messages [37, 38]. We have the option to use either CDP or zCDP to preserve privacy in the proposed algorithm, but for simplicity, we choose to use zCDP.

Since the proposed PGFL algorithm is iterative in nature, it is crucial to control privacy protection at every iteration and consider the privacy leakage for the entire

Algorithm 4 PGFL

Initialization: $\mathbf{w}_k^{(0)} = \mathbf{0}$ and $\mathbf{w}_{s,(q)}^{(0)} = 0, \forall k, q, s$
– Procedure at server s –
For iteration $n = 1, 2, \ldots$
  Receive $\{\widetilde{\mathbf{w}}_k^{(n)}, \boldsymbol{\varphi}_k^{(n-1)}; \forall k \in \mathcal{C}_s\}$
  Update $\widetilde{\mathbf{w}}_{s,(q)}^{(n)}$ as in (5.8)
  Share $\widetilde{\mathbf{w}}_{s,(q)}^{(n)}, \forall q$ with each server p in $\mathcal{N}_s^-$
  Receive $\widetilde{\mathbf{w}}_{p,(q)}^{(n)}, \forall q$ from each server p in $\mathcal{N}_s^-$
  Aggregate $\widehat{\mathbf{w}}_{s,(q)}^{(n)}$ as in (5.9)
  Compute $\mathbf{w}_{s,(q)}^{(n)}$ as in (5.10)
  Share $\mathbf{w}_{s,(q)}^{(n)}$ with clients in $\mathcal{C}_s$
EndFor
– Procedure at client $k \in \mathcal{C}_s$ –
For iteration $n = 1, 2, \ldots$
  Update $\mathbf{w}_k^{(n)}$ as in (5.4)
  Share $\mathbf{w}_k^{(n)}$ and $\boldsymbol{\varphi}_k^{(n-1)}$ with server s
  Receive $\mathbf{w}_{s,(q)}^{(n)}$ from server s
  Update $\boldsymbol{\varphi}_k^{(n)}$ as in (5.5)
EndFor

learning process. For this purpose, we adjust the privacy protection dynamically per iteration, as developed in [36], to control the total privacy leakage of the algorithm throughout the computation. In practice, instead of sharing the exact local estimate $\mathbf{w}_k^{(n)}$, a client k shares with its server at iteration n the perturbed estimate $\widetilde{\mathbf{w}}_k^{(n)}$, given by

$$\widetilde{\mathbf{w}}_k^{(n)} = \mathbf{w}_k^{(n)} + \boldsymbol{\xi}_k^{(n)}, \tag{5.11}$$

where the perturbation noise follows a Gaussian mechanism, $\boldsymbol{\xi}_k^{(n)} \sim \mathcal{N}(\mathbf{0}, \delta_k^{2(n)}\mathbf{I})$, with $\delta_k^{2(n)}$ being the variance of the perturbation noise at iteration n.

As seen previously in the thesis, the privacy protection in dynamic zCDP is governed by $\phi_k^{(0)}$, the initial privacy value, and $\zeta \in (0, 1)$, the exponential decay factor of the perturbation noise variance that adjusts the iteration-specific privacy protection dynamically.

Here, for each client, $k \in \mathcal{C}$, the initial noise perturbation variance $\delta_k^{2(0)}$ is fixed,

Algorithm 5 Privacy-preserving PGFL

Initialization: $\mathbf{w}_k^{(0)} = \mathbf{0}$ and $\mathbf{w}_{s,(q)}^{(0)} = \mathbf{0}, \forall k, q, s$
– Procedure at server s –
For iteration $n = 1, 2, \ldots$
Receive $\{\widetilde{\mathbf{w}}_k^{(n)}, \boldsymbol{\varphi}_k^{(n-1)}; \forall k \in \mathcal{C}_s\}$
Update $\widetilde{\mathbf{w}}_{s,(q)}^{(n)}$ as in (5.12)
Share $\widetilde{\mathbf{w}}_{s,(q)}^{(n)}, \forall q$ with each server p in $\mathcal{N}_s^-$
Receive $\widetilde{\mathbf{w}}_{p,(q)}^{(n)}, \forall q$ from each server p in $\mathcal{N}_s^-$
Aggregate $\widehat{\mathbf{w}}_{s,(q)}^{(n)}$ as in (5.9)
Compute $\mathbf{w}_{s,(q)}^{(n)}$ as in (5.10)
Share $\mathbf{w}_{s,(q)}^{(n)}$ with clients in $\mathcal{C}_s$
EndFor
– Procedure at client $k \in \mathcal{C}_s$ –
For iteration $n = 1, 2, \ldots$
Update $\mathbf{w}_k^{(n)}$ as in (5.4)
Perturb $\mathbf{w}_k^{(n)}$ into $\widetilde{\mathbf{w}}_k^{(n)}$ as in (5.11)
Share $\widetilde{\mathbf{w}}_k^{(n)}$ and $\boldsymbol{\varphi}_k^{(n-1)}$ with server s
Receive $\mathbf{w}_{s,(q)}^{(n)}$ from server s
Update $\boldsymbol{\varphi}_k^{(n)}$ as in (5.5) using $\widetilde{\mathbf{w}}_{s,(q)}^{(n)}$ and $\widetilde{\mathbf{w}}_k^{(n)}$.
EndFor

and subsequently, the variance at iteration n is updated according to the relationship $\delta_k^{2(n)} = \zeta \delta_k^{2(n-1)}$. This recursive update ensures a decreasing privacy protection as the algorithm progresses.

The server aggregation (5.8) and client dual update (5.5) are affected by the noise perturbation (5.11). The server aggregation becomes

$$\widetilde{\mathbf{w}}_{s,(q)}^{(n)} = \frac{1}{|\mathcal{C}_{s,(q)}|} \sum_{k \in \mathcal{C}_{s,(q)}} \widetilde{\mathbf{w}}_k^{(n)} - \frac{1}{\rho |\mathcal{C}_{s,(q)}|} \sum_{k \in \mathcal{C}_{s,(q)}} \boldsymbol{\varphi}_k^{(n-1)}, \tag{5.12}$$

and in the client dual update, we substitute $\mathbf{w}_{s,(q)}^{(n)}$ with $\widetilde{\mathbf{w}}_{s,(q)}^{(n)}$ and $\mathbf{w}_k^{(n)}$ with $\widetilde{\mathbf{w}}_k^{(n)}$.

The resulting privacy-preserving algorithm is summarized in Algorithm. 5. In the following sections, we provide a detailed study of the privacy protection and convergence properties of the proposed privacy-preserving PGFL algorithm.

5.3 Theoretical Results

5.3.1 Convergence Analysis

This section studies the convergence behavior of the proposed privacy-preserving PGFL algorithm. Section 5.3.1 studies the algorithm without inter-cluster learning and shows that it converges to the optimal solution of (5.2) with $\tau = 0$ in linear time. Section 5.3.2 then shows the impact of inter-cluster learning. In particular, we show that although inter-cluster learning leads to a different convergence point than intra-cluster learning, the distance between these two points is bounded by a function of the task dissimilarity and the inter-cluster learning parameter sequence. Moreover, we show that this bound can be used to design the inter-cluster learning parameter sequence to achieve a desired convergence point under mild assumptions on cluster similarity, allowing for greater accuracy control in personalized learning while leveraging the task similarity for faster convergence and improved performance.

Problem Reformulation

We consider the server update steps with $\tau^{(n)} = 0$. Then, the minimization problem solved at a client $k \in \mathcal{C}_{s,(q)}$ becomes

$$\begin{aligned} &\min_{\mathbf{w}_k} \frac{1}{D_k} \ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) + \frac{\lambda}{|\mathcal{C}_s|} R(\mathbf{w}_k) \\ &\text{s.t. } \mathbf{w}_k = \widehat{\mathbf{w}}_{s,(q)}, \end{aligned} \tag{5.13}$$

where $\widehat{\mathbf{w}}_{s,(q)}$ is the result of inter-server aggregation (5.9), defined as the average model for cluster q in $\mathcal{N}_s$. To simplify the analysis, we reformulate (5.13) as

$$\begin{aligned} &\min_{\mathbf{w}_k} f_k(\mathbf{w}_k) \\ &\text{s.t. } \mathbf{w}_k = \mathbf{e}_{k,l}, \mathbf{w}_l = \mathbf{e}_{k,l}, \forall l \in \sum_{p \in \mathcal{N}_s} \mathcal{C}_{p,(q)}, \end{aligned} \tag{5.14}$$

where $f_k(\mathbf{w}_k)$ is given by

$$f_k(\mathbf{w}_k) = \frac{1}{D_k} \ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) + \frac{\lambda}{|\mathcal{C}_s|} R(\mathbf{w}_k), \tag{5.15}$$

and the auxiliary variables $\{\mathbf{e}_{k,l}\}, \forall k, l \in \sum_{p \in \mathcal{N}_s} \mathcal{C}_{p,(q)}$ enforce consensus. To reformulate (5.14) further, we introduce the following:

$$\begin{aligned}
&\mathbf{w} = [\mathbf{w}_1^\intercal, \ldots, \mathbf{w}_k^\intercal, \ldots \mathbf{w}_K^\intercal]^\intercal, \\
&\widetilde{\mathbf{w}} = [\widetilde{\mathbf{w}}_1^\intercal, \ldots, \widetilde{\mathbf{w}}_k^\intercal, \ldots \widetilde{\mathbf{w}}_K^\intercal]^\intercal = \mathbf{w} + \boldsymbol{\xi} \\
&\boldsymbol{\varphi} = [\boldsymbol{\varphi}_1^\intercal, \ldots, \boldsymbol{\varphi}_k^\intercal, \ldots, \boldsymbol{\varphi}_K^\intercal]^\intercal, \\
&F(\mathbf{w}) = \sum_{k \in \mathcal{C}} f_k(\mathbf{w}_k),
\end{aligned} \tag{5.16}$$

where $\boldsymbol{\xi}$ is the concatenation of the noise added to the local models to ensure privacy. In addition, we introduce the vector $\mathbf{e} \in \mathbb{R}^{2Md}$ concatenating the vectors $\mathbf{e}_{k,l}, \mathbf{e}_{l,k}, \forall (k,l) \in \{1, \ldots, K\} : k \neq l$, where d is the dimension of the models and M is the number of constraints in (5.14). We can then reformulate (5.14) as

$$\begin{aligned}
&\min_{\mathbf{w}} F(\mathbf{w}) \\
&\text{s.t. } \mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{e} = \mathbf{0}.
\end{aligned} \tag{5.17}$$

where $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2]$ and $\mathbf{B} = [-\mathbf{I}_{2Md}, -\mathbf{I}_{2Md}]$. The matrices $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{2Md \times Kd}$ are composed of $d \times d$-sized blocks. Given a couple of connected clients (k,l), their associated auxiliary variable $\mathbf{e}_{k,l}$, and its corresponding index in $\mathbf{e}$, q; the blocks $\left(\mathbf{A}_1\right)_{q,k}$ and $\left(\mathbf{A}_2\right)_{q,l}$ are equal to the identity matrix $\mathbf{I}_d$, all other blocks are null.

From the above definitions, one can express $\sum_{\mathbf{e}_{k,l} \in \mathbf{e}} \|\mathbf{w}_k - \mathbf{e}_{k,l}\|^2 + \|\mathbf{w}_l - \mathbf{e}_{k,l}\|^2 = \|\mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{e}\|^2$ and, for $\boldsymbol{\lambda} \in \mathbb{R}^{4Md}$, $\sum_{k \in \mathcal{C}} \sum_{l \in \mathcal{N}_k} (\langle \mathbf{w}_k - \mathbf{e}_{k,l}, \boldsymbol{\lambda}_q \rangle + \langle \mathbf{w}_l - \mathbf{e}_{k,l}, \boldsymbol{\lambda}_{2e+q} \rangle) = \langle \mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{e}, \boldsymbol{\lambda} \rangle$.

Therefore, the Lagrangian can be rewritten as

$$\mathcal{L}_\rho(\mathcal{V}_q, \mathcal{M}) = F(\mathbf{w}) + \langle \mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{e}, \boldsymbol{\lambda} \rangle + \frac{\rho}{2} \|\mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{e}\|^2. \tag{5.18}$$

Convergence Proof

We make the following assumptions to continue the analysis.

Assumption 1. *The functions $f_k(\cdot), k \in \{1, \ldots, K\}$, are convex and smooth.*

Using (5.18), and under Assumption 1, the steps of the PGFL algorithm without inter-cluster learning can be expressed as follows:

$$\begin{aligned}\nabla F(\mathbf{w}^{(n+1)}) + \mathbf{A}^\mathsf{T}\boldsymbol{\lambda}^{(n)} + \rho\mathbf{A}^\mathsf{T}\Big(\mathbf{A}\mathbf{w}^{(n+1)} + \mathbf{B}\mathbf{e}^{(n)}\Big) = 0,\\ \mathbf{B}^\mathsf{T}\boldsymbol{\lambda}^{(n)} + \rho\mathbf{B}^\mathsf{T}\Big(\mathbf{A}\widetilde{\mathbf{w}}^{(n+1)} + \mathbf{B}\mathbf{e}^{(n+1)}\Big) = 0,\\ \boldsymbol{\lambda}^{(n+1)} - \boldsymbol{\lambda}^{(n)} + \rho\Big(\mathbf{A}\widetilde{\mathbf{w}}^{(n+1)} + \mathbf{B}\mathbf{e}^{(n+1)}\Big) = 0.\end{aligned} \tag{5.19}$$

Similarly to [46], we introduce the following to simplify (5.19):

$$\begin{aligned}\mathbf{H}_+ &= \mathbf{A}_1^\mathsf{T} + \mathbf{A}_2^\mathsf{T}, & \mathbf{H}_- &= \mathbf{A}_1^\mathsf{T} - \mathbf{A}_2^\mathsf{T},\\ \mathbf{L}_+ &= \frac{1}{2}\mathbf{H}_+\mathbf{H}_+^\mathsf{T}, & \mathbf{L}_- &= \frac{1}{2}\mathbf{H}_-\mathbf{H}_-^\mathsf{T},\\ \boldsymbol{\alpha} &= \mathbf{H}_-^\mathsf{T}\mathbf{w}, & \mathbf{M} &= \frac{1}{2}(\mathbf{L}_+ + \mathbf{L}_-).\end{aligned}$$

Then, as derived in [46, Section II.B], (5.19) becomes

$$\begin{aligned}\nabla F(\mathbf{w}^{(n+1)}) + \boldsymbol{\alpha}^{(n)} + 2\rho\mathbf{M}\mathbf{w}^{(n+1)} - \rho\mathbf{L}_+\widetilde{\mathbf{w}}^{(n)} = 0,\\ \boldsymbol{\alpha}^{(n+1)} - \boldsymbol{\alpha}^{(n)} - \rho\mathbf{L}_-\widetilde{\mathbf{w}}^{(n+1)} = 0.\end{aligned} \tag{5.20}$$

As in [47, Lemma 1], the equations in (5.20) can be combined to obtain

$$\begin{aligned}\mathbf{w}^{(n+1)} =& \frac{\mathbf{M}^{-1}\nabla F(\mathbf{w}^{(n+1)})}{2\rho} + \frac{\mathbf{M}^{-1}\mathbf{L}_+\widetilde{\mathbf{w}}^{(n)}}{2}\\ &- \frac{\mathbf{M}^{-1}\mathbf{L}_-}{2}\sum_{s=0}^{n}\widetilde{\mathbf{w}}^{(s)}.\end{aligned} \tag{5.21}$$

Similarly to [47], by introducing the following:

$$\begin{aligned}\mathbf{Q} &= \sqrt{\mathbf{L}_-/2}, & \mathbf{r}^{(n)} &= \sum_{s=0}^{n}\mathbf{Q}\widetilde{\mathbf{w}}^{(s)},\\ \mathbf{q}^{(n)} &= \begin{pmatrix}\mathbf{r}^{(n)}\\ \widetilde{\mathbf{w}}^{(n)}\end{pmatrix}, & \mathbf{G} &= \begin{bmatrix}\rho\mathbf{I} & 0\\ 0 & \rho\mathbf{L}_+/2\end{bmatrix},\end{aligned}$$

(5.21) can be reformulated using [47, Lemma 2] as

$$\frac{\nabla F(\mathbf{w}^{(n+1)})}{\rho} + 2\mathbf{Q}\mathbf{r}^{(n+1)} + \mathbf{L}_+\Big(\mathbf{w}^{(n+1)} - \widetilde{\mathbf{w}}^{(n)}\Big) = 2\mathbf{M}\boldsymbol{\xi}^{(t+1)}. \tag{5.22}$$

Theorem 5.1. Let Assumption 1 hold true, if $\tau^{(n)} = \tau = 0, \forall n$, the proposed PGFL algorithm converges to the optimal solution of (5.2) in linear time for each cluster.

Proof. Under Assumption 1, $F(\mathbf{w})$ is convex and smooth by composition and, therefore, differentiable. Using [39, Lemma 6] and [39, Theorem V] with a convex and smooth function $F(\mathbf{w})$ demonstrates that the proposed PGFL algorithm, without inter-cluster learning ($\tau = 0$), converges to the optimal solution of (5.2) in linear time for any given cluster. □

5.3.2 Inter-cluster learning Analysis

In situations with limited data, as demonstrated in Section V, employing inter-cluster learning ($\tau \neq 0$) can enhance performance compared to $\tau = 0$. This section establishes an upper bound on the disparity between the resulting cluster-specific personalized models obtained in scenarios with and without inter-cluster learning. It is worth noting that this bound can be controlled by properly choosing the sequence $\tau(n)$.

To do so, it is necessary to reformulate the client primal update using Assumption 1. The primal update for client $k \in \mathcal{C}_{s,(q)}$ is expressed as follows:

$$\mathbf{w}_k^{(n+1)} = \arg\min_{\mathbf{w}} f_k(\mathbf{w}) - \left\langle \boldsymbol{\varphi}_k^{(n)}, \mathbf{w} - \mathbf{w}_{s,(q)}^{(n)} \right\rangle + \frac{\rho}{2}||\mathbf{w} - \mathbf{w}_{s,(q)}^{(n)}||^2, \tag{5.23}$$

which, under Assumption 1, is equivalent to

$$\nabla f_k(\mathbf{w}_k^{(n+1)}) - \boldsymbol{\varphi}_k^{(n)} + \rho\Big(\mathbf{w}_k^{(n+1)} - \mathbf{w}_{s,(q)}^{(n)}\Big) = 0. \tag{5.24}$$

Further reformulation leads to the following:

$$\mathbf{w}_k^{(n+1)} = \mathbf{w}_{s,(q)}^{(n)} + \frac{1}{\rho}\boldsymbol{\varphi}_k^{(n)} - \frac{1}{\rho}\nabla f_k(\mathbf{w}_k^{(n+1)}). \tag{5.25}$$

By replacing $\mathbf{w}_k^{(n+1)}$ with (5.25) in (5.8), we obtain

$$\widehat{\mathbf{w}}_{s,(q)}^{(n)} = \frac{1}{|\mathcal{N}_s|}\sum_{p\in\mathcal{N}_s}\frac{1}{|\mathcal{C}_{p,(q)}|}\sum_{k\in\mathcal{C}_{p,(q)}}\Big(\mathbf{w}_{p,(q)}^{(n-1)} - \frac{1}{\rho}\nabla f_k(\mathbf{w}_k^{(n)})\Big). \tag{5.26}$$

Next, we investigate the effect of inter-cluster learning by comparing the performance of models obtained using the PGFL algorithm with and without inter-cluster learning. We shall prove that the difference between the resulting models is bounded and depends on both the inter-cluster learning parameter and the similarity of models between clusters.

Theorem 5.2. Given a sufficiently large penalty parameter ρ, for all iterations, server $s \in \mathcal{S}$ and cluster $q \in \mathcal{Q}$, the impact of inter-cluster learning after n iterations is bounded by

$$\mathbb{E}\Big[||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}||_2^2\Big] \leqslant \sum_{i=1}^{n}\Big(\prod_{j=i+1}^{n}\big(1-\tau^{(j)}\big)\Big)\tau^{(i)}\eta, \tag{5.27}$$

where the expectation is taken with respect to the privacy-related noise added in (5.11) and the data observation noise, $\bar{\mathbf{w}}_{s,(q)}^{(n)}$ denotes the model obtained by the algorithm without inter-cluster learning, and η is the maximum cluster model distance, defined as:

$$\eta = \max_{q,r\in\mathcal{Q}} \Big\|\mathbf{w}_{(q)}^{*} - \mathbf{w}_{(r)}^{*}\Big\|_2^2, \tag{5.28}$$

with the models $\mathbf{w}_{(q)}^{*}, q \in \mathcal{Q}$ being the cluster-specific solutions of (5.2) with $\tau = 0$.

Proof. See **P5**, page 6. □

Corollary. *If $\tau^{(i)} = 0, \forall i < n$ and $\tau^{(n)} \neq 0$, the impact of a single iteration of inter-cluster learning is bounded by*

$$\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}||_2^2 \leqslant \tau^{(n)}\eta, \tag{5.29}$$

where $\bar{\mathbf{w}}_{s,(q)}^{(n)}$ denotes a model obtained without inter-cluster learning, η is as defined in Theorem 5.2, and the expectation is taken with respect to the privacy-related and observation noises.

Theorem 5.2 bounds the difference in the resulting models with and without inter-cluster learning. Combining Theorems 5.1 and 5.2, the resulting models obtained by the algorithms are guaranteed to reside within a neighborhood of the optimal solution of (5.2) with $\tau = 0$. The size of this neighborhood can be adjusted by selecting the sequence $\tau^{(n)}$. When ample data is available, the algorithm converges to a satisfactory solution within this neighborhood. However, in cases of limited data, the solution of (5.2) with $\tau = 0$ may be inadequate. In such situations, inter-cluster learning becomes crucial, allowing the proposed algorithm to achieve higher accuracy, as demonstrated in Section V. By exploiting inter-cluster learning, the algorithm effectively overcomes the limitations imposed by scarce data, leading to improved performance.

5.3.3 Privacy Analysis

This section focuses on quantifying the local privacy protection provided by the proposed PGFL algorithm. To achieve this, we begin by calculating the l_2-norm sensitivity, which quantifies the variation in output resulting from a change in an individual data sample. Once we have established the l_2-norm sensitivity, we proceed to adjust the noise variance added to the primal variables, ensuring satisfactory protection.

Definition 1. *The l_2-norm sensitivity is defined by*

$$\Delta_{k,2}^{(n)} = \max_{\mathcal{D}_k, \mathcal{D}_l} \left\| \mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}_l}^{(n)} \right\| \tag{5.30}$$

where $\mathbf{w}_{k,\mathcal{D}_k}^{(n)}$ and $\mathbf{w}_{k,\mathcal{D}_l}^{(n)}$ denote the local primal variables obtained from two neighboring data sets $\mathcal{D}_k$ and $\mathcal{D}_l$, which differ in only one data sample.

Assumption 3. *The functions $\ell_k(\cdot)$, $k \in \mathcal{C}$, have bounded gradients. That is, for $k \in \mathcal{C}$ there exists a constant C_k such that $||\nabla \ell_k(\cdot)|| \leqslant C_k$.*

Lemma 5.1. *Let Assumption 3 hold true, the l_2-norm sensitivity for a client k is given by*

$$\Delta_{k,2}^{(n)} = \max_{\mathcal{D}_k, \mathcal{D}_l} ||\mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}_l}^{(n)}|| = \frac{2C_k}{\rho D_k}. \tag{5.31}$$

Proof. See **P5**, page 7. □

With the l_2-norm sensitivity, we can establish the relation between the noise variance added in (5.11) and the privacy parameter $\phi_k^{(n)}$ as well as prove the privacy guarantee of the algorithm in terms of zCDP.

Theorem 5.3. Let Assumption 3 hold true, the Privacy-preserving PGFL algorithm satisfies dynamic $\phi_k^{(n)}$-zCDP with the relation between the privacy parameter and the perturbation noise variance given by

$$\delta_k^{2(n)} = \frac{\Delta_{k,2}^{(n)2}}{2\phi_k^{(n)}}. \tag{5.32}$$

Proof. See **P5**, page 8. □

Theorem 5.3 gives the relationship between the noise perturbation variance and the privacy protection at a given iteration. Since the proposed algorithm is iterative

in nature and models are exchanged several times with the servers, one should consider the total privacy loss throughout the learning process. To this aim, we establish the following theorem.

Theorem 5.4. *Let Assumption 3 hold true, for a final iteration N, the PGFL algorithm satisfies ϕ_k^{total}-zCDP throughout the entire computation for each client k, with ϕ_k^{total} given by*

$$\phi_k^{\text{total}} = \sum_{n=1}^{N} \phi_k^{(n)}. \tag{5.33}$$

Proof. See **P5**, page 8. □

5.4 Numerical Results

This section illustrates the performance of the proposed PGFL algorithm for solving regression and classification tasks.

5.4.1 Experiments for Regression

We consider a graph federated network consisting of $|\mathcal{S}| = 10$ servers, each having access to $|\mathcal{C}_s| = 15$ clients, for a total of $K = 150$ clients. The set of servers and their communication channels form a random connected graph where the average node degree is three. Each client has access to a random number of noisy data samples between $D_k = 2$ and $D_k = 9$, each composed of a vector $\mathbf{x}_{k,i}$ of dimension $d = 60$ and a response scalar $y_{k,i}$. Doing so, each cluster is globally observable but not locally at any given client or set $\mathcal{C}_s, s \in \mathcal{S}$. The servers implement random scheduling of clients to reduce the communication load [12]. In particular, at every global iteration, each server randomly selected a subset of three clients to participate in the learning process.

The clients of the network are randomly split between $Q = 3$ clusters. Clients of a given cluster solve the ridge regression problem with data generated from an original model $\mathbf{w}_{(q)}^*$, obtained with $\mathbf{w}_{(q)}^* = \mathbf{w}_0^* + \gamma \mathbf{w}_0^*$ with $\gamma \sim \mathcal{U}(-0.15, 0.15)$, where $\mathbf{w}_0^*$ is a base model. In doing so, the learning tasks of the different clusters share the same objective functions but have different, related data distributions. The loss and regularizer functions are given by

$$\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{y}_k - \mathbf{X}_k \mathbf{w}_k||^2, R(\mathbf{w}_k) = ||\mathbf{w}_k||^2. \tag{5.34}$$

Performance is evaluated by computing the normalized mean squared deviation (NMSD) of the local models with respect to the corresponding cluster-specific

original model used to generate the data, $\mathbf{w}_{(q)}^*$ for $k \in \mathcal{C}_{(q)}$. It is given by:

$$\gamma^{(n)} = \frac{1}{K} \sum_{q=1}^{|\mathcal{Q}|} \sum_{k \in \mathcal{C}_{(q)}} \frac{\left\| \mathbf{w}_k^{(n)} - \mathbf{w}_{(q)}^* \right\|_2^2}{\left\| \mathbf{w}_{(q)}^* \right\|_2^2}, \tag{5.35}$$

where the result is averaged over several Monte Carlo iterations. The proposed algorithm is compared with various existing algorithms. The ClusterFL algorithm, defined in [88], implements conventional personalized FL with inter-cluster learning. For a fair comparison, the ClusterFL algorithm has been modified to leverage similarity among tasks in the same manner as the PGFL algorithm. The GFL algorithm, defined in [9], implements single-task graph FL in a privacy-preserving manner. To ensure a fair comparison, the ClusterFL and GFL algorithms have been modified to ensure privacy in the same manner as the PGFL algorithm. Furthermore, the algorithms are set to observe the same initial convergence rate whenever possible. For most experiments, the learning curves are displayed as plots of the NMSD versus the iteration index.

We first consider an ideal setting wherein all algorithms are evaluated without privacy considerations ($\boldsymbol{\xi}^{(n)} = \mathbf{0},\ \forall n$) and client scheduling. In this scenario, the inter-cluster parameter $\tau^{(n)}$ of the PGFL algorithm was kept fixed throughout the learning, specifically, $\tau^{(n)} = 0$ and $\tau^{(n)} = 0.4$. Figure 5.1 (a) shows the learning curves for the GFL, ClusterFL, and PGFL algorithms. The results illustrate the superiority of the proposed PGFL algorithm over GFL, as cluster-specific learning tasks benefit significantly from personalized models tailored to each cluster. We also see that incorporating inter-cluster learning results in improved convergence speed and steady-state accuracy. Furthermore, the performance of the ClusterFL algorithm is notably poor in this setting, emphasizing the importance of using the graph federated architecture when data is scarce. Leveraging the model similarities improves learning speed and accuracy by compensating for data scarcity. In addition, isolated servers whose clients lack sufficient data to achieve satisfactory accuracy independently reinforce the necessity of the graph federated architecture.

Next, we modify the setting to incorporate client scheduling and evaluate the aforementioned algorithms with reduced communication load. Figure 5.1 (b) shows the learning curves for the GFL, ClusterFL, and PGFL algorithms with client scheduling. In this figure and the ones below, 3 clients out of 15 are randomly selected to participate by each server at every iteration, reducing the communication load by 80% for every algorithm. We observe that the PGFL algorithm exhibits slower convergence and higher steady-state NMSD when utilizing client scheduling. And we note that GFL performs better with client scheduling. The performance degradation for the PGFL algorithm is due to the lower client participation resulting

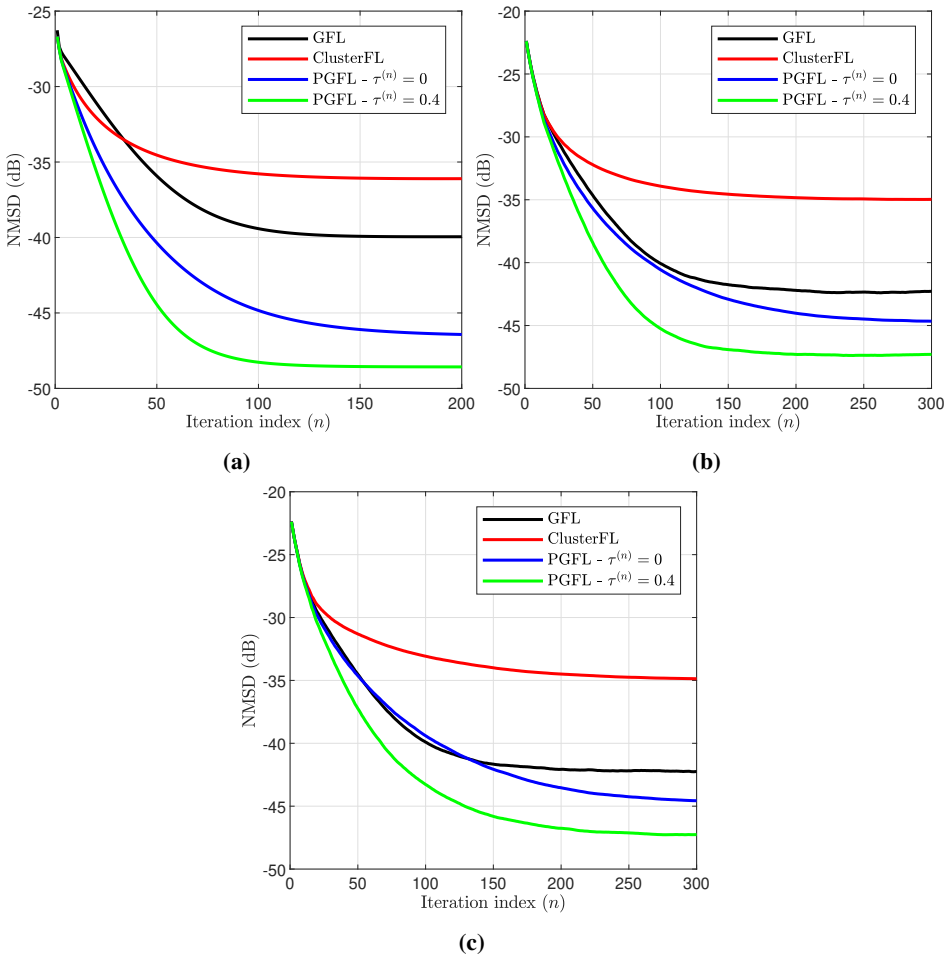


Figure 5.1: Learning curves of the PGFL algorithm with a fixed inter-cluster learning parameter. (a) without client scheduling or privacy, (b) with client scheduling without privacy, (c) with client scheduling and privacy.

in a smaller quantity of data being utilized. The better performance of GFL in this setting is due to the imbalance of cluster representation in the universal model, which benefits the participating clients on average.

Finally, we evaluate the aforementioned algorithms in a setting with client scheduling and privacy protection. All of the algorithms utilize zCDP with the noise perturbation presented in (5.11) and the parameters $\phi_k^{(0)} = 0.001, \forall k$ and $\zeta = 0.99$. Hence, all the algorithms satisfy ϕ_k^{final}-zCDP throughout the computation with $\phi_k^{\text{final}} = 0.095, \forall k$. Figure 5.1 (c) shows the learning curves for the GFL,

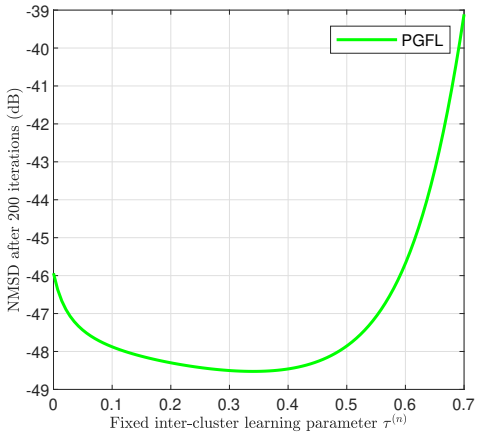


Figure 5.2: NMSD after 200 iterations vs. fixed inter-cluster learning parameter $\tau^{(n)}$ values for the PGFL algorithm with client scheduling and privacy .

ClusterFL, and PGFL algorithms with client scheduling and privacy. We observe that the noise perturbation associated with differential privacy significantly reduces the convergence speed of all the simulated algorithms. However, we note that the NMSD after 300 iterations is nearly identical to the one in Fig. 5.1 (b). This behavior is explained by the use of zCDP, in which the variance of the noise perturbation starts high and decreases linearly throughout the learning process.

Further, we illustrate the importance of carefully choosing the value of the inter-cluster learning parameter. In Fig. 5.2, we simulated the proposed PGFL algorithm for various fixed $\tau^{(n)}$ values and displayed the NMSD after 200 iterations. For instance, the NMSD for $\tau^{(n)} = 0.4$ corresponds to the result obtained in Fig. 5.1 (c). This figure confirms that inter-cluster learning has the potential to increase learning performance by alleviating data scarcity, as the PGFL algorithm achieves lower NMSD with $\tau^{(n)} \in (0.1, 0.5)$ than with $\tau^{(n)} = 0$. It also shows that the inter-cluster learning parameter must be carefully selected, as a value too large for the setting leads to performance degradation.

We then illustrate an alternative use of inter-cluster learning. For this experiment, the difference between the data distribution of the different clusters has been increased. Precisely, the datasets were simulated with the models obtained by $\mathbf{w}_{(q)} = \mathbf{w}_0 + \gamma \mathbf{w}_0$ with $\gamma \sim \mathcal{U}(-0.5, 0.5)$. The learning curves are presented in Fig. 5.3. We observed that, because of the higher cluster dissimilarity, inter-cluster learning degrades steady-state NMSD; this is observed in the learning curves for PGFL with $\tau^{(n)} = 0$ and $\tau^{(n)} = 0.4$. However, by mitigating data scarcity within a cluster, inter-cluster learning improves the initial convergence rate. To bene-

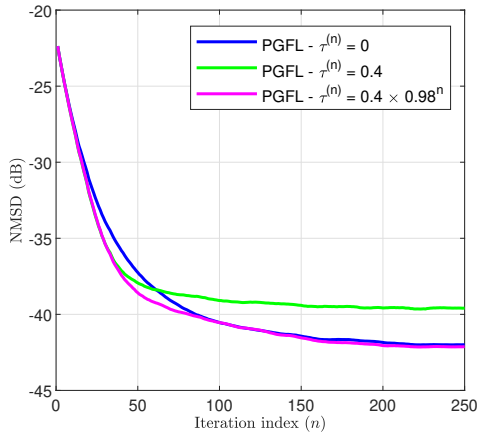
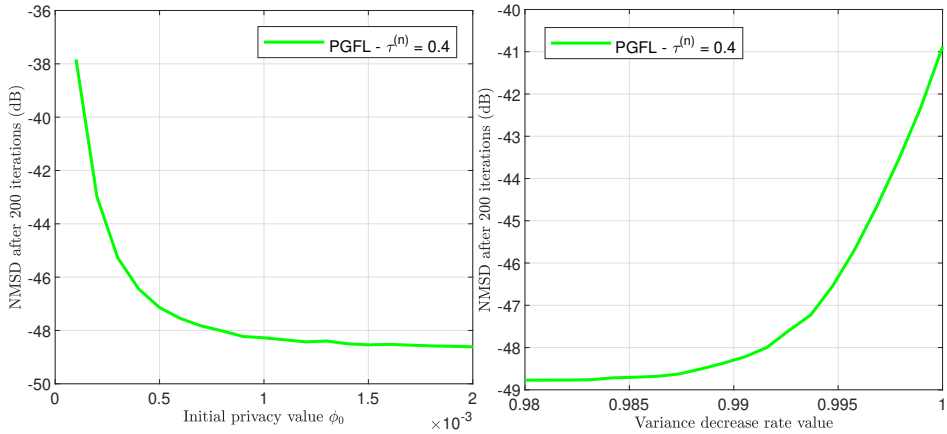


Figure 5.3: Learning curves of the PGFL algorithm with fixed and time-varying inter-cluster learning parameter $\tau^{(n)}$ in a setting with low cluster similarity, considering client scheduling and privacy.

fit from an improved initial convergence rate and avoid steady-state performance degradation, it is possible to reduce the inter-cluster learning parameter progressively. Doing so, the PGFL algorithm with time-varying $\tau^{(n)} = 0.4 \times 0.98^n$ has the same initial convergence rate as the PGFL algorithm with fixed $\tau = 0.4$ and attains near-identical steady-state NMSD as the PGFL algorithm with fixed $\tau = 0$.



(a) **(b)**

Figure 5.4: Privacy-accuracy trade-off of the PGFL algorithm with a fixed inter-cluster learning parameter, considering client scheduling. (a) for various ζ, (b) for various $\phi^{(0)}$.

Finally, we study the impact of privacy protection on the steady-state NMSD of

the PGFL algorithm. Fig. 5.4 (a) shows the NMSD after 200 iterations versus the initial value of the privacy parameter ϕ_0 for a decaying rate of $\zeta = 0.99$. Note that, as seen in Theorem III, a lower value of ϕ_0 ensures more privacy. We observe that for smaller values of ϕ_0, the steady-state NMSE of the PGFL algorithm is higher. In fact, a lower total privacy loss bound leads to higher perturbation noise variance and diminishes the learning performance of the algorithm. Similarly, Fig. 5.4 (b) shows the NMSD after 200 iterations versus the variance decrease rate ζ for an initial privacy value of $\phi_0 = 0.001$. The lower the decrease rate, the faster the privacy protection weakens, and the lower the steady-state NMSE of the algorithm as more information is exchanged among clients. On the other hand, a decrease rate close to 1 ensures better privacy protection but comes at the cost of lower accuracy.

5.4.2 Experiments for Classification on the MNIST Dataset

The following experiments were conducted on the MNIST handwritten digits dataset [89]. In those experiments, the learning tasks of the clients associated with different clusters share the same data but have different, related, objective functions. The structure of the server network, as well as the number of clients per server, are identical to the experiments for regression. In the following experiments, the clients of a given cluster use the ADMM for logistic regression to differentiate between two classes. The loss function for the logistic regression is given by

$$\log[\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k)] = \frac{-1}{D_k} \sum_{i=1}^{D_k} \Big(y_{k,i} \log[y'_{k,i}] + (1 - y_{k,i}) \log[1 - y'_{k,i}] \Big), \tag{5.36}$$

with

$$y'_{k,i} = \frac{1}{1 + \exp(-\mathbf{w}_k^\intercal \mathbf{x}_{k,i})}. \tag{5.37}$$

We simulated the PGFL algorithm in the context of classification with client scheduling, privacy, a fixed inter-cluster learning parameter $\tau^{(n)} = \tau = 0.4$, and without inter-cluster learning $\tau^{(n)} = 0$. Figure 5.5 (a) shows the test accuracy versus iteration index in a setting the clients of a given cluster must differentiate between two classes composed of a single digit. Each client receives between $D_k = 2$ and $D_k = 4$ data samples composed of two MNIST images. The clients of cluster 1 have access to images of the digits $\{1\}$ and $\{8\}$. The clients of clusters 2 and 3 have access to images of the digits $\{1\}$ and $\{9\}$, and $\{7\}$ and $\{8\}$, respectively. Given that the clients of different clusters must differentiate between different digits, the similarity between the learning task is limited. Nevertheless, we observe

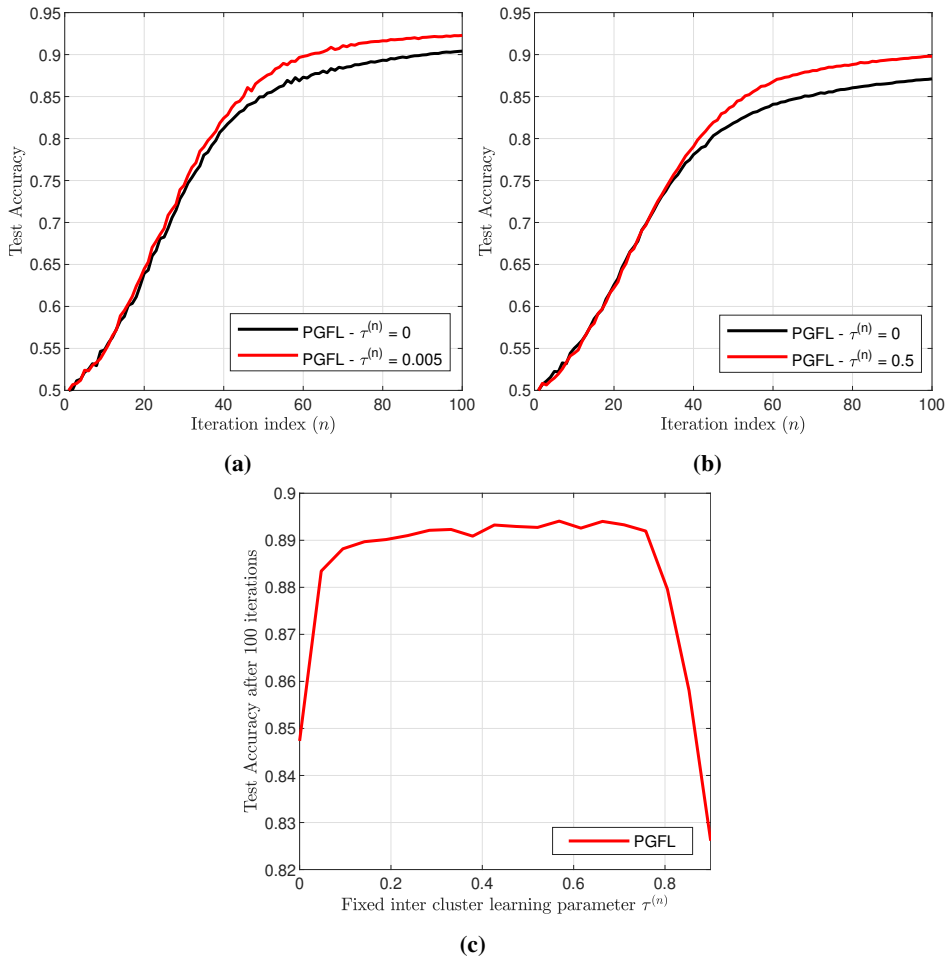


Figure 5.5: Performance of the PGFL algorithm in the MNIST classification task, considering client scheduling and privacy. Test accuracy curves with and without inter-cluster learning (a) with low task similarity, (b) with high task similarity, and (c) accuracy after 100 iterations as a function of $\tau^{(n)}$.

that inter-cluster learning does improve the accuracy of the PGFL algorithm in this setting.

Further, we modified the setting so that the clusters exhibit more similarity. Figure 5.5 (b) shows the test accuracy versus iteration index in a setting where the clients of a given cluster must differentiate between two classes composed of triplets of digits. Each client receives between $D_k = 6$ and $D_k = 12$ data samples, each composed of two triplets of MNIST images. The clients of cluster 1 must differ-

entiate between the classes $\{1, 2, 3\}$ and $\{6, 7, 8\}$, the clients of cluster 2 between $\{1, 2, 3\}$ and $\{7, 8, 9\}$, and the clients of cluster 3 between $\{1, 2, 3\}$ and $\{6, 8, 9\}$. We observe that, in this setting, inter-cluster learning significantly improves the accuracy of the PGFL algorithm.

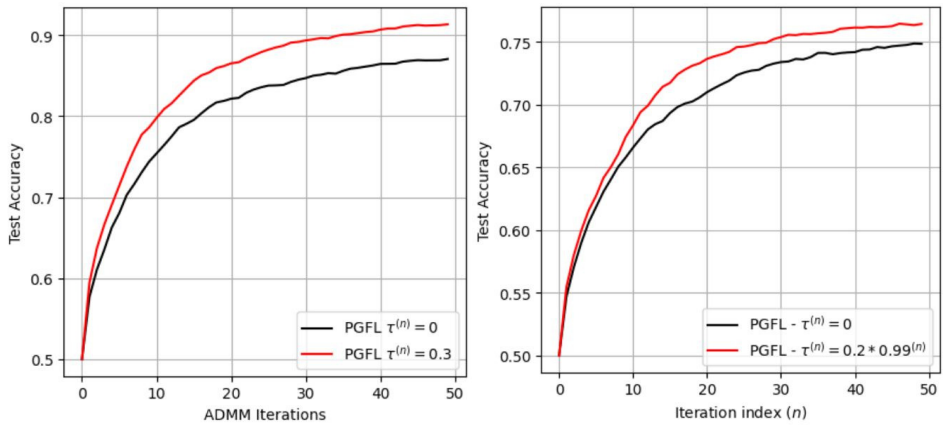
Finally, we utilize the previous setting and evaluate the impact of the value of the inter-cluster learning parameter $\tau^{(n)}$ on the accuracy achieved by the PGFL algorithm in the context of classification. Figure 5.5 (c) displays the accuracy achieved by the PGFL algorithm after 100 iterations versus the value of the inter-cluster learning parameter in the context of the classification task of Fig. 5.5 (b). We observe that, in this setting where the similarity among the learning tasks is high, medium and large fixed values for $\tau^{(n)}$ lead to significant accuracy improvement. However, very large values lead to performance degradation, similar to Fig. 5.2.

5.4.3 Experiments for Classification on the MedMNIST Dataset

To demonstrate the proposed method of utilizing inter-cluster learning to palliate data scarcity and improve learning performance in real-life applications, two experiments are conducted on the OrganAMNIST dataset, part of the biomedical MedMNIST dataset [90]. The OrganAMNIST dataset contains lightweight images of 11 different organs labeled by type. It comprises more than 58000 data samples split into training, validation, and testing data. We use the proposed method to improve classification accuracy in the following setting. The server network and the loss function are identical to previous experiments; however, only three clients are associated with each server, each client having access to two data samples. In both experiments, clients of a given cluster are tasked with differentiating between two types of organs. Different clusters are associated with different pairs of organs, and inter-cluster learning is utilized to improve classification accuracy by leveraging the similarity between some of the organs.

In the first experiment, the three clusters are given similar learning tasks. In particular, one of the elements of each pair of organs is identical. Cluster 1 differentiates between the right lung and the left lung, cluster 2 between the liver and the left lung, and cluster 3 between the right kidney and the left lung. Figure 5.6 (a) shows the test accuracy versus iteration index. We observe that a large amount of inter-cluster learning leads to significantly improved performances, increasing classification accuracy by about 5%.

In the next experiment, the learning tasks associated with each cluster are less similar than in the previous experiment. They share only the vague shape of the classified organs. Cluster 1 differentiates between the spleen and the left lung,



(a) **(b)**

Figure 5.6: Test accuracy curve of the PGFL algorithm with a fixed inter-cluster learning parameter on MedMNIST, considering privacy. (a) with high cluster similarity, (b) with low cluster similarity.

cluster 2 between the left kidney and the bladder, and cluster 3 between the right kidney and the right lung. Due to the lower cluster similarity, we utilize a decaying inter-cluster learning parameter to preserve steady-state accuracy. Figure 5.6 (b) shows the test accuracy versus iteration index. We observe that a medium decay rate of the inter-cluster learning parameter can improve the learning speed, boosting classification accuracy by about 2%.

5.5 Summary

This chapter presented the PGFL framework and its privacy-preserving implementation. The purpose of the PGFL framework is to enable large-scale personalized FL by removing the arbitrary limitation on the number of clients and, therefore, the number of models learned caused by the use of a single-server architecture. In PGFL, distributedly connected servers collaborate with each other and their associated clients to learn cluster-specific personalized models. Furthermore, the similarities among clusters are leveraged to improve learning speed and alleviate data scarcity. The presented algorithm utilizes zCDP to propose a secure and ready-to-use implementation of PGFL. Its mathematical analysis showed that this algorithm converges to the exact optimal solution for each cluster in linear time and that utilizing inter-cluster learning leads to an alternative output whose distance to the original solution is bounded by a value that can be adjusted with the inter-cluster learning parameter sequence. Finally, numerical simulations showed that the proposed method is capable of leveraging the graph federated architec-

ture and the similarity between the clusters learning tasks to improve learning performance. We have observed that the inter-cluster learning parameter sequence is essential for analytical and numerical performances. For this reason, ensuring its value is as close to the optimum as possible is key. This is the purpose of the next chapter.

Chapter 6

Networked Federated Learning with Reinforcement Learning Based Personalization

This chapter, which presents the results of publication **P7**, addresses personalized FL in a networked architecture by using reinforcement learning. In personalized federated learning, each client or group of clients learns a client- or cluster-specific model personalized for their local needs. A single client or cluster often does not possess sufficient data to learn a satisfactory model, leading to performance degradation. As seen previously, it is possible to leverage the similarities often present between the various learning tasks to alleviate data scarcity and enhance learning performance. While this can be done among clusters as a whole in most architectures, the use of a networked architecture presents additional challenges. In fact, given a distributed network of devices performing various learning tasks, both the data distribution and the cluster representation are likely to be uneven. Without a central coordinator, clients must rely on their direct neighborhood to perform their learning tasks, which may or may not include sufficient cluster-specific data. Therefore, uncontrolled inter-cluster learning may lead to performance degradation due to over- or under-usage of local task similarity. In light of this issue, an intelligent mechanism that performs inter-cluster learning based on client-specific needs is required. This chapter introduces such a mechanism, using reinforcement learning principles to control device-specific inter-cluster learning in real-time.

6.1 Motivation

Inter-cluster learning is frequently used in personalized learning as it enables the learning of task-specific personalized models despite the lack of data associated with these specific tasks [14, 15]. In addition, when enough data is available, it can be used to increase learning speed [17]. However, the fundamental behavior of inter-cluster learning is to enforce similarity across models designed for non-identical tasks. Therefore, inter-cluster learning can degrade accuracy if not adequately tuned and controlled over time. For instance, as observed in the previous chapter, it may be necessary to progressively reduce the inter-cluster learning parameter to avoid steady-state performance degradation when used to increase learning speed. In a distributed network, both data and clusters may be unevenly distributed, leading to widely varying amounts of cluster-relevant information being available within a client's neighborhood. Furthermore, clients must rely solely on their neighborhood to complete their learning tasks. For this reason, two clients of the same cluster may require different degrees of inter-cluster learning. Therefore, in the networked architecture, inter-cluster learning must be controlled in space as well as in time. This raises the need for an intelligent mechanism that controls client-specific inter-cluster learning parameters in real-time throughout the computation so that it is only used when it improves performance.

The work in [17] proposes a rule to disable inter-cluster learning if it becomes detrimental rather than scaling it according to client requirements. For this purpose, every model received from neighbors belonging to a different cluster is tested against the local training dataset. This process results in a substantial computational cost that grows with the network density, thus prohibitive for practical usage. In contrast, this chapter introduces a reinforcement learning-based mechanism that locally controls the inter-cluster learning parameter according to the client's real-time requirements to maximize the added value of inter-cluster learning. The inter-cluster learning parameter is modified on-the-fly as the computation takes place to cater to the time-evolving needs of the clients. Each client's parameter is controlled locally and independently from other clients' parameters to ensure that it is adapted to this client's neighborhood and data availability. Finally, the reinforcement learning policies used to control the inter-cluster learning parameters are computationally inexpensive not to slow down the learning process. This chapter will illustrate how this mechanism can greatly increase learning accuracy in a distributed network composed of clients with various tasks and data availability by providing them with personalized and time-adapting inter-cluster learning parameters.

6.2 Proposed Method

6.2.1 Networked Personalized Federated Learning

We consider a networked architecture modeled as an undirected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C}$ is the set of clients and $\mathcal{E}$ is the set of edges such that $(k, l) \in \mathcal{E}$ if and only if the clients k and l are neighbors. A client can only communicate with its neighbors, we denote $\mathcal{N}_k$ the set of the neighbors of client k. Further, clients are grouped into Q clusters, and the clients grouped in a cluster q, denoted by $\mathcal{C}_{(q)}$, for $q \in \{1, \dots, Q\}$, solve the same learning task, i.e., they aim to learn the same model. For $r \neq q, (r, q) \in \{1, \dots, Q\}$, the tasks associated with cluster q and r are different, but exhibit similarities.

Each client $k \in \mathcal{C}$ has access to a local dataset $(\mathbf{X}_k, \mathbf{y}_k)$ composed of a matrix $\mathbf{X}_k = [\mathbf{x}_{k,1}, \dots, \mathbf{x}_{k,D_k}]^\mathsf{T}$ and a response vector $\mathbf{y}_k = [y_{k,1}, \dots, y_{k,D_k}]^\mathsf{T}$, where D_k is the number of data samples available to client k. The objective is to estimate each cluster-specific model as accurately as possible without moving the data. This leads to the following optimization problem for a given cluster q:

$$\min_{\mathbf{w}_{(q)}} \sum_{k \in \mathcal{C}_{(q)}} \Big(\frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{(q)}) \Big) + \lambda R(\mathbf{w}_{(q)}) + \frac{\tau}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathrm{q}} \left\| \mathbf{w}_{(r)} - \mathbf{w}_{(q)} \right\|^2, \tag{6.1}$$

where ℓ_k denotes the loss function of the task performed by client k, R denotes the regularizer function, and $\lambda > 0$ is the regularization parameter. The term on the second line corresponds to the enforcement of similarity between the cluster-specific models, it is controlled by the global parameter τ. A larger τ enforces more similarity, leading to more similar cluster-specific models.

The above optimization problem is centralized and uses a global model $\mathbf{w}_{(q)}$ for each cluster. In a networked architecture, the learning process relies on the clients' models and enforces consensus among these models. To do so, the auxiliary variables $\mathbf{z}_k^l, \forall (k, l) \in \mathcal{E}$ are introduced. The distributed optimization problem for a given client k belonging to cluster q is then given by

$$\begin{aligned} \min_{\mathbf{w}_{k,(q)}} \quad & \frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{k,(q)}) + \lambda R(\mathbf{w}_{k,(q)}) \\ & + \frac{\tau_k}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathrm{q}} \left\| \hat{\mathbf{w}}_{k,(r)} - \mathbf{w}_{k,(q)} \right\|^2, \\ \text{s.t.} \quad & \mathbf{w}_{k,(q)} = \mathbf{z}_k^l, \mathbf{w}_{l,(q)} = \mathbf{z}_k^l; \forall l \in \mathcal{N}_k \bigcap \mathcal{C}_{(q)}, \end{aligned} \tag{6.2}$$

where $\mathbf{w}_{k,(q)}$ denotes the model of client k belonging to cluster q, and the constraints enforce intra-cluster consensus. The global parameter τ is replaced by client-specific parameters τ_k that control inter-cluster learning locally. The vector $\hat{\mathbf{w}}_{k,(r)}$ denotes the best available estimate of the model for cluster r available at client k. This corresponds to the average of the models of the neighboring clients from the cluster in question, given by

$$\hat{\mathbf{w}}_{k,(r)} = \frac{1}{|\mathcal{N}_k \bigcap \mathcal{C}_{(r)}|} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_{(r)}} \mathbf{w}_{l,(r)}. \tag{6.3}$$

It is possible to derive the augmented Lagrangian for a given cluster q with the set of primal variables $\mathcal{V}_q = \{\mathbf{w}_{k,(q)}\}$, Lagrange multipliers $\mathcal{M} = (\{\boldsymbol{\mu}_k^l\}, \{\boldsymbol{\gamma}_k^l\})$, and auxiliary variables $\mathcal{Z} = \{\mathbf{z}_k^l\}$ as

$$\begin{aligned}
&\mathcal{L}_{\rho,q}(\mathcal{V}_q, \mathcal{M}, \mathcal{Z}) = \\
&\sum_{k \in \mathcal{C}_{(q)}} \Bigg(\frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{k,(q)}) + \frac{\lambda}{|\mathcal{C}_{(q)}|} R(\mathbf{w}_{k,(q)}) \\
&+ \Big\| \frac{\tau_k}{Q-1} \sum_{r \in \{1,\ldots,Q\} \backslash \mathrm{q}} (\hat{\mathbf{w}}_{k,(r)} - \mathbf{w}_{k,(q)}) \Big\|^2 \Bigg) \\
&+ \sum_{k \in \mathcal{C}_{(q)}} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_{(q)}} \Big(\boldsymbol{\mu}_k^{l\mathsf{T}} (\mathbf{w}_{k,(q)} - \mathbf{z}_k^l) + \boldsymbol{\gamma}_k^{l\mathsf{T}} (\mathbf{w}_{l,(q)} - \mathbf{z}_k^l) \Big) \\
&+ \frac{\rho}{2} \sum_{k \in \mathcal{C}_{(q)}} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_{(q)}} \Big(\|\mathbf{w}_{k,(q)} - \mathbf{z}_k^l\|^2 + \|\mathbf{w}_{l,(q)} - \mathbf{z}_k^l\|^2 \Big),
\end{aligned} \tag{6.4}$$

where ρ is the penalty parameter. Given that the Lagrange multipliers are initialized to zero, by using the Karush-Kuhn-Tucker conditions of optimality and setting $\boldsymbol{\gamma}_k = 2 \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_{(q)}} \boldsymbol{\gamma}_k^l$, it can be shown that the Lagrange multipliers $\boldsymbol{\mu}_k^l$ and the auxiliary variables $\mathcal{Z}$ are eliminated [42]. From the Lagrangian, the local update steps of the ADMM for a given client k belonging to cluster q can be derived as:

- **Primal update**

$$\begin{aligned}\mathbf{w}_{k,(q)}^{(n)} = \arg\min_{\mathbf{w}} & \frac{1}{D_k}\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}) + \frac{\lambda}{|\mathcal{C}_{(q)}|}R(\mathbf{w}) \\ & + \left\| \frac{\tau_k}{Q-1} \sum_{r\in\{1,\ldots,Q\}\setminus \mathrm{q}} \left(\hat{\mathbf{w}}_{k,(r)}^{(n-1)} - \mathbf{w}\right) \right\|^2 \\ & + \mathbf{w}^{\mathsf{T}} \boldsymbol{\gamma}_{k,(q)}^{(n-1)} \\ & + \rho \sum_{l\in\mathcal{N}_k \bigcap \mathcal{C}_{(q)}} \left\| \mathbf{w} - \frac{\mathbf{w}_{k,(q)}^{(n-1)} + \mathbf{w}_{l,(q)}^{(n-1)}}{2} \right\|^2, \end{aligned} \tag{6.5}$$

- **Dual update**

$$\boldsymbol{\gamma}_{k,(q)}^{(n)} = \boldsymbol{\gamma}_{k,(q)}^{(n-1)} + \rho \sum_{l\in\mathcal{N}_k \bigcap \mathcal{C}_{(q)}} (\mathbf{w}_{l,(q)}^{(n)} - \mathbf{w}_{k,(q)}^{(n)}), \tag{6.6}$$

where the superscript (n) denotes the iteration number.

The choice of the inter-cluster learning parameters τ_k will greatly impact the performance of the proposed solution. A common and fixed value would fail to accommodate the heterogeneous local data and neighborhood quality within the network and would not remain relevant throughout the various learning stages of the clients. To address this, we adopt the principles of reinforcement learning to control time-varying and client-specific inter-cluster learning parameters $\tau_k^{(n)}$ in the next section.

6.2.2 Controlled Inter-Cluster Learning

Obtaining the optimal values for all the times series $\tau_k^{(n)}$ would require prior knowledge of the network topology and data distribution as well as extensive computational power, neither of which is available in a realistic scenario. Instead, we use computationally inexpensive reinforcement learning to leverage the information available within a client's neighborhood with the purpose of improving the value of the inter-cluster learning parameters. Each client k controls the real-time evolution of the parameter $\tau_k^{(n)}$ using local data and information received from neighbors.

At any given moment, the state of the reinforcement learning process, at a client k in cluster q, is given by the primal variable, $\mathbf{w}_{k,(q)}^{(n)}$. The action a corresponds to the modification of the local inter-cluster learning parameter $\tau_k^{(n)}$. We denote

$\mathbf{w}_{k,(q)}^{(n)}(\cdot)$, the function taking a value for the inter-cluster learning parameter as input and giving the corresponding alternative primal variable as output. The original primal variable in (6.5) corresponds to $\mathbf{w}_{k,(q)}^{(n)} = \mathbf{w}_{k,(q)}^{(n)}(\tau_k^{(n)})$. For an action a, the corresponding alternative primal variable is given by $\mathbf{w}_{k,(q)}^{(n)}(a)$.

The state- and action-value functions correspond to the error on the local test dataset of the initial and alternative primal variables, respectively. For instance, the state-value function is given by

$$V_t\Big(\mathbf{w}_{k,(q)}^{(n)}\Big) = \frac{1}{D_k}\ell_k\Big(\mathbf{X}_{k,t}, \mathbf{y}_{k,t}; \mathbf{w}_{k,(q)}^{(n)}\Big) + \frac{\lambda}{|\mathcal{C}_{(q)}|}R\Big(\mathbf{w}_{k,(q)}^{(n)}\Big), \tag{6.7}$$

where $(\mathbf{X}_{k,t}, \mathbf{y}_{k,t})$ denotes the test dataset. To avoid over-fitting, it is preferable not to use the test dataset in the reinforcement learning process. Instead, estimates of the state- and action-value functions are computed on a validation dataset $(\mathbf{X}_{k,v}, \mathbf{y}_{k,v})$. The estimate of the state-value function is given by $V_v(\mathbf{w}_{k,(q)}^{(n)})$, and the estimate of the action-value function by $V_v(\mathbf{w}_{k,(q)}^{(n)}(a))$.

Policy gradient [91, 92] and deterministic policy gradient [93] are among the most popular policies for continuous action reinforcement learning. They propose a gradient ascent alternative to the greedy maximization of the action-value function given by

$$\tau_k^{(n+1)} = \arg\max_a V_v\Big(\mathbf{w}_{k,(q)}^{(n)}(a)\Big). \tag{6.8}$$

In its simplest form, deterministic policy gradient relies on the gradient of the policy reward with respect to the policy parameter at the current state. In the proposed setting, this corresponds to the derivative of $V_v(\mathbf{w}_{k,(q)}^{(n)}(a))$ with respect to a taken at $\tau_k^{(n)}$, where the sign of the gradient is inverted since the reward corresponds to the error. The policy parameter update is given by

$$\tau_k^{(n+1)} - \tau_k^{(n)} \propto \frac{\partial V_v\Big(\mathbf{w}_{k,(q)}^{(n)}(a)\Big)}{\partial a}, \tag{6.9}$$

where $\propto$ denotes proportionality. Given the primal update (6.5), the computation of this derivative is impossible in the general case. However, it can be possible when the loss and regularizer functions are known.

The second proposed policy is a stochastic actor-critic mechanism that takes a random action and compares the action-value function with the state-value function to decide on the next value of the policy parameter. This policy offers better policy

parameter exploration than the deterministic policy gradient [94]. First, the policy proposes a random direction $\alpha \sim \mathcal{U}[\frac{-\nu_{\text{SAC}}}{2}, \frac{\nu_{\text{SAC}}}{2}]$ for the policy parameter $\tau_k^{(n)}$ so that $a = \tau_k^{(n)} + \alpha$. $\mathcal{U}(\cdot)$ denotes the uniform distribution, and ν_{SAC} is a hyper-parameter for the policy. The alternative primal variable $\mathbf{w}_{k,(q)}^{(n)}(a)$ is computed so that the action-value function can be compared with the state-value function. The policy parameter update is given by

$$\tau_k^{(n+1)} - \tau_k^{(n)} \propto -\text{sign}\left(V_v\left(\mathbf{w}_{k,(q)}^{(n)}(a)\right) - V_v\left(\mathbf{w}_{k,(q)}^{(n)}\right)\right)\alpha.$$

Both of the aforementioned policies can be modified to implement batch reinforcement learning [95]. In batch reinforcement learning, the policies take several successive actions per iteration step to better refine the policy parameter. However, this requires the computation of several gradients for the policy gradient and several action-value functions for the actor-critic policy, increasing the computational load associated with the reinforcement learning process.

6.3 Numerical Results

As a proof of concept, we use the proposed framework to solve the ridge regression problem. In ridge regression, the loss and regularizer functions used in (6.2) are given by

$$\ell(\mathbf{X}_k, \mathbf{y}_k, \mathbf{w}) = \|\mathbf{y}_k - \mathbf{X}_k\mathbf{w}\|^2, R(\mathbf{w}) = \|\mathbf{w}\|^2. \tag{6.10}$$

The primal variable update for ridge regression is obtained by substituting $\ell(\mathbf{X}_k, \mathbf{y}_k, \mathbf{w})$ and $R(\mathbf{w})$ with their above values in equation (6.5). Doing so, it is possible to compute the gradient of the term in the arg min with respect to the primal variable $\mathbf{w}$, and, setting it to zero, we obtain

$$\begin{aligned}\mathbf{w}_{k,(q)}^{(n)}(\tau_k^{(n)}) = &\left(\frac{\mathbf{X}_k^\mathsf{T}\mathbf{X}_k}{D_k} + \left(\frac{\lambda}{|\mathcal{C}_{(q)}|} + \tau_k^{(n)} + \rho|\mathcal{N}_k|\right)\mathbf{I}\right)^{-1} \\ &\left(\frac{\mathbf{X}_k^\mathsf{T}\mathbf{y}_k}{D_k}\frac{\tau_k^{(n)}}{(Q-1)}\sum_{r\in\{1,\ldots,Q\}\setminus \mathfrak{q}} \hat{\mathbf{w}}_{k,(r)}^{(n-1)}\right. \\ &\left. + \frac{\rho}{2}\sum_{l\in\mathcal{N}_k \bigcap \mathcal{C}_{(q)}}\left(\mathbf{w}_{k,(q)}^{(n-1)} + \mathbf{w}_{l,(q)}^{(n-1)}\right) - \frac{\boldsymbol{\gamma}_{k,(q)}^{(n-1)}}{2}\right).\end{aligned} \tag{6.11}$$

The alternative primal variable $\mathbf{w}_{k,(q)}^{(n)}(a)$ for an action a can be computed in the same manner, by replacing $\tau_k^{(n)}$ with a in (6.11). Using this alternative primal

variable and the values of the loss and regularizer function for ridge regression in (6.10), the action-value function can be expressed as

$$V_v\left(\mathbf{w}_{k,(q)}^{(n)}(a)\right) = \frac{1}{D_k}\left\|\mathbf{y}_{k,v} - \mathbf{X}_{k,v}\mathbf{w}_{k,(q)}^{(n)}(a)\right\|^2 + \frac{\lambda}{|\mathcal{C}_{(q)}|}\left\|\mathbf{w}_{k,(q)}^{(n)}(a)\right\|^2. \quad (6.12)$$

Using the expression for the primal variable (6.11) and the action-value function specific to ridge regression in (6.12), it is possible to compute the derivative of the policy reward with respect to the policy parameter $\partial V_v(\mathbf{w}_{k,(q)}^{(n)}(\tau_k^{(n)}))/\partial\tau_k^{(n)}$. Given that the action-value function is to be minimized, the policy parameter update step for the *deterministic policy gradient*, which we refer to as (DPG), is given by:

$$\tau_k^{(n+1)} = \tau_k^{(n)} - \delta_{\text{DPG}}\frac{\partial V_v\left(\mathbf{w}_{k,(q)}^{(n)}(a)\right)}{\partial a}, \quad (6.13)$$

where δ_{DPG} is the learning rate.

For a random direction α and corresponding action $a = \tau_k^{(n)} + \alpha$, using (6.12) for the action- and state-value functions, the update step of the policy parameter for the *stochastic actor-critic policy*, which we refer to as (SAC), is given by

$$\tau_k^{(n+1)} = \tau_k^{(n)} - \delta_{\text{SAC}}\text{sign}\left(V_v\left(\mathbf{w}_{k,(q)}^{(n)}(a)\right) - V_v\left(\mathbf{w}_{k,(q)}^{(n)}\right)\right)\alpha, \quad (6.14)$$

where the sign is negated to minimize the action-value function and δ_{SAC} is the learning rate.

The resulting algorithms, referred to as Networked Personalized Federated learning using Reinforcement Learning (NPFL-RL) are summarized in Algorithm 6.

We considered a distributed network composed of $K = 30$ clients with an average of 6 neighbors per client. The clients are randomly grouped into $Q = 3$ clusters. The goal is to estimate cluster-specific tasks given by $\mathbf{w}_q = \mathbf{w}_0 + \delta_q\mathbf{w}_0$, with $\delta_q \sim \mathcal{U}(-0.5, 0.5)$. $\mathcal{U}$ denotes the uniform distribution, and $\mathbf{w}_0$ is a randomly chosen base model. Each client k possesses a training dataset $(\mathbf{X}_k, \mathbf{y}_k)$ where $\mathbf{X}_k \in \mathbb{R}^{D_k\times 60}$ and $\mathbf{y}_k \in \mathbb{R}^{D_k\times 1}$ with $D_k \sim \mathcal{U}(5, 35)$, as well as identically distributed testing and validation datasets. The data is generated as $\mathbf{y}_k = \mathbf{X}_k\mathbf{w}_q + \mathbf{n}_k$, with $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \eta_k)$, where η_k is the client-specific noise variance and $\mathbf{w}_q$ is its cluster model. Finally, the Lagrangian penalty parameter is set to $\rho = 3$. We

Algorithm 6 NPFL-RL for ridge regression

Initialization: $\mathbf{w}_{k,(q)}^{(0)}$ and $\boldsymbol{\gamma}_{k,(q)}^{(0)},\ k \in \mathcal{C}$ are set to $\mathbf{0}$, the parameters $\tau_k^{(0)}$ are set to a given value within $(0, 1)$.

– Procedure at client k –

For $n = 1, 2, \ldots, N$

if $n > 1$

(DPG) τ_k is updated as in (6.13).

(SAC) τ_k is updated as in (6.14).

end if

Primal update: $\mathbf{w}_{k,(q)}^{(n)}$ takes the value in (6.11).

Client k shares $\mathbf{w}_{k,(q)}^{(n)}$ with its neighbors in $\mathcal{N}_k$.

Dual update:

$\boldsymbol{\gamma}_{k,(q)}^{(n)} = \boldsymbol{\gamma}_{k,(q)}^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_{(q)}} (\mathbf{w}_{l,(q)}^{(n)} - \mathbf{w}_{k,(q)}^{(n)})$.

End For

considered the normalized mean squared error on the testing data set (Test MSE) as the performance metric for comparison of the algorithms. It is given by

$$\text{Test MSE} = \frac{1}{K} \sum_{q \in \mathcal{Q}} \sum_{k \in \mathcal{C}_q} \frac{\left\| \mathbf{w}_{k,(q)} - \mathbf{w}_{(q)} \right\|_2^2}{\left\| \mathbf{w}_{(q)} \right\|}, \tag{6.15}$$

where $\{\mathbf{w}_{k,(q)}, k \in \mathcal{C}_{(q)}, q \in \mathcal{Q}\}$ are the models of the considered method. The simulation results presented in the following are obtained by averaging the results of 10 independent experiments. To ensure a fair comparison, the algorithms are tuned to have the same initial convergence rate in Fig. (6.2).

The first experiment studied the impact of the inter-cluster learning parameter τ on the learning behavior of conventional networked FL algorithms. For this purpose, we simulated the following algorithms:

- **NFL:** is the traditional networked FL that learns one universal model for the whole network.
- **NPFL:** is conventional personalized networked FL that learns cluster-specific models. The global parameter τ is fixed throughout the learning process, so that $\forall k \in \mathcal{C}, n > 0; \tau_k = \tau$. Inter-cluster learning is absent when $\tau = 0$.

The learning curves (i.e., Test MSE in dB vs. iteration index n) of the above algorithms are presented in Fig. 6.1. The figure shows that the NFL algorithm does

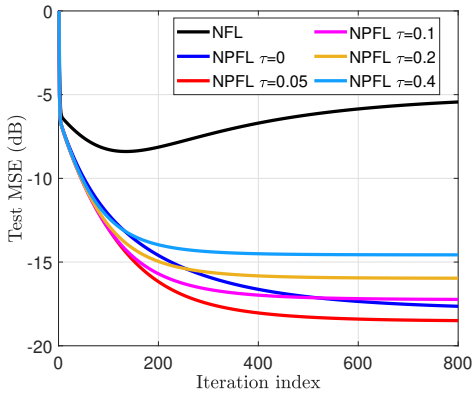


Figure 6.1: Learning curves of the NFL and NPFL algorithms for various values of τ.

not achieve satisfactory accuracy since it tries to learn a single universal model that cannot accommodate client-specific tasks. NPFL with $\tau = 0$ implies that each cluster independently builds its own model by relying solely on the cooperation among cluster members. Its performance can, therefore, be regarded as a benchmark for networked personalized federated learning. As the value of τ increases (e.g., $\tau = 0.05$), the NPFL performance improves, as it enforces similarity between the cluster-specific models. However, as more similarity is enforced between models for non-identical tasks, the steady-state accuracy decreases, as can be seen with NPFL $\tau = 0.1$, $\tau = 0.2$, and $\tau = 0.4$. This confirms that inter-cluster learning can be beneficial but leads to performance degradation when over-used.

In the second experiment, we demonstrated the effectiveness of the proposed NPFL-RL in the learning of personalized models. For this purpose, we simulated the following algorithms:

- **NPFL-RL (DPG):** uses the deterministic policy gradient in (6.13) with $\delta_{\text{DPG}} = 0.001$.
- **NPFL-BRL (DPG):** uses the deterministic policy gradient and batch reinforcement learning with 3 epochs and $\delta_{\text{DPG}} = 0.003$.
- **NPFL-RL (SAC):** uses the stochastic actor-critic policy in (6.14) with $\nu_{\text{SAC}} = 0.05$ and $\delta_{\text{SAC}} = 0.1$.
- **NPFL-BRL (SAC):** uses the stochastic actor-critic policy and batch reinforcement learning with 3 epochs, $\nu_{\text{SAC}} = 0.05$ and $\delta_{\text{SAC}} = 0.04$.

The learning curves of these algorithms are presented in Fig. 6.2. For compar-

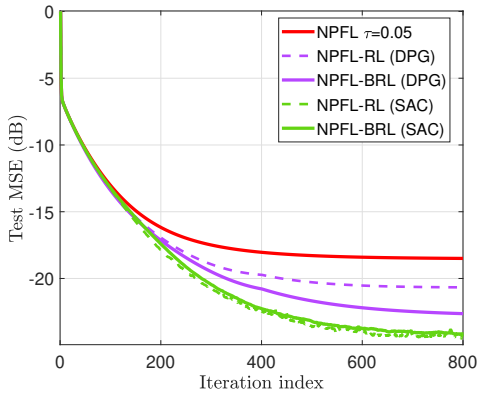


Figure 6.2: Learning curves of the NPFL-RL and NPFL-BRL with different policies. Also plotted is the learning curve of NPFL for $\tau = 0.05$.

ison purposes, the learning curve of NPFL with $\tau = 0.05$ is also displayed. We see that all the versions of NPFL-RL exhibit better performance in initial learning speed and steady-state accuracy compared to NPFL operating with a fixed τ value. Since the clients have device-specific requirements, usage of a fixed universal τ is impractical. Whereas the proposed NFL-RL controls the amount of inter-cluster learning locally by learning the device-specific parameters $\tau_k^{(n)}$ in real time. Further, we also see that NPFL-RL (SAC) exhibites enhanced accuracy over NPFL-RL (DPG). The reason for this is that the stochastic nature of NPFL-RL (SAC) ensures sufficient exploration of the policy parameters τ_k, which is not the case for NPFL-RL (DPG). This stochastic nature also leads to extensive randomness in the convergence of NPFL-RL (SAC), batch reinforcement learning attenuates this issue as can be seen with NPFL-BRL (SAC). In the case of deterministic policy gradient, batch reinforcement learning increases the learning accuracy. It is important to note that batch reinforcement learning comes with a computational cost proportional to the number of epochs performed.

Finally, we illustrate the evolution of the inter/cluster learning parameters $\tau_k^{(n)}$ for NPFL-RL (SAC) and NPFL-RL (DPG) in Fig. 6.3. We selected three clients 22, 3, 30, having access to large, moderate, and small amounts of data, respectively. Therefore, these clients have different local requirements for inter-cluster learning. From Fig. 6.3, we observe that the evolution of $\tau_k^{(n)}$ is very smooth when using NPFL-RL (DPG) but fails to quickly adapt. On the other hand, the NPFL-RL (SAC) algorithm provides sufficient exploration of the policy parameter, ensuring that the $\tau_k^{(n)}$ parameters evolve quickly at the cost of extensive randomness (BRL helps to overcome this issue). Furthermore, we also see that $\tau_k^{(n)}$ evolves according

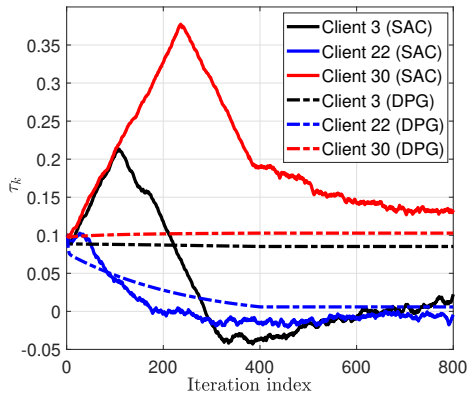


Figure 6.3: Evolution of $\tau_k^{(n)}$ for the 3rd, 22nd, and 30th clients.

to the needs of clients. Since client 30 has access to a small amount of data, $\tau_{30}^{(n)}$ increases linearly at first to enforce higher inter-cluster learning. After reaching near-convergence, the $\tau_{30}^{(n)}$ value decreases to reduce the amount of inter-cluster learning to avoid its harmful effect. In contrast, since client 22 has access to a large amount of data, the $\tau_{22}^{(n)}$ parameter decreases almost immediately and stabilizes around 0 as inter-cluster learning is not valuable for this client. Finally, client 3 has access to an average amount of data. $\tau_3^{(n)}$ increases at first as similarity enforcement allows for faster initial convergence, but decreases afterwards and stabilizes around 0 to avoid performance degradation.

6.4 Summary

Personalized federated learning suffers from data scarcity within clusters; this can be alleviated by leveraging the similarity between the learning tasks. How much these similarities are utilized is controlled by the inter-cluster learning parameter. Optimizing the temporal sequence of inter-cluster learning parameters is hard, especially in a networked setting where the network topology and data distribution impact how much each specific client needs to rely on inter-cluster learning. This chapter introduced a networked personalized FL algorithm that uses reinforcement learning to control client-specific evolving inter-cluster learning parameters. In this algorithm, each local parameter is controlled in real-time by the reinforcement learning process in a computationally inexpensive manner to accommodate the clients' varied and evolving needs, ensuring that model similarity is enforced only as much as what is beneficial for local learning. Numerical simulations showed that the proposed method successfully controls device-specific parameters and offers better learning performance than existing solutions.

Chapter 7

Conclusions and Future Work

This thesis developed solutions for various application scenarios with the purpose of enabling the artificial intelligence of distributed devices. Through those applications, various challenges faced when implementing distributed machine learning have been tackled. In particular, we have studied how to utilize an architecture that scales with the growing number of participating devices, how to preserve the participants' confidentiality, how to handle straggler devices, and how to improve performance by carefully leveraging both global and local data.

Chapter 2 provided an overview of federated learning, its alternative architectures that can scale with a large number of participants, and its particular implementations for real-time and personalized learning. In addition, it introduced the notion of differential privacy and illustrated how it can be used in an iterative process such as federated learning to protect the client data.

Chapter 3 presented a networked federated learning algorithm where clients collaborate in a peer-to-peer manner to reach a consensus on the best solution to a learning problem. As not all devices may be trusted, this algorithm protects client confidentiality and privacy by incorporating differential privacy, ensuring that private data is not compromised during the learning process. Finally, this algorithm can be used on smooth and nonsmooth objectives, increasing the range of learning problems it can tackle.

In Chapter 4, we focused on resource-constrained devices in federated learning. For this purpose, nonlinear learning was considered as it often necessitates the exchange of large-dimensional models, increasing the network strain. In addition, we considered an online FL setting, where timely learning is paramount, and developed an algorithm capable of handling unpredictable participation and mitig-

ating the adverse effect of delayed updates on accuracy using a weight-decreasing mechanism. Additionally, the proposed algorithm reduces the communication load by implementing partial-sharing-based communications, which presents the additional advantage of reducing the negative impact of delayed updates on accuracy.

Chapter 5 developed the personalized graph federated learning framework. The graph federated architecture scales easily with the growing number of participants while maintaining a fast learning rate, making it appealing in most applications. The proposed method uses intra- and inter-cluster learning within and across servers to ensure fast and accurate learning of personalized models. Finally, this framework is implemented in a privacy-preserving manner to provide a secure approach to scalable and personalized learning.

In Chapter 6, we further explored inter-cluster learning in personalized FL. To this aim, we considered a networked architecture in which the network topology and data distribution affect the clients' local need for inter-cluster learning. In such an architecture, global control of inter-cluster learning is inadequate, while local optimization is prohibitively expensive. To address this challenge, we used computationally inexpensive reinforcement learning to control inter-cluster learning locally and in real-time.

While the second half of this thesis focused on enabling and optimizing the use of both inter- and intra-cluster learning, it assumed that the client distribution among clusters was known. Detecting the initial and/or evolving distribution of clients among clusters in personalized learning would enable its use in many applications where this information cannot be known beforehand. In particular, the proposed reinforcement learning mechanism used in Chapter 6 could be adapted to behave without *a priori* cluster information on the neighbors and classify those according to the value of the inter-cluster learning parameter.

Bibliography

[1] P. Sethi and S. R. Sarangi, "Internet of things: architectures, protocols, and applications," *J. elec. comput. eng.*, Jan. 2017.

[2] B. Ali and A. I. Awad, "Cyber and physical security vulnerability assessment for IoT-based smart homes," *sensors*, vol. 18, p. 817, Mar. 2018.

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

[4] Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, H. Ludwig, and Y. Cheng, "Towards taming the resource and data heterogeneity in federated learning.," in *OpML*, pp. 19–21, 2019.

[5] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[6] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala, "Asynchronous federated learning for sensor data with concept drift," *arXiv preprint arXiv:2109.00151*, Sep. 2021.

[7] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, Mar. 2019.

[8] Y. Sarcheshmehpour, M. Leinonen, and A. Jung, "Federated learning from big data over networks," *in Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, pp. 3055–3059, Jan. 2021.

[9] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," *in Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun.*, pp. 131–135, Sep. 2021.

[10] X. Qiu, T. Parcollet, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "Can federated learning save the planet?," *arXiv preprint arXiv:2010.06537*, Oct. 2020.

[11] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, pp. 1935–1949, Nov. 2020.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial intel. statis.*, pp. 1273–1282, Apr. 2017.

[13] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, Mar. 2022.

[14] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, pp. 4129–4144, Jun. 2014.

[15] V. C. Gogineni and M. Chakraborty, "Diffusion affine projection algorithm for multitask networks," *in Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, pp. 201–206, Nov. 2018.

[16] A. Kuh, S. Huang, and C. Chen, "Personalized learning using multiple kernel models," *in Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, pp. 2085–2088, Dec. 2021.

[17] V. C. Gogineni and M. Chakraborty, "Improving the performance of multitask diffusion APA via controlled inter-cluster cooperation," *IEEE Trans. Circuits Sys. I: Reg. Papers*, vol. 67, pp. 903–912, Dec. 2019.

[18] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, pp. 40–47, Apr. 2021.

[19] S. Boll and J. Meyer, "Health-X dataLOFT: A Sovereign Federated Cloud for Personalized Health Care Services," *IEEE MultiMedia*, vol. 29, pp. 136–140, May 2022.

[20] O. Dekel, P. M. Long, and Y. Singer, "Online multitask learning," in *Int. Conf. Comput. Learn. Theory*, pp. 453–467, 2006.

[21] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," pp. 265–284, 2006.

[22] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," *IEEE Int. Workshop Signal Process. Adv. Wireless Commun.*, pp. 131–135, Sep. 2021.

[23] S. Wang, S. Hosseinalipour, M. Gorlatova, C. G. Brinton, and M. Chiang, "UAV-assisted online machine learning over multi-tiered networks: A hierarchical nested personalized federated learning approach," *IEEE Trans. Netw. Service Manage.*, Oct. 2022.

[24] F.-Z. Lian, J.-D. Huang, J.-X. Liu, G. Chen, J.-H. Zhao, and W.-X. Kang, "FedFV: A personalized federated learning framework for finger vein authentication," *Machine Intel. Research*, pp. 1–14, Jan. 2023.

[25] V. C. Gogineni, S. Werner, F. Gauthier, Y.-F. Huang, and A. Kuh, "Personalized online federated learning for IoT/CPS: challenges and future directions," *IEEE Internet Things Mag.*, 2022.

[26] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: a meta-learning approach," *arXiv preprint arXiv:2002.07948*, Feb. 2020.

[27] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *arXiv preprint arXiv:2003.13461*, Mar. 2020.

[28] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, pp. 4129–4144, Jun. 2014.

[29] S. Ben-David and R. Schuller, "Exploiting task relatedness for multiple task learning," *Conf. Learn. Theory Kernel Workshop*, pp. 567–580, Aug. 2003.

[30] R. H. Weber, "Internet of Things–New security and privacy challenges," *Computer Law & Security Review*, vol. 26, pp. 23–30, Jan. 2010.

[31] A. Narayanan and V. Shmatikov, "How to break anonymity of the netflix prize dataset," *arXiv preprint cs/0610105*, 2006.

[32] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, pp. 1322–1333, Oct. 2015.

[33] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, pp. 211–407, Aug. 2014.

[34] C. Dwork, N. Kohli, and D. Mulligan, "Differential Privacy in Practice: Expose your Epsilons!," *J. Privacy and Confidentiality*, vol. 9, Oct. 2019.

[35] A. D. Sarwate and K. Chaudhuri, "Signal processing and machine learning with differential privacy: algorithms and challenges for continuous data," *IEEE Signal Proc. Mag.*, vol. 30, pp. 86–94, Sep. 2013.

[36] T. Zhang and Q. Zhu, "Dynamic differential privacy for ADMM-based distributed classification learning," *IEEE Trans. Inf. Forens. Security*, vol. 12, pp. 172–187, Jan. 2017.

[37] C. Dwork and G. N. Rothblum, "Concentrated differential privacy," *arXiv preprint arXiv:1603.01887*, 2016.

[38] M. Bun and T. Steinke, "Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds," in *Theory of Cryptography Conf.*, pp. 635–658, Springer, 2016.

[39] J. Ding, Y. Gong, M. Pan, and Z. Han, "Optimal differentially private ADMM for distributed machine learning," *IEEE Int. Conf. Big Data*, pp. 1302–1311, Dec. 2019.

[40] C. Chen and J. Lee, "Rényi Differentially Private ADMM for Non-Smooth Regularized Optimization," *Proc. Tenth ACM Conf. Data and Appl. Secur. and Privacy*, pp. 319–328, 2020.

[41] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "DP-ADMM: ADMM-based distributed learning with differential privacy," *IEEE Trans. Inf. Forens. Security*, vol. 15, pp. 1002–1012, Jul. 2019.

[42] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, "Splitting Methods in Communication, Imaging, Science, and Engineering," in *Scientific Computation*, pp. 461–497, Springer Int. Publishing, 2016.

[43] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed Ridge Regression with Feature Partitioning," *Proc. Asilomar Conf. Signals Syst. Comput.*, pp. 1423–1427, 2018.

[44] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. optim.*, vol. 19, no. 4, pp. 1574–1609, 2009.

[45] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *in Proc. 2015 Int. Conf. Distrib. Comput. Netw.*, pp. 1–10, Jan. 2015.

[46] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, 2014.

[47] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, "Robust decentralized learning using ADMM with unreliable agents," *IEEE Trans. Signal Process.*, Jun. 2022.

[48] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, pp. 48–61, Jan. 2009.

[49] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, pp. 1–122, Jan. 2010.

[50] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1." `http://cvxr.com/cvx`, Mar. 2014.

[51] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, pp. 1935–1949, Mar. 2021.

[52] Z. Zhao, C. Feng, W. Hong, J. Jiang, C. Jia, T. Q. S. Quek, and M. Peng, "Federated learning with non-IID data in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, pp. 1927–1942, Mar. 2022.

[53] E. Ozfatura, K. Ozfatura, and D. Gündüz, "FedADC: accelerated federated learning with drift control," in *Proc. IEEE Int. Symp. Inf. Theory*, pp. 467–472, Jul. 2021.

[54] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, pp. 50–60, May 2020.

[55] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, Jul. 2019.

[56] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[57] Z. Lian, W. Wang, and C. Su, "COFEL: Communication-efficient and optimized federated learning with local differential privacy," in *Proc. IEEE Int. Conf. Commun.*, pp. 1–6, Jun. 2021.

[58] Y. Lu, Z. Liu, and Y. Huang, "Parameters compressed mechanism in federated learning for edge computing," in *Proc. IEEE Int. Conf. Cyber Secur. Cloud Comput.*, pp. 161–166, Jun. 2021.

[59] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "Communication-efficient federated learning through 1-bit compressive sensing and analog aggregation," in *Proc. IEEE Int. Conf. Commun. Workshops*, pp. 1–6, 2021.

[60] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *Proc. IEEE Int. Conf. Big Data*, pp. 15–24, Dec. 2020.

[61] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. Quek, "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, Mar. 2022.

[62] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: a communication-efficient federated learning method with asynchronous tiers under non-iid data," *arXiv preprint arXiv:2010.05958*, Oct. 2020.

[63] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, pp. 4229–4238, Dec. 2019.

[64] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, pp. 3400–3413, Sep. 2020.

[65] 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, pp. 3510–3522, Jul. 2014.

[66] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning framework for nonlinear regression," *IEEE Int. Conf. Acoust., Speech and Signal Process.*, May 2022.

[67] O. Dekel, P. M. Long, and Y. Singer, "Online multitask learning," in *Int. Conf. Comput. Learn. Theory*, pp. 453–467, 2006.

[68] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, pp. 543–554, Jan. 2008.

[69] V. C. Gogineni, V. R. Elias, W. A. Martins, and S. Werner, "Graph diffusion kernel LMS using random Fourier features," *in Proc. 54th Asilomar Conf. Signals, Syst., Computers*, pp. 1528–1532, Nov. 2020.

[70] A. Rahimi, B. Recht, *et al.*, "Random features for large-scale kernel machines.," in *Proc. Conf. on Neural Inf. Proc. Syst.*, vol. 3, pp. 1–5, Dec. 2007.

[71] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: a random Fourier feature perspective," in *Proc. IEEE Stat. Signal Process. Workshop*, pp. 1–5, Jun. 2016.

[72] 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, Apr. 1991.

[73] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, pp. 1–1, Nov. 2022.

[74] S. Dane, "CalCOFI, Over 60 years of oceanographic data." Available at: https://www.kaggle.com/sohier/calcofi?select=bottle.csv.

[75] R. Jin, Y. Huang, X. He, H. Dai, and T. Wu, "Stochastic-sign SGD for federated learning with theoretical guarantees," *arXiv preprint arXiv:2002.10940*, Feb. 2020.

[76] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Ind. Eng.*, vol. 149, p. 106854, 2020.

[77] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Sys.*, Mar. 2022.

[78] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, pp. 3710–3722, Aug. 2020.

[79] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Adv. neural inf. process. syst.*, vol. 30, . 2017.

[80] R. Li, F. Ma, W. Jiang, and J. Gao, "Online federated multitask learning," *in Proc. IEEE Int. Conf. Big Data*, pp. 215–220, Dec. 2019.

[81] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Adv. Neural Info. Pro. Syst.*, vol. 33, pp. 19586–19597, 2020.

[82] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," *in Proc.IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 2749–2758, 2021.

[83] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Decentralized graph federated multitask learning for streaming data," *in Proc. Annu. Conf. Inf. Sciences Sys.*, pp. 101–106, Mar. 2022.

[84] S. Zhou and G. Y. Li, "Federated learning via inexact ADMM," *IEEE Trans. Pattern Anal. Machine Intell.*, Feb. 2023.

[85] Y. Chen, R. S. Blum, and B. M. Sadler, "Communication efficient federated learning via ordered ADMM in a fully decentralized setting," *Conf. Inf. Sciences Syst.*, pp. 96–100, Mar. 2022.

[86] S. Yue, J. Ren, J. Xin, S. Lin, and J. Zhang, "Inexact-ADMM based federated meta-learning for fast and continual edge learning," in *Proc. Int. Symp. Theory Algorithmic Found. Protocol Des. Mobile Netw. Mobile Comput.*, p. 91–100, Assoc. Comput. Mach., Jul. 2021.

[87] J. Ding, X. Zhang, M. Chen, K. Xue, C. Zhang, and M. Pan, "Differentially Private Robust ADMM for Distributed Machine Learning," in *2019 IEEE Int. Conf. Big Data*, pp. 1302–1311, Dec. 2019.

[88] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "ClusterFL: A similarity-aware federated learning system for human activity recognition," *Proc. 19th Annu. Int. Conf. Mobile Syst. Applications Services*, pp. 54–66, Jun. 2021.

[89] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.

[90] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "MedMNIST v2-A large-scale lightweight benchmark for 2D and 3D biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

[91] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Adv. Neural Inf. Process. Syst.*, vol. 12, 1999.

[92] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, Jul. 2017.

[93] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *in Proc. Int. Conf. Mach. Learn.*, pp. 387–395, Jan. 2014.

[94] H. Wang, T. Zariphopoulou, and X. Y. Zhou, "Reinforcement learning in continuous time and space: a stochastic control approach," *J. Mach. Learn. Res.*, vol. 21, pp. 1–34, Jan. 2020.

[95] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," *Reinforcement Learn.*, pp. 45–73, Springer, 2012.

Appendix A

Publication 1

P1 F. Gauthier, C. Gratton, N. K. D. Venkategowda and S. Werner, "Privacy-Preserving Distributed Learning with Nonsmooth Objective Functions", in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, November, 2020.

Appendix B

Publication 2

P2 F. Gauthier, C. Gratton, N. K. D. Venkategowda and S. Werner, "Private Networked Federated Learning for Nonsmooth Objectives", submitted to *Elsevier Signal Processing*.

The following is the submitted manuscript.

Private Networked Federated Learning for Nonsmooth Objectives

François Gauthier[a], Cristiano Gratton[a], Naveen K. D. Venkategowda[b], Stefan Werner[a]

[a]*Department of Electronic Systems, NTNU, Norway*
[b]*Department of Science and Technology, Linköping University, Sweden*

Abstract

This paper develops a networked federated learning algorithm to solve nonsmooth objective functions. To guarantee the confidentiality of the participants with respect to each other and potential eavesdroppers, we use the zero-concentrated differential privacy notion (zCDP). Privacy is achieved by perturbing the outcome of the computation at each client with a variance-decreasing Gaussian noise. ZCDP allows for better accuracy than the conventional (ϵ, δ)-DP and stronger guarantees than the more recent Rényi-DP by assuming adversaries aggregate all the exchanged messages. The proposed algorithm relies on the distributed Alternating Direction Method of Multipliers (ADMM) and uses the approximation of the augmented Lagrangian to handle nonsmooth objective functions. The developed private networked federated learning algorithm has a competitive privacy accuracy trade-off and handles nonsmooth and non-strongly convex problems. We provide complete theoretical proof for the privacy guarantees and the algorithm's convergence to the exact solution. We also prove under additional assumptions that the algorithm converges in $O(1/n)$ ADMM iterations. Finally, we observe the performance of the algorithm in a series of numerical simulations.

Keywords: Federated Learning, Networked Architecture, Differential Privacy, Nonsmooth Objective Functions.

1. Introduction

Federated learning (FL) [1] has garnered significant research attention recently because of its capacity to process massive amounts of data over a network of clients [2, 3]. It has many applications, such as smart healthcare [4], autonomous vehicles [5] and drones [6], and industrial engineering [7]. Networked FL [8], also called decentralized FL, proposes a collaborative approach to FL in which the problem is decomposed into many sub-problems that network clients solve by interacting with their immediate neighbors in a peer-to-peer fashion without involving a central coordinator. [9–11]. Networked FL is receiving growing interest as it resolves the limitations of using a single server in FL, such as communication and computation bottlenecks, while maintaining its advantages [12].

In many applications, the data held by clients is sensitive, and adversaries may try to extract private information from the information exchanged between the clients in the network. Therefore, it is imperative to mitigate information leakage during the client-interaction process in FL [13]. In this context, differential privacy (DP) [14] provides a mechanism that protects individual privacy by ensuring minimal changes in the algorithm output, regardless of whether an individual data sample is present during the computation [15, 16]. Local DP protects the data of the clients with respect to each other and external eavesdroppers. This presents the advantage of protecting clients from honest-but-curious clients who form part of the network but share the information available to them with a third party. However, achieving good accuracy while providing high privacy guarantees in privacy-preserving FL is challenging, especially when several messages are exchanged.

To meet the demand for better privacy accuracy trade-off, concentrated differential privacy (CDP) was introduced in [17] as a relaxation of the conventional (ϵ, δ)-DP. Its purpose is to enable higher accuracy while maintaining identical pri-

vacy protection under the assumption that an adversary may aggregate all the exchanged messages [18, 19]. CDP was relaxed into zero-concentrated differential privacy (zCDP) in [20], which is easier to use and offers similar benefits. Finally, dynamic-DP, introduced in [21] can be used with zCDP to better suit iterative processes such as an ADMM algorithm. It enables iteration-specific privacy budgets. More recently, CDP has been relaxed into Rényi-DP [22], which concentrates on a single moment of a privacy loss variable. In contrast, CDP and zCDP provide a linear bound on all positive moments and, therefore, a stronger privacy guarantee. In this work, we use dynamic zCDP.

Existing networked FL and distributed learning solutions mainly comprise (sub)gradient-based and ADMM-based algorithms. The former typically converge at a rate of $O(1/\sqrt{n})$ [23], and the latter usually converge at a rate of $O(1/n)$ [24]. Although networked FL is recent, privacy-preserving distributed learning has been thoroughly studied [21, 24–29]. Among these works, we can classify two relevant groups to the problem at hand. The first group comprises solutions that use a networked architecture and assume the objective functions to be smooth and convex [21, 24–28], which may not be a valid assumption in practice. In fact, many compelling objectives cannot be accurately modeled in this way [30, 31]. Among the above works, [24] offers a convergence rate of $O(1/n)$ and, in [29], the regularizer function can be nonsmooth, but the loss function is assumed smooth and differentiable. In addition, the convergence rate of the algorithm in [29] is $O(1/\sqrt{n})$. The second group comprises solutions relying on a single server and handling nonsmooth and non-strongly convex objectives [32–36]. We note that the works in [32–34] do not consider privacy, and the work in [36] can only accommodate nonsmooth regularizer functions. The solution proposed in [35] handles nonsmooth and non-strongly convex objectives but converges in $O(1/\sqrt{n})$ and relies on a server to aggregate the local models.

This paper proposes a privacy-preserving networked FL algorithm that handles nonsmooth and non-strongly convex objective functions. Furthermore, the proposed algorithm is proven to converge to the exact solution with a rate of $O(1/n)$. We consider a distributed network of clients that solve an optimization problem collaboratively. Each client iteratively updates its local model using its local data and the models received from its neighbors. To ensure the confidentiality of its local data, a client perturbs its model before communication with white Gaussian noise. The variance of the noise added to the models is such that the total privacy leakage of the clients throughout the computation is bounded under the zCDP metric. The ADMM used to solve the optimization problem is distributed to comply with the networked setting. Further, the clients' primal updates use an approximation of the augmented Lagrangian obtained by taking the first-order approximation of the objective function. This enables the proposed method to handle nonsmooth objective functions. Mathematical analysis shows that the proposed algorithm converges to the optimal point in $O(1/n)$. Numerical simulations compare the proposed algorithm with the most closely related distributed learning solutions.

The rest of the manuscript is organized as follows. Section II introduces the proposed zero-Concentrated Differentially Private Networked Federated Learning (zCDP-NFL) algorithm. Sections III and IV contain the mathematical analysis of the privacy guarantees and convergence properties of the proposed algorithm, respectively. Section V compares zCDP-NFL with existing methods in a series of numerical simulations.

Mathematical notations: Matrices, column vectors, and scalars will be respectively denoted by bold uppercase, bold lowercase, and lowercase letters. The set of natural integers is denoted by $\mathbb{N}$ and the set of real numbers by $\mathbb{R}$. The operators $(\cdot)^\intercal$ denotes the transpose of a matrix. $||\cdot||$ represents the Euclidean norm,

$||\cdot||_1$ the L_1 norm, and $||\boldsymbol{x}||^2_{\boldsymbol{G}} = \langle \boldsymbol{x}, \boldsymbol{G}\boldsymbol{x}\rangle$ for any couple of vector $\boldsymbol{x}$ and matrix $\boldsymbol{G}$. The inner product between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is denoted by $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$. The statistical expectation operator is represented by $\mathbb{E}[\cdot]$ and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. If a random variable A follows the law $\mathcal{B}$, we will write $A \sim \mathcal{B}$. The identity matrix in $\mathbb{R}^{n\times n}$ is denoted by $\boldsymbol{I}_n$ and subgradient of a function $g(\cdot)$ is denoted by $g'(\cdot)$. The nonzero smallest and largest singular values of a semidefinite matrix $\boldsymbol{A}$ are denoted by $\Phi_{\min}(\boldsymbol{A})$ and $\Phi_{\max}(\boldsymbol{A})$.

2. Approximated Private Networked Federated Learning

2.1. Distributed Empirical Risk Minimization

We consider a connected network of clients modeled as an undirected graph $\mathcal{G}(\mathcal{C}, \mathcal{E})$ where vertex set $\mathcal{C} = \{1, \ldots, K\}$ corresponds to the clients and edge set $\mathcal{E}$ contains the $|\mathcal{E}| = E$ undirected communication links. The set $\mathcal{N}_k$ contains the indexes of the neighbors of client k.

Each client $k \in \mathcal{C}$ has a private data set $\mathcal{D}_k \coloneqq \{(\mathbf{X}_k, \mathbf{y}_k) : \mathbf{X}_k = [\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,M_k}]^\mathsf{T} \in \mathbb{R}^{M_k\times P},\ \mathbf{y}_k = [y_{k,1}, \ldots, y_{k,M_k}]^\mathsf{T} \in \mathbb{R}^{M_k}\}$, where M_k is the number of data samples and P the number of features in the data.

To fit with the networked architecture, we consider the distributed empirical risk minimization problem with local primal variables $\mathcal{V} \coloneqq \{\mathbf{w}_k\}_{k=1}^K$:

$$\begin{aligned} \min_{\{\mathbf{w}_k\}} \quad & \sum_{k=1}^{K}\Big(\frac{1}{M_k}\sum_{j=1}^{M_k}\ell(\mathbf{x}_{k,j}, \mathbf{y}_{k,j}; \mathbf{w}_k) + \frac{\lambda}{K}R(\mathbf{w}_k)\Big) \\ \text{s.t.} \quad & \mathbf{w}_k = \mathbf{z}_k^l,\ \mathbf{w}_l = \mathbf{z}_k^l, \quad l \in \mathcal{N}_k, \quad \forall k \in \mathcal{C}, \end{aligned} \tag{1}$$

where $\ell : \mathbb{R}^P \to \mathbb{R}$ is the loss function, $R : \mathbb{R}^P \to \mathbb{R}$ is the regularizer function, $\lambda > 0$ is the regularization parameter, and the equality constraints enforce consensus. The auxiliary variables $\mathcal{Z} \coloneqq \{\mathbf{z}_k^l\}_{l\in\mathcal{N}_k}$ are only used to derive the local recursions and are eventually eliminated. In the following, we consider the learning problem where $\ell(\cdot)$ and $R(\cdot)$ are convex, but not necessarily strongly convex or smooth.

2.2. Approximate Augmented Lagrangian

The augmented Lagrangian associated with (1) is given by

$$\begin{aligned}\mathcal{L}_\rho(\mathcal{V},\mathcal{M},\mathcal{Z}) &= \sum_{k=1}^{K}\Big(\frac{\ell(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k)}{M_k}+\frac{\lambda R(\mathbf{w}_k)}{K}\Big)\\ &+\sum_{k=1}^{K}\sum_{l\in\mathcal{N}_k}\Big[\boldsymbol{\mu}_k^{l\mathsf{T}}(\mathbf{w}_k-\mathbf{z}_k^l)+\boldsymbol{\gamma}_k^{l\mathsf{T}}(\mathbf{w}_l-\mathbf{z}_k^l)\Big]\\ &+\frac{\rho}{2}\sum_{k=1}^{K}\sum_{l\in\mathcal{N}_k}\Big(||\mathbf{w}_k-\mathbf{z}_k^l||^2+||\mathbf{w}_l-\mathbf{z}_k^l||^2\Big)\end{aligned} \tag{2}$$

where $\rho>0$ is a penalty parameter and $\mathcal{M}\coloneqq\{\{\boldsymbol{\mu}_k^l\}_{l\in\mathcal{N}_k},\{\boldsymbol{\gamma}_k^l\}_{l\in\mathcal{N}_k}\}_{k=1}^K$ are the Lagrange multipliers associated with the constraints in (1).

Given that the Lagrange multipliers $\mathcal{M}$ are initialized to zero, by using the Karush-Kuhn-Tucker conditions of optimality for (1) and setting $\boldsymbol{\gamma}_k^{(n)}=2\sum_{l\in\mathcal{N}_k}(\boldsymbol{\gamma}_k^l)^{(n)}$, it can be shown that the Lagrange multipliers $\{\boldsymbol{\mu}_k^l\}_{l\in\mathcal{N}_k}$ and the auxiliary variables $\mathcal{Z}$ are eliminated [37, 38]. The resulting algorithm reduces to the following iterative steps at client k.

$$\begin{aligned}\mathbf{w}_k^{(n)} &= \arg\min_{\mathbf{w}_k}\left[f_k(\mathbf{w}_k)+\mathbf{w}_k^{\mathsf{T}}\boldsymbol{\gamma}_k^{(n-1)}+\rho\sum_{l\in\mathcal{N}_k}\left\|\mathbf{w}_k-\frac{\mathbf{w}_k^{(n-1)}+\mathbf{w}_l^{(n-1)}}{2}\right\|^2\right]\\ \boldsymbol{\gamma}_k^{(n)} &= \boldsymbol{\gamma}_k^{(n-1)}+\rho\sum_{l\in\mathcal{N}_k}\Big(\mathbf{w}_k^{(n)}-\mathbf{w}_l^{(n)}\Big)\end{aligned} \tag{3}$$

where n is the iteration index and

$$f_k(\mathbf{w}_k)=\frac{\ell(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k)}{M_k}+\frac{\lambda R(\mathbf{w}_k)}{K}. \tag{4}$$

To handle nonsmooth $\ell(\cdot)$ and $R(\cdot)$ functions, we take the first-order approximation of f_k with an l_2-norm prox function, denoted as $\hat{f}_k$. Similarly as in [35, 39], such an approximation is given by

$$\begin{aligned}\hat{f}_k(\mathbf{w}_k;\mathcal{V}^{(n)}) &= \frac{\ell(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k^{(n)})}{M_k}+\frac{\lambda R(\mathbf{w}_k^{(n)})}{K}+\frac{\left\|\mathbf{w}_k-\mathbf{w}_k^{(n)}\right\|^2}{2\eta_k^{(n+1)}}\\ &+\Big(\mathbf{w}_k-\mathbf{w}_k^{(n)}\Big)^{\mathsf{T}}\Big(\frac{\ell'(\mathbf{X}_k,\mathbf{y}_k;\mathbf{w}_k^{(n)})}{M_k}+\frac{\lambda R'(\mathbf{w}_k^{(n)})}{K}\Big)\end{aligned} \tag{5}$$

where $\mathcal{V}^{(n)} = \{\mathbf{w}_k^{(n)}, k \in \mathcal{C}\}$, and $\eta_k^{(n)}$ is a time-varying step size.

Taking the first-order approximation of f_k leads to an inexact update at a given iteration; however, the algorithm does not need to solve the problem with high precision at each iteration to guarantee overall accuracy [35]. In the end, considering $\hat{f}_k$ instead of f_k in the primal update makes the algorithm capable of solving nonsmooth objectives with a minimal impact on overall accuracy. Unlike the method used in [23] to deal with nonsmooth objective functions, the approach taken here is compatible with the algorithm's convergence to the exact objective value.

2.3. Privacy Preservation

To prevent the leakage of private information, we introduce local differential privacy to the algorithm via message perturbation. For this purpose, each client k shares at iteration n with its neighbors the perturbed estimate

$$\widetilde{\mathbf{w}}_k^{(n)} = \mathbf{w}_k^{(n)} + \boldsymbol{\xi}_k^{(n)} \tag{6}$$

with $\boldsymbol{\xi}_k^{(n)} \sim \mathcal{N}(\mathbf{0}, \sigma_k^2(n)\mathbf{I}_P)$. We denote $\widetilde{\mathcal{V}}^{(n)} = \{\widetilde{\mathbf{w}}_k^{(n)}, k \in \mathcal{C}\}$.

The value of the noise perturbation variance, $\sigma_k^2(n)$, in (6) dictates the privacy protection of the algorithm. To guarantee convergence to the optimal solution, as opposed to a neighborhood of it, the variance must decrease with the iterations [40]. This is made possible by using dynamic zCDP, where the privacy budget is iteration-specific.

The proposed zero-Concentrated Differentially Private Networked Federated Learning (zCDP-NFL) algorithm is developed in algorithm 1. It is a networked federated learning algorithm that protects the privacy of the clients with local dynamic differential privacy and can handle nonsmooth objective functions. In the following sections, we conduct mathematical analysis to quantify its privacy protection and establish convergence guarantees.

Algorithm 1 zCDP-NFL

Initialization: $\mathbf{w}_k^{(0)} = \mathbf{0}$, $\boldsymbol{\gamma}_k^{(0)} = \mathbf{0}$, $\forall k \in \mathcal{K}$

– *Procedure at client* k –

For iteration $n = 1, 2, \ldots$:

$$\mathbf{w}_k^{(n)} = \arg\min_{\mathbf{w}_k} \hat{f}_k(\mathbf{w}_k; \widetilde{\mathcal{V}}^{(n-1)}) + \mathbf{w}_k^{\mathsf{T}} \boldsymbol{\gamma}_k^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k} \left\| \mathbf{w}_k - \frac{\widetilde{\mathbf{w}}_k^{(n-1)} + \widetilde{\mathbf{w}}_l^{(n-1)}}{2} \right\|^2 \tag{7}$$

$$\widetilde{\mathbf{w}}_k^{(n)} = \mathbf{w}_k^{(n)} + \boldsymbol{\xi}_k^{(n)} \tag{8}$$

$$\boldsymbol{\gamma}_k^{(n)} = \boldsymbol{\gamma}_k^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k} \left(\widetilde{\mathbf{w}}_k^{(n)} - \widetilde{\mathbf{w}}_l^{(n)} \right) \tag{9}$$

End For

3. Privacy Analysis

The first step in the quantification of differential privacy is to measure the impact of an individual data sample on the output of the local training process. For this purpose, we define the l_2-norm sensitivity as follows.

Definition I. *The l_2-norm sensitivity is given by*

$$\Delta_{k,2} = \max_{\mathcal{D}_k, \mathcal{D}'_k} \left\| \mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}'_k}^{(n)} \right\| \tag{10}$$

where $\mathbf{w}_{k,\mathcal{D}_k}^{(n)}$ and $\mathbf{w}_{k,\mathcal{D}'_k}^{(n)}$ denote the local primal variable updates from two neighboring data sets $\mathcal{D}_k$ and $\mathcal{D}'_k$ differing in only one data sample $(\mathbf{x}'_{k,M_k}, y'_{k,M_k})$, i.e., $\mathcal{D}'_k \coloneqq \{(\mathbf{X}'_k, \mathbf{y}'_k) : \mathbf{X}'_k = [\mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \ldots, \mathbf{x}_{k,M_k-1}, \mathbf{x}'_{k,M_k}]^{\mathsf{T}} \in \mathbb{R}^{M_k \times P},\ \mathbf{x}_{k,j} \in \mathbb{R}^P,\ j = 1, \ldots, M_k, \mathbf{y'}_k = [y_{k,1}, y_{k,2}, \ldots, y_{k,M_k-1}, y'_{k,M_k}]^{\mathsf{T}} \in \mathbb{R}^{M_k}\}$.

Two parameters govern privacy protection in dynamic zCDP. The initial privacy value, $\varphi_k^{(0)}$, and the variance decrease rate, τ. To establish a relation between the privacy value at a given iteration, $\varphi_k^{(n)}$, and the noise perturbation, it is necessary to take the following assumption.

Assumption 1. *The functions $\ell_k(\cdot)$ have bounded gradient, that is, there exists a constant c_1 such that $||\ell_k'(\cdot)|| \leqslant c_1, \forall k \in \mathcal{C}$.*

We now quantify the l_2-norm sensitivity in the following result.

Lemma I. *Under Assumption 1, the l_2-norm sensitivity is given by*

$$\Delta_{k,2}(n) = \max_{\mathcal{D},\mathcal{D}'} ||\mathbf{w}_{k,\mathcal{D}}^{(n)} - \mathbf{w}_{k,\mathcal{D}'}^{(n)}|| \leqslant \frac{2c_1}{M_k(2\rho|\mathcal{N}_k| + \frac{1}{\eta^{(n)}})}. \tag{11}$$

Proof. See Appendix A. □

With the l_2-norm sensitivity, we can establish the relation between the noise perturbation in (8) and the privacy value $\varphi_k^{(n)}$, quantifying the local privacy guarantee of the algorithm in terms of zCDP.

Theorem I. *Under Assumption 1, zCDP-NFL satisfies dynamic $\varphi_k^{(n)}$-zCDP with the relation between $\varphi_k^{(n)}$ and $\sigma_k^2(n)$ given by*

$$\sigma_k^2(n) = \frac{\Delta_{k,2}^2(n)}{2\varphi_k^{(n)}}. \tag{12}$$

Proof. See Appendix B. □

Using the result above, it is possible to obtain the total privacy guarantee throughout the computation in terms of (ϵ, δ)-DP using [20, Lemma 1.7]. We establish the following.

Corollary. *For any $\tau \in (0,1)$ and $\delta \in (0,1)$, zCDP-NFL guarantees (ϵ, δ)-DP throughout the computation with $\epsilon = \max_{k \in \mathcal{C}} \epsilon_k$, where $\epsilon_k = \varphi_k^{(1)} \frac{1-\tau^T}{\tau^{T-1}-\tau^T} + 2\sqrt{\varphi_k^{(1)} \frac{1-\tau^T}{\tau^{T-1}-\tau^T} \log \frac{1}{\delta}}$, and T is the final iteration index.*

Proof. See Appendix C. □

4. Convergence Analysis

This section proves that the zCDP-NFL algorithm converges to the optimal value in $O(1/n)$ iterations under the following assumption that the objective func-

tion $f(\cdot)$ is convex. Additionally, we derive the privacy-accuracy trade-off bound of the algorithm.

Assumption 2. *The objective function $f(\cdot)$ is convex.*

4.1. Alternative Representation

We begin by transforming the minimization problem (1) into (13) by reformulating the conditions. We denote by $\boldsymbol{\mathfrak{w}} = [\mathbf{w}_1^\mathsf{T}, \mathbf{w}_2^\mathsf{T}, ..., \mathbf{w}_K^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{KP}$, and $\boldsymbol{\mathfrak{z}} = [(\boldsymbol{z}_k^l)^\mathsf{T}, (\boldsymbol{z}_l^k)^\mathsf{T}; \forall (k,l) \in \mathcal{E}]^\mathsf{T} \in \mathbb{R}^{2EP}$ the vectors of the concatenated vectors $\mathbf{w}_k$ and $\boldsymbol{z}_k^l$ respectively. We also introduce the matrices $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathbb{R}^{2EP \times KP}$ composed of $P \times P$-sized blocks. Given a couple of connected clients $(k,l) \in \mathcal{E}$, their associated auxiliary variable $\boldsymbol{z}_{k,l}$, and its corresponding index in $\boldsymbol{\mathfrak{z}}$, q; the blocks $\big(\mathbf{A}_1\big)_{q,k}$ and $\big(\mathbf{A}_2\big)_{q,l}$ are equal to the identity matrix $\mathbf{I}_d$, all other blocks are null. Finally, we set $\boldsymbol{A} = [\boldsymbol{A}_1; \boldsymbol{A}_2] \in \mathbb{R}^{4EP \times KP}$ and $\boldsymbol{B} = [-\boldsymbol{I}_{2EP}; -\boldsymbol{I}_{2EP}] \in \mathbb{R}^{4EP \times 2EP}$. Hence, we can reformulate (1) as

$$\begin{aligned} \min_{\boldsymbol{\mathfrak{w}}} \quad & \sum_{k=1}^{K} \Big(\frac{1}{M_k} \sum_{j=1}^{M_k} \ell(\mathbf{x}_{k,j}, \mathbf{y}_{k,j}; \mathbf{w}_k) + \frac{\lambda}{K} R(\mathbf{w}_k) \Big). \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}} = 0 \end{aligned} \tag{13}$$

The newly introduced matrices can be used to reformulate the Lagrangian, the objective function, and the ADMM steps. The conventional augmented Lagrangian in (2) can be expressed as

$$\mathcal{L}_\rho = f(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)}) + \langle \boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}, \boldsymbol{\lambda} \rangle + \frac{\rho}{2} ||\boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}||^2$$

where $f(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)}) = \sum_{k=1}^{K} f(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$. Similarly, the augmented Lagrangian, corresponding to the use of the first-order approximation of the objective function in (5), can be expressed as

$$\hat{\mathcal{L}}_\rho = \hat{f}(\boldsymbol{\mathfrak{w}}, \widetilde{\mathcal{V}}^{(n)}) + \langle \boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}, \boldsymbol{\lambda} \rangle + \frac{\rho}{2} ||\boldsymbol{A}\boldsymbol{\mathfrak{w}} + \boldsymbol{B}\boldsymbol{\mathfrak{z}}||^2$$

where $\hat{f}(\mathfrak{w}, \widetilde{\mathcal{V}}^{(n)}) = \sum_{k=1}^{K} \hat{f}_k(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$ with $\hat{f}_k(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$, as defined in (5).

From now on, we will denote $\hat{f}(\mathfrak{w}, \widetilde{\mathcal{V}}^{(n)})$ and $\hat{f}_k(\mathbf{w}_k, \widetilde{\mathcal{V}}^{(n)})$ by $\hat{f}(\mathfrak{w})$ and $\hat{f}_k(\mathbf{w}_k)$, respectively. Further, we let $\widetilde{\mathfrak{w}}^{(n)}$, $\mathfrak{w}^{(n)}$, and $\boldsymbol{\xi}^{(n)}$ denote the concatenation of $\widetilde{\mathbf{w}}_k^{(n)}$, $\mathbf{w}_k^{(n)}$, and $\boldsymbol{\xi}_k^{(n)}$, respectively, such that $\widetilde{\mathfrak{w}}^{(n)} = \mathfrak{w}^{(n)} + \boldsymbol{\xi}^{(n)}$.

We introduce the diagonal matrix $\boldsymbol{D}^{(n+1)} \in \mathbb{R}^{K \times K}$ comprising the time-varying step sizes, i.e., $[\boldsymbol{D}^{(n+1)}]_{k,k} = \frac{1}{\sqrt{2\eta_k^{(n+1)}}}$, and reformulate $\hat{f}(\mathfrak{w}^{(n+1)})$ in matrix form:

$$\begin{aligned}\hat{f}(\mathfrak{w}^{(n+1)}) =& f(\widetilde{\mathfrak{w}}^{(n)}) + ||\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\mathfrak{w}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})||^2 \\ &+ (\mathfrak{w}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})^{\mathsf{T}} f'(\widetilde{\mathfrak{w}}^{(n)}) \end{aligned} \tag{14}$$

The resulting function $\hat{f}$ is convex with respect to $\mathfrak{w}$. That is, it satisfies $\hat{f}(\widetilde{\mathfrak{w}}^{(n)}) - \hat{f}(\mathfrak{w}) \leqslant \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}, \hat{f}'(\widetilde{\mathfrak{w}}^{(n)}) \rangle$, where the subgradient $\hat{f}'(\mathfrak{w}^{(n+1)}) \in \mathbb{R}^{KP}$ is given by $\hat{f}'(\mathfrak{w}^{(n+1)}) = 2\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\mathfrak{w}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}) + f'(\widetilde{\mathfrak{w}}^{(n)})$.

The steps of the ADMM, consisting of the minimization of $\hat{\mathcal{L}}_\rho$ with respect to $\mathfrak{w}, \mathfrak{z}$ and $\boldsymbol{\lambda}$ alternatively, can now be reformulated with the newly introduced variables as follows:

$$\begin{aligned}\hat{f}'(\mathfrak{w}^{(n+1)}) + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{\lambda}^{(n)} + \rho \boldsymbol{A}^{\mathsf{T}}(\boldsymbol{A}\mathfrak{w}^{(n+1)} + \boldsymbol{B}\mathfrak{z}^{(n)}) &= 0 \\ \boldsymbol{B}^{\mathsf{T}}\boldsymbol{\lambda}^{(n)} + \rho \boldsymbol{B}^{\mathsf{T}}(\boldsymbol{A}\widetilde{\mathfrak{w}}^{(n+1)} + \boldsymbol{B}\mathfrak{z}^{(n+1)}) &= 0 \\ \boldsymbol{\lambda}^{(n+1)} - \boldsymbol{\lambda}^{(n)} + \rho(\boldsymbol{A}\widetilde{\mathfrak{w}}^{(n+1)} + \boldsymbol{B}\mathfrak{z}^{(n+1)}) &= 0 \end{aligned} \tag{15}$$

We introduce the following auxiliary matrices in order to reduce (15) to two steps, similarly as in [41]: $\boldsymbol{H}_+ = \boldsymbol{A}_1^{\mathsf{T}} + \boldsymbol{A}_2^{\mathsf{T}}$, $\boldsymbol{H}_- = \boldsymbol{A}_1^{\mathsf{T}} - \boldsymbol{A}_2^{\mathsf{T}}$, $\boldsymbol{\alpha} = \boldsymbol{H}_-^{\mathsf{T}}\mathfrak{w}$, $\boldsymbol{L}_+ = \frac{1}{2}\boldsymbol{H}_+\boldsymbol{H}_+^{\mathsf{T}}$, $\boldsymbol{L}_- = \frac{1}{2}\boldsymbol{H}_-\boldsymbol{H}_-^{\mathsf{T}}$ and $\boldsymbol{M} = \frac{1}{2}(\boldsymbol{L}_+ + \boldsymbol{L}_-)$. We note that $\boldsymbol{L}_+$ and $\boldsymbol{L}_-$ correspond to the signless Laplacian and signed Laplacian matrices of the network, respectively. Hence, $\boldsymbol{L}_-$ is positive semi-definite with the nullspace given by $Null(\boldsymbol{L}_-) = span\{1\}$. Then, as derived in [41, Section II.B], (15) becomes

$$\begin{aligned}\hat{f}'(\mathfrak{w}^{(n+1)}) + \boldsymbol{\alpha}^{(n)} + 2\rho \boldsymbol{M}\mathfrak{w}^{(n+1)} - \rho \boldsymbol{L}_+\widetilde{\mathfrak{w}}^{(n)} &= 0 \\ \boldsymbol{\alpha}^{(n+1)} - \boldsymbol{\alpha}^{(n)} - \rho \boldsymbol{L}_-\widetilde{\mathfrak{w}}^{(n+1)} &= 0 \end{aligned} \tag{16}$$

The last reformulation step is based on the work in [42]. We introduce the matrix $\boldsymbol{Q} = \sqrt{\boldsymbol{L}_-/2}$, note that by construction $Null(\boldsymbol{Q}) = span\{1\}$, the auxiliary sequence $\boldsymbol{r}^{(n)} = \sum_{s=0}^{n} \boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(s)}$, vector $\boldsymbol{q}^{(n)} = \begin{pmatrix} \boldsymbol{r}^{(n)} \\ \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} \end{pmatrix}$, and matrix $\boldsymbol{G} = \begin{pmatrix} \rho\boldsymbol{I} & 0 \\ 0 & \rho\boldsymbol{L}_+/2 \end{pmatrix}$. Combining both equations in (16), as in [42, Lemma 1], and reformulating the result, see see [42, Lemma 2], we obtain

$$\frac{\hat{f}'(\boldsymbol{\mathfrak{w}}^{(n+1)})}{\rho} + 2\boldsymbol{Q}\boldsymbol{r}^{(n+1)} + \boldsymbol{L}_+(\boldsymbol{\mathfrak{w}}^{(n+1)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}) = 2\boldsymbol{M}\boldsymbol{\xi}^{(n+1)}. \tag{17}$$

4.2. Convergence Proof

We start by establishing a bound for the distance to the optimal solution, denoted $\boldsymbol{\mathfrak{w}}^*$, at a given iteration.

Lemma II. *For any $\boldsymbol{r} \in \mathbb{R}^{KP}$ and at any iteration n, we have*

$$\begin{aligned} &\frac{f(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}) - f(\boldsymbol{\mathfrak{w}}^*)}{\rho} + \langle \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}, 2\boldsymbol{Q}\boldsymbol{r}\rangle \qquad (18)\\ &\leqslant \frac{1}{\rho}(||\boldsymbol{q}^{(n-1)} - \boldsymbol{q}^*||_{\boldsymbol{G}}^2 - ||\boldsymbol{q}^{(n)} - \boldsymbol{q}^*||_{\boldsymbol{G}}^2) - 2\langle \boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}, \boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)}\rangle - ||\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n-1)}||_{\frac{\boldsymbol{L}_+}{2}}^2 \\ &\quad + \langle \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} - \boldsymbol{\mathfrak{w}}^*, \boldsymbol{L}_+(2\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n-1)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)})\rangle + \frac{4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)}{\Phi_{\min}(\boldsymbol{L}_-)}||\boldsymbol{\xi}^{(n+1)}||_2^2 \\ &\quad + \langle \widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} - \boldsymbol{\mathfrak{w}}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)} - \widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)})\rangle \end{aligned}$$

where $\boldsymbol{q}^* = [\boldsymbol{r}^T, (\boldsymbol{\mathfrak{w}}^*)^T]$.

Proof. See Appendix D. □

Following the result of Lemma II, we can establish the following theorem from which we will derive the converge results.

Theorem II. *Under Assumption 2, and given the final iteration $T > 0$, we can*

bound the expected error of the zCDP-NFL algorithm as

$$\begin{aligned}\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] \leqslant \frac{\rho}{T}\sum_{n=1}^{T}\Big(&-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}} \\ &+ \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle \\ &- \langle \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle + ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+}\Big) \\ &+ \frac{1}{T}\frac{\rho P 4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)\sum_{k=1}^{K}\sigma_k^{2(0)}}{\Phi_{\min}(\boldsymbol{L}_-)(1-\tau)} \\ &+ \frac{\langle \widetilde{\mathfrak{w}}^{(1)}, \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(1)} - \widetilde{\mathfrak{w}}^{(0)})\rangle}{T} + \frac{\rho||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(0)}||_2^2}{T} + \frac{\rho||\widetilde{\mathfrak{w}}^{(0)} - \mathfrak{w}^*||^2_{\frac{\boldsymbol{L}_-}{2}}}{T}. \end{aligned} \tag{19}$$

where $\hat{\mathfrak{w}}^{(T)} = \frac{1}{T}\sum_{n=1}^{T}\widetilde{\mathfrak{w}}^{(n)}$*, and the expectation is taken with respect to the noise. Since* $\mathfrak{w}^*$ *is the optimal solution,* $\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)]$ *is positive.*

Proof. See Appendix E. □

4.3. Convergence Properties

We can derive three important results from Theorem II. The first is that the zCDP-NFL algorithm converges to the exact solution of (1). The second is the rate of this convergence. The third result is the privacy accuracy trade-off bound of the algorithm. First, we define the required assumptions for convergence.

Assumption 3. *We require that* $\lim_{n\to+\infty}\eta_k^{(n)} = 0, \forall k \in \mathcal{C}$*. This will enforce the asymptotic stability of the local estimates.*

Theorem III. *Under Assumptions 2 and 3, the zCDP-NFL algorithm defined by the steps* (7)-(9)*, converges to the exact solution.*

Proof. We can simplify the result of Theorem II into the following:

$$\begin{aligned}\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] \leqslant \frac{\rho}{T}\sum_{n=1}^{T}\Big(&-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle \\ &+ \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle + ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+}\Big) \\ &+ \frac{1}{T}\frac{\rho P 4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)\sum_{k=1}^{K}\sigma_k^{2(0)}}{\Phi_{\min}(\boldsymbol{L}_-)(1-\tau)} \\ &+ \frac{\langle \widetilde{\mathfrak{w}}^{(1)}, \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(1)} - \widetilde{\mathfrak{w}}^{(0)})\rangle}{T} + \frac{\rho||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(0)}||_2^2}{T} + \frac{\rho||\widetilde{\mathfrak{w}}^{(0)} - \mathfrak{w}^*||^2_{\frac{\boldsymbol{L}_-}{2}}}{T}. \end{aligned} \tag{20}$$

We will consider the terms separately in their order of appearance. We first prove that $\lim_{n\to+\infty} \frac{\rho\sum_{n=1}^{T} -2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle}{T} = 0$.

We can now note that

$$\begin{aligned} &-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle = -\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}\rangle - \langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})\rangle \\ &\quad - \langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle - \langle \boldsymbol{Q}(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)}), \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle \\ &= -||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}||_2^2 - ||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}||_2^2 + ||\boldsymbol{Q}(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})||_2^2 \\ &\leqslant -||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}||_2^2 - ||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}||_2^2 + ||\boldsymbol{Q}||_2^2||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_2^2. \end{aligned} \tag{21}$$

As seen in (7), $\mathfrak{w}^{(n+1)}$ minimizes a function where all terms are bounded except the term $\frac{\left\|\mathfrak{w} - \widetilde{\mathfrak{w}}^{(n)}\right\|^2}{2\eta_k^{(n+1)}}$. Therefore, under Assumption 3, $\lim_{n\to+\infty} ||\mathfrak{w}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_2^2 = 0$. Since $\widetilde{\mathfrak{w}}^{(n+1)}$ is defined as $\widetilde{\mathfrak{w}}^{(n+1)} = \mathfrak{w}^{(n+1)} + \boldsymbol{\xi}^{(n+1)}$ with $\lim_{n\to+\infty} ||\boldsymbol{\xi}^{(n+1)}|| = 0$, we have $\lim_{n\to+\infty} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_2^2 = 0$.

This implies that $-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle$ is bounded by a series converging to 0. Therefore, since $\mathbb{E}[\hat{f}(\hat{\mathfrak{w}}^{(T)}) - \hat{f}(\mathfrak{w}^*)]$ is positive, $\frac{\rho\sum_{n=1}^{T} -2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle}{T}$ converges to 0.

Next, under Assumption 3, we have $\lim_{n\to+\infty} \frac{\rho\sum_{n=1}^{T}\langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle}{T} = 0$ since $[\boldsymbol{D}^{(n+1)}]_{k,k} = \frac{1}{\sqrt{2\eta_k^{(n+1)}}}$. We now consider $\frac{\rho}{T}\sum_{n=1}^{T} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+}$. As we have shown that $\lim_{n\to+\infty} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_2^2 = 0$, we have $\lim_{n\to+\infty} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+} = 0$,

and therefore, the sum $\sum_{n=1}^{T} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+}$ is a Cauchy sequence. Hence we have $\lim_{n\to+\infty} \frac{\rho}{T} \sum_{n=1}^{T} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+} = 0$.

Finally, the terms outside of the summation trivially converge to 0 as $T \to +\infty$. This concludes the proof. □

We now introduce the required assumption to establish the convergence rate of the algorithm.

Assumption 4. *The $\eta_k^{(n)}, k \in \mathcal{C}$ are chosen such that $||\boldsymbol{D}^{(n+1)}||_2^2$ is a convergent series. This assumption, stronger than Assumption 3, is necessary to guarantee the exponential stability of the local estimates.*

Theorem IV. *Under Assumptions 2 and 4, the zCDP-NFL algorithm converges with a rate of $\mathcal{O}(1/n)$ iterations.*

Proof. In the following, we assume that the optimal solution $\mathfrak{w}^* \neq \mathbf{0}$. If $\mathfrak{w}^* = \mathbf{0}$, one could add a nonzero artificial dimension and proceed.

In order to prove this result, we will show that the expectation of the error is bounded by a bounded term divided by T. Notably, we will show that the sum in (20) converges. We consider the terms in their order of appearance in Theorem II. We will also use the result of Theorem III, $\lim_{n\to+\infty} \widetilde{\mathfrak{w}}^{(n)} = \mathfrak{w}^*$, for which Assumption 3 is satisfied by Assumption 4.

To begin, we consider $\frac{\rho}{T} \sum_{n=1}^{T} \Big(-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)} \rangle - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}} \Big)$. Since $\widetilde{\mathfrak{w}}^{(n)}$ converges to $\widetilde{\mathfrak{w}}^*$, $-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)} \rangle$ converges to $-2||\boldsymbol{Q}\mathfrak{w}^*||_2^2$ that is strictly negative. Therefore, there exist an iteration n_0 after which all terms $-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)} \rangle$ are negative. Hence, $\frac{\rho}{T} \sum_{n=1}^{T} \Big(-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)} \rangle - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}} \Big) \leqslant \frac{\rho}{T} \sum_{n=1}^{n_0} -2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)} \rangle$.

Next, we can bound $\langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)}) \rangle$ by $||\widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*||_2^2 \frac{2}{\rho} ||\boldsymbol{D}^{(n+1)}||_2^2 ||\boldsymbol{I}_P||_2^2 ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)}||_2^2$ and thus $\rho \sum_{n=1}^{T} \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)}) \rangle$ by $2 \sum_{n=1}^{T} ||\widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*||_2^2 ||\boldsymbol{D}^{(n+1)}||_2^2 ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)}||_2^2$. Using

$\lim_{n\to+\infty} \widetilde{\mathfrak{w}}^{(n)} = \mathfrak{w}^*$, there exist constants α_0 and α_1 such that $\forall n \in \mathbb{N}, ||\widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*||_2^2 \leqslant \alpha_0$ and $||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)}||_2^2 \leqslant \alpha_1$. This leads to $2\sum_{n=1}^{T} ||\widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*||_2^2 ||\boldsymbol{D}^{(n+1)}||_2^2 ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)}||_2^2 \leqslant 2\alpha_0\alpha_1 \sum_{n=1}^{T} ||\boldsymbol{D}^{(n+1)}||_2^2$, which is a convergent series under Assumption 4. Therefore, $\frac{\rho}{T}\sum_{n=1}^{T}\langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle$ converges to zero in $O(1/n)$.

We can bound $\langle \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle$ by $2||\mathfrak{w}^*||_2^2||\boldsymbol{L}_+||_2^2(||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_2^2 + ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||_2^2)$. Using $\lim_{n\to+\infty} \widetilde{\mathfrak{w}}^{(n)} = \mathfrak{w}^*$, the series $\sum_{n=1}^{\infty} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_2^2$ and $\sum_{n=1}^{\infty} ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_{\boldsymbol{L}_+}^2$ converge to values that we denote α_2 and α_3, respectively. We have $\sum_{n=1}^{T}(-\langle \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle + ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||_{\boldsymbol{L}_+}^2) \leqslant 4||\mathfrak{w}^*||_2^2||\boldsymbol{L}_+||_2^2\alpha_2 + \alpha_3$.

Finally, we prove that all terms outside of the sum are bounded by a constant with respect to T. The only one requiring further analysis is $\langle \widetilde{\mathfrak{w}}^{(1)}, \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(1)} - \widetilde{\mathfrak{w}}^{(0)})\rangle$ and it can be bounded by $||\widetilde{\mathfrak{w}}^{(1)}||_2^2||\boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(1)} - \widetilde{\mathfrak{w}}^{(0)})||_2^2$.

Each term has either been bounded by a constant with respect to T, divided by T; this concludes the proof. □

Remark: In practice, Assumption 4 can be relaxed in most cases.

4.4. Privacy Accuracy Trade-off

The last result established by Theorem II is the privacy accuracy trade-off bound. The privacy accuracy trade-off quantifies how ensuring more privacy deteriorates the accuracy of the algorithm and is one of the most important parameters of a privacy-preserving algorithm. Under Assumption 4, we can reformulate (19) as

$$\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] \leqslant \frac{\alpha}{T} + \frac{\alpha_{\boldsymbol{\xi}}}{T}\frac{\sum_{k=1}^{K}\sigma_k^2(0)}{1-\tau} \tag{22}$$

where α is a constant with respect to T and the noise perturbation and $\alpha_{\boldsymbol{\xi}} = \frac{\rho P 4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)}{\Phi_{\min}(\boldsymbol{L}_-)}$.

By combining this result with Theorem I, we obtain

$$\mathbb{E}[f(\hat{\mathbf{w}}^{(T)}) - f(\mathbf{w}^*)] \leqslant \frac{\alpha}{T} + \frac{\alpha_{\boldsymbol{\xi}}}{T} \frac{\sum_{k=1}^{K} \frac{\Delta_{k,2}^2(0)}{2\varphi_k^{(1)}}}{1-\tau} \tag{23}$$

In the common case where the privacy parameter $\varphi_k^{(1)}$ is identical for all clients, i.e., $\varphi_k^{(1)} = \varphi^{(1)}, \forall k \in \mathcal{C}$, we have

$$\mathbb{E}[f(\hat{\mathbf{w}}^{(T)}) - f(\mathbf{w}^*)] \leqslant \frac{\alpha}{T} + \frac{\alpha_{\boldsymbol{\xi}}}{T} \frac{\sum_{k=1}^{K} \Delta_{k,2}^2(0)}{2K\varphi^{(1)}(1-\tau)} \tag{24}$$

With this result, we see that ensuring more privacy, which can be done by decreasing $\varphi^{(1)}$ or having τ closer to 1, would result in a less restrictive convergence bound for the algorithm.

5. Numerical Simulations

This section presents simulation results to evaluate the performance and privacy accuracy trade-off of the proposed zCDP-NFL. To compare the different DP implementations, we introduce the (ϵ, δ)DP-NFL, identical to zCDP-NFL except for the fact that it uses conventional (ϵ, δ)-DP rules to control the local noise perturbation. On nonsmooth objective functions, we benchmark the proposed algorithm against conventional subgradient-based networked FL, as presented in [23], modified to use zCDP and denoted zCDP-grad-NFL. On smooth objective functions, we benchmark the proposed algorithm against the existing distributed learning algorithm P-ADMM, presented in [24] that uses a networked FL architecture, zCDP, and the ADMM, but is constrained to smooth objective functions. In the following, we consider the elastic net, least absolute deviation, and ridge regression problems, all presented in [43].

For a fair comparison, the algorithms are tuned to provide the same total privacy guarantees throughout the computation - this is made possible by the corollary of Theorem I. This corollary provides (ϵ, δ)-DP guarantees for an algorithm

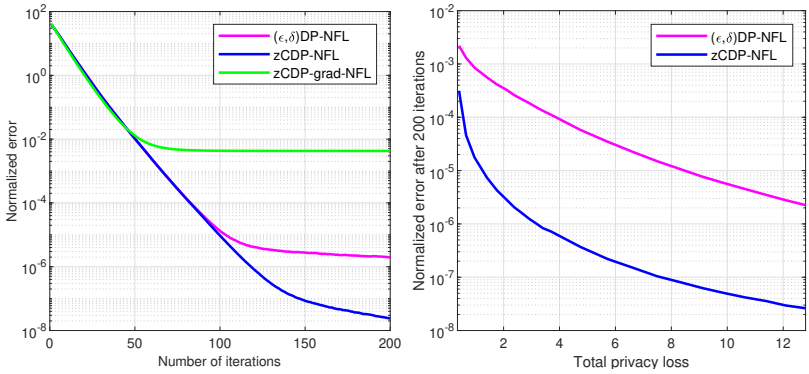


Figure 1: Learning curves (left) and privacy-accuracy trade-off (right) on the elastic net problem.

using zCDP with both $\varphi^{(1)}$ and τ as parameters. Furthermore, the algorithms are tuned to observe identical initial convergence speeds when possible.

In the following, we consider a network with a random topology comprising $K = 50$ nodes, where each node connects to 3 other nodes on average. Each node k possesses $M_k = 50$ local noisy observations of the unknown parameter $\mathbf{w}$ of dimension $P = 8$. The proposed simulations are performed on synthetic data and performance is evaluated by computing the normalized error defined as $\sum_{k=1}^{K} ||\mathbf{w}_k^{(n)} - \mathbf{w}_c||^2 / ||\mathbf{w}_c||^2$, $\mathbf{w}_c$ being the centralized solution obtained by the CVX toolbox [44].

Figure 1 contains the learning curve, i.e., normalized error versus iteration index, and privacy-accuracy trade-off, i.e., normalized error after 200 iterations versus total privacy loss, on the elastic net problem, defined by a smooth loss function $\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{X}_k \mathbf{w}_k - \mathbf{y}_k||^2$ and a nonsmooth regularizer function $R(\mathbf{w}_k) = \lambda_1 ||\mathbf{w}_k||_1 + \lambda_2 ||\mathbf{w}_k||^2$ with $\lambda_1 = 0.001 ||\mathbf{X}^\mathsf{T} \mathbf{y}||_\infty$, as in [43], and $\lambda_2 = 1$. We observe that the proposed ADMM-based algorithm significantly outperforms the subgradient-based algorithm. Furthermore, we can see that the use of the zCDP notion as opposed to the (ϵ, δ)-DP notion allows for better accuracy given

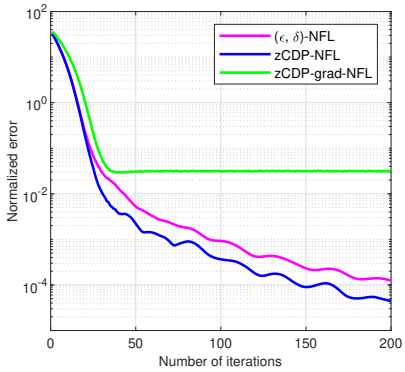


Figure 2: Learning curve on the least absolute deviation problem.

the same privacy budget throughout the computation. This same fact is illustrated for various total privacy losses on the privacy-accuracy trade-off curve.

Figure 2 contains the learning curve on the least absolute deviation problem, solely composed of a nonsmooth loss $\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{X}_k\mathbf{w}_k - \mathbf{y}_k||_1$. We observe once again that the proposed algorithm significantly outperforms its subgradient-based counterpart, as the learning rate required for the zCDP-grad-NFL algorithm to attain a similar initial convergence speed to the ADMM-based algorithms does not allow it to reach high accuracy. In addition, we observe that the use of zCDP allows for better accuracy than (ϵ, δ)-DP. In the end, the proposed zCDP-NFL algorithm significantly improves over existing methods on nonsmooth objectives.

Figure 3 contains the learning curve and privacy-accuracy trade-off on the ridge regression problem, defined by a smooth loss function $\ell(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) = ||\mathbf{X}_k\mathbf{w}_k - \mathbf{y}_k||^2$ and a smooth regularizer function $R(\mathbf{w}_k) = ||\mathbf{w}_k||^2$. Since this objective is smooth, we can present the performance of the P-ADMM algorithm. We observe that the proposed zCDP-NFL algorithm slightly outperforms the P-ADMM algorithm on smooth objectives, despite the inexact ADMM update required to handle nonsmooth objectives. As can be observed on the privacy-

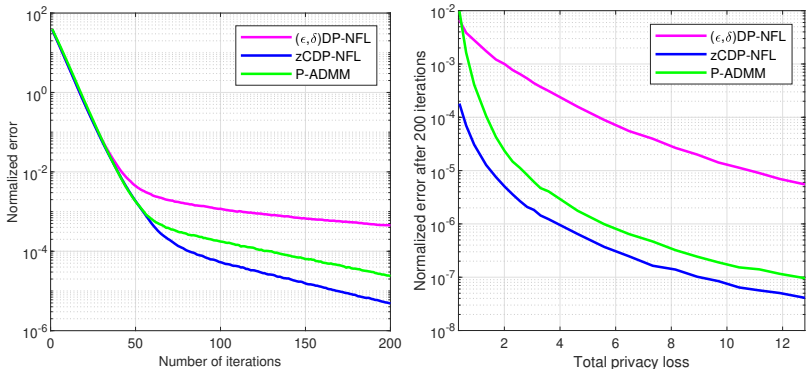


Figure 3: Learning curves (left) and privacy-accuracy trade-off (right) on the smooth ridge regression problem.

accuracy trade-off curve, this is the case for all total privacy losses. This better performance is due to the use of the time-varying step size η in the proposed algorithm. Even though the zCPD-NFL algorithm is designed for nonsmooth objective functions, it offers very good performances on smooth objective functions.

6. Conclusions and Future Directions

The proposed zCDP-NFL algorithm is a networked, privacy-preserving algorithm that accommodates nonsmooth and non-strongly convex objective functions. Each client is protected by local differential privacy with guarantees provided in terms of zCDP. We provided mathematical proofs of the privacy guarantee and convergence to the optimal point in $O(1/n)$, as well as an analysis of the privacy-accuracy trade-off to quantify the accuracy loss caused by increased privacy. Numerical simulations show that the proposed zCDP-NFL algorithm significantly outperforms existing networked algorithms on nonsmooth objectives while offering very good performances on smooth objectives. Future work includes communication efficient implementations and robustness to model poisoning.

Appendix A. Proof of Lemma I

Proof. We consider two neighboring data sets $\mathcal{D}$ and $\mathcal{D}'$ and their respective primal updates for the client k whose data set contains the difference. We will denote $\mathcal{D}_k$ and $\mathcal{D}'_k$ the local data set of client k corresponding to the use of $\mathcal{D}$ and $\mathcal{D}'$, respectively. Moreover, we will denote $\mathbf{w}_{k,\mathcal{D}_k}^{(n)}$ and $\mathbf{w}_{k,\mathcal{D}'_k}^{(n)}$ the estimates computed by client k using the data sets $\mathcal{D}_k$ and $\mathcal{D}'_k$, respectively. These can be computed as

$$\mathbf{w}_{k,\mathcal{D}_k}^{(n)} = \frac{1}{2\rho|\mathcal{N}_k| + \frac{1}{\eta^{(n)}}}\Big(\frac{\widetilde{\mathbf{w}}_k^{(n-1)}}{\eta^{(n)}} + \frac{\rho}{2}\sum_{i\in\mathcal{N}_k}(\widetilde{\mathbf{w}}_k^{(n-1)} - \widetilde{\mathbf{w}}_i^{(n-1)}) + \frac{\boldsymbol{\gamma}_k^{(n-1)}}{2} - \sum_{j=1}^{M_k}\frac{\ell'(\mathbf{x}_{k,j}, y_{k,j}; \widetilde{\mathbf{w}}_k)}{M_k} - \frac{\lambda R'(\mathbf{w}_k)}{K}\Big), \tag{A.1}$$

$$\mathbf{w}_{k,\mathcal{D}'_k}^{(n)} = \frac{1}{2\rho|\mathcal{N}_k| + \frac{1}{\eta^{(n)}}}\Big(\frac{\widetilde{\mathbf{w}}_k^{(n-1)}}{\eta^{(n)}} + \frac{\rho}{2}\sum_{i\in\mathcal{N}_k}(\widetilde{\mathbf{w}}_k^{(n-1)} - \widetilde{\mathbf{w}}_i^{(n-1)}) + \frac{\boldsymbol{\gamma}_k^{(n-1)}}{2} - \sum_{j=1}^{M_k-1}\frac{\ell'(\mathbf{x}_{k,j}, y_{k,j}; \widetilde{\mathbf{w}}_k)}{M_k} - \frac{\ell'(\mathbf{x}'_{k,M_k}, y'_{k,M_k}; \widetilde{\mathbf{w}}_k)}{M_k} - \frac{\lambda R'(\mathbf{w}_k)}{K}\Big). \tag{A.2}$$

We notice that the primal updates corresponding with $\mathcal{D}$ and $\mathcal{D}'$ differ only for the ℓ-update, where for the index M_k, the vector $\mathbf{x}_{k,M_k}$ and the scalar y_{k,M_k} are different from $\mathbf{x}'_{k,M_k}$ and y'_{k,M_k}. Thus, for any neighboring data set $\mathcal{D}$ and $\mathcal{D}'$, the following holds:

$$||\mathbf{w}_{k,\mathcal{D}}^{(n)} - \mathbf{w}_{k,\mathcal{D}'}^{(n)}|| = \Big|\Big|\frac{1}{2\rho|\mathcal{N}_k| + \frac{1}{\eta^{(n)}}}\frac{1}{M_k} \Big(\ell'(\mathbf{x}_{k,M_k}, y_{k,M_k}, \widetilde{\mathbf{w}}_k^{(n-1)}) - \ell'(\mathbf{x}'_{k,M_k}, y'_{k,M_k}, \widetilde{\mathbf{w}}_k^{(n-1)})\Big)\Big|\Big|. \tag{A.3}$$

Since we assumed that $||\ell'(\cdot)||$ is bounded by c_1, the l_2-norm sensitivity is given by

$$\max_{\mathcal{D},\mathcal{D}'} ||\mathbf{w}_{k,\mathcal{D}}^{(n)} - \mathbf{w}_{k,\mathcal{D}'}^{(n)}|| \leqslant \frac{2c_1}{M_k(2\rho|\mathcal{N}_k| + \frac{1}{\eta^{(n)}})}. \quad \square$$

Appendix B. Proof of Theorem I

Proof. For any client k, at any step t, we add to the primal update a white Gaussian noise of variance $\sigma_k^2(n)\boldsymbol{I}_P$, that is equivalent to $\widetilde{\mathbf{w}}_k^{(n)} \sim \mathcal{N}(\mathbf{w}_k^{(n)}, \sigma_k^2(n)\boldsymbol{I}_P)$. Hence, for two neighboring data sets $\mathcal{D}$ and $\mathcal{D}'$, we have $\widetilde{\mathbf{w}}_{k,\mathcal{D}}^{(n)} \sim \mathcal{N}(\mathbf{w}_{k,\mathcal{D}}^{(n)}, \sigma_k^2(n)\boldsymbol{I}_P)$ and $\widetilde{\mathbf{w}}_{k,\mathcal{D}'}^{(n)} \sim \mathcal{N}(\mathbf{w}_{k,\mathcal{D}'}^{(n)}, \sigma_k^2(n)\boldsymbol{I}_P)$.

Therefore, using [20, Lemma 17], which states that $D_\alpha(N(\mu, \sigma^2\boldsymbol{I}_d)||N(\nu, \sigma^2\boldsymbol{I}_d)) = \frac{\alpha||\mu-\nu||_2^2}{2\sigma^2}$, $\forall\alpha \in [1, \infty)$; we obtain, $\forall\alpha \in [1, \infty)$, the following Rényi divergence and simplification using Lemma I:

$$D_\alpha(\widetilde{\mathbf{w}}_{k,\mathcal{D}}^{(n)}||\widetilde{\mathbf{w}}_{k,\mathcal{D}'}^{(n)}) = \frac{\alpha||\mathbf{w}_{k,\mathcal{D}}^{(n)} - \mathbf{w}_{k,\mathcal{D}'}^{(n)}||_2^2}{2\sigma_k^2(n)} \leqslant \frac{\alpha\Delta_k^2(n)}{2\sigma_k^2(n)}. \tag{B.1}$$

We now consider the privacy loss of $\widetilde{\mathbf{w}}_k^{(n)}$ at output λ:

$$\boldsymbol{z}_k^{(n)}(\widetilde{\mathbf{w}}_{k,\mathcal{D}}^{(n)}||\widetilde{\mathbf{w}}_{k,\mathcal{D}'}^{(n)}) = \log\frac{P(\widetilde{\mathbf{w}}_{k,\mathcal{D}}^{(n)} = \lambda)}{P(\widetilde{\mathbf{w}}_{k,\mathcal{D}'}^{(n)} = \lambda)}. \tag{B.2}$$

As $D_\alpha(\cdot) \leqslant \epsilon + \rho\alpha \iff E(e^{(\alpha-1)Z(\cdot)}) \leqslant e^{(\alpha-1)(\epsilon+\rho\alpha)}$, we have:

$$E(e^{(\alpha-1)\boldsymbol{z}_k^{(n)}(\lambda)}) \leqslant e^{(\alpha-1)\frac{\alpha\Delta_k^2(n)}{2\sigma_k^2(n)}}.$$

Thus, the zCDP-NFL algorithm satisfies the dynamic $\varphi_k^{(n)}$-zCDP with $\varphi_k^{(n)} = \frac{\Delta_k^2(n)}{2\sigma_k^2(n)}$. □

Appendix C. Proof of Corollary

Proof. Using [20, Lemma 7] and Theorem I, each client k of the network has zCDP with φ parameter $\sum_{0<n<T}\varphi_k^{(n)}$, T being the final iteration index.
Since $\varphi_k^{(n+1)} = \varphi_k^{(n)}/\tau$, we have

$$\sum_{0<n<T}\varphi_k^{(n)} = \varphi_k^{(1)}\frac{1-\tau^T}{\tau^{T-1}-\tau^T}.$$

Using [20, Prop. 3], zCDP-NFL provides, $\forall \delta \in (0,1)$, each client k with (ϵ_k, δ)-DP, where $\epsilon_k = \varphi_k^{(1)} \frac{1-\tau^T}{\tau^{T-1}-\tau^T} + 2\sqrt{\varphi_k^{(1)} \frac{1-\tau^T}{\tau^{T-1}-\tau^T} log\frac{1}{\delta}}$. Thus, the total privacy of the algorithm can be given in the DP metric with parameters (ϵ, δ), $\forall \delta \in (0,1)$, $\epsilon = \max_{k\in\mathcal{C}} \epsilon_k$. □

Appendix D. Proof of Lemma II

Proof. Using the convexity of $f(\cdot)$ we have:

$f(\widetilde{\mathfrak{w}}^{(n)}) - f(\mathfrak{w}^*) \leqslant \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, f'(\widetilde{\mathfrak{w}}^{(n)})\rangle$. And since $\hat{f}'(\widetilde{\mathfrak{w}}^{(n+1)}) = 2\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}) + f'(\widetilde{\mathfrak{w}}^{(n)})$, we have $f'(\widetilde{\mathfrak{w}}^{(n)}) = \hat{f}'(\widetilde{\mathfrak{w}}^{(n+1)}) - 2\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})$.

Combining both equations we obtain:

$$f(\widetilde{\mathfrak{w}}^{(n)}) - f(\mathfrak{w}^*) \leqslant \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \hat{f}'(\widetilde{\mathfrak{w}}^{(n+1)}) - 2\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})\rangle. \tag{D.1}$$

Employing (17) in (D.1), followed by some algebraic manipulations, yields

$$\begin{aligned}
&\frac{f(\widetilde{\mathfrak{w}}^{(n)}) - f(\mathfrak{w}^*)}{\rho} + \langle \widetilde{\mathfrak{w}}^{(n)}, 2\boldsymbol{Q}\boldsymbol{r}\rangle \\
&\leqslant \langle \widetilde{\mathfrak{w}}^{(n)}, 2\boldsymbol{Q}\boldsymbol{r}\rangle + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, 2\boldsymbol{M}\boldsymbol{\xi}^{(n+1)} - 2\boldsymbol{Q}\boldsymbol{r}^{(n+1)} \\
&\quad - \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}) - \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})\rangle, \\
&\leqslant \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, 2\boldsymbol{Q}(\boldsymbol{r} - \boldsymbol{r}^{(n+1)}) + \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^{(n+1)}) \\
&\quad + \boldsymbol{L}_-(\widetilde{\mathfrak{w}}^{(n+1)} - \mathfrak{w}^{(n+1)}) - \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)})\rangle.
\end{aligned} \tag{D.2}$$

It follows that

$$\begin{aligned}
||\boldsymbol{q}^{(n)} - \boldsymbol{q}^*||_{\boldsymbol{G}}^2 &= \left\langle \begin{pmatrix} \boldsymbol{r}^{(n)} - \boldsymbol{r} \\ \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^* \end{pmatrix}, \begin{pmatrix} \rho(\boldsymbol{r}^{(n)} - \boldsymbol{r}) \\ \frac{\rho \boldsymbol{L}_+}{2}(\widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*) \end{pmatrix} \right\rangle \\
&= \rho||\boldsymbol{r}^{(n)} - \boldsymbol{r}||_2^2 + ||\widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*||_{\frac{\rho \boldsymbol{L}_+}{2}}^2.
\end{aligned}$$

In particular, we obtain the equality:

$$\begin{aligned}
&\frac{1}{\rho}(||\boldsymbol{q}^{(n-1)}-\boldsymbol{q}^*||_{\boldsymbol{G}}^2-||\boldsymbol{q}^{(n)}-\boldsymbol{q}^*||_{\boldsymbol{G}}^2-||\boldsymbol{q}^{(n)}-\boldsymbol{q}^{(n-1)}||_{\boldsymbol{G}}^2)\\
&=\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*,2\boldsymbol{Q}(\boldsymbol{r}-\boldsymbol{r}^{(n)})\rangle+\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*,\boldsymbol{L}_+(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n-1)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})\rangle. \qquad \text{(D.3)}
\end{aligned}$$

Which we use to reformulate the second term of (D.2), after which it is combined with (D.3). Using the fact that $\boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)}=\boldsymbol{r}^{(n+1)}-\boldsymbol{r}^{(n)}$, we obtain:

$$\begin{aligned}
&\frac{f(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})-f(\boldsymbol{\mathfrak{w}}^*)}{\rho}+\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)},2\boldsymbol{Q}\boldsymbol{r}\rangle\leqslant\frac{1}{\rho}(||\boldsymbol{q}^{(n-1)}-\boldsymbol{q}^*||_{\boldsymbol{G}}^2-||\boldsymbol{q}^{(n)}-\boldsymbol{q}^*||_{\boldsymbol{G}}^2)\\
&-\frac{\Phi_{\min}(\boldsymbol{L}_-)}{2}||\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*||_2^2-||\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n-1)}||_{\frac{\boldsymbol{L}_+}{2}}^2-2\langle\boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)},\boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)}\rangle\\
&+\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*,\boldsymbol{L}_+(2\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n-1)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)})\rangle\\
&+||\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*||_2||\boldsymbol{L}_+\boldsymbol{\xi}^{(n+1)}||_2+||\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*||_2||\boldsymbol{L}_-\boldsymbol{\xi}^{(n+1)}||_2\\
&+\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*,\frac{2}{\rho}\boldsymbol{D}^{(n+1)}\otimes\boldsymbol{I}_P(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)})\rangle
\end{aligned}$$

Using the inequality $||\boldsymbol{a}||||\boldsymbol{b}||\leqslant m||\boldsymbol{a}||+\frac{1}{m}||\boldsymbol{b}||$ for $m>0$ with $m=\Phi_{\min}(\boldsymbol{L}_-)$, it can be established that

$$\begin{aligned}
&\frac{f(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)})-f(\boldsymbol{\mathfrak{w}}^*)}{\rho}+\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)},2\boldsymbol{Q}\boldsymbol{r}\rangle\leqslant\frac{1}{\rho}(||\boldsymbol{q}^{(n-1)}-\boldsymbol{q}^*||_{\boldsymbol{G}}^2-||\boldsymbol{q}^{(n)}-\boldsymbol{q}^*||_{\boldsymbol{G}}^2)\\
&-2\langle\boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)},\boldsymbol{Q}\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)}\rangle-||\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n-1)}||_{\frac{\boldsymbol{L}_+}{2}}^2\\
&+\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*,\boldsymbol{L}_+(2\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n-1)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)})\rangle\\
&+\frac{4(\Phi_{\max}(\boldsymbol{L}_-)^2+\Phi_{\max}(\boldsymbol{L}_+)^2)}{\Phi_{\min}(\boldsymbol{L}_-)}||\boldsymbol{\xi}^{(n+1)}||_2^2\\
&+\langle\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\boldsymbol{\mathfrak{w}}^*,\frac{2}{\rho}\boldsymbol{D}^{(n+1)}\otimes\boldsymbol{I}_P(\widetilde{\boldsymbol{\mathfrak{w}}}^{(n)}-\widetilde{\boldsymbol{\mathfrak{w}}}^{(n+1)})\rangle. \qquad \square
\end{aligned}$$

Appendix E. Proof of Theorem II

Proof. We first take the sum of the result of Lemma II from $n = 1$ to $n = T$ to obtain a bound given by

$$\begin{aligned}
&\frac{1}{\rho}(\sum_{n=1}^{T} f(\widetilde{\mathfrak{w}}^{(n)}) - f(\mathfrak{w}^*)) + \langle 2\boldsymbol{r}, \sum_{n=1}^{T} \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}\rangle \qquad \text{(E.1)}\\
&\leqslant \sum_{n=1}^{T}\Big(\frac{4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)}{\Phi_{\min}(\boldsymbol{L}_-)}||\boldsymbol{\xi}^{(n+1)}||_2^2 - 2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle\\
&\quad - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}} + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle\\
&\quad + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle\Big) + \frac{1}{\rho}||\boldsymbol{q}^{(0)} - \boldsymbol{q}^*||^2_{\boldsymbol{G}}.
\end{aligned}$$

After setting $r = 0$ and performing some algebraic manipulations, we obtain:

$$\begin{aligned}
&\frac{1}{\rho}(\sum_{n=1}^{T} f(\widetilde{\mathfrak{w}}^{(n+1)}) - f(\mathfrak{w}^*)) \leqslant \sum_{n=1}^{T}\Big(\frac{4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)}{\Phi_{\min}(\boldsymbol{L}_-)}||\boldsymbol{\xi}^{(n+1)}||_2^2\\
&\quad - 2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}}\\
&\quad + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle\\
&\quad - \langle \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle + ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+}\Big)\\
&\quad + \langle \widetilde{\mathfrak{w}}^{(1)}, \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(1)} - \widetilde{\mathfrak{w}}^{(0)})\rangle + ||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(0)}||_2^2 + ||\widetilde{\mathfrak{w}}^{(0)} - \mathfrak{w}^*||^2_{\frac{\boldsymbol{L}_-}{2}}. \qquad \text{(E.2)}
\end{aligned}$$

Using Jensen's inequality on the expectation of (E.2), we obtain:

$$\begin{aligned}
&\mathbb{E}[f(\hat{\mathfrak{w}}^{(T)}) - f(\mathfrak{w}^*)] \leqslant \frac{\rho}{T}\sum_{n=1}^{T}\Big(-2\langle \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n)}, \boldsymbol{Q}\widetilde{\mathfrak{w}}^{(n+1)}\rangle - ||\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)}||^2_{\frac{\boldsymbol{L}_+}{2}}\\
&\quad + \langle \widetilde{\mathfrak{w}}^{(n)} - \mathfrak{w}^*, \frac{2}{\rho}\boldsymbol{D}^{(n+1)} \otimes \boldsymbol{I}_P(\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle\\
&\quad - \langle \mathfrak{w}^*, \boldsymbol{L}_+(2\widetilde{\mathfrak{w}}^{(n)} - \widetilde{\mathfrak{w}}^{(n-1)} - \widetilde{\mathfrak{w}}^{(n+1)})\rangle\\
&\quad + ||\widetilde{\mathfrak{w}}^{(n+1)} - \widetilde{\mathfrak{w}}^{(n)}||^2_{\boldsymbol{L}_+}\Big) + \frac{\langle \widetilde{\mathfrak{w}}^{(1)}, \boldsymbol{L}_+(\widetilde{\mathfrak{w}}^{(1)} - \widetilde{\mathfrak{w}}^{(0)})\rangle}{T} + \frac{\rho||\boldsymbol{Q}\widetilde{\mathfrak{w}}^{(0)}||_2^2}{T}\\
&\quad + \frac{1}{T}\frac{\rho P 4(\Phi_{\max}(\boldsymbol{L}_-)^2 + \Phi_{\max}(\boldsymbol{L}_+)^2)\sum_{k=1}^{K}\sigma_k^{2(0)}}{\Phi_{\min}(\boldsymbol{L}_-)(1-\tau)} + \frac{\rho||\widetilde{\mathfrak{w}}^{(0)} - \mathfrak{w}^*||^2_{\frac{\boldsymbol{L}_-}{2}}}{T}. \qquad \square
\end{aligned}$$

References

[1] J. Konečnỳ, H. B. McMahan, D. Ramage, P. Richtárik, Federated optimization: distributed machine learning for on-device intelligence, arXiv preprint arXiv:1610.02527 (Oct. 2016).

[2] T. Li, A. K. Sahu, A. Talwalkar, V. Smith, Federated learning: challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.

[3] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Elsevier Knowledge-Based Syst. 216 (2021) 106775.

[4] S. Boll, J. Meyer, Health-X dataLOFT: A Sovereign Federated Cloud for Personalized Health Care Services, IEEE MultiMedia 29 (1) (2022) 136–140.

[5] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, Z. Han, Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions, IEEE Wireless Commun. 28 (2) (2021) 40–47.

[6] A. S. Bedi, A. Koppel, K. Rajawat, Asynchronous online learning in multi-agent systems with proximity constraints, IEEE Trans. Signal Inf. Process. over Networks 5 (3) (2019) 479–494.

[7] L. Li, Y. Fan, M. Tse, K.-Y. Lin, A review of applications in federated learning, Elsevier Comput. & Indus. Eng. 149 (2020) 106854.

[8] Y. SarcheshmehPour, Y. Tian, L. Zhang, A. Jung, Networked federated learning, arXiv preprint arXiv:2105.12769 (2021).

[9] A. Lalitha, S. Shekhar, T. Javidi, F. Koushanfar, Fully decentralized federated learning, in: NeurIPS Third workshop Bayesian deep learning, Vol. 2, 2018.

[10] C. Li, G. Li, P. K. Varshney, Decentralized federated learning via mutual knowledge transfer, in: IEEE Internet Things J., Vol. 2, 2021, pp. 1136–1147.

[11] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, C. Wachinger, Braintorrent: A peer-to-peer environment for decentralized federated learning, arXiv preprint arXiv:1905.06731 (May 2019).

[12] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, A. H. Celdrán, Decentralized federated learning: Fundamentals, state-of-the-art, frameworks, trends, and challenges, arXiv preprint arXiv:2211.08413 (2022).

[13] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, H. V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3454–3469.

[14] C. Dwork, F. McSherry, K. Nissim, A. Smith, Calibrating Noise to Sensitivity in Private Data Analysis, in: Theory of Cryptography, Springer Berlin Heidelberg, 2006, pp. 265–284.

[15] C. Dwork, A. Roth, The Algorithmic Foundations of Differential Privacy, Found. Trends Theor. Comput. Sci. 9 (2014) 211–407.

[16] J. Cortés, G. E. Dullerud, S. Han, J. Le Ny, S. Mitra, G. J. Pappas, Differential privacy in control and network systems, in: IEEE 55th Conf. Decision Control, 2016, pp. 4252–4272.

[17] C. Dwork, G. N. Rothblum, Concentrated differential privacy, arXiv preprint arXiv:1603.01887 (2016).

[18] B. Jayaraman, D. Evans, Evaluating differentially private machine learning in practice, in: 28th USENIX Secur. Symp., 2019, pp. 1895–1912.

[19] C. Dwork, N. Kohli, D. Mulligan, Differential Privacy in Practice: Expose your Epsilons!, J. Privacy and Confidentiality 9 (2) (Oct. 2019).

[20] M. Bun, T. Steinke, Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds, in: Theory of Cryptography Conf., Springer, 2016, pp. 635–658.

[21] T. Zhang, Q. Zhu, A Dual Perturbation Approach for Differential Private ADMM-Based Distributed Empirical Risk Minimization, Proc. ACM Workshop Artif. Intell. Secur. (2016) 129–137.

[22] I. Mironov, Rényi differential privacy, in: IEEE Comput. Secur. Found. Symp., IEEE, 2017, pp. 263–275.

[23] A. Nedic, A. Ozdaglar, Distributed subgradient methods for multi-agent optimization, IEEE Trans. Autom. Control 54 (1) (2009) 48–61.

[24] J. Ding, Y. Gong, M. Pan, Z. Han, Optimal differentially private ADMM for distributed machine learning, IEEE Int. Conf. Big Data (2019) 1302–1311.

[25] T. Zhang, Q. Zhu, Dynamic differential privacy for ADMM-based distributed classification learning, IEEE Trans. Inf. Forens. Security 12 (1) (2017) 172–187.

[26] T. Zhang, Q. Zhu, Distributed privacy-preserving collaborative intrusion detection systems for VANETs, IEEE Trans. Signal Inf. Process. Netw. 4 (1) (2018) 148–161.

[27] Communication efficient privacy-preserving distributed optimization using adaptive differential quantization, Elsevier Signal Processing 194 (2022) 108456.

[28] J. Ding, X. Zhang, M. Chen, K. Xue, C. Zhang, M. Pan, Differentially Private Robust ADMM for Distributed Machine Learning, in: 2019 IEEE Int. Conf. Big Data, 2019, pp. 1302–1311.

[29] C. Chen, J. Lee, Rényi Differentially Private ADMM for Non-Smooth Regularized Optimization, Proc. Tenth ACM Conf. Data and Appl. Secur. and Privacy (2020) 319–328.

[30] J. E. Gentle, Least absolute values estimation: An introduction, Commun. Statist.-Simul. Comput. 6 (4) (1977) 313–328.

[31] V. Roth, The Generalized LASSO, IEEE Trans. Neural Netw. 15 (1) (2004) 16–28.

[32] N. K. Dhingra, S. Z. Khong, M. R. Jovanović, The proximal augmented Lagrangian method for nonsmooth composite optimization, IEEE Trans. Autom. Control 64 (7) (2018) 2861–2868.

[33] H. Zhu, X. Zhang, D. Chu, L.-Z. Liao, Nonconvex and nonsmooth optimization with generalized orthogonality constraints: An approximate augmented Lagrangian method, J. Scientific Comput. 72 (1) (2017) 331–372.

[34] Y. Wang, W. Yin, J. Zeng, Global convergence of ADMM in nonconvex nonsmooth optimization, J. Scientific Comput. 78 (1) (2019) 29–63.

[35] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, Y. Gong, DP-ADMM: ADMM-based distributed learning with differential privacy, IEEE Trans. Inf. Forens. Security 15 (2019) 1002–1012.

[36] Y. Hu, P. Liu, L. Kong, D. Niu, Learning privately over distributed features: An ADMM sharing approach, arXiv preprint arXiv:1907.07735 (Jul. 2019).

[37] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, Splitting Methods in Communication, Imaging, Science, and Engineering, in: Scientific Computation, Springer Int. Publishing, 2016, pp. 461–497.

[38] C. Gratton, N. K. D. Venkategowda, R. Arablouei, S. Werner, Distributed Ridge Regression with Feature Partitioning, Proc. Asilomar Conf. Signals Syst. Comput. (2018) 1423–1427.

[39] A. Nemirovski, A. Juditsky, G. Lan, A. Shapiro, Robust stochastic approximation approach to stochastic programming, SIAM J. optim. 19 (4) (2009) 1574–1609.

[40] Z. Huang, S. Mitra, N. Vaidya, Differentially private distributed optimization, in: Proc. 2015 Int. Conf. Distrib. Comput. Netw., 2015, pp. 1–10.

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

[42] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, P. K. Varshney, Robust decentralized learning using ADMM with unreliable agents, IEEE Trans. Signal Process. (Jun. 2022).

[43] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (1) (2010) 1–122.

[44] M. Grant, S. Boyd, CVX: Matlab software for disciplined convex programming, version 2.1, `http://cvxr.com/cvx` (Mar. 2014).

Appendix C

Publication 3

P3 F. Gauthier, V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Resource-aware asynchronous online federated learning for nonlinear regression", in *Proceedings of IEEE International Conference on Communications*, May, 2022.

Appendix D

Publication 4

P4 F. Gauthier, V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Asynchronous Online Federated Learning with Reduced Communication Requirements", published in *IEEE Internet of Things Journal*.

The following is the accepted manuscript.

Asynchronous Online Federated Learning with Reduced Communication Requirements

Francois Gauthier, *Member*, *IEEE*, Vinay Chakravarthi Gogineni, *Senior Member*, *IEEE*, Stefan Werner, *Fellow*, *IEEE*, Yih-Fang Huang, *Life Fellow*, *IEEE*, Anthony Kuh, *Fellow*, *IEEE*

***Abstract*—Online federated learning (FL) enables geographically distributed devices to learn a global shared model from locally available streaming data. Most online FL literature considers a best-case scenario regarding the participating clients and the communication channels. However, these assumptions are often not met in real-world applications. Asynchronous settings can reflect a more realistic environment, such as heterogeneous client participation due to available computational power and battery constraints, as well as delays caused by communication channels or straggler devices. Further, in most applications, energy efficiency must be taken into consideration. Using the principles of partial-sharing-based communications, we propose a communication-efficient asynchronous online federated learning (PAO-Fed) strategy. By reducing the communication load of the participants, the proposed method renders participation more accessible and efficient. In addition, the proposed aggregation mechanism accounts for random participation, handles delayed updates and mitigates their effect on accuracy. We study the first and second-order convergence of the proposed PAO-Fed method and obtain an expression for its steady-state mean square deviation. Finally, we conduct comprehensive simulations to study the performance of the proposed method on both synthetic and real-life datasets. The simulations reveal that in asynchronous settings, the proposed PAO-Fed is able to achieve the same convergence properties as that of the online federated stochastic gradient while reducing the communication by 98 percent.**

***Index Terms*—Asynchronous behavior, communication efficiency, online federated learning, partial-sharing-based communications, nonlinear regression.**

This work was supported by the Research Council of Norway.

Francois Gauthier and Stefan Werner are with the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway (e-mail: {francois.gauthier, stefan.werner}@ntnu.no). Stefan Werner is also with the Department of Information and Communications Engineering, Aalto University, 00076, Finland.

Vinay Chakravarthi Gogineni is with the SDU Applied AI and Data Science, the Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Denmark (e-mail: vigo@mmmi.sdu.dk). Previously, he was with the Norwegian University of Science and Technology, Trondheim, Norway.

Yih-Fang Huang is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: huang@nd.edu).

Anthony Kuh is with the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, HI 96822 USA (e-mail: kuh@hawaii.edu). Anthony Kuh acknowledges support in part by NSF Grant 2142987

The current paper significantly advances and expands upon the conference precursor [1], offering a more comprehensive treatment of the subject matter with a greatly expanded scope, analysis, and simulation examples. Specifically, it comprehensively illustrates and further develops the behavior of partial-sharing-based communication, presents rigorous second-order mathematical analysis, and substantially extends the numerical simulations. The latter includes hyperparameter selection analysis, communication reduction comparison, simulations on a real-world IoT dataset, and comparisons with existing communication-efficient algorithms.

I. Introduction

A myriad of intelligent devices, such as smartphones, smartwatches, and smart home appliances, are becoming an integral part of our daily lives, and an enormous amount of data is available on those devices. Unfortunately, this data is primarily unused, and we need to develop tools that can process this data to extract information that can improve our daily lives while, at the same time, ensuring our privacy. Federated learning (FL) [2] provides an adaptive large-scale collaborative learning framework suitable for this task. In FL, a server aggregates information received from distributed devices referred to as clients to train a global shared model; the clients do not share any private data with the server, only their local model parameters or gradients learned from this data [2], [3]. When data becomes progressively available to clients, it is possible to perform decentralized learning in real-time (implementing, e.g., online FL [4]) for applications that include environmental monitoring and condition monitoring using sensor networks [5], internet-of-medical-things (IoMT) based healthcare applications [6] (e.g., cardio rhythm monitoring), and autonomous vehicles [7]. In online FL, the server aggregates the local models learned on the streaming data of the clients [8]. However, in many applications, the participating clients might have heterogeneous energy supply and limited communication capacity that can be intermittently unavailable or subject to failure. Therefore, such edge devices cannot participate in typical federated learning implementations.

In most real-world implementations of FL, it is essential to consider statistical heterogeneity, system heterogeneity, and imperfect communication channels between clients and the server. Statistical heterogeneity implies that data are imbalanced and not independent and identically distributed (non-i.i.d.) [9] across devices, while system heterogeneity refers to their various computational and communication capacities. Finally, imperfect communication channels cause delays in the exchanged messages. Although many FL approaches can handle statistical heterogeneity, there is relatively little research addressing the remaining complications above. In particular, existing FL methods commonly assume a best-case scenario concerning the client availability and performance as well as perfect channel conditions [2], [10]–[19]. However, several additional aspects need attention for efficient FL in a realistic setting. First, clients cannot be expected to have the same participation frequency, e.g., due to diverse resource constraints, channel availability, or concurrent solicitations [20]–[23]. Furthermore, clients may become unavailable for a certain period during the learning process, i.e., some clients

are malfunctioning or not reachable by the server [20], [21]. In addition, physical constraints such as distance or overload introduce delays in the communication between the clients and the server, making their contribution arrive later than expected [21]–[24]. These constraints, frequently occurring in practice, impair the efficiency of FL and complicate the design of methods tailored for asynchronous settings [20]–[26].

Energy efficiency is an essential aspect of distributed machine learning algorithms and one of the original motivations for FL [27]. The communication of high-dimensional models is energy-onerous for distributed devices. For this reason, it is crucial to cut the communication cost for clients [15], [19]. Further, such reduction can facilitate more frequent participation of resource-constrained devices, or stragglers, in the learning process. In addition, in asynchronous settings, where power and communication are restricted, ensuring communication efficiency reduces the risk of bottlenecks in the communication channels or power-related shutdown of clients due to excessive resource usage.

We can find a considerable amount of research in the literature on communication-efficient FL [14]–[19], [28]–[31] and asynchronous FL [20]–[26], [32]–[40]; however, only a few works consider both aspects within the same framework. The classical federated averaging (FedAvg) [19], developed for ideal conditions, reduces the communication cost by selecting a subset of the clients to participate at each iteration. In a perfect setting, this allows clients to space out their participation while maintaining a consistent participation rate. In asynchronous settings, however, clients may already participate sporadically because of their inherent limitations. Hence, subsampling comes with an increased risk of discarding valuable information. The work of [35] proposes a smart selection system to address this issue, but this is associated with an additional computational burden on the server, and only lessens the information loss associated with client scheduling. The works in [24], [29] reduce communication in uplink via compressed client updates. Aside from the accuracy penalty associated with the sparsification and projection used, the resulting extra computational burden on the clients of these non-trivial operations is not appealing for resource-constrained clients. Moreover, the work in [29] did not consider asynchronous settings. Although the work in [24] considers various participation frequencies for the clients, it assumes they are constant throughout the learning process. The works in [25], [40] reduce the communication load of clients in asynchronous settings; however, they are specific to neural networks and lack mathematical analysis. In addition, the considered asynchronous settings do not include communication delays. We note that structure and sketch update methods suffer the same accuracy cost and additional computational burden as compressed updates; and in all three, the simultaneous unpacking of all the received updates at the server can form a computational bottleneck. Another option explored recently for distributed learning is the partial-sharing of model parameters [41]. The partial-sharing-based online FL (PSO-Fed) algorithm [28] features reduced communications in FL, but only in ideal settings.

This paper proposes a partial-sharing-based asynchronous online federated learning (PAO-Fed) algorithm for nonlinear regression in asynchronous settings. The proposed approach reduces communication significantly while retaining fast convergence. In order to perform nonlinear regression, we use random Fourier feature space (RFF) [42], [43], where inner products in a fixed-dimensional space approximate the nonlinear relationship between the input and output data. Consequently, given the constant communication and computational load, RFF is more suitable for decentralized learning than traditional dictionary-based solutions whose model order depends on the sample size. In addition, RFF presents the advantage of being resilient to model change during the learning process, which is key in online FL. Further, we implement partial-sharing-based communications to reduce the communication load of the algorithm. Compared to the other available methods, partial-sharing does not incur an additional computational load and only transfers a fraction of the model parameters between clients and the server. This allows clients to participate more frequently while maintaining minimal communication without additional computational burden. The proposed aggregation mechanism handles delayed updates and calibrates their contribution to the global shared model. We provide first- and second-order convergence analyses of the PAO-Fed algorithm in a setting where client participation is random, and communication links suffer delays. Finally, we conducted simulation studies using synthetic and real-life data to examine and compare the proposed algorithm with existing methods.

The paper is organized as follows. Section II introduces FL for nonlinear regression as well as partial-sharing-based communications. Section III defines the considered asynchronous settings and introduces the proposed method. Section IV provides the first and second-order convergence analysis of the PAO-Fed algorithm. Section V presents numerical results for the proposed method and compares it with existing ones. Finally, Section VI concludes the paper.

II. Preliminaries and Problem Formulation

This section presents the nonlinear regression problem in the context of FL. Further, a brief overview of the most closely related existing algorithms is proposed. Finally, the behavior of partial-sharing-based communications is presented.

A. Online Federated Learning for Nonlinear Regression

We consider a federated network where a server is connected to a set $\mathcal{K}$ of $|\mathcal{K}| = K$ geographically distributed devices, referred to as clients. In the online FL setting [4], used when real-time computation is desirable, the entire dataset of a client is not immediately available. Instead, it is made available to the client progressively throughout the learning process. We denote the continuous streaming data appearing at client $k \in \mathcal{K}$ at iteration n by $\mathbf{x}_{k,n} \in \mathbb{R}^L$, the corresponding output $y_{k,n}$ is given by:

$$y_{k,n} = f(\mathbf{x}_{k,n}) + \eta_{k,n}, \tag{1}$$

where $f(\cdot) : \mathbb{R}^L \longrightarrow \mathbb{R}$ is a nonlinear model and $\eta_{k,n}$ is the observation noise. The objective is that the server and clients

learn a global shared nonlinear model from the data available at each client, without this data being shared amongst clients or with the server. To this aim, the clients periodically share with the server their local model, learned from local data, and the server shares its global model with the clients.

Several adaptive methods can be used to handle nonlinear model estimation problems, e.g., [42]–[45]. The conventional kernel least-mean-square (KLMS) algorithm [44] is one of the most popular choices but suffers from a growing dimensionality problem, leading to prohibitive computation and communication requirements. Coherence-check-based methods [45] sparsify the original dictionary by selecting the regressors using a coherence measure. Although feasible, this method is not attractive for online FL, especially in asynchronous settings, since it requires that each new dictionary element be made available throughout the network, inducing a significant communication overhead, especially if the underlying model changes. The random Fourier feature (RFF) space method [42], [43] approximates the kernel function evaluation by projecting the model into a pre-selected fixed-dimensional space. The selected RFF space does not change throughout the computation, and, given that the chosen dimension is large enough, the obtained linearizations can be as precise as desired. Therefore, we use RFF-based KLMS for the nonlinear regression task, as it is data-independent, resilient to model change, and does not require extra communication overhead, unlike conventional or coherence-check-based KLMS.

In the following, we approximate the nonlinear model by projecting it on a D-dimensional RFF-space, in which the function $f(\cdot)$ is approximated by the linear model $\mathbf{w}^*$. To estimate the global shared model using the local streaming data, we solve the following problem:

$$\min_{\mathbf{w}} \mathcal{J}(\mathbf{w}), \tag{2}$$

where $\mathcal{J}(\mathbf{w})$ is given by:

$$\begin{aligned} \mathcal{J}(\mathbf{w}) &= \frac{1}{K}\sum_{k\in\mathcal{K}} \mathcal{J}_k(\mathbf{w}) \\ \mathcal{J}_k(\mathbf{w}) &= \mathbb{E}[|y_{k,n} - \mathbf{w}^\mathsf{T}\mathbf{z}_{k,n}|^2], \end{aligned} \tag{3}$$

and $\mathbf{z}_{k,n}$ is the mapping of $\mathbf{x}_{k,n}$ into the D-dimentional RFF-space.

B. Existing Algorithms

The Online-Fed algorithm, an online FL version of the conventional FedAvg algorithm [19] solves the above estimation problem as follows. At each iteration, n, the server selects a subset of the clients $\mathcal{K}_n \subseteq \mathcal{K}$ to participate in the learning task and shares the global shared model $\mathbf{w}_n$ with them. Then the selected clients in $\mathcal{K}_n$ perform the local learning process on their local estimates $\mathbf{w}_{k,n}$ as

$$\mathbf{w}_{k,n+1} = \mathbf{w}_n + \mu\mathbf{z}_{k,n}e_{k,n}, \tag{4}$$

where μ is the learning rate and $e_{k,n}$ is the *a priori* error of the global model on the local data given by

$$e_{k,n} = y_{k,n} - \mathbf{w}_n^\mathsf{T}\mathbf{z}_{k,n}. \tag{5}$$

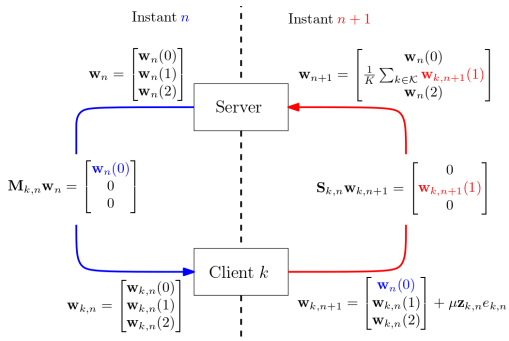


Fig. 1: Partial sharing in a simple scenario.

The clients then share their updated models with the server, which aggregates them as

$$\mathbf{w}_{n+1} = \frac{1}{|\mathcal{K}_n|}\sum_{k\in\mathcal{K}_n} \mathbf{w}_{k,n+1}, \tag{6}$$

where $|\mathcal{K}_n|$ denotes the cardinality of $\mathcal{K}_n$. In the particular case where $\forall n, \mathcal{K}_n = \mathcal{K}$, i.e., all the clients participate at each iteration, we denote the algorithm Online-FedSGD.

The PSO-Fed algorithm proposed in [28] uses partial-sharing-based communications to reduce further the communication load of the Online-Fed algorithm. Additionally, PSO-Fed allows clients who are not participating in the current iteration to perform local learning on their new data. By doing so, this algorithm drastically reduces communication without compromising the convergence speed.

C. Partial-sharing-based Communications

In partial-sharing-based communications, as defined in [41], the server and the clients exchange only a portion of their respective models instead of the entire model. The portion is extracted prior to communication by multiplication with a diagonal selection matrix with main diagonal elements being either 0 or 1, where the locations of the latter specify the model parameters to share. This operation is computationally trivial and, therefore, does not induce delay on the communication, unlike compressed update methods, e.g., [24], [29]. Here, m denotes the number of nonzero elements in the selection matrices; this is the number of model parameters shared at each iteration. The selection matrix $\mathbf{M}_{k,n}$ is used for server-to-client communication at time n and the selection matrix $\mathbf{S}_{k,n}$ for client k's response, as can be seen on Fig. 1 where the simple case where $m = D/3$ is illustrated.

The usual aggregation step in (6) cannot be used with partial-sharing-based communications, and needs to be adapted. The expression of $\mathbf{w}_{n+1}$ in Fig. 1 is the aggregation step for coordinated partial-sharing in perfect settings. Coordinated partial-sharing is the special case where all clients send the same portion of the model at a given iteration.

For the clients to participate in the learning of the whole model, and to ensure consistency across models, it is necessary that the selection matrices evolve. To this aim, we set:

$$\text{diag}(\mathbf{M}_{k,n+1}) = \text{circshift}(\text{diag}(\mathbf{M}_{k,n}), m) \tag{7}$$

$$\mathbf{S}_{k,n} = \mathbf{M}_{k,n+1} \tag{8}$$

where circshift denotes a circular shift operator. $\mathbf{S}_{k,n}$ is set to be equal to $\mathbf{M}_{k,n+1}$ rather than $\mathbf{M}_{k,n}$ in order to share a portion of the client's model further refined by the local learning process. As can be seen in Fig. 1, $\mathbf{w}_{k,n+1}(0)$ contains information from a single local learning step of client k, while $\mathbf{w}_{k,n+1}(1)$ contains information from three (since it is equal to $\mathbf{w}_{n-2}(1)$ refined thrice by the local learning process). For smaller values of m, the additional information is greater, but the original value of the portion is also older.

D. Motivation

The above-mentioned algorithms offer significant communication savings but do not consider practical network environments and client resources. When performing federated learning in real-world applications, clients may be unavailable for various reasons, message exchanges may be delayed or blocked, and straggler clients may be present. For this reason, it is essential to tailor the developed algorithms to asynchronous settings. Those environments impact the algorithm design and optimization. For instance, we will see that many choices made for the PSO-Fed algorithm in an ideal setting are unsuitable for asynchronous settings.

III. Proposed Method

This section presents the proposed communication-efficient Partial-sharing-based Asynchronous Online Federated Learning (PAO-Fed) algorithm and the asynchronous settings for which it is developed.

A. Asynchronous Settings

The following features are necessary for an online FL method to operate successfully in realistic environments.

- The capability to handle non-IID and unevenly distributed data.
- The capability to handle heterogeneous, time-varying, and unpredictable client participation, including possible downtimes. In most real-world applications, the computation and communication capacity of a specific task are heterogeneous and time-varying. In addition, clients are unreliable as they may experience many issues (low battery, software failure, physical threat, etc.). Moreover, when dealing with many clients, an infrequently occurring failure is likely experienced at least once. Lastly, it is unlikely for the server to know in advance when a client will be unavailable or suffer a failure, so even the most reliable clients may suffer downtimes.
- The capability to weigh the importance of delayed messages. Model parameters with the same timestamp may arrive at different instants at the server. In practice, communication channels are unreliable, and although most messages arrive within a short window, some may take longer, especially when the communication channels are strained. In addition, straggler clients may not be able to complete the learning task in the given time frame, and although their update may not be delayed, it will arrive late at the server. Therefore, the developed method must be robust to a delay spread in the received parameters.
- The capability to reduce the likelihood of straggler-like behavior. Resource-constrained devices may induce latency or run out of power, resulting in reduced information sharing. It is, therefore, not sufficient to consider stragglers-like behavior [22]; it is preferable to improve their operational environments, e.g., by reducing their computation and communication load.

The first step to address those challenges is to model the presented behaviors properly. To this aim, the clients' participation is modeled by participation probabilities. At an iteration n, the Bernoulli trial on the probability $p_{k,n}$ dictates if client k is able to participate. The use of probabilities for participation allows the model to address all the behaviors presented in the second point, unlike the commonly used tier-based model for participation (e.g., [24]), where each tier is expected to behave optimally given a tailored frequency. In fact, heterogeneity and time-dependency are handled by giving clients various evolving probabilities $p_{k,n}$, and unpredictability and downtimes are naturally present when ensuring that all probabilities are lower than one. In addition, any communication sent by a client to the server may be delayed by one or several iterations.

With the proposed model, the limitations of real-world applications and the heterogeneity of the computational power and communication capacity of the available devices are taken into consideration. Those asynchronous settings diminish the potential performance of an FL method, especially in the online setting, where data not shared in time is lost. The proposed method ensures communication efficiency and, in turn, some extend energy efficiency in order, notably, to avoid downward cycles in the asynchronous behavior of the participating clients. For instance, a weaker device may take longer to perform the learning process, struggle to send a long message, and need time to save enough power to participate again. Therefore, performing less computation and exchanging shorter messages will reduce the burden on the clients and the communication channels, making further complications or delays less likely. For this reason, a communication-efficient method tailored for the asynchronous settings can perform above its expectations in a real-life scenario.

B. Delayed Updates

The consequence of the introduced delays is that not all updates sent by clients participating at a given iteration will arrive at the server simultaneously. Precisely, we denote $\mathcal{K}_n$ the set of all the clients who sent an update that arrived at the server at iteration n. This set can be decomposed as:

$$\mathcal{K}_n = \bigcup_{l=0}^{\infty} \mathcal{K}_{n,l} \tag{9}$$

where $\mathcal{K}_{n,l}$ denotes the set of the clients who sent an update at iteration $n-l$ which reached the server at iteration n, the subscript l corresponds to the number of iterations during which the update was delayed. A delayed update will naturally lose value the longer it is delayed, as it becomes outdated. To improve the learning accuracy of the proposed algorithm, we propose a weight-decreasing mechanism that weights down delayed updates. By doing so, we diminish the negative impact of outdated data on the convergence. This mechanism is different from age of update mechanisms found in [46]–[48] where weights are dictated by the amount of data, independently from communication delays. We denote $\alpha_l \in [0,1]$ the weight given to the updates sent by the clients in $\mathcal{K}_{n,l}$. This work only considers potential delays in client-to-server communications. Although delays in server-to-client communications also affect performance, they do not require further modification of the aggregation mechanism. Additionally, such delays are less likely to occur in IoT/CPS applications, where the server is typically a powerful device that broadcasts messages to resource-constrained clients.

C. PAO-Fed

The proposed PAO-Fed algorithm is tailored to the asynchronous settings; notably, its novel aggregation step is designed to handle delayed updates. PAO-Fed makes use of all the available clients at a given iteration. To reduce the amount of communication associated with the learning, it uses partial-sharing-based communications, which is well adapted to the asynchronous settings as it does not lay any additional computational burden on the participating clients. Further, the aggregation step is refined with a weight-decreasing mechanism to diminish the negative impact of delayed updates on convergence. The algorithm is as follows.

During iteration n, the server shares a portion of the global shared model, i.e., $\mathbf{M}_{k,n}\mathbf{w}_n$, to all the available clients. The selection matrix $\mathbf{M}_{k,n}$ dictates which portion of the model is sent to client k. The available client k receives its portion of the global shared model, and uses it to update its local model, the new local model is given by $\mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I}-\mathbf{M}_{k,n})\mathbf{w}_{k,n}$. Afterward, the available client k refines its local model by performing the process of local learning on its newly available data as follows.

$$\mathbf{w}_{k,n+1} = \mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I}-\mathbf{M}_{k,n})\mathbf{w}_{k,n} + \mu\mathbf{z}_{k,n}e_{k,n}, \quad (10)$$

where $e_{k,n}$ is the *a priori* error of the local model on the local data given by:

$$e_{k,n} = y_{k,n} - (\mathbf{M}_{k,n}\mathbf{w}_n + (\mathbf{I}-\mathbf{M}_{k,n})\mathbf{w}_{k,n})^{\mathsf{T}}\mathbf{z}_{k,n}. \quad (11)$$

When a client is unavailable at a given iteration but receives new data and is not malfunctioning, it refines its local model autonomously. For example, this can be a case where a client is well functioning but does not have communication capacity at the time. This local update step, identical to the one used in [28], is performed as

$$\mathbf{w}_{k,n+1} = \mathbf{w}_{k,n} + \mu\mathbf{z}_{k,n}e_{k,n}, \quad (12)$$

where $e_{k,n}$ in that case is given by

$$e_{k,n} = y_{k,n} - \mathbf{w}_{k,n}^{\mathsf{T}}\mathbf{z}_{k,n}. \quad (13)$$

This update is computationally trivial for most devices and does not involve communication. Its purpose is for the client to share better-refined model parameters during the next participation. Naturally, this additional information only reaches the server if the model parameters are not overwritten before being communicated, further motivating the choice of selection matrices made in (8).

After this local update step, all available clients communicate a portion of their updated local models to the server. A client k communicates the portion of the model dictated by the selection matrix $\mathbf{S}_{k,n}$, that is, $\mathbf{S}_{k,n}\mathbf{w}_{k,n+1}$. Those updates may arrive at the present iteration or at a later one if they are delayed.

At the server, we consider the previously introduced set $\mathcal{K}_n$ consisting of the clients whose updates arrive at the current iteration. This set comprises the sets $\mathcal{K}_{n,l}, 0 \leqslant l < \infty$ that consist of the clients whose update was sent at iteration $n-l$ and arrives at the current iteration. The set $\mathcal{K}_{n,0}$ consists of the available clients at the current iteration whose updates have not been delayed. Note that a client may appear twice in the set $\mathcal{K}_n$ if two of its updates arrive at the same iteration. The deviation from the current global model engendered by the updates received from a non-empty set $\mathcal{K}_{n,l}$ is given by

$$\Delta_{n,l} = \frac{1}{|\mathcal{K}_{n,l}|}\sum_{k\in\mathcal{K}_{n,l}} \mathbf{S}_{k,n-l}(\mathbf{w}_{k,n+1-l} - \mathbf{w}_n). \quad (14)$$

If a set $\mathcal{K}_{n,l}$ is empty, we set by convention $\Delta_{n,l} = \mathbf{0}$

The aggregation step of the proposed algorithm uses a weight-decreasing mechanism for delayed updates. A client's participation that has been delayed for l iterations will be given the weight $\alpha_l \in [0,1]$. By convention, we set the weight of the updates that are not delayed to $\alpha_0 = 1$. The resulting aggregation mechanism is given by:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \sum_{l=0}^{\infty}\alpha_l\Delta_{n,l}. \quad (15)$$

When $l > l_{\max}$, the maximum effective delay, the aggregation mechanism discards the corresponding updates by setting $\alpha_l = 0$, $l > l_{\max}$. It is possible to replace ∞ by $l_{\max}$ in (15) without changing the aggregation mechanism. Note that in the eventuality where several updates from clients in $\mathcal{K}_n$ update the same model parameter, only the most recent updates are considered, the selection matrices of the remaining updates are adjusted accordingly prior to computing (15). The resulting algorithm is presented in Algorithm 1.

D. Partial-sharing in Asynchronous Settings

In coordinated partial sharing, all participating clients share the same portion of the model so that the server's model is aggregated from a large number of clients, thus improving accuracy. For this reason, coordinated partial-sharing is used in most algorithms assuming perfect settings. In practice, however, delayed updates partially overwrite the previously

Algorithm 1 PAO-Fed

Initialization: $\mathbf{w}_0$ and $\mathbf{w}_{k,0}, k \in \mathcal{K}$ set to $\mathbf{0}$
Procedure at Local client k
for iteration $n = 1, 2, \ldots, N$ **do**
  if Client k receives new data at time n **then**
    if k is available **then**
      Receive $\mathbf{M}_{k,n}\mathbf{w}_n$ from the server.
      Compute $\mathbf{w}_{k,n+1}$ as in (10).
      Share $\mathbf{S}_{k,n}\mathbf{w}_{k,n+1}$ with the server.
    else
      Update $\mathbf{w}_k$ as in (12).
    end if
  end if
end for
Procedure at Central Server
for iteration $n = 1, 2, \ldots, N$ **do**
  Receive client updates from subset $\mathcal{K}_n \subset \mathcal{K}$.
  Compute $\mathbf{w}_{n+1}$ as in (15).
  Share $\mathbf{M}_{k,n+1}\mathbf{w}_{n+1}$ with the available clients.
end for

aggregated portion, as can be seen in (15), thus negating the added value of coordination.

To tackle this issue, one can either use a weight-decreasing mechanism such as the one presented above or use uncoordinated partial sharing. Besides, uncoordinated partial-sharing is ideal when dealing with underlying model changes, as the server's model uniformly steers towards its new steady-state value, instead of doing so portion by portion as with coordinated partial-sharing.

IV. Convergence Analysis

In this section, we examine the convergence behavior of the proposed PAO-Fed algorithm that uses partial-sharing-based communications and evolves in asynchronous settings such as the ones presented in Section III. We prove mathematically that the proposed PAO-Fed algorithm converges to the exact model in the RFF space and exhibits stable extended mean square displacement under certain general assumptions.

Before proceeding to the analysis, we introduce auxiliary matrices to express an entire iteration of the algorithm in the matrix form. Similar to [49], we define the extended model vector $\mathbf{w}_{e,n}$, local update matrix $\mathbf{A}_{e,n}$, and mapping of the data into the RFF-space $\mathbf{Z}_{e,n}$ as

$$\begin{aligned}\mathbf{w}_{e,n} &= \text{col}\{\mathbf{w}_n, \mathbf{w}_{1,n}, \ldots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n} \ldots, \mathbf{w}_{K,n}, \mathbf{w}_{1,n-1}, \\ &\qquad \ldots, \mathbf{w}_{K,n-1}, \ldots, \mathbf{w}_{1,n-l_{\max}}, \ldots, \mathbf{w}_{K,n-l_{\max}}\}, \\ \mathbf{A}_{e,n} &= \text{blockdiag}\{\mathbf{A}_n, \mathbf{I}_{DK}, \ldots, \mathbf{I}_{DK}\}, \\ \mathbf{Z}_{e,n} &= \text{blockdiag}\{\mathbf{Z}_n, \mathbf{0}_{DK\times K}, \ldots, \mathbf{0}_{DK\times K}\}, \end{aligned} \tag{16}$$

with

$$\mathbf{A}_n = \begin{bmatrix} \mathbf{I} & \mathbf{0}_D & \cdots & \mathbf{0}_D \\ a_{1,n}\mathbf{M}_{1,n} & \mathbf{I} - a_{1,n}\mathbf{M}_{1,n} & & \vdots \\ \vdots & \mathbf{0}_D & \ddots & \mathbf{0}_D \\ a_{K,n}\mathbf{M}_{K,n} & \vdots & & \mathbf{I} - a_{K,n}\mathbf{M}_{K,n} \end{bmatrix},$$

$$\mathbf{Z}_n = \text{blockdiag}\{\mathbf{0}_D, \mathbf{z}_{1,n}, \ldots, \mathbf{z}_{K,n}\}, \tag{17}$$

where $a_{k,n} = 1$ if the client k is available at iteration n and 0 otherwise, col$\{\cdot\}$ and blockdiag$\{\cdot\}$ represent column-wise stacking and block diagonalization operators, respectively. We can now express the extended observation vector $\mathbf{y}_{e,n} = \text{col}\{0, y_{1,n}, y_{2,n}, \ldots, y_{K,n}, \mathbf{0}_{K\times 1}, \ldots, \mathbf{0}_{K\times 1}\}$ as

$$\mathbf{y}_{e,n} = \mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{w}_e^* + \boldsymbol{\eta}_{e,n}, \tag{18}$$

where $\mathbf{w}_e^* = \mathbf{1}_{(K+1)l_{\max}+1} \otimes \mathbf{w}^*$ and the extended observation noise $\boldsymbol{\eta}_{e,n} = \text{col}\{0, \eta_{1,n}, \eta_{2,n}, \ldots, \eta_{K,n}, \mathbf{0}_{K\times 1}, \ldots, \mathbf{0}_{K\times 1}\}$. We then can express the extended estimation error vector as

$$\mathbf{e}_{e,n} = \mathbf{y}_{e,n} - \mathbf{Z}_{e,n}^{\mathsf{T}}\mathbf{A}_{e,n}\mathbf{w}_{e,n}. \tag{19}$$

Therefore, the recursion of the extended model vector $\mathbf{w}_{e,n}$ is given by

$$\mathbf{w}_{e,n+1} = \mathbf{B}_{e,n}(\mathbf{A}_{e,n}\mathbf{w}_{e,n} + \mu\mathbf{Z}_{e,n}\mathbf{e}_{e,n}), \tag{20}$$

with

$$\mathbf{B}_{e,n} = \begin{bmatrix} \mathbf{B}_n & \mathbf{B}_{0,n} & \mathbf{0}_{D\times DK} & \mathbf{B}_{1,n} & \cdots & \mathbf{B}_{l_{\max},n} \\ \mathbf{0}_{D\times 1} & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \cdots & \mathbf{0}_{DK} \\ \vdots & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \cdots & \mathbf{0}_{DK} \\ \vdots & \mathbf{0}_{DK} & \mathbf{I}_{DK} & \mathbf{0}_{DK} & \cdots & \mathbf{0}_{DK} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \mathbf{0}_{DK} \\ \mathbf{0}_{D\times 1} & \mathbf{0}_{DK} & \cdots & \mathbf{0}_{DK} & \mathbf{I}_{DK} & \mathbf{0}_{DK} \end{bmatrix}$$

$$\mathbf{B}_n = \mathbf{I} - \sum_{l=0}^{l_{\max}} \alpha_l \sum_{k\in\mathcal{K}_{n,l}} \frac{b_{k,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{k,n-l}$$

$$\mathbf{B}_{l,n} = [\frac{\alpha_l b_{1,n,l}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{1,n-l}, \cdots, \frac{\alpha_l b_{K,n,1}}{|\mathcal{K}_{n,l}|}\mathbf{S}_{K,n-l}]. \tag{21}$$

where $b_{k,n,l} = 1$ if $k \in \mathcal{K}_{n,l}$ and 0 otherwise.

In the following, we present a detailed convergence analysis of the PAO-Fed algorithm both in mean and mean-square senses. To this end, we make the following assumptions:

Assumption 1: The mapped data vectors $\mathbf{z}_{k,n}$ are drawn at each time step from a WSS multivariate random sequence with correlation matrix $\mathbf{R}_k = \mathbb{E}[\mathbf{z}_{k,n}\mathbf{z}_{k,n}^{\mathsf{T}}]$.

Assumption 2: The observation noise $\eta_{k,n}$ is assumed to be zero mean white Gaussian, and independent of all input and output data.

Assumption 3: At each client, the model parameter vector is assumed to be independent of the input data.

Assumption 4: The selection matrices are assumed independent from each other, and of any other data.

Assumption 5: The learning rate μ is small enough for terms involving higher-order powers of μ to be neglected.

It is important to note that no assumption is taken on the α_l variables because $l_{\max}$ is a fixed value in our analysis.

A. First-order Analysis

This subsection examines the mean convergence of the proposed PAO-Fed algorithm.

Theorem 1. *Let **Assumptions 1–4** hold true. Then, The proposed PAO-Fed converges in mean if and only if*

$$0 < \mu < \frac{2}{\max\limits_{\forall k,i} \lambda_i(\mathbf{R}_k)}. \tag{22}$$

Proof: Denoting the model error vector $\tilde{\mathbf{w}}_{e,n} = \mathbf{w}_e^* - \mathbf{w}_{e,n}$, and using the fact that $\mathbf{w}_e^* = \mathbf{B}_{e,n}\mathbf{A}_{e,n}\mathbf{w}_e^*$ (by construction, all rows in $\mathbf{B}_{e,n}$ and $\mathbf{A}_{e,n}$ sum to 1), from (20), we can recursively express $\tilde{\mathbf{w}}_{e,n}$ as

$$\begin{aligned}
\tilde{\mathbf{w}}_{e,n+1} &= \mathbf{w}_e^* - \mathbf{w}_{e,n+1} \\
&= \mathbf{w}_e^* - \mathbf{B}_{e,n}\mathbf{A}_{e,n}\mathbf{w}_{e,n} - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\mathbf{e}_{e,n} \\
&= \mathbf{B}_{e,n}\mathbf{A}_{e,n}\tilde{\mathbf{w}}_{e,n} - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n} \\
&\quad - \mathbf{B}_{e,n}\mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^\mathsf{T}(\mathbf{w}_e^* - \mathbf{A}_{e,n}\mathbf{w}_{e,n}) \\
&= \mathbf{B}_{e,n}(\mathbf{I} - \mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^\mathsf{T})\mathbf{A}_{e,n}\tilde{\mathbf{w}}_{e,n} \\
&\quad - \mu\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n}.
\end{aligned} \tag{23}$$

Taking the statistical expectation $\mathbb{E}[\cdot]$ on both sides of (23) and using **Assumptions** 1–4, we obtain

$$\begin{aligned}
\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}] &= \mathbb{E}[\mathbf{B}_{e,n}]\mathbb{E}[\mathbf{I} - \mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^\mathsf{T}]\mathbb{E}[\mathbf{A}_{e,n}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}] \\
&= \mathbb{E}[\mathbf{B}_{e,n}](\mathbf{I} - \mu\mathbf{R}_e)\mathbb{E}[\mathbf{A}_{e,n}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}],
\end{aligned} \tag{24}$$

where $\mathbf{R}_e = \text{blockdiag}\{\mathbf{0}_D, \mathbf{R}_1, \mathbf{R}_1, \cdots, \mathbf{R}_K, \mathbf{0}_{DKl_{\max}}\}$. The quantities $\mathbb{E}[\mathbf{A}_{e,n}]$ and $\mathbb{E}[\mathbf{B}_{e,n}]$ are evaluated in Appendix A.

Further, we consider the vectors and matrices reduced to the subspace between the index $D+1$ and $D(K+1)$. We denote the reduction of $\mathbf{x}$ by $\mathbf{x}|_{\text{sel}}$. Using the reduced definitions, (24) becomes: $\mathbb{E}[\tilde{\mathbf{w}}_{e,n+1}|_{\text{sel}}] = (\mathbf{I} - \mu\mathbf{R}_e|_{\text{sel}})\mathbb{E}[\mathbf{A}_{e,n}|_{\text{sel}}]\mathbb{E}[\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}]$, where the block $\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}$ is defined as a linear sequence of order 1 in a normed algebra. To prove the convergence of $\mathbb{E}[\tilde{\mathbf{w}}_{e,n}|_{\text{sel}}]$, we use the properties of the block maximum norm [50]. From Appendix A, we have $||\mathbb{E}[\mathbf{A}_{e,n}|_{\text{sel}}]||_{b,\infty} = 1$. Then the convergence condition reduces to $||\mathbf{I} - \mu\mathbf{R}_e|_{\text{sel}}||_{b,\infty} < 1$, equivalently, $|1 - \mu\lambda_i(\mathbf{R}_k)| < 1$, $\forall k, i$, where $\lambda_i(\cdot)$ is the ith eigenvalue of the argument matrix. This leads to the convergence condition given by (22).

□

B. Second-order Analysis

In this subsection, we present the second-order analysis of the proposed PAO-Fed algorithm. For the given arbitrary positive semidefinite matrix $\boldsymbol{\Sigma}$, the weighted norm-square of $\tilde{\mathbf{w}}_{e,n}$ is given by $||\tilde{\mathbf{w}}_{e,n}||_{\boldsymbol{\Sigma}}^2 = \tilde{\mathbf{w}}_{e,n}^\mathsf{T}\boldsymbol{\Sigma}\tilde{\mathbf{w}}_{e,n}$. From (23), we can obtain

$$\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||_{\boldsymbol{\Sigma}}^2] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||_{\boldsymbol{\Sigma}'}^2] + \mu^2\mathbb{E}[\boldsymbol{\eta}_{e,n}^\mathsf{T}\mathbf{Y}_n^{\boldsymbol{\Sigma}}\boldsymbol{\eta}_{e,n}], \tag{25}$$

where the cross terms are null under **Assumption 2** and the matrices $\boldsymbol{\Sigma}'$ and $\mathbf{Y}^{\boldsymbol{\Sigma}}$ are given by

$$\begin{aligned}
\boldsymbol{\Sigma}' = \mathbb{E}[\mathbf{A}_{e,n}^\mathsf{T}(\mathbf{I} - \mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^\mathsf{T})\mathbf{B}_{e,n}^\mathsf{T}\,\boldsymbol{\Sigma} \\
\mathbf{B}_{e,n}(\mathbf{I} - \mu\mathbf{Z}_{e,n}\mathbf{Z}_{e,n}^\mathsf{T})\mathbf{A}_{e,n}],
\end{aligned} \tag{26}$$

$$\mathbf{Y}_n^{\boldsymbol{\Sigma}} = \mathbf{Z}_{e,n}^\mathsf{T}\mathbf{B}_{e,n}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{B}_{e,n}\mathbf{Z}_{e,n}. \tag{27}$$

Using **Assumption 3** and the properties of the block Kronecker product, and the block vectorization operator $\text{bvec}\{\cdot\}$ [51], we can establish a relationship between $\boldsymbol{\sigma} = \text{bvec}\{\boldsymbol{\Sigma}\}$ and $\boldsymbol{\sigma}' = \text{bvec}\{\boldsymbol{\Sigma}'\}$ as

$$\boldsymbol{\sigma}' = \boldsymbol{\mathcal{F}}^\mathsf{T}\boldsymbol{\sigma}, \tag{28}$$

where

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{Q}}_{\mathbf{B}}\boldsymbol{\mathcal{Q}}_{\mathbf{A}} - \mu\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{I} \otimes_b \mathbf{R}_e)\boldsymbol{\mathcal{Q}}_{\mathbf{A}} - \mu\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{R}_e \otimes_b \mathbf{I})\boldsymbol{\mathcal{Q}}_{\mathbf{A}},$$

where the higher-order powers of μ are neglected under **Assumption 5**. In the above

$$\begin{aligned}
\boldsymbol{\mathcal{Q}}_{\mathbf{A}} &= \mathbb{E}[\mathbf{A}_{e,n} \otimes_b \mathbf{A}_{e,n}], \\
\boldsymbol{\mathcal{Q}}_{\mathbf{B}} &= \mathbb{E}[\mathbf{B}_{e,n} \otimes_b \mathbf{B}_{e,n}].
\end{aligned} \tag{29}$$

In Appendix B, we evaluate the matrices $\boldsymbol{\mathcal{Q}}_{\mathbf{A}}$ and $\boldsymbol{\mathcal{Q}}_{\mathbf{B}}$, and prove that all their entries are real, non-negative, and add up to unity on each row. This implies that both matrices are right-stochastic, and thus, their spectral radius is equal to one.

We will now evaluate the term $\mathbb{E}[\boldsymbol{\eta}_{e,n}^\mathsf{T}\mathbf{Y}_n^{\boldsymbol{\Sigma}}\boldsymbol{\eta}_{e,n}]$ as follows:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\eta}_{e,n}^\mathsf{T}\mathbf{Y}_n^{\boldsymbol{\Sigma}}\boldsymbol{\eta}_{e,n}] &= \mathbb{E}[\boldsymbol{\eta}_{e,n}^\mathsf{T}\mathbf{Z}_{e,n}^\mathsf{T}\mathbf{B}_{e,n}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n}] \\
&= \mathbb{E}[\text{trace}(\boldsymbol{\eta}_{e,n}^\mathsf{T}\mathbf{Z}_{e,n}^\mathsf{T}\mathbf{B}_{e,n}^\mathsf{T}\boldsymbol{\Sigma}\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\boldsymbol{\eta}_{e,n})] \\
&= \text{trace}(\mathbb{E}[\mathbf{B}_{e,n}\mathbf{Z}_{e,n}\mathbb{E}[\boldsymbol{\eta}_{e,n}^\mathsf{T}\boldsymbol{\eta}_{e,n}]\mathbf{Z}_{e,n}^\mathsf{T}\mathbf{B}_{e,n}^\mathsf{T}]\boldsymbol{\Sigma}) \\
&= \text{trace}(\mathbb{E}[\mathbf{B}_{e,n}\boldsymbol{\Phi}_n\mathbf{B}_{e,n}^\mathsf{T}]\boldsymbol{\Sigma}),
\end{aligned} \tag{30}$$

with $\boldsymbol{\Phi}_n = \mathbf{Z}_{e,n}\boldsymbol{\Lambda}_\eta\mathbf{Z}_{e,n}^\mathsf{T}$, where $\boldsymbol{\Lambda}_\eta = \mathbb{E}[\boldsymbol{\eta}_{e,n}^\mathsf{T}\boldsymbol{\eta}_{e,n}]$ is a diagonal matrix having the noise variances of all clients on its main diagonal. Note that we used **Assumption 2** in the last line of (30). Finally, using the properties of the block Kronecker product, we have

$$\text{trace}(\mathbb{E}[\mathbf{B}_{e,n}\boldsymbol{\Phi}_n\mathbf{B}_{e,n}^\mathsf{T}]\boldsymbol{\Sigma}) = \mathbf{h}^\mathsf{T}\boldsymbol{\sigma}, \tag{31}$$

with

$$\begin{aligned}
\mathbf{h} &= \text{bvec}\{\mathbb{E}[\mathbf{B}_{e,n}\boldsymbol{\Phi}_n\mathbf{B}_{e,n}^\mathsf{T}]\} \\
&= \boldsymbol{\mathcal{Q}}_{\mathbf{B}}\text{bvec}\{\mathbb{E}[\boldsymbol{\Phi}_n]\}.
\end{aligned} \tag{32}$$

Combining (25), (28), and (30), we can write the recursion for the weighted extended mean square displacement of the PAO-Fed algorithm as:

$$\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}^2] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||_{\text{bvec}^{-1}\{\boldsymbol{\mathcal{F}}^\mathsf{T}\boldsymbol{\sigma}\}}^2] + \mu^2\mathbf{h}^\mathsf{T}\boldsymbol{\sigma}, \tag{33}$$

where $\text{bvec}^{-1}\{\cdot\}$ represents the reverse operation of block vectorization.

Theorem 2. *Let **Assumptions 1–5** hold true. Then, the PAO-Fed algorithm exhibits stable mean square displacement if and only if:*

$$0 < \mu < \frac{1}{\max\limits_{\forall k,i} \lambda_i(\mathbf{R}_k)}. \tag{34}$$

Proof: Iterating (33) backwards to $n = 0$, we get

$$\begin{aligned}
\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}^2] =& \mathbb{E}[||\tilde{\mathbf{w}}_{e,0}||_{\text{bvec}^{-1}\{(\boldsymbol{\mathcal{F}}^\mathsf{T})^{n+1}\boldsymbol{\sigma}\}}^2] \\
&+ \mu^2\mathbf{h}^\mathsf{T}(\mathbf{I} + \sum_{j=1}^{n}(\boldsymbol{\mathcal{F}}^\mathsf{T})^j)\boldsymbol{\sigma}.
\end{aligned} \tag{35}$$

To prove the convergence of $\mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\boldsymbol{\Sigma}}] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$, we need to prove that the spectral radius of $\boldsymbol{\mathcal{F}}$ is less than one, i.e. $\rho(\boldsymbol{\mathcal{F}}) < 1$. Using the properties of the block maximum norm [50], we have

$$\begin{aligned}\rho(\boldsymbol{\mathcal{F}}) &\leqslant ||\boldsymbol{\mathcal{Q}}_{\mathbf{B}}(\mathbf{I}-\mu(\mathbf{I}\otimes_b \mathbf{R}_e)-\mu(\mathbf{R}_e\otimes_b \mathbf{I}))\boldsymbol{\mathcal{Q}}_{\mathbf{A}}||_{b,\infty},\\ &\leqslant ||\boldsymbol{\mathcal{Q}}_{\mathbf{B}}||_{b,\infty}||\boldsymbol{\mathcal{Q}}_{\mathbf{A}}||_{b,\infty}\\ &\quad ||(\mathbf{I}-\mu(\mathbf{I}\otimes_b \mathbf{R}_e)-\mu(\mathbf{R}_e\otimes_b \mathbf{I}))||_{b,\infty}. \end{aligned} \tag{36}$$

Since the matrices $\boldsymbol{\mathcal{Q}}_{\mathbf{A}}$ and $\boldsymbol{\mathcal{Q}}_{\mathbf{B}}$ are right stochastic, we have $||\boldsymbol{\mathcal{Q}}_{\mathbf{A}}||_{b,\infty} = ||\boldsymbol{\mathcal{Q}}_{\mathbf{B}}||_{b,\infty} = 1$. Therefore the condition $||(\mathbf{I}-\mu(\mathbf{I}\otimes_b \mathbf{R}_e)-\mu(\mathbf{R}_e\otimes_b \mathbf{I}))||_{b,\infty} < 1$, equivalently, $|1-\mu(\lambda_i(\mathbf{R}_e)+\lambda_j(\mathbf{R}_e))| < 1,\ \forall i,j$, is sufficient to guarantee the convergence of $||\tilde{\mathbf{w}}_{e,n}||^2_{\boldsymbol{\Sigma}}$. This simplification leads to the convergence condition in (34). □

C. Transient and Steady-state Mean Square Deviation

From (33), we can express the relation between $\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ and $\mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$ as

$$\begin{aligned}\mathbb{E}[||\tilde{\mathbf{w}}_{e,n+1}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}] &= \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]\\ &\quad + \mathbb{E}[||\tilde{\mathbf{w}}_{e,0}||^2_{\text{bvec}^{-1}\{(\boldsymbol{\mathcal{F}}^{\mathsf{T}}-\mathbf{I})(\boldsymbol{\mathcal{F}}^{\mathsf{T}})^n\boldsymbol{\sigma}\}}]\\ &\quad + \mu^2\mathbf{h}^{\mathsf{T}}(\boldsymbol{\mathcal{F}}^{\mathsf{T}})^n\boldsymbol{\sigma}. \end{aligned} \tag{37}$$

If we set $\boldsymbol{\sigma} = \text{bvec}\{\text{blockdiag}\{\mathbf{I_D}, \mathbf{0}, \ldots, \mathbf{0}\}\}$, we obtain the transient expression for the mean square deviation of the global model at iteration n: $\mathbb{E}[||\tilde{\mathbf{w}}_n||^2] = \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{\boldsymbol{\sigma}\}}]$.

Under (34), by letting $n \to \infty$ in (33), we obtain the expression of the steady-state mean square deviation (MSD) for the PAO-Fed algorithm.

$$\lim_{n\to\infty} \mathbb{E}[||\tilde{\mathbf{w}}_{e,n}||^2_{\text{bvec}^{-1}\{(\mathbf{I}-\boldsymbol{\mathcal{F}}^{\mathsf{T}})\boldsymbol{\sigma}\}}] = \mu^2\mathbf{h}^{\mathsf{T}}\boldsymbol{\sigma}. \tag{38}$$

By setting $\boldsymbol{\sigma} = (\mathbf{I}-\boldsymbol{\mathcal{F}}^{\mathsf{T}})^{-1}\text{bvec}\{\text{blockdiag}\{\mathbf{I_D}, \mathbf{0}, \ldots, \mathbf{0}\}\}$, the steady-state MSD expression of the global model can be obtained.

V. Numerical Simulations

This section demonstrates the performance of the proposed PAO-Fed algorithm through a series of numerical experiments. In these experiments, we compare the performance of the PAO-Fed algorithm with existing methods, specifically, Online-FedSGD, Online-Fed [19], and PSO-Fed [28].

A. Simulation Setup

We considered a federated network comprising $K = 256$ clients connected to a server. Synthetic data is progressively made available to the clients in an imbalanced and non-IID manner. For this purpose, the clients are separated into 4 data groups for which training sets are composed of 500, 1000, 1500, and 2000 samples, respectively. A single data sample is of the form $\{\mathbf{x}_{k,n}, y_{k,n}\}$, and related by the following nonlinear relation $\mathbb{R}^4 \longrightarrow \mathbb{R}$:

$$\begin{aligned}y_{k,n} =& \sqrt{\mathbf{x}^2_{k,n}[1] + \sin^2(\pi \mathbf{x}_{k,n}[4])}\\ &+ (0.8 - 0.5\exp(-\mathbf{x}^2_{k,n}[2])\mathbf{x}_{k,n}[3]) + \eta_{k,n}, \end{aligned} \tag{39}$$

where $\mathbf{x}_{k,n}[i]$ denotes the ith element of vector $\mathbf{x}_{k,n} = [x_{k,n}, x_{k,n-1}, x_{k,n-4}, x_{k,n-3}]$. A first-order autoregressive model is used to produce the non-IID input signal $x_{k,n} = \theta_k\, x_{k,n-1} + \sqrt{1-\theta_k^2}\, u_{k,n}$, with $u_{k,n} \in \mathcal{N}(\mu_k, \sigma^2_{u_k})$, and, for a given client k, $\theta_k \in \mathcal{U}(0.2, 0.9)$, $\mu_k \in \mathcal{U}(-0.2, 0.2)$, and $\sigma^2_{u_k} \in \mathcal{U}(0.2, 1.2)$. The observation noise $\nu_{k,n}$ is assumed to be white Gaussian with variance $\sigma^2_{\nu_k} \in \mathcal{U}(0.005, 0.03)$. Further, the cosine feature function is used to map $\mathbf{x}_{k,n}$ from dimension $L = 4$ into the RFF space of dimension $D = 200$.

As discussed in Section III.A, client participation is modeled using the probabilities $p_{k,n}, k \in \mathcal{K}$. Note that a client can only participate in an iteration if it receives new data; otherwise, the probability is set to 0. The clients of each data group are further separated into 4 availability groups, dictating their probability $p_{k,n}$ of participating at each iteration. The Bernoulli trial on $p_{k,n}$ dictates if a client is available or not at a given iteration. Unless stated otherwise, the participation probabilities given to the four availability groups are 0.25, 0.1, 0.025, and 0.005. Finally, each communication to the server will be delayed by more than l iterations with probability $\delta^l, 0 < l < l_{\max}$, with, unless stated otherwise, $\delta = 0.2$ and $l_{\max} = 10$. This probability is assumed to be the same for all clients.

The performance of the algorithms is evaluated on a test dataset with the mean squared error (MSE) given at iteration n by:

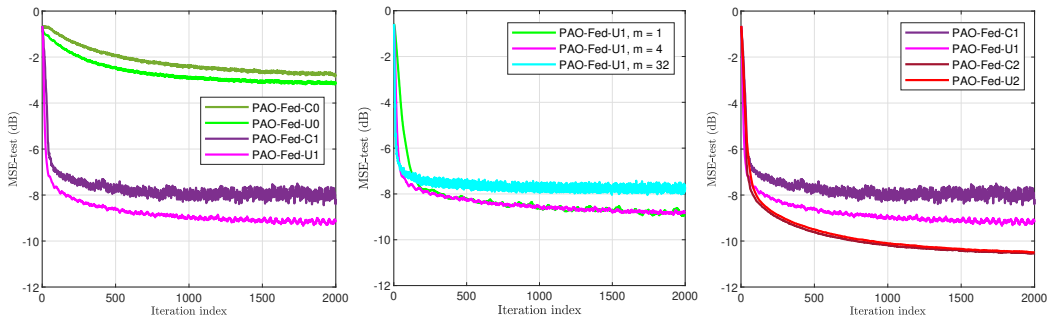
$$\text{MSE-test} = \frac{1}{\text{MC}}\sum_{e=1}^{\text{MC}} \frac{||\mathbf{y}^e_{\text{test}} - (\mathbf{Z}^e_{\text{test}})^{\mathsf{T}}\mathbf{w}^e_n||_2^2}{T}, \tag{40}$$

where MC is the number of Monte Carlo iterations, T is the size of the test dataset, $\mathbf{y}^e_{\text{test}}$ and $\mathbf{Z}^e_{\text{test}}$ are the realization of the data for a given Monte Carlo iteration, and $\mathbf{w}^e_n$ is the server's model vector for the considered method. When comparing the PAO-Fed algorithm with other methods, the learning rates were set to yield identical initial convergence rates so that steady-state values may be compared. Some algorithms were not able to reach this common convergence rate, but since their steady-state accuracy is lower, comparison is still possible. All the learning rates satisfy the convergence conditions obtained in Section IV for PAO-Fed, and are available in [19], [28] for Online-Fed, Online-FedSGD, and PSO-Fed. For instance, in Fig. 2, 3, and 4, the step-size for the PAO-Fed algorithm is set to $\mu = 0.4$ with $\max_{\forall k,i} \lambda_i(\mathbf{R}_k) = 1.02$.

In the simulations, we implement uncoordinated partial-sharing-based communications from the server to the clients with $\text{diag}(\mathbf{M}_{k,n}) = \text{circshift}(\text{diag}(\mathbf{M}_{1,n}), mk)$ and $\text{diag}(\mathbf{M}_{1,n}) = \text{circshift}(\text{diag}(\mathbf{M}_{1,0}), mn)$. This, in turn, dictates the portion of the model sent by the clients to the server (see Section II C) so that, on average, all portions are equally represented in the aggregation. We recall that m is the number of model parameters shared at each iteration by both the server and the clients, and dictates the communication savings in partial-sharing-based communications.

We consider different versions of the PAO-Fed algorithm.

- **PAO-Fed-C0** and **PAO-Fed-U0** utilize coordinated and uncoordinated partial-sharing, respectively, without employing the weight-decreasing mechanism in (15), that



(a) (b) (c)

Fig. 2: Optimization of the PAO-Fed method. (a) Utilizing local updates and coordinated/uncoordinated partial-sharing, (b) Communication savings, (c) Utilizing a weight-decreasing mechanism for delayed updates.

is, $\alpha_l = 1, 0 \leqslant l \leqslant l_{\max}$. Further, the clients share the last received server model portion, refined once by the local update process.

- **PAO-Fed-C1** and **PAO-Fed-U1** utilize coordinated and uncoordinated partial-sharing, respectively, without employing the weight-decreasing mechanism in (15). Their selection matrices evolve as described in Section II C.
- **PAO-Fed-C2** and **PAO-Fed-U2** utilize coordinated and uncoordinated partial-sharing, respectively, and employ the weight-decreasing mechanism in (15) with $\alpha_l = 0.2^l, 0 \leqslant l \leqslant l_{\max}$. Their selection matrices evolve as described in Section II C.

Unless explicitly specified, each PAO-Fed implementation shares $m = 4$ model parameters per communication round, resulting in a 98% reduction in communication.

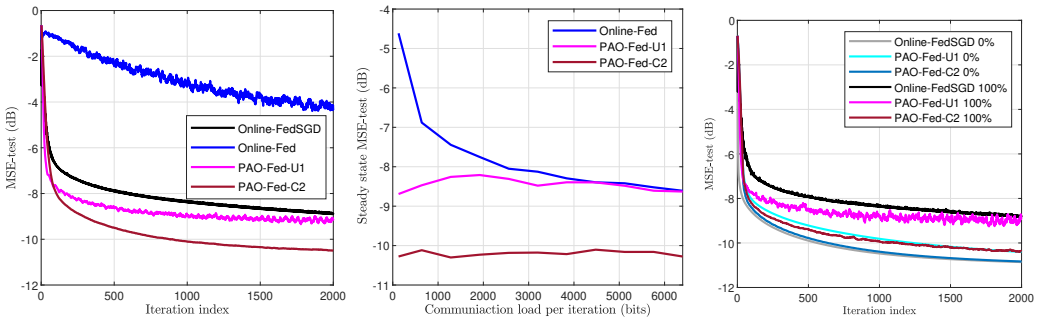
B. Hyper Parameters Selection

In the first experiments, we study the impact of the hyperparameters on the convergence properties of the PAO-Fed algorithm. Specifically, we investigate the impact of the choice of the selection matrices, the number of model parameters shared, and the scale of the weight-decreasing mechanism for delayed updates. The corresponding learning curves in Fig. 2 display the MSE-test in dB versus the iteration index.

First, we examined how the choice of the selection matrices $\mathbf{M}_{k,n}$ and $\mathbf{S}_{k,n}$ impact the convergence properties of the PAO-Fed algorithm. These matrices select the model portion to be shared between the server and clients (see Section II C). The versions PAO-Fed-C0 and PAO-Fed-U0 are set with $\mathbf{S}_{k,n} = \mathbf{M}_{k,n}$; that is, the last received portion from the server is updated once by the local learning process at the clients before being sent back to the server. On the contrary, the versions PAO-Fed-C1 and PAO-Fed-U1 are set as in (7) and (8); that is, the received portions from the server will be updated several times by the local learning process to accumulate information, in a manner similar to batch learning, before being sent back to the server. We observe in Fig. 2 (a) that the versions PAO-Fed-(C/U)1 outperform the versions PAO-Fed-(C/U)0. For this reason, we will only consider the versions of the PAO-Fed algorithm making full use of the local updates in the following. We also notice in this experiment that it is best to use uncoordinated partial-sharing in asynchronous settings, this contradicts the behavior of partial-sharing-based communications in ideal settings, where coordinated partial-sharing performs slightly better than uncoordinated, as explained in [28].

Second, we studied the impact of the number of model parameters m shared by participating clients and the server during the learning process. Fig. 2 (b) shows the performance of the PAO-Fed-U1 algorithm (uncoordinated, making use of local updates) for different values of m, namely $m = 1$, $m = 4$, and $m = 32$. Although sharing more model parameters increases the initial convergence speed, we observed that it decreases the final accuracy for larger m values. This contradicts previous results in the literature about the behavior of partial-sharing in ideal settings [28]. In fact, sharing more model parameters increases the potential negative impact of one single delayed update carrying outdated information, decreasing the overall accuracy. Sharing a small number of model parameters limits the impact of a given update, providing some level of protection against outdated information, and ensuring better model fitting [52]. We chose to set $m = 4$ as a baseline, as it presents a good compromise between initial convergence speed, steady-state accuracy, and communication reduction.

Finally, to reduce the harmful effect of delayed updates on the convergence properties of the algorithm, we introduce the weight-decreasing mechanism for delayed updates proposed in (15) in the versions PAO-Fed-C2 and PAO-Fed-U2. We set $\alpha_l = 0.2^l, 0 \leqslant l \leqslant l_{\max}$. In Fig. 2 (c), we display the performance of these methods alongside PAO-Fed-C1 and PAO-Fed-U1. We observe that decreasing the weight of the delayed updates significantly improves the performance of the PAO-Fed algorithm on the considered asynchronous settings. The proposed mechanism considers the relevance of delayed and potentially outdated updates by effectively reducing their impact on the server model, especially for substantial delays. By doing so, the negative effect of delayed updates is mitigated; in particular, when using the aforementioned weight-



(a) (b) (c)

Fig. 3: Comparison of PAO-Fed with existing methods. (a) Learning curves, (b) Steady-state MSE vs. communication load, (c) Impact of straggler clients.

decreasing mechanism, PAO-Fed-C2 using coordinated partial sharing and PAO-Fed-U2 using uncoordinated partial sharing exhibit the same performance.

C. Comparison of PAO-Fed with Existing Algorithms

In the following experiments, we compare the performance of the PAO-Fed algorithm with existing online FL methods in the literature. Figs. 3 (a) and (c) display the MSE-test in dB versus the iteration index, and Fig. 3 (b) displays accuracy variation versus communication savings.

First, we compared PAO-Fed-U1 and PAO-Fed-U2 with Online-Fed [19], and Online-FedSGD. Fig. 3 (a) displays the corresponding learning curves. First, we observe that Online-Fed performs poorly; sub-sampling the already reduced pool of available clients is not a viable solution to reduce communication in asynchronous settings. Then, we observe that both PAO-Fed-U1 and PAO-Fed-U2 outperform Online-FedSGD while using 98% less communication. The reason for this very good performance is twofold. First, using the local and autonomous local updates in the PAO-Fed algorithm allows it to extract more information from the sparsely participating clients. Second, partial-sharing-based communication provides the PAO-Fed algorithm with an innate resilience to the negative impact of delayed updates; this resilience is further increased in the PAO-Fed-U2 algorithm with the weight-decreasing mechanism, hence its better performance.

Second, we study the relationship between communication load and accuracy. Figure 3 (b) shows the steady-state mean squared error on the test dataset versus the average communication load per iteration when the clients employ either PAO-Fed-U1, PAO-Fed-C2, or Online-Fed algorithms. The communication load is obtained by multiplying the average number of model parameters shared by a client during a given iteration, corresponding to m for the PAO-Fed algorithms, by 32, which is the number of bits on which a model parameter is stored. We find the MSE reached after 2000 iterations in the previous figure by the three algorithms in this figure for a communication load of 128 bits. Similarly, we find the MSE reached after 2000 iterations in the previous figure by Online-FedSGD in this figure for the Online-Fed algorithm with a communication load of 6400 bits. Further, we observe that the higher the communication load is, the better the performance of Online-Fed is. However, the performances of the algorithms using partial-sharing-based communication vary very little with the communication load, as the lower amount of communication is compensated by the use of local updates and the resilience to delayed communications.

Finally, to observe the impact of the straggler clients on the convergence properties of the algorithms, we compare the performance of the algorithms in the proposed settings (100% of clients are potential stragglers) to their performance in an ideal setting where clients are always available when they receive new data and their communication channels do not suffer from delays (0% of clients are potential stragglers). The learning curves are shown in Fig. 3 (c). We observe that, in the absence of straggler clients, the methods using coordinated partial-sharing achieve greater accuracy, almost identical to methods with no communication reduction, while the methods using uncoordinated partial-sharing have slightly worse performance, this corresponds to the results obtained in [28]. Furthermore, we see that the PAO-Fed-C2 algorithm used on straggler clients has convergence properties almost similar to the ones of algorithms in a perfect setting.

D. Performance on a Real-world Dataset

Fig. 4 shows the performance of the proposed PAO-Fed algorithm on the real-world California Cooperative Oceanic Fisheries Investigations (CalCOFI) dataset [53]. This dataset comprises oceanographic data from seawater samples collected at various stations and contains more than 800,000 samples. Each sample contains parameters such as temperature, salinity, O_2 saturation, etc. The salinity of the water is linked in a nonlinear manner to the other available parameters, and we employed the proposed method to learn this nonlinear model relating the salinity level in a decentralized manner. For the purpose of the experiment, we consider only 80,000 samples that we distribute progressively and unevenly to the 256 clients throughout the learning process (to ensure non-IID and

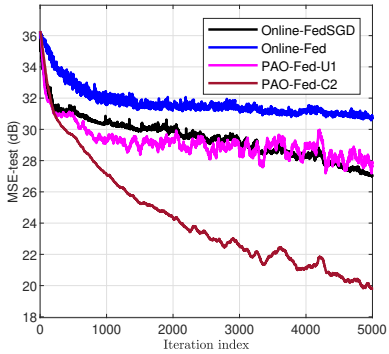


Fig. 4: Learning curves on the CalCOFI dataset.

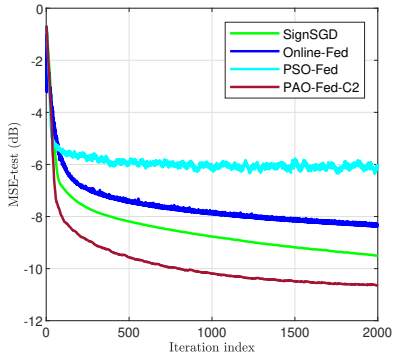


Fig. 5: Learning curves of PAO-Fed, PSO-Fed, Online-Fed, and SignSGD.

imbalanced data settings). Further, we simulated the straggler-like behavior of the clients as mentioned above (availability groups are 0.25, 0.1, 0.025, and 0.005; each communication to the server will be delayed by more than l iterations with probability $\delta^l, 0 < l < l_{\max}$, with $\delta = 0.2$ and $l_{\max} = 10$). We observe similar performance for the PAO-Fed, Online-Fed, and Online-FedSGD algorithms to the experiments on synthetic datasets. The PAO-Fed-U1 algorithm is able to achieve the same accuracy as Online-FedSGD while using 98% less communications, and the PAO-Fed-C2 algorithm, also using 98% less communications, is able to outperform all other methods.

E. Comparison of Various Communication Reduction Methods in Asynchronous Settings

In this simulation, we compare the performance of the proposed method with the PSO-Fed [28], Online-Fed [19], and SignSGD [54] algorithms. The PSO-Fed algorithm combines client scheduling and partial-sharing-based communications. For a fair comparison, it has been tailored to reduce the overall communication load by 98%, similar to the proposed PAO-Fed-C2 algorithms. By design, the SignSGD drastically reduces the communication load from clients to server but does not reduce the communication load from server to clients. Its communication load reduction is, therefore, less than 50%. For this reason, the Online-Fed algorithm has been tailored to reduce the communication load by only 50%. The learning curves are displayed in Fig. 5. We observe that reducing the communication load via a combination of client scheduling and partial-sharing-based communication, as in PSO-Fed, is not desirable in asynchronous settings. Furthermore, we see that the SignSGD achieves significantly better performance than Online-Fed for a similar communication load reduction, making it a viable alternative to partial-sharing-based communication in asynchronous settings. However, it would need to be complemented by server-to-client communication reduction and a weight-decreasing mechanism to achieve the same accuracy and communication load reduction as the proposed PAO-Fed-C2.

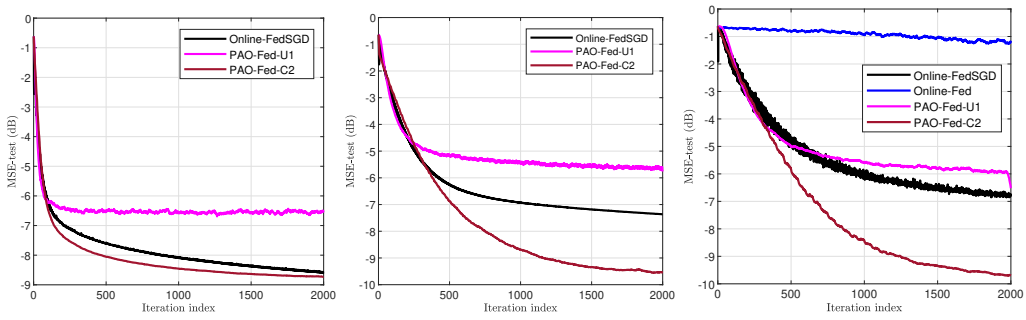
F. Impact of the Environment on Convergence Properties

In these last experiments, we study the impact that a change in the external environment can have on the convergence properties of the proposed algorithms and existing methods. The corresponding learning curves are shown in Fig. 6.

First, we studied in Fig. 6 (a) the importance of using partial-sharing-based communications both at the server and at the clients. The algorithms using partial-sharing-based communications have been altered in this simulation with $\mathbf{M}_{k,n} = \mathbf{I}, \forall k, n$; that is, the server sends its entire model to the participating clients at each iteration. This modification can be appealing if the server is not subject to power constraints. The clients behave normally and only send a portion of their local model; however, unlike in the other simulations, the received global model replaces the local model at each participant, see (10). In such a case, we observe that the performance of the partial-sharing-based methods is drastically reduced. It is the information kept by the clients in the not-yet-shared portions of their local models that allows partial-sharing-based methods to outperform Online-FedSGD. We note that clients may choose to ignore part of the received model to avoid this downfall.

Second, we studied the algorithm behaviors in an environment where most communications are delayed, but delays cannot be too lengthy. To this aim, the delay probability has been significantly increased, and the maximum possible delay reduced ($\delta = 0.8$ and $l_{\max} = 5$). We observe in Fig. 6 (b) that the limited maximum delay allows Online-FedSGD to outperform PAO-Fed-U1, as the benefit of partial-sharing against data of poor quality does not out-weight the smaller amount of communication available to PAO-Fed-U1. To compensate for the fact that most incoming information is weighted down by the weight-decreasing mechanism of PAO-Fed-C2, its learning rate has been increased to near its maximum value obtained in **Theorem 2**. Despite this, the PAO-Fed-C2 algorithm reaches very low steady-state error and significantly outperforms Online-FedSGD.

Finally, we modeled an environment where availability groups are given the probabilities 0.025, 0.01, 0.0025, and



(a) (b) (c)

Fig. 6: Learning curves in different environments. (a) Full server communication, (b) Common delays, (c) Increased straggler behavior.

0.0005; communications to the server have a probability $\delta = 0.4$ to be delayed. Further, delays last for more than l iterations, l taking the values $10i, 0 \leqslant i \leqslant 6$, with probability $\delta^{\frac{l}{10}}$; $l_{\max}$ is set to 60. This notably implies that, in this environment, delayed updates have a greater probability of arriving after a non-delayed update coming from the same client. Such an environment where clients are less likely to be available to participate, communications are more likely to be delayed, and delays last for more iterations, is less favorable to learning. An application relying on edge devices that are poorly available and unreliable would evolve in an environment similar to this. Fig. 6 (c) presents the learning curves of Online-Fed, Online-FedSGD, and the PAO-Fed algorithm in this new environment to see how it may impact the convergence properties of the algorithms. We observe that, in this environment, reducing the weight given to the delayed updates gains importance as the accuracy difference between PAO-Fed-C2 and PAO-Fed-U1 increases. In fact, delayed updates may carry information that is significantly outdated and, therefore, prevent the algorithms not using a weight-decreasing mechanism for delayed updates to reach satisfactory steady-state error. For this reason, the PAO-Fed-C2 algorithm achieves significantly better accuracy than Online-FedSGD in this environment.

VI. Conclusions

This paper proposed a communication-efficient FL algorithm adapted to a realistic environment. The proposed FL algorithm operates with significantly reduced communication requirements and can cope with an unevenly distributed system with poor client availability, potential failures, and communication delays. The proposed partial sharing mechanism reduces the communication overhead and diminishes the negative impact of delayed updates on accuracy. We further proposed a weight-decreasing aggregation mechanism that emphasizes more recent updates to improve performance in environments suffering from substantial delays, poor participation, and straggler devices. Our numerical results showed that the proposed algorithm outperforms standard FL methods in an asynchronous environment while reducing the communication overhead by 98 percent. The proposed approach is ideal for extracting information in real-time from diverse geographically dispersed devices without overloading the system, making it highly desirable in IoT applications in particular. Future works include expanding the proposed algorithm to a multi-server or networked architecture to alleviate the strain on the single server in applications with many clients.

Appendix A
Evaluation of $\mathbb{E}[\mathbf{A}_{e,n}]$ and $\mathbb{E}[\mathbf{B}_{e,n}]$

The matrix $\mathbf{A}_{e,n}$ is composed of $D \times D$-sized blocks $\mathbf{A}_{i,j,n}$, given by:

$$\mathbf{A}_{i,j,n} = \begin{cases} \mathbf{I}_D & \text{if } i = j \wedge (i = 1 \vee i > K+1), \\ a_{k,n}\mathbf{M}_{k,n} & \text{if } i \in [|2, \ldots, K+1|] \wedge j = 1, \\ \mathbf{I}_D - a_{k,n}\mathbf{M}_{k,n} & \text{if } i \in [2, \ldots, K+1] \wedge i = j, \\ \mathbf{0}_D & \text{otherwise,} \end{cases}$$

where $k = i - 1$.

We note that $\mathbb{E}[a_{k,n}\mathbf{M}_{k,n}] = p_{k,n}p_m\mathbf{I}_D$, with $p_{k,n}$ being the probability that client k participates at iteration n, and p_m being the probability that a given model parameter is selected by the selection matrix (i.e., the density of the selection: $\frac{m}{D}$). Since $0 \leqslant p_{k,n}p_m \leqslant 1$, and given the above decomposition, matrix $\mathbb{E}[\mathbf{A}_{e,n}]$ is right stochastic.

Further, we note that by construction, $(a_{k,n}\mathbf{M}_{k,n})^2 = a_{k,n}\mathbf{M}_{k,n}$; therefore, under **Assumption 3**, we have

$$\mathbb{E}[a_{k,n}\mathbf{M}_{k,n}a_{k',n'}\mathbf{M}_{k',n'}] = \begin{cases} p_{k,n}p_m\mathbf{I}_D & \text{if } k = k' \wedge n = n', \\ p_{k,n}p_{k',n'}p_m^2\mathbf{I}_D & \text{otherwise.} \end{cases}$$

Similarly, we decompose the matrix $\mathbf{B}_{e,n}$ in $D \times D$-sized blocks $\mathbf{B}_{i,j,n}$ as follows:

$$\mathbf{B}_{i,j,n} = \begin{cases} \mathbf{B}_n & \text{if } i = j = 1, \\ \mathbf{B}_{0,n}^{(j-1)} & \text{if } i = 1 \wedge j \in [2, K+1], \\ \mathbf{B}_{\lceil \frac{j-1}{K} \rceil - 3, n}^{(j-1 \mod K)} & \text{if } i = 1 \wedge j \in \\ & [|3K+2, \ldots, (l_{\max}+3)K+1|], \\ \mathbf{I}_D & \text{if } i \in [|1,2|] \wedge j = 2, \\ \mathbf{I}_D & \text{if } i > 3 \wedge j > 2 \wedge i = j+1, \\ \mathbf{0}_D & \text{otherwise.} \end{cases}$$

The blocks are given by:

$$\begin{aligned} \mathbf{B}_n &= \mathbf{I} - \sum_{l=0}^{l_{\max}} \alpha_l \sum_{k \in \mathcal{K}_{n,l}} \frac{b_{k,n,l}}{|\mathcal{K}_{n,l}|} \mathbf{S}_{k,n-l}, \\ \mathbf{B}_{l,n}^{(k)} &= \mathbf{B}_{l,n}[k], \\ \mathbf{B}_{l,n} &= \left[\frac{\alpha_l b_{1,n,l}}{|\mathcal{K}_{n,l}|} \mathbf{S}_{1,n-l}, \cdots, \frac{\alpha_l b_{K,n,1}}{|\mathcal{K}_{n,l}|} \mathbf{S}_{K,n-l} \right]. \end{aligned}$$

We note that by construction,

$$\mathbf{B}_n + \sum_{l=1}^{l_{\max}} \sum_{k=1}^{K} \mathbf{B}_{l,n}^{(k)} = \mathbf{I},$$

hence, the matrix $\mathbb{E}[\mathbf{B}_{e,n}]$ is right stochastic as well.

Appendix B
Evaluation of $\mathcal{Q}_{\mathbf{A}}$ and $\mathcal{Q}_{\mathbf{B}}$

We decompose matrix $\mathcal{Q}_{\mathbf{A}}$ into $D \times D$-sized blocks and prove the property by computing the Kronecker product $\mathbf{A}_{e,n} \otimes_b \mathbf{A}_{e,n}$ before taking the expectation. In particular, we have

$$\mathcal{Q}_{\mathbf{A}} = [\mathbb{E}[\mathbf{A}_{i,j,n} \otimes_b \mathbf{A}_{e,n}], (i,j) \in [|1, \ldots, K(l_{\max}+1)+1|]^2],$$

and we note that $\mathcal{Q}_{\mathbf{A}}$ can be proven to be right stochastic one block-row at a time, considering sets of D rows indexed by i in the above equation.

The property is easy to prove on the block-rows $i = 1$ and $i > K+1$. On those block-rows, we have

$$\mathbf{A}_{i,j,n} = \begin{cases} \mathbf{I}_D & \text{if } i = j \\ \mathbf{0}_D & \text{otherwise} \end{cases},$$

therefore, since $\mathbb{E}[\mathbf{A}_{e,n}]$ satisfies the property, it is satisfied on those block-rows.

We now consider the remaining block-rows. For this purpose, let $i \in [|2, \ldots, K+1|]$. According to the decomposition of the left-hand side $\mathbf{A}_{e,n}$, the block-row i of $\mathcal{Q}_{\mathbf{A}}$ reduces to only two non-zero elements, $\mathbb{E}[\mathbf{A}_{i,1,n} \otimes_b \mathbf{A}_{e,n}]$ and $\mathbb{E}[\mathbf{A}_{i,i,n} \otimes_b \mathbf{A}_{e,n}]$. Hence we can compute:

$$\begin{aligned} &\mathbb{E}[\mathbf{A}_{i,1,n} \otimes_b \mathbf{A}_{e,n}] + \mathbb{E}[\mathbf{A}_{i,i,n} \otimes_b \mathbf{A}_{e,n}] \\ &\quad = \mathbb{E}[a_{i-1,n} \mathbf{M}_{i-1,n} \otimes_b \mathbf{A}_{e,n}] \\ &\quad + \mathbb{E}[(\mathbf{I}_D - a_{i-1,n} \mathbf{M}_{i-1,n}) \otimes_b \mathbf{A}_{e,n}] \\ &\quad = \mathbb{E}[\mathbf{I}_D \otimes_b \mathbf{A}_{e,n}], \end{aligned}$$

and conclude that the block-row i satisfies the property.

Similarly, we decompose the matrix $\mathcal{Q}_{\mathbf{B}}$ into $D \times D$-sized blocks and prove that it is right stochastic by computing the Kronecker product $\mathbf{B}_{e,n} \otimes_b \mathbf{B}_{e,n}$ before taking the expectation.

$$\mathcal{Q}_{\mathbf{B}} = [\mathbb{E}[\mathbf{B}_{i,j,n} \otimes_b \mathbf{B}_{e,n}], (i,j) \in [1, \ldots, K(l_{\max}+1)+1]^2].$$

The evaluation is trivial for the block-rows $i \in [2, \ldots, K(l_{\max}+1)+1]$, where the decomposition of the left-hand side $\mathbf{B}_{e,n}$ reduces to only one non-zero element: $\mathbf{I}$. Therefore, since $\mathbf{B}_{e,n}$ satisfies the property, it is satisfied on those block-rows.

We now consider the block-row $i = 1$ and compute the sum of the elements as:

$$\begin{aligned} &\mathbb{E}[\mathbf{B}_n \otimes_b \mathbf{B}_{e,n} + \sum_{l=1}^{l_{\max}} \sum_{k=1}^{K} \mathbf{B}_{l,n}^{(k)} \otimes_b \mathbf{B}_{e,n}] \\ &\quad = \mathbb{E}[\mathbf{I}_D \otimes_b \mathbf{B}_{e,n}], \end{aligned}$$

by construction of the $\mathbf{B}_{l,n}^{(k)}$ matrices. We conclude that the block-row $i = 1$ satisfies the property as well.

We have proven that both $\mathcal{Q}_{\mathbf{A}}$ and $\mathcal{Q}_{\mathbf{B}}$ are right stochastic matrices.

References

[1] F. Gauthier, V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Resource-aware asynchronous online federated learning for nonlinear regression," *arXiv preprint arXiv:2111.13931*, 2021.

[2] 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, May 2020.

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

[4] O. Dekel, P. M. Long, and Y. Singer, "Online multitask learning," in *Int. Conf. Comput. Learn. Theory*, 2006, pp. 453–467.

[5] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Ind. Eng.*, vol. 149, p. 106854, 2020.

[6] S. Boll and J. Meyer, "Health-X dataLOFT: A Sovereign Federated Cloud for Personalized Health Care Services," *IEEE MultiMedia*, vol. 29, no. 1, pp. 136–140, May 2022.

[7] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

[8] 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, May 2022.

[9] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[10] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, Mar. 2021.

[11] Z. Zhao, C. Feng, W. Hong, J. Jiang, C. Jia, T. Q. S. Quek, and M. Peng, "Federated Learning With Non-IID Data in Wireless Networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2022.

[12] E. Ozfatura, K. Ozfatura, and D. Gündüz, "FedADC: accelerated federated learning with drift control," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2021, pp. 467–472.

[13] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, Jul. 2019.

[14] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[15] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Nov. 2020.

[16] Z. Lian, W. Wang, and C. Su, "COFEL: Communication-efficient and optimized federated learning with local differential privacy," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[17] Y. Lu, Z. Liu, and Y. Huang, "Parameters compressed mechanism in federated learning for edge computing," in *Proc. IEEE Int. Conf. Cyber Secur. Cloud Comput.*, Jun. 2021, pp. 161–166.

[18] X. Fan, Y. Wang, Y. Huo, and Z. Tian, "Communication-efficient federated learning through 1-bit compressive sensing and analog aggregation," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2021, pp. 1–6.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial Intel. Statis.*, pp. 1273–1282, Apr. 2017.

[20] Y. Chen, Z. Chai, Y. Cheng, and H. Rangwala, "Asynchronous federated learning for sensor data with concept drift," *arXiv preprint arXiv:2109.00151*, Sep. 2021.

[21] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, Mar. 2019.

[22] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2020, pp. 15–24.

[23] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. Quek, "Asynchronous Federated Learning over Wireless Communication Networks," *IEEE Trans. Wireless Commun.*, Mar. 2022.

[24] Z. Chai, Y. Chen, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: a communication-efficient federated learning method with asynchronous tiers under non-iid data," *arXiv preprint arXiv:2010.05958*, Oct. 2020.

[25] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Dec. 2019.

[26] Z. Wang, Z. Zhang, and J. Wang, "Asynchronous federated learning over wireless communication networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–7.

[27] X. Qiu, T. Parcollet, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "Can federated learning save the planet?" *arXiv preprint arXiv:2010.06537*, Oct. 2020.

[28] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning framework for nonlinear regression," *IEEE Int. Conf. Acoust., Speech and Signal Process.*, May 2022.

[29] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[30] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, Oct. 2016.

[31] M. Chen, N. Shlezinger, H. V. Poor, Y. C. Eldar, and S. Cui, "Communication-efficient federated learning," *Proc. Nat. Acad. Sciences*, vol. 118, no. 17, Apr. 2021.

[32] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," *Proc. Int. Conf. Artificial Intel. Statis.*, pp. 3581–3607, May 2022.

[33] R. Wang and W.-T. Tsai, "Asynchronous federated learning system based on permissioned blockchains," *Sensors*, vol. 22, no. 4, p. 1672, Feb. 2022.

[34] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. Quek, "Asynchronous federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 6961–6978, Mar. 2022.

[35] H. Zhu, Y. Zhou, H. Qian, Y. Shi, X. Chen, and Y. Yang, "Online client selection for asynchronous federated learning with fairness consideration," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2493–2506, Oct. 2022.

[36] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," *Proc. Int. Symp. High-Perform. Parallel Distrib. Comput.*, pp. 125–136, Jun. 2020.

[37] X. Zhang, Y. Liu, J. Liu, A. Argyriou, and Y. Han, "D2D-Assisted Federated Learning in Mobile Edge Computing Networks," *IEEE Wireless Commun. Netw. Conf.*, pp. 1–7, Mar. 2021.

[38] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: a semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Computers*, vol. 70, no. 5, pp. 655–668, May 2020.

[39] S. Ko, K. Lee, H. Cho, Y. Hwang, and H. Jang, "Asynchronous federated learning with directed acyclic graph-based blockchain in edge computing: Overview, design, and challenges," *Elsevier Expert Syst. Applications*, p. 119896, Mar. 2023.

[40] L. You, S. Liu, Y. Chang, and C. Yuen, "A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24 199–24 211, Jul. 2022.

[41] 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.

[42] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: a random Fourier feature perspective," in *Proc. IEEE Stat. Signal Process. Workshop*, Jun. 2016, pp. 1–5.

[43] A. Rahimi, B. Recht *et al.*, "Random features for large-scale kernel machines." in *Proc. Conf. on Neural Inf. Proc. Syst.*, vol. 3, no. 4, Dec. 2007, pp. 1–5.

[44] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Jan. 2008.

[45] V. C. Gogineni, V. R. Elias, W. A. Martins, and S. Werner, "Graph diffusion kernel LMS using random Fourier features," *54th Asilomar Conf. Signals, Syst., Computers*, pp. 1528–1532, Nov. 2020.

[46] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: an introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, Mar. 2021.

[47] H. H. Yang, A. Arafa, T. Q. Quek, and H. V. Poor, "Age-based scheduling policy for federated learning in mobile edge networks," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, May 2020, pp. 8743–8747.

[48] C.-H. Hu, Z. Chen, and E. G. Larsson, "Scheduling and aggregation design for asynchronous federated learning over wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 874–886, Jul. 2023.

[49] V. C. Gogineni, S. P. Talebi, and S. Werner, "Performance of clustered multitask diffusion lms suffering from inter-node communication delays," *IEEE Trans. on Circuits and Syst. II: Express Briefs*, vol. 68, no. 7, pp. 2695–2699, 2021.

[50] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[51] 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, Apr. 1991.

[52] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-efficient online federated learning strategies for kernel regression," *IEEE Internet Things J.*, Nov. 2022.

[53] S. Dane, "CalCOFI, Over 60 years of oceanographic data," available at: https://www.kaggle.com/sohier/calcofi?select=bottle.csv.

[54] R. Jin, Y. Huang, X. He, H. Dai, and T. Wu, "Stochastic-sign SGD for federated learning with theoretical guarantees," *arXiv preprint arXiv:2002.10940*, Feb. 2020.

Francois Gauthier (Member, IEEE) received both the B.Sc. and M.Sc. degree in mathematics and computer science from École Nationale Supérieure d'Informatiques et de Mathématiques Appliquées de Grenoble. He is pursuing a Ph.D. degree at the Department of Electronic Systems at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. His research focuses on federated learning, differential privacy, communication efficiency, personalized learning, and reinforcement learning.

Vinay Chakravarthi Gogineni (Senior Member, IEEE) received the Bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University, Andhra Pradesh, India, in 2005, the Master's degree in communication engineering from VIT University, India, in 2008, and the Ph.D. degree in electronics and electrical communication engineering from Indian Institute of Technology Kharagpur, India in 2019. Currently, he is an Assistant Professor at SDU Applied AI and Data Science, The Maersk Mc-Kinney Moller Institute, University of Southern Denmark. Prior to this, he worked as a postdoctoral research fellow at NTNU and Simula, Norway. From 2008 to 2011, he was with a couple of MNCs in India. His research interests include statistical signal processing, distributed machine learning, geometric deep learning, and their application in healthcare. He was a recipient of the ERCIM Alain Bensoussan Fellowship in 2019 and the Best Paper Award at APSIPA ASC-2021, Tokyo, Japan. He is a member of the editorial board for the IEEE Sensors Journal.

Stefan Werner (Fellow, IEEE) received the M.Sc. Degree in electrical engineering from the Royal Institute of Technology, Stockholm, Sweden, in 1998, and a D.Sc. degree (Hons.) in electrical engineering from the Signal Processing Laboratory, Helsinki University of Technology, Espoo, Finland, in 2002. He is a Professor at the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), Director of IoT@NTNU, and Adjunct Professor at Aalto University in Finland. He was a visiting Melchor Professor with the University of Notre Dame during the summer of 2019 and an Adjunct Senior Research Fellow with the Institute for Telecommunications Research, University of South Australia, from 2014 to 2020. He held an Academy Research Fellowship, funded by the Academy of Finland, from 2009 to 2014. His research interests include adaptive and statistical signal processing, wireless communications, and security and privacy in cyber-physical systems. He is a member of the editorial boards for the EURASIP Journal of Signal Processing and the IEEE Transactions on Signal and Information Processing over Networks.

Yih-Fang Huang (Life Fellow, IEEE) is Professor of Electrical Engineering and Special Advisor to the Dean of the College of Engineering. He received his B.S.E.E. degree from National Taiwan University, M.S.E.E. degree from University of Notre Dame, M.A. and Ph.D. degrees from Princeton University. He was chair of Notre Dame's Electrical Engineering department from 1998 to 2006, and was Senior Associate Dean for Education and Undergraduate Programs for the College of Engineering from 2013 to 2023. His research lies in the area of statistical and adaptive signal processing and employs principles in mathematical statistics to solve signal detection and estimation problems that arise in various applications, including wireless communications, distributed sensor networks, smart electric power grid, etc. Dr. Huang received the Golden Jubilee Medal of the IEEE Circuits and Systems Society in 1999. He also served as Vice President in 1997-98 and was a Distinguished Lecturer for the same society in 2000-2001. He served as the lead Guest Editor for a Special Issue on Signal Processing in Smart Electric Power Grid of the IEEE Journal of Selected Topics in Signal Processing, December 2014. At the University of Notre Dame, he received Presidential Award in 2003, the Electrical Engineering department's Outstanding Teacher Award in 1994 and in 2011, the Rev. Edmund P. Joyce, CSC Award for Excellence in Undergraduate Teaching in 2011, and the Engineering College's Outstanding Teacher of the Year Award in 2013. In Spring 1993, Dr. Huang received the Toshiba Fellowship and was Toshiba Visiting Professor at Waseda University, Tokyo, Japan. From April to July 2007, he was a visiting professor at the Munich University of Technology, Germany. In Fall, 2007, Dr. Huang was awarded the Fulbright-Nokia scholarship for lectures/research at Helsinki University of Technology in Finland. He was appointed Honorary Professor in the College of Electrical Engineering and Computer Science at National Chiao-Tung University, Hsinchu, Taiwan, in 2014. Dr. Huang is a Life Fellow of the IEEE and a Fellow of the AAAS.

Anthony Kuh (Fellow, IEEE) received his B.S. in Electrical Engineering and Computer Science at the University of California, Berkeley in 1979, an M.S. in Electrical Engineering from Stanford University in 1980, and a Ph.D. in Electrical Engineering from Princeton University in 1987. He previously worked at AT&T Bell Laboratories and has been on the faculty in the Department of Electrical and Computer Engineering at the University of Hawai'i since 1986. He is currently a Professor and previously served as Department Chair. His research is in the area of neural networks and machine learning, adaptive signal processing, sensor networks, and renewable energy and smart grid applications. He won a National Science Foundation (NSF) Presidential Young Investigator Award and is an IEEE Fellow. He is currently serving as a program director for NSF in the Electrical, Communications, and Cyber Systems (ECCS) division working in the Energy, Power, Control, and Network (EPCN) group. He previously served for the IEEE Signal Processing Society on the Board of Governors as a Regional Director-at-Large Regions1-6, as a senior editor for the IEEE Journal of Selected Topics in Signal Processing, and as a member of the Awards Board. He previously also served as President of the Asia Pacific Signal and Information Processing Association (APSIPA).

Appendix E

Publication 5

P5 F. Gauthier, V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Clustered Graph Federated Personalized Learning", in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, October, 2022.

Therefore, the Lagrangian can be rewritten as

$$\mathcal{L}_\rho(\mathcal{V}_q, \mathcal{M}) = F(\mathbf{w}) + \langle \mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{e}, \boldsymbol{\lambda} \rangle + \frac{\rho}{2} \|\mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{e}\|^2. \tag{19}$$

B. Convergence Proof

We make the following assumptions to continue the analysis.

Assumption 1. *The functions $f_k(\cdot), k \in \{1, \ldots, K\}$, are convex and smooth.*

Using (19), and under Assumption 1, the steps of the PGFL algorithm without inter-cluster learning can be expressed as follows:

$$\begin{aligned} \nabla F(\mathbf{w}^{(n+1)}) + \mathbf{A}^\mathsf{T}\boldsymbol{\lambda}^{(n)} + \rho \mathbf{A}^\mathsf{T}\Big(\mathbf{A}\mathbf{w}^{(n+1)} + \mathbf{B}\mathbf{e}^{(n)}\Big) = 0, \\ \mathbf{B}^\mathsf{T}\boldsymbol{\lambda}^{(n)} + \rho \mathbf{B}^\mathsf{T}\Big(\mathbf{A}\widetilde{\mathbf{w}}^{(n+1)} + \mathbf{B}\mathbf{e}^{(n+1)}\Big) = 0, \\ \boldsymbol{\lambda}^{(n+1)} - \boldsymbol{\lambda}^{(n)} + \rho\Big(\mathbf{A}\widetilde{\mathbf{w}}^{(n+1)} + \mathbf{B}\mathbf{e}^{(n+1)}\Big) = 0. \end{aligned} \tag{20}$$

Similarly to [41], we introduce the following to simplify (20):

$$\begin{aligned} \mathbf{H}_+ &= \mathbf{A}_1^\mathsf{T} + \mathbf{A}_2^\mathsf{T}, & \mathbf{H}_- &= \mathbf{A}_1^\mathsf{T} - \mathbf{A}_2^\mathsf{T}, \\ \mathbf{L}_+ &= \frac{1}{2}\mathbf{H}_+\mathbf{H}_+^\mathsf{T}, & \mathbf{L}_- &= \frac{1}{2}\mathbf{H}_-\mathbf{H}_-^\mathsf{T}, \\ \boldsymbol{\alpha} &= \mathbf{H}_-^\mathsf{T}\mathbf{w}, & \mathbf{M} &= \frac{1}{2}(\mathbf{L}_+ + \mathbf{L}_-). \end{aligned}$$

Then, as derived in [41, Section II.B], (20) becomes

$$\begin{aligned} \nabla F(\mathbf{w}^{(n+1)}) + \boldsymbol{\alpha}^{(n)} + 2\rho\mathbf{M}\mathbf{w}^{(n+1)} - \rho\mathbf{L}_+\widetilde{\mathbf{w}}^{(n)} = 0, \\ \boldsymbol{\alpha}^{(n+1)} - \boldsymbol{\alpha}^{(n)} - \rho\mathbf{L}_-\widetilde{\mathbf{w}}^{(n+1)} = 0. \end{aligned} \tag{21}$$

As in [47, Lemma 1], the equations in (21) can be combined to obtain

$$\begin{aligned} \mathbf{w}^{(n+1)} =& \frac{\mathbf{M}^{-1}\nabla F(\mathbf{w}^{(n+1)})}{2\rho} + \frac{\mathbf{M}^{-1}\mathbf{L}_+\widetilde{\mathbf{w}}^{(n)}}{2} \\ & - \frac{\mathbf{M}^{-1}\mathbf{L}_-}{2}\sum_{s=0}^{n}\widetilde{\mathbf{w}}^{(s)}. \end{aligned} \tag{22}$$

Similarly to [47], by introducing the following:

$$\begin{aligned} \mathbf{Q} &= \sqrt{\mathbf{L}_-/2}, & \mathbf{r}^{(n)} &= \sum_{s=0}^{n}\mathbf{Q}\widetilde{\mathbf{w}}^{(s)}, \\ \mathbf{q}^{(n)} &= \begin{pmatrix} \mathbf{r}^{(n)} \\ \widetilde{\mathbf{w}}^{(n)} \end{pmatrix}, & \mathbf{G} &= \begin{bmatrix} \rho\mathbf{I} & 0 \\ 0 & \rho\mathbf{L}_+/2 \end{bmatrix}, \end{aligned}$$

(22) can be reformulated using [47, Lemma 2] as

$$\frac{\nabla F(\mathbf{w}^{(n+1)})}{\rho} + 2\mathbf{Q}\mathbf{r}^{(n+1)} + \mathbf{L}_+\Big(\mathbf{w}^{(n+1)} - \widetilde{\mathbf{w}}^{(n)}\Big) = 2\mathbf{M}\boldsymbol{\xi}^{(t+1)}. \tag{23}$$

Theorem I. *Under Assumption 1, if $\tau^{(n)} = \tau = 0, \forall n$, the proposed PGFL algorithm converges to the optimal solution of (2) in linear time for each cluster.*

Proof. Under Assumption 1, $F(\mathbf{w})$ is convex and smooth by composition and, therefore, differentiable. Using [48, Lemma 6] and [48, Theorem V] with a convex and smooth function $F(\mathbf{w})$ demonstrates that the proposed PGFL algorithm, without inter-cluster learning ($\tau = 0$), converges to the optimal solution of (2) in linear time for any given cluster. □

C. Impact of Inter-Cluster Learning

In situations with limited data, as demonstrated in Section V, employing inter-cluster learning ($\tau \neq 0$) can enhance performance compared to $\tau = 0$. This section establishes an upper bound on the disparity between the resulting cluster-specific personalized models obtained in scenarios with and without inter-cluster learning. It is worth noting that this bound can be controlled by properly choosing the sequence $\tau(n)$.

To do so, it is necessary to reformulate the client primal update using Assumption 1. The primal update for client $k \in \mathcal{C}_{s,(q)}$ is expressed as follows:

$$\begin{aligned} \mathbf{w}_k^{(n+1)} =& \arg\min_{\mathbf{w}} f_k(\mathbf{w}) - \Big\langle \boldsymbol{\varphi}_k^{(n)}, \mathbf{w} - \mathbf{w}_{s,(q)}^{(n)} \Big\rangle \\ & + \frac{\rho}{2}||\mathbf{w} - \mathbf{w}_{s,(q)}^{(n)}||^2, \end{aligned} \tag{24}$$

which, under Assumption 1, is equivalent to

$$\nabla f_k(\mathbf{w}_k^{(n+1)}) - \boldsymbol{\varphi}_k^{(n)} + \rho\Big(\mathbf{w}_k^{(n+1)} - \mathbf{w}_{s,(q)}^{(n)}\Big) = 0. \tag{25}$$

Further reformulation leads to the following:

$$\mathbf{w}_k^{(n+1)} = \mathbf{w}_{s,(q)}^{(n)} + \frac{1}{\rho}\boldsymbol{\varphi}_k^{(n)} - \frac{1}{\rho}\nabla f_k(\mathbf{w}_k^{(n+1)}). \tag{26}$$

By replacing $\mathbf{w}_k^{(n+1)}$ with (26) in (8), we obtain

$$\widehat{\mathbf{w}}_{s,(q)}^{(n)} = \frac{1}{|\mathcal{N}_s|}\sum_{p\in\mathcal{N}_s}\frac{1}{|\mathcal{C}_{p,(q)}|}\sum_{k\in\mathcal{C}_{p,(q)}}\Big(\mathbf{w}_{p,(q)}^{(n-1)} - \frac{1}{\rho}\nabla f_k(\mathbf{w}_k^{(n)})\Big). \tag{27}$$

Next, we investigate the effect of inter-cluster learning by comparing the performance of models obtained using the PGFL algorithm with and without inter-cluster learning. We shall prove that the difference between the resulting models is bounded and depends on both the inter-cluster learning parameter and the similarity of models between clusters.

Theorem II. *Given a sufficiently large penalty parameter ρ, for all iterations, server $s \in \mathcal{S}$ and cluster $q \in \mathcal{Q}$, the impact of inter-cluster learning after n iterations is bounded by*

$$\mathbb{E}\Big[||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}||_2^2\Big] \leqslant \sum_{i=1}^{n}\Big(\prod_{j=i+1}^{n}\big(1-\tau^{(j)}\big)\Big)\tau^{(i)}\eta, \tag{28}$$

where the expectation is taken with respect to the privacy-related noise added in (12) and the data observation noise, $\bar{\mathbf{w}}_{s,(q)}^{(n)}$ denotes the model obtained by the algorithm without inter-cluster learning, and η is the maximum cluster model distance, defined as:

$$\eta = \max_{q,r\in\mathcal{Q}}\Big\|\mathbf{w}_{(q)}^* - \mathbf{w}_{(r)}^*\Big\|_2^2, \tag{29}$$

with the models $\mathbf{w}_{(q)}^*, q \in \mathcal{Q}$ being the cluster-specific solutions of (2) with $\tau = 0$.

Proof. We prove this theorem by induction. With initial values $\mathbf{w}_{s,(q)}^{(0)} = \mathbf{0}$ and $\bar{\mathbf{w}}_{s,(q)}^{(0)} = \mathbf{0}$, one can write.

$$\begin{aligned} \mathbf{w}_{s,(q)}^{(1)} &= \Big(1-\tau^{(1)}\Big)\widehat{\mathbf{w}}_{s,(q)}^{(1)} + \frac{\tau^{(1)}}{Q-1}\sum_{r\in\mathcal{Q}\setminus q}\widehat{\mathbf{w}}_{s,(r)}^{(1)}, \\ \bar{\mathbf{w}}_{s,(q)}^{(1)} &= \frac{1}{|\mathcal{N}_s|}\sum_{p\in\mathcal{N}_s}\frac{1}{|\mathcal{C}_{p,(q)}|}\sum_{k\in\mathcal{C}_{p,(q)}}\Big(\bar{\mathbf{w}}_{p,(q)}^{(0)} - \frac{1}{\rho}\nabla f_k(\bar{\mathbf{w}}_k^{(1)})\Big), \end{aligned} \tag{30}$$

where, given that $\bar{\mathbf{w}}_{p,(q)}^{(0)} = \mathbf{w}_{p,(q)}^{(0)}$ and $\bar{\mathbf{w}}_{k}^{(0)} = \mathbf{w}_{k}^{(0)}$, and using (27), we have $\widehat{\mathbf{w}}_{s,(q)}^{(1)} = \bar{\mathbf{w}}_{s,(q)}^{(1)}$. Hence,

$$\bar{\mathbf{w}}_{s,(q)}^{(1)} - \mathbf{w}_{s,(q)}^{(1)} = \frac{\tau^{(1)}}{Q-1} \sum_{r \in \mathcal{Q}\setminus q} \Big(\bar{\mathbf{w}}_{s,(q)}^{(1)} - \widehat{\mathbf{w}}_{s,(r)}^{(1)} \Big). \tag{31}$$

Taking the expectation with respect to the privacy-related and observation noises, we can express this difference as a function of the inter-cluster learning parameter and the maximum cluster model distance.

$$\mathbb{E}[||\bar{\mathbf{w}}_{s,(q)}^{(1)} - \mathbf{w}_{s,(q)}^{(1)}||_2^2] \leqslant \tau^{(1)}\eta. \tag{32}$$

Further, we assume that (28) is satisfied for all iterations up to iteration $n-1$. For iteration n, we have

$$\begin{aligned}
\mathbf{w}_{s,(q)}^{(n)} &= \Big(1-\tau^{(n)}\Big)\widehat{\mathbf{w}}_{s,(q)}^{(n)} + \frac{\tau^{(n)}}{Q-1} \sum_{r \in \mathcal{Q}\setminus q} \widehat{\mathbf{w}}_{s,(r)}^{(n)}, \\
\bar{\mathbf{w}}_{s,(q)}^{(n)} &= \frac{1}{|\mathcal{N}_s|} \sum_{p \in \mathcal{N}_s} \frac{1}{|\mathcal{C}_{p,(q)}|} \sum_{k \in \mathcal{C}_{p,(q)}} \Big(\bar{\mathbf{w}}_{p,(q)}^{(n-1)} - \frac{1}{\rho} \nabla f_k(\bar{\mathbf{w}}_{k}^{(n)}) \Big),
\end{aligned} \tag{33}$$

where $\widehat{\mathbf{w}}_{s,(q)}^{(n)} \neq \bar{\mathbf{w}}_{s,(q)}^{(n)}$ since

$$\widehat{\mathbf{w}}_{s,(q)}^{(n)} = \frac{1}{|\mathcal{N}_s|} \sum_{p \in \mathcal{N}_s} \frac{1}{|\mathcal{C}_{p,(q)}|} \sum_{k \in \mathcal{C}_{p,(q)}} \Big(\mathbf{w}_{p,(q)}^{(n-1)} - \frac{1}{\rho} \nabla f_k(\mathbf{w}_{k}^{(n)}) \Big). \tag{34}$$

The difference is given by

$$\begin{aligned}
\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)} &= \Big(1-\tau^{(n)}\Big)\Big(\bar{\mathbf{w}}_{s,(q)}^{(n)} - \widehat{\mathbf{w}}_{s,(q)}^{(n)} \Big) \\
&\quad + \frac{\tau^{(n)}}{Q-1} \sum_{r \in \mathcal{Q}\setminus q} \Big(\bar{\mathbf{w}}_{s,(q)}^{(n)} - \widehat{\mathbf{w}}_{s,(r)}^{(n)} \Big),
\end{aligned} \tag{35}$$

with

$$\begin{aligned}
\bar{\mathbf{w}}_{s,(q)}^{(n)} - \widehat{\mathbf{w}}_{s,(q)}^{(n)} &= \frac{1}{|\mathcal{N}_s|} \sum_{p \in \mathcal{N}_s} \frac{1}{|\mathcal{C}_{p,(q)}|} \sum_{k \in \mathcal{C}_{p,(q)}} \Big(\bar{\mathbf{w}}_{p,(q)}^{(n-1)} \\
&\quad - \mathbf{w}_{p,(q)}^{(n-1)} - \frac{1}{\rho} \nabla f_k(\bar{\mathbf{w}}_{k}^{(n)}) + \frac{1}{\rho} \nabla f_k(\mathbf{w}_{k}^{(n)}) \Big).
\end{aligned} \tag{36}$$

We note that the expectation of $||\bar{\mathbf{w}}_{p,(q)}^{(n-1)} - \mathbf{w}_{p,(q)}^{(n-1)}||_2^2$ with respect to the privacy-related and observation noises is identical for all servers. Therefore, since (28) is satisfied for iteration $n-1$ for all servers, given a sufficiently large penalty parameter ρ, and taking the expectation with respect to the privacy-related and observation noises, we have

$$\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \widehat{\mathbf{w}}_{s,(q)}^{(n)}||_2^2 \leqslant \mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n-1)} - \mathbf{w}_{s,(q)}^{(n-1)}||_2^2. \tag{37}$$

Combining (35) and (37), we will have

$$\begin{aligned}
\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}||_2^2 &\leqslant (1-\tau^{(n)})\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n-1)} - \mathbf{w}_{s,(q)}^{(n-1)}||_2^2 \\
&\quad + \frac{\tau^{(n)}}{Q-1} \sum_{r \in \mathcal{Q}\setminus q} \mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \widehat{\mathbf{w}}_{s,(r)}^{(n)}||_2^2,
\end{aligned} \tag{38}$$

which, using the maximum cluster model distance, yields

$$\begin{aligned}
\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}||_2^2 &\leqslant \Big(1-\tau^{(n)}\Big)\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n-1)} - \mathbf{w}_{s,(q)}^{(n-1)}||_2^2 \\
&\quad + \tau^{(n)}\eta.
\end{aligned} \tag{39}$$

Given (28) for iteration $n-1$, we have

$$\begin{aligned}
\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}||_2^2 &\leqslant \Big(1-\tau^{(n)}\Big) \sum_{i=1}^{n-1} \Big(\prod_{j=i+1}^{n-1} (1-\tau^{(j)}) \Big) \tau^{(i)}\eta \\
&\quad + \tau^{(n)}\eta, \\
&\leqslant \sum_{i=1}^{n} \Big(\prod_{j=i+1}^{n} (1-\tau^{(j)}) \Big) \tau^{(i)}\eta.
\end{aligned} \tag{40}$$

That is, (28) is satisfied for iteration n.

By the principle of induction, (28) is satisfied for all iterations, server $s \in \mathcal{S}$ and cluster $q \in \mathcal{Q}$.

□

Corollary. *If $\tau^{(i)} = 0, \forall i < n$ and $\tau^{(n)} \neq 0$, the impact of a single iteration of inter-cluster learning is bounded by*

$$\mathbb{E}||\bar{\mathbf{w}}_{s,(q)}^{(n)} - \mathbf{w}_{s,(q)}^{(n)}||_2^2 \leqslant \tau^{(n)}\eta, \tag{41}$$

where $\bar{\mathbf{w}}_{s,(q)}^{(n)}$ denotes a model obtained without inter-cluster learning, η is as defined in Theorem II, and the expectation is taken with respect to the privacy-related and observation noises.

Theorem II bounds the difference in the resulting models with and without inter-cluster learning. Combining Theorems I and II, the resulting models obtained by the algorithms are guaranteed to reside within a neighborhood of the optimal solution of (2) with $\tau = 0$. The size of this neighborhood can be adjusted by selecting the sequence $\tau^{(n)}$. When ample data is available, the algorithm converges to a satisfactory solution within this neighborhood. However, in cases of limited data, the solution of (2) with $\tau = 0$ may be inadequate. In such situations, inter-cluster learning becomes crucial, allowing the proposed algorithm to achieve higher accuracy, as demonstrated in Section V. By exploiting inter-cluster learning, the algorithm effectively overcomes the limitations imposed by scarce data, leading to improved performance.

IV. Privacy Analysis

This section focuses on quantifying the local privacy protection provided by the proposed PGFL algorithm. To achieve this, we begin by calculating the l_2-norm sensitivity, which quantifies the variation in output resulting from a change in an individual data sample. Once we have established the l_2-norm sensitivity, we proceed to adjust the noise variance added to the primal variables, ensuring satisfactory protection.

Definition. *The l_2-norm sensitivity is defined by*

$$\Delta_{k,2}^{(n)} = \max_{\mathcal{D}_k, \mathcal{D}_l} \Big\| \mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}_l}^{(n)} \Big\| \tag{42}$$

where $\mathbf{w}_{k,\mathcal{D}_k}^{(n)}$ and $\mathbf{w}_{k,\mathcal{D}_l}^{(n)}$ denote the local primal variables obtained from two neighboring data sets $\mathcal{D}_k$ and $\mathcal{D}_l$, which differ in only one data sample.

Assumption 3. *The functions $\ell_k(\cdot)$, $k \in \mathcal{C}$, have bounded gradients. That is, for $k \in \mathcal{C}$ there exists a constant C_k such that $||\nabla \ell_k(\cdot)|| \leqslant C_k$.*

Lemma 1. *Under Assumption* 3*, the l_2-norm sensitivity for a client k is given by*

$$\Delta_{k,2}^{(n)} = \max_{\mathcal{D}_k, \mathcal{D}_l} ||\mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}_l}^{(n)}|| = \frac{2C_k}{\rho D_k}. \tag{43}$$

Proof. We consider two neighboring data sets for a client k, $\mathcal{D}_k$ and $\mathcal{D}_l$, both of cardinality D_k. For simplicity, we assume that they differ on the last data sample. We denote $\mathbf{w}_{k,\mathcal{D}_k}^{(n)}$ the model obtained using the initial data set, and $\mathbf{w}_{k,\mathcal{D}_l}^{(n)}$ the model obtained using the alternative data set. Those are obtained, according to (4), by:

$$\begin{aligned}\mathbf{w}_{k,\mathcal{D}_k}^{(n)} &= \arg\min_{\mathbf{w}} \frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}) + \frac{\lambda}{|\mathcal{C}_s|} R(\mathbf{w}) \\ &\quad - \left\langle \boldsymbol{\varphi}_k^{(n-1)}, \mathbf{w} - \mathbf{w}_{s,(q)}^{(n-1)} \right\rangle + \frac{\rho}{2} ||\mathbf{w} - \mathbf{w}_{s,(q)}^{(n-1)}||^2, \\ \mathbf{w}_{k,\mathcal{D}_l}^{(n)} &= \arg\min_{\mathbf{w}} \frac{\lambda}{|\mathcal{C}_s|} R(\mathbf{w}) \\ &\quad + \frac{1}{D_k} \Big(\sum_{i=1}^{D_k - 1} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}) + \ell_k(\mathbf{x}'_{k,D_k}, y'_{k,D_k}; \mathbf{w}) \Big) \\ &\quad - \left\langle \boldsymbol{\varphi}_k^{(n-1)}, \mathbf{w} - \mathbf{w}_{s,(q)}^{(n-1)} \right\rangle + \frac{\rho}{2} ||\mathbf{w} - \mathbf{w}_{s,(q)}^{(n-1)}||^2.\end{aligned}$$

Using (26), that we recall:

$$\mathbf{w}_k^{(n)} = \mathbf{w}_{s,(q)}^{(n-1)} + \frac{1}{\rho} \boldsymbol{\varphi}_k^{(n-1)} - \frac{1}{\rho} \nabla f_k(\mathbf{w}_k^{(n)}), \tag{44}$$

we can derive:

$$\begin{aligned}&||\mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}_l}^{(n)}|| = \\ &\left\| \frac{1}{\rho D_k} (\nabla \ell_k(\mathbf{x}_{k,D_k}, y_{k,D_k}; \mathbf{w}_k) - \nabla \ell_k(\mathbf{x}'_{k,D_k}, y'_{k,D_k}; \mathbf{w}_k)) \right\|,\end{aligned} \tag{45}$$

which, under Assumption 3, provides a value for the l_2-norm sensitivity:

$$\max_{\mathcal{D}_k, \mathcal{D}_l} ||\mathbf{w}_{k,\mathcal{D}_k}^{(n)} - \mathbf{w}_{k,\mathcal{D}_l}^{(n)}|| = \frac{2C_k}{\rho D_k}. \tag{46}$$

□

With the l_2-norm sensitivity, we can establish the relation between the noise variance added in (12) and the privacy parameter $\phi_k^{(n)}$ as well as prove the privacy guarantee of the algorithm in terms of zCDP.

Theorem III. *Under Assumption* 3*, PGFL satisfies dynamic $\phi_k^{(n)}$-zCDP with the relation between the privacy parameter and the perturbation noise variance given by*

$$\delta_k^{2(n)} = \frac{\Delta_{k,2}^{2(n)}}{2\phi_k^{(n)}}. \tag{47}$$

Proof. For any client k and iteration n, the perturbed primal update is obtained with (12). That is, it is equivalent to $\widetilde{\mathbf{w}}_k^{(n)} \sim \mathcal{N}(\mathbf{w}_k^{(n)}, \delta_k^{2(n)} \mathbf{I})$. The result in [36, Proposition 6], states that a sensitivity-Δ query q releasing an output $\mathcal{N}(q(x), \delta^2)$ from an input x satisfies $(\Delta^2/2\delta^2) - zCDP$. Thus, the PGFL algorithm satisfies the dynamic $\phi_k^{(n)}$-zCDP with $\phi_k^{(n)} = \frac{\Delta_{k,2}^{2(n)}}{2\delta_k^{2(n)}}$. □

Theorem III gives the relationship between the noise perturbation variance and the privacy protection at a given iteration. Since the proposed algorithm is iterative in nature and models are exchanged several times with the servers, one should consider the total privacy loss throughout the learning process. To this aim, we establish the following theorem.

Theorem IV. *Under Assumption* 3 *and for a final iteration N, the PGFL algorithm satisfies ϕ_k^{total}-zCDP throughout the entire computation for each client k, with ϕ_k^{total} given by*

$$\phi_k^{\text{total}} = \sum_{n=1}^{N} \phi_k^{(n)}. \tag{48}$$

Proof. This theorem results from the use of [36, Lemma 7] N times over. □

V. Numerical Simulations

This section illustrates the performance of the proposed PGFL algorithm for solving regression and classification tasks.

A. Experiments for Regression

We consider a graph federated network consisting of $|\mathcal{S}| = 10$ servers, each having access to $|\mathcal{C}_s| = 15$ clients, for a total of $|\mathcal{C}| = 150$ clients. The set of servers and their communication channels form a random connected graph where the average node degree is three. Each client has access to a random number of noisy data samples between $D_k = 2$ and $D_k = 9$, each composed of a vector $\mathbf{x}_{k,i}$ of dimension $d = 60$ and a response scalar $y_{k,i}$. Doing so, each cluster is globally observable but not locally at any given client or set $\mathcal{C}_s, s \in \mathcal{S}$. The servers implement random scheduling of clients to reduce the communication load [49]. In particular, at every global iteration, each server randomly selects a subset of three clients to participate in the learning process.

The clients of the network are randomly split between $Q = 3$ clusters. Clients of a given cluster solve the ridge regression problem with data generated from an original model $\mathbf{w}_{(q)}^*$, obtained with $\mathbf{w}_{(q)}^* = \mathbf{w}_0^* + \gamma \mathbf{w}_0^*$ with $\gamma \sim \mathcal{U}(-0.15, 0.15)$, where $\mathbf{w}_0^*$ is a base model. In doing so, the learning tasks of different clusters share the same objective functions but have different, albeit related, data distributions. The loss and regularizer functions are given by

$$\begin{aligned}\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k) &= ||\mathbf{y}_k - \mathbf{X}_k \mathbf{w}_k||^2, \\ R(\mathbf{w}_k) &= ||\mathbf{w}_k||^2.\end{aligned} \tag{49}$$

Performance is evaluated by computing the normalized mean squared deviation (NMSD) of the local models with respect to the corresponding cluster-specific original model used to generate the data, $\mathbf{w}_{(q)}^*$ for $k \in \mathcal{C}_{(q)}$. It is given by:

$$\gamma^{(n)} = \frac{1}{|\mathcal{C}|} \sum_{q=1}^{|\mathcal{Q}|} \sum_{k \in \mathcal{C}_{(q)}} \frac{\left\| \mathbf{w}_k^{(n)} - \mathbf{w}_{(q)}^* \right\|_2^2}{\left\| \mathbf{w}_{(q)}^* \right\|_2^2}, \tag{50}$$

where the result is averaged over several Monte Carlo iterations. The proposed algorithm is compared with various existing algorithms. The ClusterFL algorithm, defined in [50], implements conventional personalized FL with inter-cluster learning. For a fair comparison, the ClusterFL algorithm has been modified to leverage similarity among tasks in the same manner as the PGFL algorithm. The GFL algorithm, defined in [5], implements single-task graph FL in a privacy-preserving manner. To ensure a fair comparison, the ClusterFL and GFL algorithms have been modified to ensure privacy in the same manner as the PGFL algorithm. Furthermore, the algorithms are set to observe the same initial convergence rate whenever possible. For most experiments, the learning curves are displayed as plots of the NMSD versus the iteration index.

We first consider an ideal setting wherein all algorithms are evaluated without privacy considerations ($\boldsymbol{\xi}^{(n)} = \mathbf{0}$, $\forall n$)) and client scheduling. In this scenario, the inter-cluster parameter $\tau^{(n)}$ of the PGFL algorithm was kept fixed throughout the learning, specifically, $\tau^{(n)} = 0$ and $\tau^{(n)} = 0.4$. Figure 1 shows the learning curves for the GFL, ClusterFL, and PGFL algorithms. The results illustrate the superiority of the proposed PGFL algorithm over GFL, as cluster-specific learning tasks benefit significantly from personalized models tailored to each cluster. We also see that incorporating inter-cluster learning results in improved convergence speed and steady-state accuracy. Furthermore, the performance of the ClusterFL algorithm is notably poor in this setting, emphasizing the importance of using the graph federated architecture when data is scarce. Leveraging the model similarities improves learning speed and accuracy by compensating for data scarcity. In addition, isolated servers whose clients lack sufficient data to achieve satisfactory accuracy independently reinforce the necessity of the graph federated architecture.

Next, we modify the setting to incorporate client scheduling and evaluate the aforementioned algorithms with reduced communication load. Figure 2 shows the learning curves for the GFL, ClusterFL, and PGFL algorithms with client scheduling. In this figure and the ones below, 3 clients out of 15 are randomly selected to participate by each server at every iteration, reducing the communication load by 80% for every algorithm. We observe that the PGFL algorithm exhibits slower convergence and higher steady-state NMSD when utilizing client scheduling. And we note that GFL performs better with client scheduling. The performance degradation for the PGFL algorithm is due to the lower client participation resulting in a smaller quantity of data being utilized. The better performance of GFL in this setting is due to the imbalance of cluster representation in the universal model, which benefits the participating clients on average.

Finally, we evaluate the aforementioned algorithms in a setting with client scheduling and privacy protection. All of the algorithms utilize zCDP with the noise perturbation presented in (12) and the parameters $\phi_k^{(0)} = 0.001, \forall k$ and $\zeta = 0.99$. Hence, all the algorithms satisfy ϕ_k^{final}-zCDP throughout the computation with $\phi_k^{\text{final}} = 0.095, \forall k$. Figure 3 shows the learning curves for the GFL, ClusterFL, and PGFL algorithms with client scheduling and privacy. We observe that the noise perturbation associated with differential privacy significantly

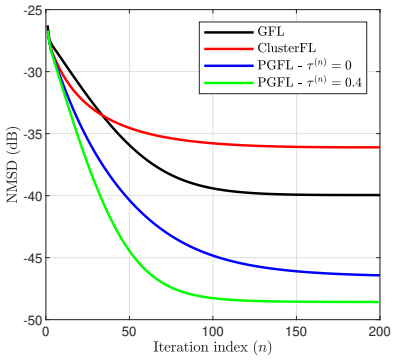


Fig. 1: Learning curves of the PGFL algorithm with a fixed inter-cluster learning parameter and the FedAvg algorithm, without client scheduling or privacy.

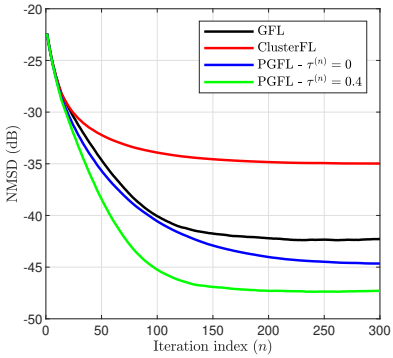


Fig. 2: Learning curves of the PGFL algorithm with a fixed inter-cluster learning parameter and the FedAvg algorithm, considering client scheduling and without privacy.

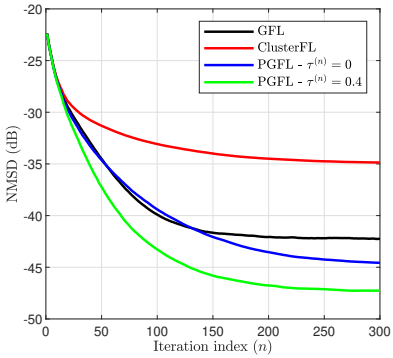


Fig. 3: Learning curves of the PGFL algorithm with a fixed inter-cluster learning parameter and the FedAvg algorithm, considering client scheduling and privacy.

reduces the convergence speed of all the simulated algorithms. However, we note that the NMSD after 300 iterations is nearly identical to the one in Fig. 2. This behavior is explained by the use of zCDP, in which the variance of the noise perturbation starts high and decreases linearly throughout the learning process.

Further, we illustrate the importance of carefully choosing the value of the inter-cluster learning parameter. In Fig. 4, we simulated the proposed PGFL algorithm for various fixed $\tau^{(n)}$ values and displayed the NMSD after 200 iterations. For instance, the NMSD for $\tau^{(n)} = 0.4$ corresponds to the result obtained in Fig. 3. This figure confirms that inter-cluster learning has the potential to increase learning performance by alleviating data scarcity, as the PGFL algorithm achieves lower NMSD with $\tau^{(n)} \in (0.1, 0.5)$ than with $\tau^{(n)} = 0$. It also shows that the inter-cluster learning parameter must be carefully selected, as a value too large for the setting leads to performance degradation.

We then illustrate an alternative use of inter-cluster learning. For this experiment, the difference between the data distribution of the different clusters has been increased. Precisely, the datasets were simulated with the models obtained by $\mathbf{w}_{(q)} = \mathbf{w}_0 + \gamma \mathbf{w}_0$ with $\gamma \sim \mathcal{U}(-0.5, 0.5)$. The learning curves are presented in Fig. 5. We observed that, because of the higher cluster dissimilarity, inter-cluster learning degrades steady-state NMSD; this is observed in the learning curves for PGFL with $\tau^{(n)} = 0$ and $\tau^{(n)} = 0.4$. However, by mitigating data scarcity within a cluster, inter-cluster learning improves the initial convergence rate. To benefit from an improved initial convergence rate and avoid steady-state performance degradation, it is possible to reduce the inter-cluster learning parameter progressively. Doing so, the PGFL algorithm with time-varying $\tau^{(n)} = 0.4 \times 0.98^n$ has the same initial convergence rate as the PGFL algorithm with fixed $\tau = 0.4$ and attains near-identical steady-state NMSD as the PGFL algorithm with fixed $\tau = 0$.

Finally, we study the impact of privacy protection on the steady-state NMSD of the PGFL algorithm. Fig. 6 shows the NMSD after 200 iterations versus the initial value of the privacy parameter ϕ_0 for a decaying rate of $\zeta = 0.99$. Note that, as seen in Theorem III, a lower value of ϕ_0 ensures more privacy. We observe that for smaller values of ϕ_0, the steady-state NMSE of the PGFL algorithm is higher. In fact, a lower total privacy loss bound leads to higher perturbation noise variance and diminishes the learning performance of the algorithm. Similarly, Fig. 7 shows the NMSD after 200 iterations versus the variance decrease rate ζ for an initial privacy value of $\phi_0 = 0.001$. The lower the decrease rate, the faster the privacy protection weakens, and the lower the steady-state NMSE of the algorithm as more information is exchanged among clients. On the other hand, a decrease rate close to 1 ensures better privacy protection but comes at the cost of lower accuracy.

B. Experiments for Classification on the MNIST Dataset

The following experiments were conducted on the MNIST handwritten digits dataset [51]. In those experiments, the

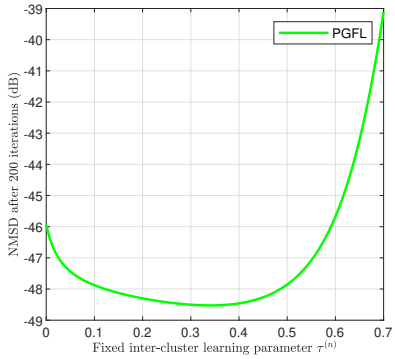


Fig. 4: NMSD after 200 iterations vs. fixed inter-cluster learning parameter $\tau^{(n)}$ values for the PGFL algorithm with client scheduling and privacy .

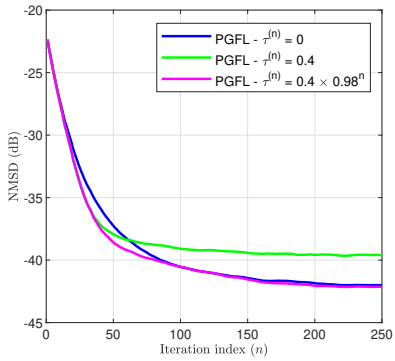


Fig. 5: Learning curves of the PGFL algorithm with fixed and time-varying inter-cluster learning parameter $\tau^{(n)}$ in a setting with low cluster similarity, considering client scheduling and privacy.

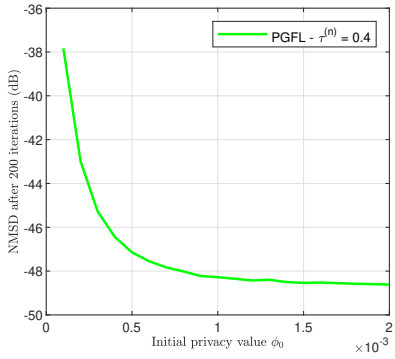


Fig. 6: Privacy-accuracy trade-off of the PGFL algorithm for ϕ_0 with a fixed inter-cluster learning parameter, considering client scheduling.

learning tasks of the clients associated with different clusters share the same data but have different, related, objective functions. The structure of the server network, as well as the number of clients per server, are identical to the experiments for regression. In the following experiments, the clients of a given cluster use the ADMM for logistic regression to differentiate between two classes. The loss function for the logistic regression is given by

$$\log[\ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}_k)] = \frac{-1}{D_k} \sum_{i=1}^{D_k} \Big(y_{k,i} \log[y'_{k,i}] + (1 - y_{k,i}) \log[1 - y'_{k,i}] \Big), \quad (51)$$

with

$$y'_{k,i} = \frac{1}{1 + \exp(-\mathbf{w}_k^{\mathsf{T}} \mathbf{x}_{k,i})}. \quad (52)$$

We simulated the PGFL algorithm in the context of classification with client scheduling, privacy, a fixed inter-cluster learning parameter $\tau^{(n)} = \tau = 0.4$, and without inter-cluster learning $\tau^{(n)} = 0$. Figure 8 shows the test accuracy versus iteration index in a setting where the clients of a given cluster must differentiate between two classes composed of a single digit. Each client receives between $D_k = 2$ and $D_k = 4$ data samples composed of two MNIST images. The clients of cluster 1 have access to images of the digits $\{1\}$ and $\{8\}$. The clients of clusters 2 and 3 have access to images of the digits $\{1\}$ and $\{9\}$, and $\{7\}$ and $\{8\}$, respectively. Given that the clients of different clusters must differentiate between different digits, the similarity between the learning task is limited. Nevertheless, we observe that inter-cluster learning does improve the accuracy of the PGFL algorithm in this setting.

Further, we modified the setting so that the clusters exhibit more similarity. Figure 9 shows the test accuracy versus iteration index in a setting where the clients of a given cluster must differentiate between two classes composed of triplets of digits. Each client receives between $D_k = 6$ and $D_k = 12$ data samples, each composed of two triplets of MNIST images. The clients of cluster 1 must differentiate between the classes $\{1, 2, 3\}$ and $\{6, 7, 8\}$, the clients of cluster 2 between $\{1, 2, 3\}$ and $\{7, 8, 9\}$, and the clients of cluster 3 between $\{1, 2, 3\}$ and $\{6, 8, 9\}$. We observe that, in this setting, inter-cluster learning significantly improves the accuracy of the PGFL algorithm.

Finally, we utilize the previous setting and evaluate the impact of the value of the inter-cluster learning parameter $\tau^{(n)}$ on the accuracy achieved by the PGFL algorithm in the context of classification. Figure 10 displays the accuracy achieved by the PGFL algorithm after 100 iterations versus the value of the inter-cluster learning parameter in the context of the classification task of Fig. 9. We observe that, in this setting where the similarity among the learning tasks is high, medium and large fixed values for $\tau^{(n)}$ lead to significant accuracy improvement. However, very large values lead to performance degradation, similar to Fig. 4.

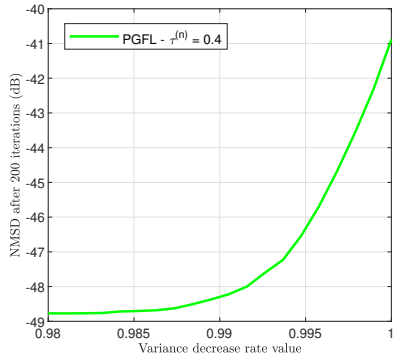


Fig. 7: Privacy-accuracy trade-off of the PGFL algorithm for ζ with a fixed inter-cluster learning parameter, considering client scheduling.

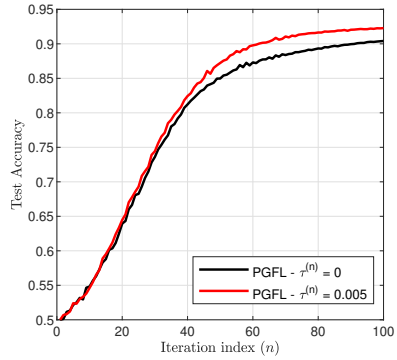


Fig. 8: Test accuracy curve of the PGFL algorithm with a fixed inter-cluster learning parameter on MNIST, considering client scheduling and privacy, with low cluster similarity.

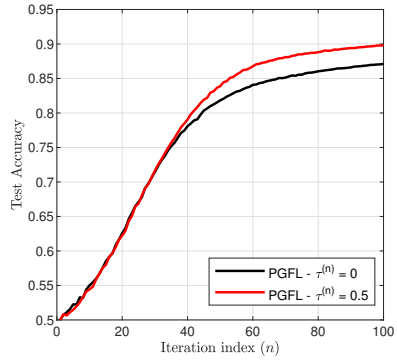


Fig. 9: Test accuracy curve of the PGFL algorithm with a fixed inter-cluster learning parameter on MNIST, considering client scheduling and privacy, with high cluster similarity.

C. Experiments for Classification on the MedMNIST Dataset

To demonstrate the proposed method of utilizing inter-cluster learning to palliate data scarcity and improve learning performance in real-life applications, two experiments are conducted on the OrganAMNIST dataset, part of the biomedical MedMNIST dataset [52]. The OrganAMNIST dataset contains lightweight images of 11 different organs labeled by type. It comprises more than 58000 data samples split into training, validation, and testing data. We use the proposed method to improve classification accuracy in the following setting. The server network and the loss function are identical to previous experiments; however, only three clients are associated with each server, each client having access to two data samples. In both experiments, clients of a given cluster are tasked with differentiating between two types of organs. Different clusters are associated with different pairs of organs, and inter-cluster learning is utilized to improve classification accuracy by leveraging the similarity between some of the organs.

In the first experiment, the three clusters are given similar learning tasks. In particular, one of the elements of each pair of organs is identical. Cluster 1 differentiates between the right lung and the left lung, cluster 2 between the liver and the left lung, and cluster 3 between the right kidney and the left lung. Figure 11 shows the test accuracy versus iteration index. We observe that a large amount of inter-cluster learning leads to significantly improved performances, increasing classification accuracy by about 5%.

In the next experiment, the learning tasks associated with each cluster are less similar than in the previous experiment. They share only the vague shape of the classified organs. Cluster 1 differentiates between the spleen and the left lung, cluster 2 between the left kidney and the bladder, and cluster 3 between the right kidney and the right lung. Due to the lower cluster similarity, we utilize a decaying inter-cluster learning parameter to preserve steady-state accuracy. Figure 12 shows the test accuracy versus iteration index. We observe that a medium decay rate of the inter-cluster learning parameter can improve the learning speed, boosting classification accuracy by about 2%.

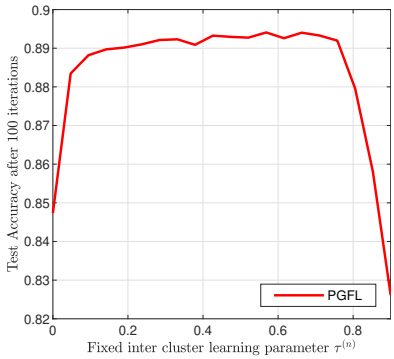


Fig. 10: Test accuracy of the PGFL algorithm on MNIST after 100 iterations vs. fixed inter-cluster learning parameter $\tau^{(n)}$, considering client scheduling and privacy..

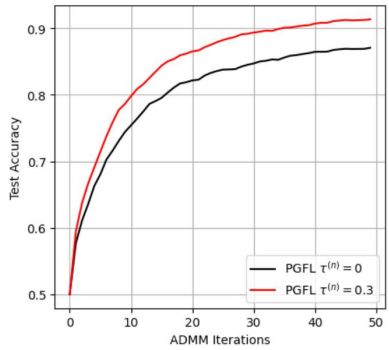


Fig. 11: Test accuracy curve of the PGFL algorithm with a fixed inter-cluster learning parameter on MedMNIST, considering privacy, with high cluster similarity.

VI. Conclusions

This paper proposed a framework for personalized graph federated learning in which distributed servers collaborate with each other and their respective clients to learn cluster-specific personalized models. The proposed framework leverages the similarities among clusters to improve learning speed and alleviate data scarcity. Further, this framework is implemented with the ADMM as a local learning process and with local zero-concentrated differential privacy to protect the participants' data from eavesdroppers. Our mathematical analysis showed that this algorithm converges to the exact optimal solution for each cluster in linear time and that utilizing inter-cluster learning leads to an alternative output whose distance to the original solution is bounded by a value that can be adjusted with the inter-cluster learning parameter sequence. Finally, numerical simulations showed that the proposed method is capable of leveraging the graph federated architecture and the similarity between the clusters' learning tasks to improve learning performance.

References

[1] F. Gauthier, V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Clustered graph federated personalized learning," *Asilomar Conf. Signals Syst. Comput.*, pp. 744–748, Oct. 2022.

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

[3] 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, May 2020.

[4] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, Jun. 2020.

[5] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," *IEEE Int. Workshop Signal Process. Advances Wireless Commun.*, pp. 131–135, Sep. 2021.

[6] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," *IEEE Int. Conf. Commun.*, pp. 1–6, Jun. 2020.

[7] Y. Sarcheshmehpour, M. Leinonen, and A. Jung, "Federated learning from big data over networks," *IEEE Int. Conf. Acoust., Speech Signal Process.*, pp. 3055–3059, Jan. 2021.

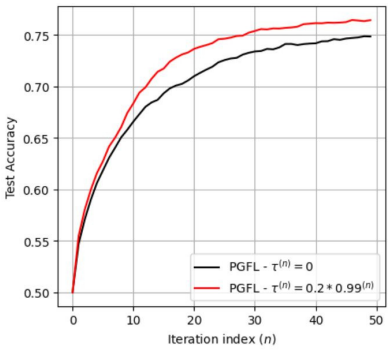


Fig. 12: Test accuracy curve of the PGFL algorithm with a varying inter-cluster learning parameter on MedMNIST, considering privacy, with low cluster similarity.

[8] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Decentralized graph federated multitask learning for streaming data," *Annu. Conf. Inf. Sciences Syst.*, pp. 101–106, Mar. 2022.

[9] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Ind. Eng.*, vol. 149, p. 106854, 2020.

[10] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

[11] S. Boll and J. Meyer, "Health-X dataLOFT: a sovereign federated cloud for personalized health care services," *IEEE MultiMedia*, vol. 29, no. 1, pp. 136–140, May 2022.

[12] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Networks Learning Syst.*, Mar. 2022.

[13] C. T Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances Neural Inf. Process. Syst.*, vol. 33, pp. 21 394–21 405, 2020.

[14] C. J. Felix, J. Ye, and A. Kuh, "Personalized learning using kernel methods and random fourier features," *Int. Joint Conf. Neural Netw.*, pp. 1–5, Jul. 2022.

[15] V. C. Gogineni, S. Werner, F. Gauthier, Y.-F. Huang, and A. Kuh, "Personalized online federated learning for IoT/CPS: challenges and future directions," *IEEE Internet Things Mag.*, 2022.

[16] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: a meta-learning approach," *arXiv preprint arXiv:2002.07948*, Feb. 2020.

[17] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *arXiv preprint arXiv:2003.13461*, Mar. 2020.

[18] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Jun. 2014.

[19] V. C. Gogineni and M. Chakraborty, "Improving the performance of multitask diffusion APA via controlled inter-cluster cooperation," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 67, no. 3, pp. 903–912, Dec. 2019.

[20] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances Neural Info. Process. Syst.*, vol. 33, pp. 19 586–19 597, 2020.

[21] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," *Proc.IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 2749–2758, 2021.

[22] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3710–3722, Aug. 2020.

[23] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances Neural Inf. Process. Syst.*, vol. 30, . 2017.

[24] R. Li, F. Ma, W. Jiang, and J. Gao, "Online federated multitask learning," *IEEE Int. Conf. Big Data*, pp. 215–220, Dec. 2019.

[25] A. Taïk and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," *IEEE Int. Conf. Commun.*, Jun. 2020.

[26] F.-Z. Lian, J.-D. Huang, J.-X. Liu, G. Chen, J.-H. Zhao, and W.-X. Kang, "FedFV: A personalized federated learning framework for finger vein authentication," *Machine Intel. Research*, pp. 1–14, Jan. 2023.

[27] S. Wang, S. Hosseinalipour, M. Gorlatova, C. G. Brinton, and M. Chiang, "UAV-assisted online machine learning over multi-tiered networks: A hierarchical nested personalized federated learning approach," *IEEE Trans. Netw. Service Manage.*, Oct. 2022.

[28] A. M. G. Salem, A. Bhattacharyya, M. Backes, M. Fritz, and Y. Zhang, "Updates-leak: Data set inference and reconstruction attacks in online learning," *29th USENIX Secur. Symp.*, pp. 1291–1308, Aug. 2020.

[29] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, Apr. 2020.

[30] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Conf. Theory Cryptography*, 2006, pp. 265–284.

[31] J. Cortés, G. E. Dullerud, S. Han, J. Le Ny, S. Mitra, and G. J. Pappas, "Differential privacy in control and network systems," in *IEEE 55th Conf. Decision Control*, Dec. 2016, pp. 4252–4272.

[32] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, pp. 211–407, Aug. 2014.

[33] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, Q. S. Tony Quek, and H. Vincent Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, Apr. 2020.

[34] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," *Int. Conf. Machine Learn.*, pp. 1376–1385, Jun. 2015.

[35] C. Dwork and G. N. Rothblum, "Concentrated differential privacy," *arXiv preprint arXiv:1603.01887*, 2016.

[36] M. Bun and T. Steinke, "Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds," in *Theory of Cryptography Conf.* Springer, 2016, pp. 635–658.

[37] T. Zhang and Q. Zhu, "A Dual Perturbation Approach for Differential Private ADMM-Based Distributed Empirical Risk Minimization," *Proc. ACM Workshop Artif. Intell. Secur.*, p. 129–137, 2016.

[38] S. Zhou and G. Y. Li, "Federated learning via inexact ADMM," *IEEE Trans. Pattern Anal. Machine Intell.*, Feb. 2023.

[39] Y. Chen, R. S. Blum, and B. M. Sadler, "Communication efficient federated learning via ordered ADMM in a fully decentralized setting," *Conf. Inf. Sciences Syst.*, pp. 96–100, Mar. 2022.

[40] S. Yue, J. Ren, J. Xin, S. Lin, and J. Zhang, "Inexact-ADMM based federated meta-learning for fast and continual edge learning," in *Proc. Int. Symp. Theory Algorithmic Found. Protocol Des. Mobile Netw. Mobile Comput.* Assoc. Comput. Mach., Jul. 2021, p. 91–100.

[41] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, 2014.

[42] J. Ding, X. Zhang, M. Chen, K. Xue, C. Zhang, and M. Pan, "Differentially Private Robust ADMM for Distributed Machine Learning," in *2019 IEEE Int. Conf. Big Data*, Dec. 2019, pp. 1302–1311.

[43] S. Ben-David and R. Schuller, "Exploiting task relatedness for multiple task learning," *Conf. Learn. Theory Kernel Workshop*, pp. 567–580, Aug. 2003.

[44] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, "Splitting Methods in Communication, Imaging, Science, and Engineering," in *Scientific Computation*. Springer Int. Publishing, Jan. 2017, pp. 461–497.

[45] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Jun. 2014.

[46] V. C. Gogineni and M. Chakraborty, "Diffusion affine projection algorithm for multitask networks," *Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, pp. 201–206, Nov. 2018.

[47] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, "Robust decentralized learning using ADMM with unreliable agents," *arXiv preprint arXiv:1710.05241*, Oct. 2017.

[48] J. Ding, Y. Gong, M. Pan, and Z. Han, "Optimal differentially private ADMM for distributed machine learning," *Available at http://arxiv.org/abs/1901.02094*, Feb. 2019.

[49] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial Intel. Statis.*, pp. 1273–1282, Apr. 2017.

[50] X. Ouyang, Z. Xie, J. Zhou, J. Huang, and G. Xing, "ClusterFL: A similarity-aware federated learning system for human activity recognition," *Proc. 19th Annu. Int. Conf. Mobile Syst. Applications Services*, p. 54–66, Jun. 2021.

[51] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, 2012.

[52] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "MedMNIST v2-A large-scale lightweight benchmark for 2D and 3D biomedical image classification," *Scientific Data*, vol. 10, no. 1, p. 41, 2023.

Appendix G

Publication 7

P7 F. Gauthier, V. C. Gogineni, and S. Werner, "Networked Personalized Federated Learning Using Reinforcement Learning", in *Proceedings of IEEE International Conference on Communications*, June, 2023.

The following is the accepted manuscript.

Networked Personalized Federated Learning Using Reinforcement Learning

Francois Gauthier, Vinay Chakravarthi Gogineni, Stefan Werner
Dept. of Electronic Systems, Norwegian University of Science and Technology, Norway
E-mails: {francois.gauthier, vinay.gogineni, stefan.werner}@ntnu.no

***Abstract*—Personalized federated learning enables every edge device or group of edge devices within the distributed network to learn a device- or cluster-specific model tailored to their local needs. Data scarcity, however, makes it difficult to learn such individual models, resulting in performance degradation. Since the device- or cluster-specific tasks are distinct but often related, leveraging these similarities through inter-cluster learning alleviates data shortage and enhances learning performance. Although inter-cluster learning can boost performance, uncontrolled inter-cluster learning may lead to performance degradation due to over- or under-usage of local similarity enforcement. In light of this issue, an intelligent mechanism that performs inter-cluster learning based on device-specific needs is required. To this end, this paper proposes adopting reinforcement learning principles to control device-specific inter-cluster learning in real-time. We propose networked personalized federated learning using reinforcement learning (NPFed-RL) as a general framework and then demonstrate its feasibility by applying it to the ridge regression problem. We conduct numerical experiments to compare the proposed method with the state-of-the-art. The proposed method successfully controls device-specific parameters and offers better learning performance than existing solutions.**

***Index Terms*—Personalized federated learning, networked federated learning, distributed learning, inter-cluster learning.**

I. Introduction

Federated learning (FL) is a distributed learning paradigm that enables geographically dispersed edge devices, often called clients, to learn a global shared model on their locally stored data without revealing it [1]. FL received enormous attention due to the ability to handle system and statistical heterogeneity. System heterogeneity refers to the various computational and communication capacities of the participating devices [2]. Statistical heterogeneity implies that data are imbalanced and non-i.i.d. across devices [3]. Several challenges associated with the practical implementation of FL, such as communication efficiency [4], privacy preservation [5], byzantine attacks [6], and asynchronous behavior of devices and communication links [7], have been studied extensively in the literature. In contrast to single-server methodologies, multi-server architectures [8], [9] and fully distributed architectures [10] have also been studied recently. This paper deals with a fully distributed architecture.

In many practical applications, a network of clients must learn more than one model. Those models might be different, yet they exhibit similarities [11]. For instance, networked vehicles need to learn and maintain customized models of surrounding environments to plan trajectories [12]. Likewise, patient-specific models in healthcare are required for better diagnosis and treatment [13]. Personalized federated learning fulfills the requirements of these applications by allowing each client, or a group of clients (referred to as a cluster), to learn client- or cluster-specific models [14], [15]. Building personalized models for clusters is challenging, as they often have access to limited data. Inter-cluster learning, i.e., cooperation across the clusters, can alleviate data scarcity [16].

Although inter-cluster learning can improve learning accuracy and speed by exploiting similarities across personalized models, it can also have the opposite effects if not adequately controlled [17]. In particular, a fixed regularization parameter cannot effectively control inter-cluster learning for all clients since system heterogeneity and network topology affect local sensitivity to data shortages. For example, clients will benefit more from inter-cluster learning if neighbors, and clients within a few hops, mainly belong to different clusters, slowing down the propagation of cluster-specific information through the network. However, as consensus is reached within a group of clients from a given cluster, the need for inter-cluster learning diminishes. For these reasons, there is a need for an intelligent mechanism that controls client-specific inter-cluster learning parameters in real-time so that it is only used when it improves performance [18]. To that end, this paper develops a reinforcement learning-based mechanism that locally controls client-specific inter-cluster learning parameters on-the-fly.

Personalized distributed learning algorithms use a fixed inter-cluster learning parameter and suffer from the above-mentioned limitations [16], [17]. In [18], an ad-hoc rule has been proposed to disable inter-cluster learning if it becomes detrimental rather than adapting its impact according to client requirements. For this purpose, every model received from neighbors belonging to a different cluster is tested against the local training dataset. This process results in a substantial computational cost that grows with the network density, thus prohibitive for practical usage. Reinforcement learning has been used in distributed single-task learning [19], [20] and multi-task learning [21]–[23], but they have yet to focus on inter-cluster learning. In the field of evolutionary computing, reinforcement learning has been proven effective for parameter control [24], suggesting its potential for real-time control of inter-cluster learning parameters in personalized FL.

This paper proposes the NPFed-RL algorithm, where clients use the alternating direction method of multipliers (ADMM) and interact with neighbors to learn cluster-specific models. Inter-cluster learning is used to improve those models further.

Each client uses reinforcement learning to adapt its inter-cluster learning parameter on-the-fly so that task similarity with neighboring clients is only leveraged when it is beneficial for local learning. Two reinforcement learning policies are developed and studied: a deterministic policy gradient and a stochastic actor-critic policy based on an estimate of the action-value function. The proposed policies are computationally inexpensive and improve the convergence properties of networked personalized federated learning. The proposed NPFed-RL will first be derived as a framework and then used to solve the ridge regression problem as proof of concept. Finally, numerical simulations will be conducted to demonstrate the performance of the proposed NPFed-RL and to observe the evolution of the inter-cluster learning parameters in the proposed method.

II. Networked Personalized Federated Learning

We consider a fully distributed network modeled as an undirected graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C}$ is the set of clients and $\mathcal{E}$ is the set of edges such that $\mathcal{E}(k,l) = 1$ if the clients k and l are neighbors and 0 otherwise. A client can only communicate with its neighbors, we denote $\mathcal{N}_k$ the set of the neighbors of client k. Further, clients are grouped into Q clusters, and the clients grouped in a cluster q, denoted by $\mathcal{C}_q$, for $q \in \{1, \dots, Q\}$, carry out the same task, i.e., they aim to learn the same model. For $r \neq q, (r,q) \in \{1, \dots, Q\}$, the tasks associated with cluster q and r are different, but similar. To warrant the use of inter-cluster learning, we take the following assumption on task similarity, where $\mathbf{w}_q^*$ denotes the optimal model for cluster q.

Assumption 1. $\left\|\mathbf{w}_q^* - \mathbf{w}_r^*\right\|_2^2 \leqslant \eta, \forall q, r \in \{1, \dots, Q\}$

Each client $k \in \mathcal{C}$ has access to a proprietary dataset $(\mathbf{X}_k, \mathbf{y}_k)$ composed of a matrix $\mathbf{X}_k = [\mathbf{x}_{k,1}, \dots, \mathbf{x}_{k,D_k}]^\mathsf{T}$ and a response vector $\mathbf{y}_k = [y_{k,1}, \dots, y_{k,D_k}]^\mathsf{T}$, where D_k is the number of data samples available to client k. The objective is to estimate each cluster-specific model as accurately as possible without leaking data. This leads to the following optimization problem for a given cluster q:

$$\min_{\mathbf{w}_q} \sum_{k \in \mathcal{C}_q} \Big(\frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_q) \Big) + \lambda R(\mathbf{w}_q) + \frac{\tau}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathfrak{q}} \left\|\mathbf{w}_r - \mathbf{w}_q\right\|^2, \tag{1}$$

where ℓ_k denotes the loss function of the task performed by client k, R denotes the regularizer function, and $\lambda > 0$ is the regularization parameter. The term on the second line corresponds to the enforcement of similarity between the cluster-specific models, it is controlled by the global parameter τ. A larger τ enforces more similarity, leading to more similar cluster-specific models. This optimization problem uses a global model $\mathbf{w}_q$ for each cluster, this is not feasible in the proposed setting.

To circumvent this difficulty, the learning process must rely on the clients' models and enforce consensus among these models. To do so, the auxiliary variables $\mathbf{z}_k^l, \forall (k,l) \in \mathcal{C}_q : \mathcal{E}(k,l) = 1$ are introduced. The optimization problem for a given client k belonging to cluster q is then given by

$$\begin{aligned} \min_{\mathbf{w}_{k,q}} \quad & \frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{k,q}) + \lambda R(\mathbf{w}_{k,q}) \\ & + \frac{\tau_k}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathfrak{q}} \left\|\hat{\mathbf{w}}_{k,r} - \mathbf{w}_{k,q}\right\|^2, \\ \text{s.t.} \quad & \mathbf{w}_{k,q} = \mathbf{z}_k^l, \mathbf{w}_{l,q} = \mathbf{z}_k^l; \forall (k,l) \in \mathcal{C}_q : \mathcal{E}(k,l) = 1, \end{aligned} \tag{2}$$

where $\mathbf{w}_{k,q}$ denotes the model of client k belonging to cluster q, and the constraints enforce intra-cluster consensus. The global parameter τ is replaced by client-specific parameters τ_k that control inter-cluster learning locally. $\hat{\mathbf{w}}_{k,r}$ denotes the best available estimate of the model for cluster r available at client k. This corresponds to the average of the models of the neighboring clients from the cluster in question, given by

$$\hat{\mathbf{w}}_{k,r} = \frac{1}{|\mathcal{N}_k \bigcap \mathcal{C}_r|} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_r} \mathbf{w}_{l,r}. \tag{3}$$

It is possible to derive the augmented Lagrangian for a given cluster q with the set of primal variables $\mathcal{V}_q = \{\mathbf{w}_{k,q}\}$, Lagrange multipliers $\mathcal{M} = (\{\boldsymbol{\mu}_k^l\}, \{\boldsymbol{\gamma}_k^l\})$, and auxiliary variables $\mathcal{Z} = \{\mathbf{z}_k^l\}$ as

$$\begin{aligned} & \mathcal{L}_{\rho,q}(\mathcal{V}_q, \mathcal{M}, \mathcal{Z}) = \\ & \sum_{k \in \mathcal{C}_q} \Bigg(\frac{1}{D_k} \sum_{i=1}^{D_k} \ell_k(\mathbf{x}_{k,i}, y_{k,i}; \mathbf{w}_{k,q}) + \frac{\lambda}{|\mathcal{C}_q|} R(\mathbf{w}_{k,q}) \\ & + \Big\| \frac{\tau_k}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathfrak{q}} (\hat{\mathbf{w}}_{k,r} - \mathbf{w}_{k,q}) \Big\|^2 \Bigg) \\ & + \sum_{k \in \mathcal{C}_q} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \Big(\boldsymbol{\mu}_k^{l\mathsf{T}} (\mathbf{w}_{k,q} - \mathbf{z}_k^l) + \boldsymbol{\gamma}_k^{l\mathsf{T}} (\mathbf{w}_{l,q} - \mathbf{z}_k^l) \Big) \\ & + \frac{\rho}{2} \sum_{k \in \mathcal{C}_q} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \Big(\left\|\mathbf{w}_{k,q} - \mathbf{z}_k^l\right\|^2 + \left\|\mathbf{w}_{l,q} - \mathbf{z}_k^l\right\|^2 \Big), \end{aligned} \tag{4}$$

where ρ is the penalty parameter. Given that the Lagrange multipliers are initialized to zero, by using the Karush-Kuhn-Tucker conditions of optimality and setting $\boldsymbol{\gamma}_k = 2 \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \boldsymbol{\gamma}_k^l$, it can be shown that the Lagrange multipliers $\boldsymbol{\mu}_k^l$ and the auxiliary variables $\mathcal{Z}$ are eliminated [25]. From the Lagrangian, the local update steps of the ADMM for a given client k belonging to cluster q can be derived as:

- **Primal update**

$$\begin{aligned} \mathbf{w}_{k,q}^{(n)} = \arg\min_{\mathbf{w}} \; & \frac{1}{D_k} \ell_k(\mathbf{X}_k, \mathbf{y}_k; \mathbf{w}) + \frac{\lambda}{|\mathcal{C}_q|} R(\mathbf{w}) \\ & + \Big\| \frac{\tau_k}{Q-1} \sum_{r \in \{1,\dots,Q\} \setminus \mathfrak{q}} (\hat{\mathbf{w}}_{k,r}^{(n-1)} - \mathbf{w}) \Big\|^2 \\ & + \mathbf{w}^\mathsf{T} \boldsymbol{\gamma}_{k,q}^{(n-1)} \\ & + \rho \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} \Big\| \mathbf{w} - \frac{\mathbf{w}_{k,q}^{(n-1)} + \mathbf{w}_{l,q}^{(n-1)}}{2} \Big\|^2, \end{aligned} \tag{5}$$

- **Dual update**

$$\boldsymbol{\gamma}_{k,q}^{(n)} = \boldsymbol{\gamma}_{k,q}^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} (\mathbf{w}_{l,q}^{(n)} - \mathbf{w}_{k,q}^{(n)}), \quad (6)$$

where the superscript (n) denotes the iteration number.

The choice of the inter-cluster learning parameters τ_k will greatly impact the performance of the proposed solution. A common and fixed value would fail to accommodate the heterogeneous data distribution and would not remain relevant throughout the various learning stages of the clients. To address this, we adopt in the next section the principles of reinforcement learning to control time-varying and client-specific inter-cluster learning parameters $\tau_k^{(n)}$.

III. Controlled Inter-Cluster Learning using Reinforcement Learning

Obtaining the optimal values for all $\tau_k^{(n)}$ would require prior knowledge of the network topology and data distribution as well as extensive computational power. Instead, each client k controls the real time evolution of the parameter $\tau_k^{(n)}$ using local data and information received from neighbors. Computationally inexpensive reinforcement learning is used for this purpose.

At any given moment, the state of the reinforcement learning process, at a client k in cluster q, is given by the primal variable, $\mathbf{w}_{k,q}^{(n)}$. The action a corresponds to the modification of the local inter-cluster learning parameter $\tau_k^{(n)}$. We denote $\mathbf{w}_{k,q}^{(n)}(\cdot)$ the function taking a value for the inter-cluster learning parameter and giving the corresponding alternative primal variable. The original primal variable in (5) corresponds to $\mathbf{w}_{k,q}^{(n)} = \mathbf{w}_{k,q}^{(n)}(\tau_k^{(n)})$. For an action a, the corresponding alternative primal variable is given by $\mathbf{w}_{k,q}^{(n)}(a)$.

The state- and action-value functions correspond to the error on the local test dataset of the initial and alternative primal variables, respectively. For instance, the state-value function is given by

$$V_t(\mathbf{w}_{k,q}^{(n)}) = \frac{1}{D_k} \ell_k(\mathbf{X}_{k,t}, \mathbf{y}_{k,t}; \mathbf{w}_{k,q}^{(n)}) + \frac{\lambda}{|\mathcal{C}_q|} R(\mathbf{w}_{k,q}^{(n)}), \quad (7)$$

where $(\mathbf{X}_{k,t}, \mathbf{y}_{k,t})$ denotes the test dataset. To avoid overfitting, it is preferable not to use the test dataset in the reinforcement learning process. Instead, estimates of the state- and action-value functions are computed on a validation dataset $(\mathbf{X}_{k,v}, \mathbf{y}_{k,v})$. The estimate of the state-value function is given by $V_v(\mathbf{w}_{k,q}^{(n)})$, and the estimate of the action-value function by $V_v(\mathbf{w}_{k,q}^{(n)}(a))$.

Policy gradient [26], [27] and deterministic policy gradient [28] are among the most popular policies for continuous action reinforcement learning. They propose a gradient ascent alternative to the greedy maximization of the action-value function given by

$$\tau_k^{(n+1)} = \arg\max_a V_v(\mathbf{w}_{k,q}^{(n)}(a)). \quad (8)$$

In its simplest form, deterministic policy gradient relies on the gradient of the policy reward with respect to the policy parameter at the current state. In the proposed setting, this corresponds to the derivative of $V_v(\mathbf{w}_{k,q}^{(n)}(a))$ with respect to a taken at $\tau_k^{(n)}$, where the sign of the gradient is inverted since the reward corresponds to the error. The policy parameter update is given by

$$\tau_k^{(n+1)} - \tau_k^{(n)} \propto \frac{\partial V_v(\mathbf{w}_{k,q}^{(n)}(a))}{\partial a}, \quad (9)$$

where $\propto$ denotes proportionality. Given the primal update (5), the computation of this derivative is impossible in the general case. However, it can be possible when the loss and regularizer functions are known.

The second proposed policy is a stochastic actor-critic mechanism that takes a random action and compares the action-value function with the state-value function to decide on the next value of the policy parameter. This policy offers better policy parameter exploration than the deterministic policy gradient [29]. First, the policy proposes a random direction $\alpha \sim \mathcal{U}[\frac{-\nu_{\text{SAC}}}{2}, \frac{\nu_{\text{SAC}}}{2}]$ for the policy parameter $\tau_k^{(n)}$ so that $a = \tau_k^{(n)} + \alpha$. $\mathcal{U}(\cdot)$ denotes the uniform distribution, and ν_{SAC} is a hyper-parameter for the policy. The alternative primal variable $\mathbf{w}_{k,q}^{(n)}(a)$ is computed so that the action-value function can be compared with the state-value function. The policy parameter update is given by

$$\tau_k^{(n+1)} - \tau_k^{(n)} \propto -\text{sign}\big(V_v(\mathbf{w}_{k,q}^{(n)}(a)) - V_v(\mathbf{w}_{k,q}^{(n)})\big)\alpha.$$

IV. NPFed-RL for Ridge Regression

As a proof of concept, we use the proposed framework to solve the ridge regression problem. In ridge regression, the loss and regularizer functions used in (2) are given by

$$\ell(\mathbf{X}_k, \mathbf{y}_k, \mathbf{w}) = \|\mathbf{y}_k - \mathbf{X}_k \mathbf{w}\|^2,$$
$$R(\mathbf{w}) = \|\mathbf{w}\|^2. \quad (10)$$

The primal variable update for ridge regression is obtained by substituting $\ell(\mathbf{X}_k, \mathbf{y}_k, \mathbf{w})$ and $R(\mathbf{w})$ with their above values in equation (5). Doing so, it is possible to compute the gradient of the term in the $\arg\min$ with respect to the primal variable $\mathbf{w}$, and, setting it to zero, we obtain

$$\begin{aligned}\mathbf{w}_{k,q}^{(n)}(\tau_k^{(n)}) = &\Big(\frac{\mathbf{X}_k^\mathsf{T}\mathbf{X}_k}{D_k} + (\frac{\lambda}{|\mathcal{C}_q|} + \tau_k^{(n)} + \rho|\mathcal{N}_k|)\mathbf{I}\Big)^{-1} \\ &\Big(\frac{\mathbf{X}_k^\mathsf{T}\mathbf{y}_k}{D_k} \frac{\tau_k^{(n)}}{(Q-1)} \sum_{r \in \{1,\dots,Q\}\setminus \mathrm{q}} \hat{\mathbf{w}}_{k,r}^{(n-1)} \\ &+ \frac{\rho}{2} \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} (\mathbf{w}_{k,q}^{(n-1)} + \mathbf{w}_{l,q}^{(n-1)}) - \frac{\boldsymbol{\gamma}_{k,q}^{(n-1)}}{2}\Big).\end{aligned} \quad (11)$$

The alternative primal variable $\mathbf{w}_{k,q}^{(n)}(a)$ for an action a can be computed in the same manner, by replacing $\tau_k^{(n)}$ with a in (11). Using this alternative primal variable and the values of

Algorithm 1 NPFed-RL for ridge regression

Initialization: $\mathbf{w}_{k,q}^{(0)}$ and $\boldsymbol{\gamma}_{k,q}^{(0)}$, $k \in \mathcal{C}$ are set to $\mathbf{0}$, the parameters $\tau_k^{(0)}$ are set to a given value within $(0,1)$.

Procedure at client k:

For $n = 1, 2, \ldots, N$

If $n > 1$

(DPG) τ_k is updated as in (13).

(SAC) τ_k is updated as in (14).

end If

Primal update: $\mathbf{w}_{k,q}^{(n)}$ takes the value in (11).

Client k shares $\mathbf{w}_{k,q}^{(n)}$ with its neighbors in $\mathcal{N}_k$.

Dual update:

$\boldsymbol{\gamma}_{k,q}^{(n)} = \boldsymbol{\gamma}_{k,q}^{(n-1)} + \rho \sum_{l \in \mathcal{N}_k \bigcap \mathcal{C}_q} (\mathbf{w}_{l,q}^{(n)} - \mathbf{w}_{k,q}^{(n)})$.

the loss and regularizer function for ridge regression in (10), the action-value function can be expressed as

$$V_v(\mathbf{w}_{k,q}^{(n)}(a)) = \frac{1}{D_k} \left\| \mathbf{y}_{k,v} - \mathbf{X}_{k,v} \mathbf{w}_{k,q}^{(n)}(a) \right\|^2 + \frac{\lambda}{|\mathcal{C}_q|} \left\| \mathbf{w}_{k,q}^{(n)}(a) \right\|^2. \quad (12)$$

Using the expression for the primal variable (11) and the action-value function specific to ridge regression in (12), it is possible to compute the derivative of the policy reward with respect to the policy parameter $\partial V_v(\mathbf{w}_{k,q}^{(n)}(\tau_k^{(n)}))/\partial \tau_k^{(n)}$. The explicit derivation is not included because of space constraints. Given that the action-value function is to be minimized, the policy parameter update step for the *deterministic policy gradient*, which we refer to as (DPG), is given by:

$$\tau_k^{(n+1)} = \tau_k^{(n)} - \delta_{\text{DPG}} \frac{\partial V_v(\mathbf{w}_{k,q}^{(n)}(a))}{\partial a}, \quad (13)$$

where δ_{DPG} is the learning rate.

For a random direction α and corresponding action $a = \tau_k^{(n)} + \alpha$, using (12) for the action- and state-value functions, the update step of the policy parameter for the *stochastic actor-critic policy*, which we refer to as (SAC), is given by

$$\tau_k^{(n+1)} = \tau_k^{(n)} - \delta_{\text{SAC}} \text{sign}\big(V_v(\mathbf{w}_{k,q}^{(n)}(a)) - V_v(\mathbf{w}_{k,q}^{(n)})\big)\alpha, \quad (14)$$

where the sign is negated to minimize the action-value function and δ_{SAC} is the learning rate.

The resulting algorithms, referred to as Networked Personalized Federated learning using Reinforcement Learning (NPFed-RL) are summarized in Algorithm 1.

V. Numerical Simulations

We considered a distributed network composed of $|\mathcal{C}| = 30$ clients with an average of 6 neighbors per client. The clients are randomly grouped into $Q = 3$ clusters. The goal is to estimate cluster-specific tasks given by $\mathbf{w}_q = \mathbf{w}_0 + \delta_q \mathbf{w}_0$, with $\delta_q \sim \mathcal{U}(-0.5, 0.5)$. $\mathcal{U}$ denotes the uniform distribution, and $\mathbf{w}_0$ is a randomly chosen base model. Each client k possesses a training dataset $(\mathbf{X}_k, \mathbf{y}_k)$ where $\mathbf{X}_k \in \mathbb{R}^{D_k \times 60}$ and $\mathbf{y}_k \in \mathbb{R}^{D_k \times 1}$ with $D_k \sim \mathcal{U}(5, 35)$, as well as identically distributed testing and validation datasets. The data is generated as $\mathbf{y}_k = \mathbf{X}_k \mathbf{w}_q + \mathbf{n}_k$, with $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \eta_k)$, where η_k is the client-specific noise variance and $\mathbf{w}_q$ is its cluster model. Finally, the Lagrangian penalty parameter is set to $\rho = 3$. We considered the mean squared error on the testing data set (Test MSE) as the performance metric for comparison of the algorithms. It is given by

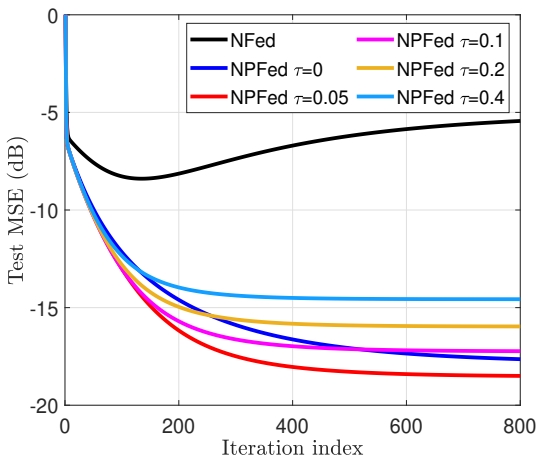


Fig. 1: Learning curves of the NFed and NPFed algorithms for various values of τ.

$$\text{Test MSE} = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{|\mathcal{C}|} \frac{\|\mathbf{w}_{k,q} - \bar{\mathbf{w}}_q\|_2^2}{\|\bar{\mathbf{w}}_q\|}, \quad (15)$$

where $\{\mathbf{w}_{k,q}, k \in \mathcal{C}_q, q \in \{1, \ldots, Q\}\}$ are the models of the considered method and $\bar{\mathbf{w}}_q$ is the optimal model for the cluster. The simulation results presented in the following are obtained by averaging the results of 10 independent experiments. To ensure a fair comparison, the algorithms are tuned to have the same initial convergence rate in Fig. (2).

The first experiment studied the impact of the inter-cluster learning parameter τ on the learning behavior of conventional networked FL algorithms. For this purpose, we simulated the following algorithms:

- **NFed:** is the traditional networked FL that learns one universal model for the whole network.
- **NPFed:** is conventional personalized networked FL that learns cluster-specific models. The global parameter τ is fixed throughout the learning process, so that $\forall k \in \mathcal{C}, n > 0; \tau_k = \tau$. Inter-cluster learning is absent when $\tau = 0$.

The learning curves (i.e., Test MSE in dB vs. iteration index n) of the above algorithms are presented in Fig. 1. The figure shows that the NFed algorithm does not achieve satisfactory

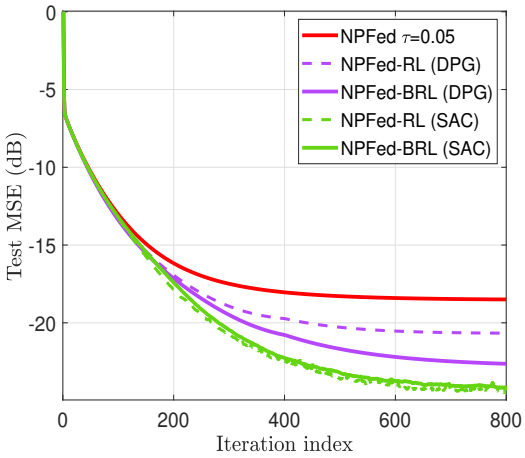


Fig. 2: Learning curves of the NPFed-RL and NPFed-BRL with different policies. Also plotted is the learning curve of NPFed for $\tau = 0.05$.

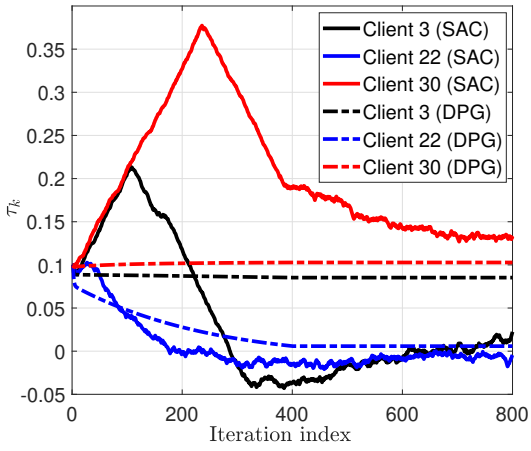


Fig. 3: Evolution of $\tau_k^{(n)}$ for the 3rd, 22nd, and 30th clients.

accuracy since it tries to learn a single universal model that cannot accommodate client-specific tasks. NPFed with $\tau = 0$ implies that each cluster independently builds its own model by relying solely on the cooperation among cluster members. Its performance can, therefore, be regarded as a benchmark for networked personalized federated learning. As the value of τ increases (e.g., $\tau = 0.05$), the NPFed performance improves, as it enforces similarity between the cluster-specific models. However, as more similarity is enforced between models for non-identical tasks, the steady-state accuracy decreases, as can be seen with NPFed $\tau = 0.1$, $\tau = 0.2$, and $\tau = 0.4$. This confirms that inter-cluster learning can be beneficial but leads to performance degradation when over-used.

In the second experiment, we demonstrated the effectiveness of the proposed NPFed-RL in the learning of personalized models. For this purpose, we simulated the following algorithms:

- **NPFed-RL (DPG):** is the proposed NPFed-RL using the deterministic policy gradient method. The policy hyperparameter was set to $\delta_{\text{DPG}} = 0.001$, and its policy parameter update step is given in (13).
- **NPFed-BRL (DPG):** is the NPFed-RL using the policy gradient and batch reinforcement learning [30] with 3 epochs and $\delta_{\text{DPG}} = 0.003$.
- **NPFed-RL (SAC):** is the proposed NPFed-RL using the stochastic actor-critic policy. The policy hyperparameters were set to $\nu_{\text{SAC}} = 0.05$ and $\delta_{\text{SAC}} = 0.1$. The policy parameter update step is given in (14).
- **NPFed-BRL (SAC):** is the NPFed-RL using the stochastic actor-critic policy and batch reinforcement learning [30] with 3 epochs, $\nu_{\text{SAC}} = 0.05$ and $\delta_{\text{SAC}} = 0.04$.

The learning curves of these algorithms are presented in Fig. 2. For comparison purposes, the learning curve of NPFed with $\tau = 0.05$ is also displayed. We see that all the versions of NPFed-RL exhibit better performance in initial learning speed and steady-state accuracy compared to NPFed operating with a fixed τ value. Since the clients have device-specific requirements, usage of a fixed universal τ is impractical. Whereas the proposed NFed-RL controls the amount of inter-cluster learning locally by learning the device-specific parameters $\tau_k^{(n)}$ in real time. Further, we also see that NPFed-RL (SAC) exhibites enhanced accuracy over NPFed-RL (DPG). The reason for this is that the stochastic nature of NPFed-RL (SAC) ensures sufficient exploration of the policy parameters τ_k, which is not the case for NPFed-RL (DPG). This stochastic nature also leads to extensive randomness in the convergence of NPFed-RL (SAC), batch reinforcement learning attenuates this issue as can be seen with NPFed-BRL (SAC). In the case of deterministic policy gradient, batch reinforcement learning increases the learning accuracy. It is important to note that batch reinforcement learning comes with a computational cost proportional to the number of epochs performed.

Finally, we illustrate the evolution of the inter/cluster learning parameters $\tau_k^{(n)}$ for NPFed-RL (SAC) and NPFed-RL (DPG) in Fig. 3. We selected three clients 22, 3, 30, having access to large, moderate, and small amounts of data, respectively. Therefore, these clients have different local requirements for inter-cluster learning. From Fig. 3, we observe that the evolution of $\tau_k^{(n)}$ is very smooth when using NPFed-RL (DPG) but fails to quickly adapt. On the other hand, the NPFed-RL (SAC) algorithm provides sufficient exploration of the policy parameter, ensuring that the $\tau_k^{(n)}$ parameters evolve quickly at the cost of extensive randomness (BRL helps to overcome this issue). Furthermore, we also see that $\tau_k^{(n)}$ evolves according to the needs of clients. Since client 30 has access to a small amount of data, $\tau_{30}^{(n)}$ increases linearly at first to enforce higher inter-cluster learning. After reaching near-convergence, the $\tau_{30}^{(n)}$ value decreases to reduce the amount of

inter-cluster learning to avoid its harmful effect. In contrast, since client 22 has access to a large amount of data, the $\tau_{22}^{(n)}$ parameter decreases almost immediately and stabilizes around 0 as inter-cluster learning is not valuable for this client. Finally, client 3 has access to an average amount of data. $\tau_{3}^{(n)}$ increases at first as similarity enforcement allows for faster initial convergence, but decreases afterwards and stabilizes around 0 to avoid performance degradation.

VI. Conclusions

Personalized federated learning suffers from data scarcity within clusters, this is alleviated by leveraging the similarity between the learning tasks. However, how much a specific client needs to rely on inter-cluster learning depends on its local data and the network topology. To accommodate the varied needs of the clients, we propose a networked personalized federated learning algorithm using reinforcement learning to control evolving client-specific inter-cluster learning parameters. Each local parameter is updated on-the-fly by the reinforcement learning process in a computationally inexpensive manner so that model similarity is enforced only as much as what is beneficial for local learning. Numerical simulations show that the proposed method has better learning performances than the state-of-the-art.

Acknowledgement

The Research Council of Norway supported this work.

References

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

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Proc. Mag.*, vol. 37, no. 3, pp. 50–60, May. 2020.

[3] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[4] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Communication-Efficient Online Federated Learning Strategies for Kernel Regression," *IEEE Internet Things J.*, pp. 1–1, Nov. 2022.

[5] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, Apr. 2020.

[6] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2168–2181, Jul. 2020.

[7] F. Gauthier, V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Resource-aware asynchronous online federated learning for nonlinear regression," *in Proc. IEEE Int. Conf. Commun.*, pp. 2828–2833, May 2022.

[8] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," *in Proc. IEEE Int. Workshop Signal Process. Advances Wireless Commun.*, pp. 131–135, Sep. 2021.

[9] V. C. Gogineni, S. Werner, Y.-F. Huang, and A. Kuh, "Decentralized graph federated multitask learning for streaming data," *in Proc. Annu. Conf. Inf. Sciences Sys.*, pp. 101–106, Mar. 2022.

[10] Y. Sarcheshmehpour, M. Leinonen, and A. Jung, "Federated learning from big data over networks," *in Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.*, pp. 3055–3059, Jan. 2021.

[11] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Jun. 2014.

[12] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

[13] S. Boll and J. Meyer, "Health-X dataLOFT: A Sovereign Federated Cloud for Personalized Health Care Services," *IEEE MultiMedia*, vol. 29, no. 1, pp. 136–140, May 2022.

[14] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Networks Learning Sys.*, Mar. 2022.

[15] V. C. Gogineni, S. Werner, F. Gauthier, Y.-F. Huang, and A. Kuh, "Personalized online federated learning for IoT/CPS: challenges and future directions," *IEEE Internet Things Mag.*, 2022.

[16] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, Jun. 2014.

[17] V. C. Gogineni and M. Chakraborty, "Diffusion affine projection algorithm for multitask networks," *in Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, pp. 201–206, Nov. 2018.

[18] ——, "Improving the performance of multitask diffusion APA via controlled inter-cluster cooperation," *IEEE Trans. Circuits Sys. I: Reg. Papers*, vol. 67, no. 3, pp. 903–912, Dec. 2019.

[19] W. Lei, Y. Ye, M. Xiao, M. Skoglund, and Z. Han, "Adaptive stochastic ADMM for decentralized reinforcement learning in edge IoT," *IEEE Internet Things J.*, Jun. 2022.

[20] M. Krouka, A. Elgabli, C. B. Issaid, and M. Bennis, "Communication-efficient and federated multi-agent reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 311–320, Nov. 2021.

[21] S. El Bsat, H. B. Ammar, and M. E. Taylor, "Scalable multitask policy gradient reinforcement learning," *in Proc. Thirty-first AAAI Conf. Artif. Intell.*, Feb. 2017.

[22] R. Tutunov, D. Kim, and H. Bou Ammar, "Distributed multitask reinforcement learning with quadratic convergence," *Advances Neural Inf. Process. Syst.*, vol. 31, 2018.

[23] Y. Ye, M. Xiao, and M. Skoglund, "Randomized neural networks based decentralized multi-task learning via hybrid multi-block ADMM," *IEEE Trans. Signal Process.*, vol. 69, pp. 2844–2857, May 2021.

[24] G. Karafotias, A. E. Eiben, and M. Hoogendoorn, "Generic parameter control with reinforcement learning," *in Proc. Annu. Conf. Genetic Evol. Comput.*, pp. 1319–1326, Jul. 2014.

[25] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, "Splitting Methods in Communication, Imaging, Science, and Engineering," in *Scientific Computation*. Springer Int. Publishing, Jan. 2017, pp. 461–497.

[26] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances Neural Inf. Process. Syst.*, vol. 12, 1999.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, Jul. 2017.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *in Proc. Int. Conf. Mach. Learn.*, pp. 387–395, Jan. 2014.

[29] H. Wang, T. Zariphopoulou, and X. Y. Zhou, "Reinforcement Learning in Continuous Time and Space: A Stochastic Control Approach," *J. Mach. Learn. Res.*, vol. 21, no. 198, pp. 1–34, Jan. 2020.

[30] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," *Reinforcement Learning*, pp. 45–73, Springer, 2012.

Appendix H

Publication 8

P8 V. C. Gogineni, S. Werner, F. Gauthier, Y. Huang, and A. Kuh, "Personalized online federated learning for IoT/CPS: challenges and future directions", in *IEEE Internet of Things Magazine*, December, 2022.

ISBN 978-82-326-7778-8 (printed ver.)
ISBN 978-82-326-7777-1 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU

Norwegian University of
Science and Technology