Doctoral theses at NTNU, 2023:12

Cristiano Gratton

Privacy-preserving distributed machine learning for artificial intelligence of things

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

Cristiano Gratton

Privacy-preserving distributed machine learning for artificial intelligence of things

Thesis for the Degree of Philosophiae Doctor

Trondheim, January 2023

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 978-82-326-5563-2 (printed ver.)
ISBN 978-82-326-6584-6 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2023:12

Printed by NTNU Grafisk senter

Abstract

This thesis proposes machine learning algorithms that can be fully distributed over ad-hoc networks of machines/agents. Developing distributed algorithms for artificial intelligence is necessary since running machine-learning-based data analytics on a single central hub may be unfeasible due to computing/communication costs. In the context of distributed learning, privacy violation risks due to curious members of the network or eavesdroppers make the development of privacy-preserving distributed algorithms imperative.

The main contributions of the thesis are around developing distributed machine learning algorithms for artificial intelligence of things including distributed algorithms with intrinsic privacy-preserving properties. In particular, the contributions can be grouped in the following categories:

- distributed learning over networks with horizontal/row partitioning of data
- intrinsically privacy-preserving distributed learning with zeroth-order optimization
- distributed learning over networks with vertical/feature partitioning of data.

In the context of distributed learning with horizontal partitioning of data, we propose a new distributed algorithm to solve the total least-squares (TLS) problem and a privacy-preserving distributed algorithm to minimize a regularized empirical risk function when the first-order information is not available.

We show that the latter algorithm has intrinsic privacy-preserving properties. Most existing privacy-preserving distributed optimization/estimation algorithms exploit some perturbation mechanism to preserve privacy, which comes at the cost of reduced accuracy. Contrarily, we exploit the inherent randomness due to the use of a zeroth-order method and show that this stochasticity is sufficient to ensure differential privacy. Moreover, we demonstrate that the proposed algorithm outperforms the existing differentially-private ones in terms of accuracy while yielding similar privacy guarantees.

In the context of distributed learning with feature partitioning of data, we develop a new distributed algorithm to solve the ridge regression problem. Subsequently, we develop a new algorithm that is designed for an ℓ_2-norm-square cost function with non-smooth regularizers. Finally, we develop a new consensus-based distributed algorithm for solving learning problems when the data is distributed among agents in feature partitions and computing the conjugate of the possibly non-smooth cost or regularizer functions is challenging or unfeasible. The proposed algorithm is designed for optimizing generic non-smooth objective functions over arbitrary graphs without using or computing any conjugate function. All the above-mentioned algorithms are fully-distributed and based on the alternating direction method of multipliers (ADMM) that is suitable for distributed optimization thanks to its scalability and robustness properties. We prove theoretically that the proposed algorithms converge. We also confirm their network-wide convergence via simulations.

Acknowledgments

First, I would like to thank my family, in particular my parents, my wife, and my son for supporting me and giving me the motivation to perform the research activities.

I would like to express my gratitude to my supervisor Professor Stefan Werner for believing in me and giving me the opportunity to pursue the PhD degree at the NTNU, for his availability and for guiding and encouraging me in this long and hard path.

Further, I would like to thank my co-supervisor Reza Arablouei for his insightful suggestions, for making available his expertise in the research fields, and for the time and effort put into producing the papers forming the thesis.

I wish to thank my co-author Naveen K. D. Venkategowda for his patience in helping me, his hints and his valuable contributions to the papers constiuting this PhD thesis.

Finally, I also thank everybody from the Department of Electronic Systems (IES) at the NTNU, in particular, the Signal Processing Group.

Contents

List of Tables

List of Figures

Abbreviations and Symbols

Abbreviations

ADMM	Alternating direction method of multipliers
BCD	Block coordinate descent
D-TLS	Distributed total least-squares
D-ZOA	Distributed zeroth-order based ADMM
DA-TLS	Distributed ADMM total least-squares
DC-ADMM	Dual consensus ADMM
DPSGD	Distributed subgradient algorithm
DVP	Dual variable perturbation
ERM	Empirical risk minimization
EVD	Eigen-decomposition
IPI-D-TLS	Inverse-power-iteration-based distributed total least-squares
KKT	Karush-Kuhn-Tucker
PVP	Primal variable perturbation
RRR	Reduced-rank regression
SDR	Semidefinite relaxation

TLS	Total least-squares
WSN	Wireless sensor network

Symbols

$(\cdot)^{\mathsf{T}}$	Transpose of a matrix
$\mathbf{I}_M$	$M \times M$ identity matrix
$\mathbb{E}[\cdot]$	Expectation operator
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of positive real numbers
$\mathbf{0}_{M \times N}$	$M \times N$ matrix with all zeros entries
$\mathbf{0}_M$	$M \times 1$ vector with all zeros entries
$\mathbf{L}$	Laplacian matrix
$\mathcal{E}$	Set of edges
$\mathcal{G}$	Graph
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\mathcal{V}$	Set of vertices
$\|\|\mathbf{x}\|\|_{\mathbf{A}}^2$	Quadratic form $\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x}$
$\otimes$	Kronecker-product operator
∂f	Subdifferential of function f
$\|\| \cdot \|\|$	Euclidean norm
$\|\| \cdot \|\|_*$	Nuclear norm
$\|\| \cdot \|\|_\infty$	Infinity norm
$\|\| \cdot \|\|_{\mathrm{F}}$	Frobenius norm
$\|\mathcal{V}\|$	Cardinality of the set $\mathcal{V}$

e_{ij}	Graph edge from agent i to agent j
f'	Subgradient of function f
f^*	Conjugate function of function f
$\lambda_{\max}(\cdot)$	Nonzero largest eigenvalue of a positive semidefinite matrix
$\lambda_{\min}(\cdot)$	Nonzero smallest eigenvalue of a positive semidefinite matrix
$\mathrm{cov}[\cdot]$	Covariance operator
$\det(\cdot)$	Determinant of a matrix
$\mathrm{tr}(\cdot)$	Trace of a matrix
$\mathrm{vec}(\cdot)$	Matrix-to-vector operator

Chapter 1

Introduction

With the recent advances in technology, large quantities of data are collected by numerous sensors, which are often geographically dispersed. Hence, performing data analytic tasks such as estimation and classification at a central processing unit in a distributed network can be infeasible due to the associated computation/communication costs. In addition, collecting all the data in a central hub creates a single point of failure. Therefore, it is necessary to develop algorithms that facilitate in-network processing and model learning using data collected by nodes/agents that are dispersed over a network. They ought to operate in a distributed fashion relying only on the available local information. In this context, each agent in the network only possesses information of a local cost function and the agents aim to collaboratively minimize the sum of the local objective functions. Distributed optimization problems pertain to several applications in statistics, signal processing, machine learning, and control [6–10].

An important issue associated with distributed learning is how the data is distributed among the agents. Horizontal partitioning of data refers to when subsets of data samples with a common set of features are distributed over the network. Examples of learning with horizontal partitioning of data can be found in [11–14]. However, many regression or classification problems encountered in machine learning deal with heterogeneous data that do not contain common features. These problems lead to the so-called feature (column) partitioning of the data where subsets of features of all data samples are distributed over the network agents. Distributed learning problems with feature partitioning also arise in several signal processing applications, e.g., bioinformatics, multi-view learning, and dictionary learning [5, 15]. Further technical details about horizontal and feature partitioning is given in Sections 2.1.1 and 2.1.2, respectively.

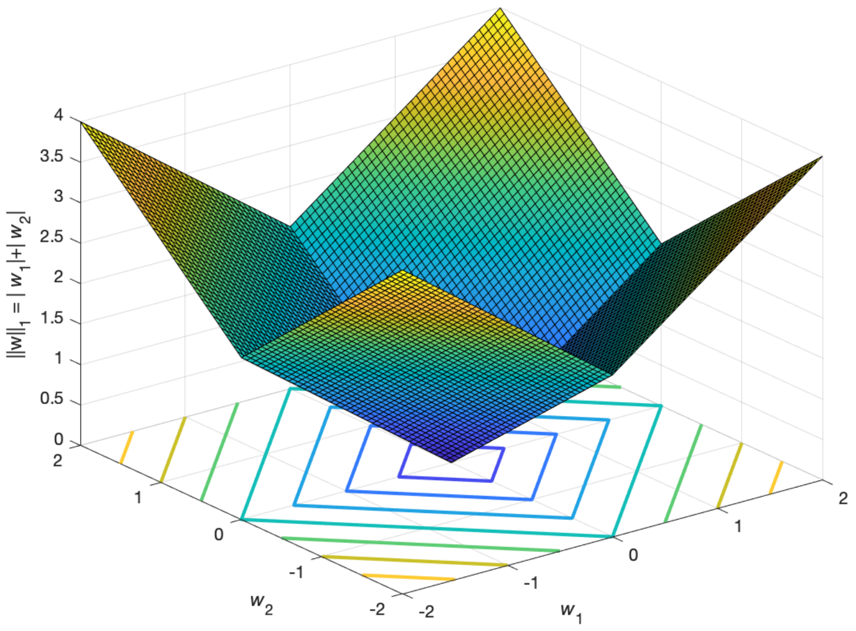


Figure 1.1: ℓ_1-norm regularization (lasso penalty).

Regarding distributed learning with horizontal partitioning of data, we consider a distributed solution to the total least-squares (TLS) problem and a distributed solution for non-smooth optimization problems when the first-order information is unavailable.

In the realm of linear estimation, the TLS method has been introduced as an alternative to the ordinary least-squares method to deal with errors-in-variables models. In such models, both independent and dependent variables are corrupted by noise or perturbation. TLS has been successfully used in several signal processing applications, e.g., frequency estimation of power systems [16–18], cognitive spectrum sensing [19], system identification [20], and wireless sensor networks [14]. The existing consensus-based approaches to distributed TLS estimation are proposed in [14, 21], however, their convergence highly depends on the choice of the step-size, whose optimal tuning requires the global knowledge of the data and network topology. In contrast, the proposed distributed ADMM TLS (DA-TLS) algorithm does not require careful tuning of any design parameter.

There have been several works developing algorithms for solving distributed convex optimization problems over ad-hoc networks. However, many existing algorithms only offer solutions for problems with smooth objective functions, see, e.g., [22–24]. In some real-world problems, obtaining first-order information is hard due to non-smooth objectives [8, 9, 11] or lack of any complete objective function. Non-smooth objectives arise, for example, in finding sparse solutions that are required, e.g., in compressed sensing and cognitive radio [25]. In this context, a common way of finding sparse solutions consists in solving the lasso problem whose non-smooth regularizer function is shown in Figure 1.1. Another method for sparse optimization, which we consider in our simulations is the reduced-rank regression (RRR). In the RRR problem, the objective function is least squares with nuclear norm regularization [7]. Nuclear norm is a non-smooth function that is used as a convex surrogate for the rank. RRR has applications in robust PCA [26], low-rank matrix decomposition [27], matrix completion [28], etc. There are some other scenarios where the complete objective function is not available and we only have access to zeroth-order information, i.e., function values, e.g., in bandit optimization [29], in simulation-based optimization [30], or in adversarial black-box machine learning [31]. This motivates the use of zeroth-order methods, which only use the values of the objective functions to approximate their gradients [4, 32, 33].

Most existing algorithms for non-smooth optimization rely on calculating subgradients, e.g., [9, 10] or proximal operators, e.g., [5, 11]. However, the computation of subgradients might be hard to achieve for some objectives and the derivation of proximal operators might not be feasible in some scenarios. The proposed distributed zeroth-order based ADMM (D-ZOA) algorithm is fully-distributed and solves optimization problems when first-order information is not available. D-ZOA is based on a zeroth-order method and, therefore, it only requires function values to approximate gradients of the non-smooth objective function.

Regarding distributed learning with feature partitioning of data, we first consider a distributed solution to the ridge regression problem. Next, we deal with a generalization of the last algorithm, which allows for distributed optimization with feature partitioning and non-smooth regularizers. Finally, we consider a further generalization allowing distributed optimization with feature partitioning of data and generic non-smooth objectives.

Shrinkage methods such as ridge regression and lasso have attracted a lot of attention since they play an important role in regularizing the learning parameters by imposing a penalty on their size to avoid overfitting [11, 34]. Moreover, ridge regression prevents the problem from being ill-posed due to possible rank deficiency of the observation matrix [35]. In the ridge regression and lasso problems, the objective functions are least squares with ℓ_2-norm regularization and ℓ_1-norm regularization, respectively. An existing work on distributed ridge regression with feature partitioning, i.e., [35], is based on the diffusion strategy. However, it suffers from relatively slow convergence. In this respect, we propose the D-Ridge algorithm that outperforms its diffusion-based contender in terms of convergence rate. Other approaches to distributed learning with feature partitioning are only designed for specific objectives, e.g., lasso in [36] and basis pursuit in [37] or assume an appropriate coloring scheme of the network and cannot be extended to a general graph labeling [38]. Some other existing approaches are not fully-distributed [5] or rely on the calculation of conjugate functions, which may be hard or unfeasible for some objective functions. We first consider an extension of [22] in [39] where we propose an algorithm to solve feature-distributed learning problems with an ℓ_2-norm-square cost function and non-smooth regularizer functions over arbitrary graphs while not relying on any conjugate of the regularizer function. Next, we consider a further generalization of the previous algorithm in [1] where we propose a fully-distributed distributed algorithm for optimizing generic non-smooth objective functions over arbitrary graphs without using or computing any conjugate function.

However, the communications between neighboring agents in a distributed network may pose privacy violation risks. An adversary may infer sensitive data of one or more agents by sniffing the communicated information. The adversary can be either a curious member of the network or an eavesdropper. Therefore, it is important to develop privacy-preserving methods that allow distributed processing of data without revealing private information. Differential privacy provides privacy protection against adversarial attacks by ensuring minimal change in the outcome of the algorithm regardless of whether or not a single individual's data is taken into account. Most existing privacy-preserving distributed optimization/estimation algorithms exploit some perturbation mechanism to preserve privacy, which comes at the cost of reduced accuracy, see, e.g., [40, 41]. Contrarily, by analyzing the inherent randomness due to the use of a zeroth-order method, we show that the proposed D-ZOA algorithm is intrinsically endowed with $(\epsilon, \delta)-$differential privacy. D-ZOA outperforms the existing differentially-private approaches in terms of accuracy while yielding similar privacy guarantee.

1.1 Objectives

The goals of the thesis are twofold. First, we aim to design robust distributed solutions for machine learning over networks with horizontal and feature partitioning of data. The second goal consists in designing an algorithm that outperforms the existing differentially-private approaches in terms of accuracy while yielding similar privacy guarantee. An itemized summary of the main objectives of the thesis is as follows:

- **O1**: Designing robust algorithms for distributed optimization with horizontal and feature partitioning of data.
- **O2**: Improving the accuracy of privacy-preserving distributed algorithms.

1.2 Methodology

The thesis proposes algorithms and methods to address the objectives **O1** and **O2** described in Section 1.1. The motivations for the thesis are in accordance with the literature on distributed optimization and privacy preserving data analysis. We employ tools from optimization and statistical modeling in order to deal with the considered learning problems in distributed optimization and private data analysis. The results and conclusions presented in the thesis stem from rigorous analysis. The proposed methods are compared with the existing approaches to show their effectiveness. Network-wide convergence of the proposed algorithms is confirmed via theoretical analysis and simulation results.

1.3 Thesis Contributions

In the thesis, we develop algorithms and methods to address **O1** and **O2**. Regarding the objective summarized in **O1** and, more specifically, the algorithms for distributed learning with horizontal partitioning, we first propose a distributed algorithm to solve the TLS problem. The proposed algorithm, called DA-TLS, is fully distributed and its performance is not sensitive to the tuning of its parameters. This makes DA-TLS more flexible and suitable for distributed deployment in comparison with the algorithms of [14, 21]. Simulation results show faster convergence of DA-TLS to the centralized solution at all agents in comparison with the existing algorithms. Next, we develop a fully-distributed algorithm to solve an optimization problem with a non-smooth convex objective function over an ad-hoc network. We utilize the alternating direction method of multipliers (ADMM) for distributed optimization and a zeroth-order method that is suitable for non-smooth objectives to obtain an approximate minimizer of the augmented Lagrangian in the ADMM's primal update step. The proposed algorithm, called D-ZOA, is fully distributed and does not compute any subgradient. It only requires the objective function values to approximate the gradient of the augmented Lagrangian. The simulations show that D-ZOA is competitive even on a problem that can be solved using a subgradient-based algorithm. Furthermore, the experiments show the usefulness of D-ZOA on a problem where calculating any subgradient is impractical.

Table 1.1: Comparative Summary for [1]

	fully-distributed	non-smooth cost function	non-smooth regularizer	no conjugate function
[39]	✓		✓	
[22]	✓			
[5]		✓	✓	
[15]	✓			✓
[37]	✓	✓		
[36]			✓	✓
[38]	✓		✓	
[42]				✓
[43]				
[44]				✓
[45]				
[46]		✓	✓	✓
[47]	✓	✓		
[35]	✓			✓
[48]			✓	
[49]	✓		✓	
[50]			✓	✓
[51]				✓
[52]	✓			✓
[53]	✓			✓
[54]	✓			✓
[55]	✓			✓
proposed [1]	✓	✓	✓	✓

Regarding the contributions within distributed learning with feature partitioning, we first propose a distributed algorithm, called D-Ridge to solve the ridge regression problem with feature partitioning of the observation matrix. D-Ridge is fully distributed and is based on the ADMM. It also converges faster than the diffusion-based algorithm of [35] and has a per-iteration per-agent computational complexity order that is linear in the sample size. Our experiments with a variety of network topologies show that D-Ridge outperforms its diffusion-based contender in terms of convergence rate. Subsequently, we develop a new fully-distributed algorithm to solve feature-distributed learning problems with an ℓ_2-norm-square cost function and non-smooth regularizer functions. The cost function cannot be written as the sum of the local agent-specific cost functions, i.e., it is not separable. To achieve separability, we formulate the dual problem associated with the underlying convex optimization problem and exploit its favorable structure that, unlike the original problem, allows us to solve it by utilizing the ADMM. By using the dual of the optimization problem associated with the ADMM primal variable update step, we devise a new strategy that does not require any conjugate function of the non-smooth regularizers, which may be infeasible or hard to obtain in some scenarios. Finally, we extend D-Ridge and the previous algorithm by developing a fully-distributed algorithm for solving learning problems when the data is distributed among agents in feature partitions and computing the conjugate of the possibly non-smooth cost or regularizer functions is challenging or unfeasible. We consider a general regularized non-smooth learning problem whose cost function is not separable. To tackle the problem, we articulate the associated dual optimization problem and utilize the ADMM to solve it as, unlike the original problem, its structure is suitable for distributed treatment via the ADMM. We then consider the dual of the optimization problem associated with the ADMM update step and solve it via the BCD algorithm. In that manner, we devise an approach that enables us to avoid the explicit computation of any conjugate function, which may be hard or infeasible for some objective functions. We demonstrate that the proposed algorithm approaches the optimal centralized solution at all agents. Our experiments show that the proposed algorithm converges to the optimal solution in various scenarios and is competitive with the relevant existing algorithms even when dealing with problems that, unlike its contenders, it is not tailored for. In Table 1.1, we provide a comparative summary of the proposed algorithm with respect to the most relevant existing ones in terms of the key features of being fully distributed, ability to handle non-smooth cost or regularization functions, and non-reliance on any conjugate function.

Regarding the objective summarized in **O2**, we show that the zeroth-order approach in D-ZOA is able to estimate the gradient in such a way that the inherent randomness in the gradient estimate can be exploited to protect privacy. Unlike the existing differentially-private algorithms that perturb the updates exchanged among the agents by adding noise, we exploit the inherent randomness due to the use of a zeroth-order method for solving the primal update optimization step to guarantee $(\epsilon, \delta)-$differential privacy. We show that the proposed D-ZOA algorithm is able to preserve privacy without requiring any explicit perturbation of the primal or dual variables. We also show that the total privacy leakage of the proposed D-ZOA algorithm grows sublinearly with the number of ADMM iterations. Moreover, our proposed D-ZOA algorithm outperforms the existing differentially-private approaches in terms of accuracy while yielding similar privacy guarantee.

Table 1.2: Comparative summary for [2]

	fully-distributed	non-strongly-convex objectives	non-smooth objectives	zeroth-order optimization	differential privacy-preserving	total privacy-preserving	intrinsic privacy-preserving
[13]	✓	✓	✓	✓			
[11]	✓	✓	✓				
[8]		✓	✓	✓			
[4]		✓	✓	✓			
[32]		✓	✓	✓			
[41]	✓				✓		
[56]	✓				✓	✓	
[57]	✓				✓	✓	
[58]	✓	✓			✓	✓	
[59]					✓	✓	
[60]	✓		✓		✓	✓	
[40]		✓	✓		✓	✓	
[61]	✓	✓	✓				✓
[62]		✓			✓		
[63]	✓						
[64]	✓	✓					
proposed [2]	✓	✓	✓	✓	✓	✓	✓

We prove that D-ZOA reaches a neighborhood of the optimal solution. Our convergence analysis also reveals a practically important trade-off between privacy and accuracy of D-ZOA. In Table 1.2 below, we have highlighted the contributions regarding **O2** in reference to the existing related literature. As illustrated in Table 1.2, most existing works assume that the objective function is smooth and strongly convex and the relevant first-order information is available. However, there are many applications where the objective functions are non-smooth or their first-order information is unavailable. This problem has been addressed, for example, in [40] by considering first-order approximations to non-smooth functions. However, in our work, motivated by the fact that only approximate gradient knowledge is sufficient for optimization, we use a zeroth-order method to solve the minimization in the ADMM primal update step and show that the randomness intrinsic to the employed zeroth-order method is sufficient to impart (ϵ, δ)-differential privacy.

1.3.1 List of Publications

This dissertation comprises a comprehensive summary of the research wherein parts have been published in the following papers, containing minute details on the algorithm derivations, proofs of theorem, and lemmas.

- **P1**: [12] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Consensus-based distributed total least-squares estimation using parametric semidefinite programming," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5227–5231, May 2019.

- **P2**: [13] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed learning with non-smooth objective functions," in *Proc. 28th European Signal Processing Conference*, pp. 2180–2184, Jan. 2021.
- **P3**: [2] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Privacy-preserved distributed learning with zeroth-order optimization," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 265-279, 2022.
- **P4**: [22] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed ridge regression with feature partitioning," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018.
- **P5**: [39] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed learning over networks with non-smooth regularizers and feature partitioning," in *Proc. European Speech and Signal Processing Conference*, Aug. 2021.
- **P6**: [1] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Decentralized optimization with distributed features and non-smooth objective functions," 2022, *arXiv: 2208.11224.*

The author of this dissertation has been responsible for the overall planning of the research work, planning the research problem, developing the original ideas, performing simulations, and writing the papers. The remaining authors have given feedback to guide the research and writing.

1.4 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we provide the background on distributed optimization and privacy. Regarding distributed optimization, we first present an essential aspect of distributed learning, namely, horizontal and feature partitioning of data. Subsequently, we present relevant optimization methods to tackle the considered learning problems and review the fundamentals of differential privacy. In Chapter 3, we consider learning problems with horizontal partitioning of data. We propose a new distributed algorithm to solve the TLS problem. Further, we develop a new distributed algorithm to solve a learning problem with non-smooth objective functions when data are distributed over a multi-agent network. In Chapter 4, we consider a privacy-preserving distributed algorithm to minimize a regularized empirical risk function when the first-order information is not available and data is distributed over a multi-agent network. We show that the proposed algorithm has intrinsic privacy-preserving properties. In Chapter 5, we develop a distributed algorithm to solve the ridge regression problem with feature partitioning of the observation matrix. Subsequently, we consider a generalization of this algorithm, which allows for distributed learning with feature partitioning and non-smooth regularizers. Finally, we devise further generalization that considers learning under distributed features with generic non-smooth objective functions. In Chapter 6, we draw conclusions and discuss some possible future work and research directions.

Chapter 2

Distributed Optimization and Privacy

2.1 Distributed Optimization

Performing optimization tasks in a centralized setting where all the data is collected at a fusion center may be prohibitive due to the shortcomings mentioned in the introductory chapter. Therefore, we consider a different setting where data is collected/owned by agents dispersed over a distributed network. In this context, each agent of the network communicates only with its immediate neighbors and no central coordination is necessary. Each agent has access only to the information of its local objective function while the agents aim to optimize the aggregate of the local objective functions collaboratively.

We consider a network with $N \in \mathbb{N}$ agents and $E \in \mathbb{N}$ edges modeled as an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V} = \{1, \ldots, N\}$ corresponds to the agents and the set $\mathcal{E}$ represents the bidirectional communication links between the pairs of neighboring agents. Edge $e_{ij} = (i, j) \in \mathcal{E}$ indicates that agent i and j are neighbors. Agent $i \in \mathcal{V}$ can communicate only with the agents in its neighborhood $\mathcal{V}_i = \{j \in \mathcal{V} \,|\, (i, j) \in \mathcal{E}\}$.

2.1.1 Horizontal Partitioning

Horizontal partitioning of data refers to the case when the data samples containing all features are distributed over the network. That is, all the agents estimate the same common model. Examples of learning with horizontal partitioning of data can be found in [6, 12–14]. Data with horizontal partitioning can also be referred to as instance-distributed data [65], data with row-partitioning [37], or homogeneous

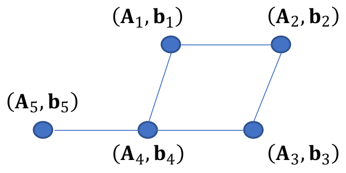

$(\mathbf{A}_1, \mathbf{b}_1)$
$(\mathbf{A}_2, \mathbf{b}_2)$
$(\mathbf{A}_5, \mathbf{b}_5)$
$(\mathbf{A}_4, \mathbf{b}_4)$
$(\mathbf{A}_3, \mathbf{b}_3)$

Figure 2.1: Horizontal partitioning of data over a network with five agents.

data [65].

Let us denote the network-wide data as an observation matrix[1] $\mathbf{A} \in \mathbb{R}^{M \times P}$ and a response vector $\mathbf{b} \in \mathbb{R}^{M \times 1}$ where M is the number of data samples and P is the total number of features across the network. As we consider the horizontal partitioning of the data, we denote the observation matrix of the ith agent by $\mathbf{A}_i \in \mathbb{R}^{M_i \times P}$ where M_i is the number of data samples specific to agent i. Accordingly, we have $M = \sum_{i=1}^{N} M_i$, $\mathbf{A}$ consists of N submatrices $\mathbf{A}_i$ as

$$\mathbf{A} = [\mathbf{A}_1^{\mathrm{T}}, \mathbf{A}_2^{\mathrm{T}}, \ldots, \mathbf{A}_N^{\mathrm{T}}]^{\mathrm{T}},$$

and the response vector $\mathbf{b}$ is a stack of N subvectors $\mathbf{b}_i$ as

$$\mathbf{b} = \left[\mathbf{b}_1^{\mathrm{T}}, \mathbf{b}_2^{\mathrm{T}}, \ldots, \mathbf{b}_N^{\mathrm{T}}\right]^{\mathrm{T}}.$$

The network-wide model vector that relates $\mathbf{A}$ and $\mathbf{b}$ is denoted by $\mathbf{x} \in \mathbb{R}^{P \times 1}$. We give an example for horizontal partitioning of data in Fig. 2.1 where data samples are distributed over a network of five agents. In the context of distributed learning with horizontal partitioning, we consider the problem when the N agents of the network solve the following minimization problem collaboratively

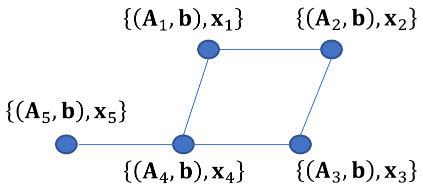
$$\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x}; \mathbf{A}_i, \mathbf{b}_i) \tag{2.1}$$

where $f_i : \mathbb{R}^P \rightarrow \mathbb{R}$ is the local cost function.

2.1.2 Feature Partitioning

When subsets of the features of all data samples are distributed over the network agents, we have feature partitioning of the data and every agent estimates a local model that is a part of the network-wide model. In the machine learning terminology, features are the descriptors or measurable characteristics of the data samples.

[1]Letter $\mathbf{A}$ is used in unified notation throughout the thesis to denote the observation matrix in accordance with common use in the literature, see, e.g., [5, 6]



$\{(\mathbf{A}_1, \mathbf{b}), \mathbf{x}_1\}$
$\{(\mathbf{A}_2, \mathbf{b}), \mathbf{x}_2\}$
$\{(\mathbf{A}_5, \mathbf{b}), \mathbf{x}_5\}$
$\{(\mathbf{A}_4, \mathbf{b}), \mathbf{x}_4\}$
$\{(\mathbf{A}_3, \mathbf{b}), \mathbf{x}_3\}$

Figure 2.2: Distributed features over a network with five agents.

In regression analysis, they may be called predictors or independent explanatory variables.

Several machine learning problems deal with heterogeneous distributed data that common features cannot describe. For example, in multi-agent systems, each agent may acquire data to learn a local model and refrains from sharing the data with other agents due to resource constraints or privacy concerns. However, the aggregate data can be exploited to enhance accuracy or augment inference due to the correlation of the data across agents. In the Internet of things, a device may only be interested in estimating its own local model parameters. However, multiple devices distributed over an ad hoc network may be able to collectively process the network-wide data and enhance the estimation/inference quality. Distributed learning problems with feature partitioning arise in several signal processing applications, e.g., bioinformatics, multi-view learning, and dictionary learning, as mentioned in [5, 15]. Data with feature partitioning can also be referred to as attribute-distributed data [65], vertically-partitioned data [66, 67], data with column-partitioning [37], or heterogeneous data [65].

In the context of distributed learning with feature partitioning of the data, we denote the observation matrix of the ith agent by $\mathbf{A}_i \in \mathbb{R}^{M\times P_i}$ and its local model vector by $\mathbf{x}_i \in \mathbb{R}^{P_i\times 1}$ where P_i is the number of features specific to agent i. Accordingly, we have $P = \sum_{i=1}^{N} P_i$ and the observation matrix $\mathbf{A}$ consists of N submatrices $\mathbf{A}_i$ as

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N].$$

The network-wide model vector $\mathbf{x} \in \mathbb{R}^{P\times 1}$ that relates $\mathbf{A}$ and the response vector $\mathbf{b}$ is also a stack of N subvectors $\mathbf{x}_i$ as

$$\mathbf{x} = \left[\mathbf{x}_1^{\mathrm{T}}, \mathbf{x}_2^{\mathrm{T}}, \ldots, \mathbf{x}_N^{\mathrm{T}}\right]^{\mathrm{T}}.$$

We give an example for feature partitioning of data in Fig. 2.2 where features are distributed over a network of five agents. We consider the problem when the N

agents of the network solve the following minimization problem collaboratively

$$\min_{\{\mathbf{x}_i\}} \sum_{i=1}^{N} f_i(\mathbf{x}_i; \mathbf{A}_i, \mathbf{b}) \tag{2.2}$$

where $f_i : \mathbb{R}^{P_i} \to \mathbb{R}$ is the local cost function.

2.2 The Alternating Direction Method of Multipliers

All the algorithms proposed in this thesis are based on the alternating direction method of multipliers (ADMM) whose use is motivated by the fact that this method is suitable for distributed optimization since it is robust and scalable with respect to both the data size and the network size. In this section, we first discuss the consensus-based reformulation of the optimization problems (2.1) and (2.2), which allows us to find a distributed solution via the ADMM. Then, we describe the ADMM procedure for solving the resulting constrained minimization problem. This iterative process consists of two steps at each iteration, i.e., the primal and the dual update steps.

2.2.1 Consensus-Based Reformulation

To solve (2.1) and (2.2) in a distributed manner, we reformulate them as the following constrained minimization problem

$$\begin{aligned} \min_{\{\mathbf{x}_i\}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i; \mathcal{X}_i) \\ \text{s.t.} \quad & \mathbf{x}_i = \mathbf{x}_j, \quad j \in \mathcal{V}_i, \quad \forall i \in \mathcal{V} \end{aligned} \tag{2.3}$$

where $\mathcal{X}_i$ represents the local information at agent i. In the case of horizontal partitioning of data, $\mathcal{X}_i$ is given by $(\mathbf{A}_i, \mathbf{b}_i)$ as in Section 2.1.1. The primal variables $\mathcal{P} := \{\mathbf{x}_i\}_{i\in\mathcal{V}}$ represent local copies of $\mathbf{x}$ at the agents. In the scenario of feature partitioning of data, $\mathcal{X}_i$ is given by $(\mathbf{A}_i, \mathbf{b})$ as in Section 2.1.2. Here, unlike the case of horizontal partitioning of data, we do not need to introduce the local variables because the primal variables are already local model vectors due to feature partitioning.

Since the network is connected, the equality constraints in (2.3) impose consensus across each agent's neighborhood $\mathcal{V}_i$. To solve (2.3) collaboratively and in a fully-distributed manner, we utilize the ADMM [7]. For this purpose, we rewrite (2.3) as

$$\begin{aligned} \min_{\{\mathbf{x}_i\}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i; \mathcal{X}_i) \\ \text{s.t.} \quad & \mathbf{x}_i = \mathbf{z}_i^j,\ \mathbf{x}_j = \mathbf{z}_i^j, \quad j \in \mathcal{V}_i, \quad \forall i \in \mathcal{V} \end{aligned} \tag{2.4}$$

where $\mathcal{A} := \{\mathbf{z}_i^j\}_{i\in\mathcal{V},j\in\mathcal{V}_i}$ are the auxiliary variables yielding an alternative but equivalent representation of the constraints in (2.3). They help decouple $\mathbf{x}_i$ in the constraints and facilitate the derivation of the local recursions before being eventually eliminated. The augmented Lagrangian function is given by

$$\mathcal{L}_\rho(\mathcal{P},\mathcal{A},\mathcal{D}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i;\mathcal{X}_i) + \sum_{i=1}^{N}\sum_{j\in\mathcal{V}_i}\left[\bar{\boldsymbol{\lambda}}_i^{j\mathsf{T}}\left(\mathbf{x}_i-\mathbf{z}_i^j\right)+\tilde{\boldsymbol{\lambda}}_i^{j\mathsf{T}}\left(\mathbf{x}_j-\mathbf{z}_i^j\right)\right] \\ +\frac{\rho}{2}\sum_{i=1}^{N}\sum_{j\in\mathcal{V}_i}\left(\left\|\mathbf{x}_i-\mathbf{z}_i^j\right\|^2+\left\|\mathbf{x}_j-\mathbf{z}_i^j\right\|^2\right) \tag{2.5}$$

where $\mathcal{D} := \{\{\bar{\boldsymbol{\lambda}}_i^j\}_{j\in\mathcal{V}_i},\{\tilde{\boldsymbol{\lambda}}_i^j\}_{j\in\mathcal{V}_i}\}_{i\in\mathcal{V}}$ are the Lagrange multipliers associated with the constraints in (2.4), and $\rho > 0$ is a penalty parameter.
The ADMM will use the augmented Lagrangian in (2.5) to solve the optimization problem in (2.3). The ADMM will require an iterative process that is described in the next subsection.

2.2.2 Distributed ADMM Algorithm

To solve the minimization problem (2.4) in a distributed fashion, the ADMM entails an iterative procedure consisting of three steps at each iteration. In the first step, the augmented Lagrangian function $\mathcal{L}_\rho$ is minimized with respect to the primal variables $\mathcal{P}$, i.e., .

$$\mathcal{P}^{(k)} = \arg\min_{\mathcal{P}} \mathcal{L}_\rho(\mathcal{P},\mathcal{A}^{(k-1)},\mathcal{D}^{(k-1)}) \tag{2.6}$$

where superscript (k) denotes the iteration index of the ADMM loop and

$$\begin{aligned}\mathcal{P}^{(k)} &:= \{\mathbf{x}_i^{(k)}\}_{i\in\mathcal{V}}\\ \mathcal{A}^{(k)} &:= \{\{(\mathbf{z}_i^j)^{(k-1)}\}_{i\in\mathcal{V},j\in\mathcal{V}_i}\\ \mathcal{D}^{(k)} &:= \{\{(\bar{\boldsymbol{\lambda}}_i^j)^{(k-1)}\}_{j\in\mathcal{V}_i},\{(\tilde{\boldsymbol{\lambda}}_i^j)^{(k-1)}\}_{j\in\mathcal{V}_i}\}_{i\in\mathcal{V}}.\end{aligned}$$

Then, $\mathcal{L}_\rho$ is minimized with respect to the auxiliary variables $\mathcal{A}$, i.e.,

$$\mathcal{A}^{(k)} = \arg\min_{\mathcal{A}} \mathcal{L}_\rho(\mathcal{P}^{(k)},\mathcal{A},\mathcal{D}^{(k-1)}). \tag{2.7}$$

In the end, the Lagrange multipliers in $\mathcal{D}$ are updated via dual gradient-ascent iterations [68], i.e.,

$$\begin{aligned}(\bar{\boldsymbol{\lambda}}_i^j)^{(k)} &= (\bar{\boldsymbol{\lambda}}_i^j)^{(k-1)}+\rho[\mathbf{x}_i^{(k)}-(\mathbf{z}_i^j)^{(k)}], \quad j\in\mathcal{V}_i, \quad \forall i\in\mathcal{V}\\ (\tilde{\boldsymbol{\lambda}}_i^j)^{(k)} &= (\tilde{\boldsymbol{\lambda}}_i^j)^{(k-1)}+\rho[\mathbf{x}_j^{(k)}-(\mathbf{z}_i^j)^{(k)}], \quad j\in\mathcal{V}_i, \quad \forall i\in\mathcal{V}.\end{aligned} \tag{2.8}$$

Thanks to the reformulation of the original problems (2.1) and (2.2) as (2.4), the augmented Lagrangian in (2.5) is decomposable both with respect to variables in $\mathcal{P}$, $\mathcal{A}$ and across agents.

By using the Karush-Kuhn-Tucker conditions of optimality for (2.4) and setting

$$\boldsymbol{\lambda}_i^{(k)} = 2\sum_{j\in\mathcal{V}_i}(\bar{\boldsymbol{\lambda}}_i^j)^{(k)},$$

it can be shown that the auxiliary variables in $\mathcal{A}$ and the Lagrange multipliers $\{\tilde{\boldsymbol{\lambda}}_i^j\}_{i\in\mathcal{V},j\in\mathcal{V}_i}$ are eliminated [7, 11, 69]. Therefore, the distributed ADMM algorithm reduces to the following iterative updates at the ith agent

$$\mathbf{x}_i^{(k)} = \arg\min_{\mathbf{x}_i}\Big\{f_i(\mathbf{x}_i;\mathcal{X}_i) + (\boldsymbol{\lambda}_i^{(k-1)})^{\mathrm{T}}\mathbf{x}_i + \rho\sum_{j\in\mathcal{V}_i}\left\|\mathbf{x}_i - \frac{\mathbf{x}_i^{(k-1)}+\mathbf{x}_j^{(k-1)}}{2}\right\|^2\Big\} \tag{2.9}$$

$$\boldsymbol{\lambda}_i^{(k)} = \boldsymbol{\lambda}_i^{(k-1)} + \rho\sum_{j\in\mathcal{V}_i}[\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}] \tag{2.10}$$

where all initial values $\{\mathbf{x}_i^{(0)}\}_{i\in\mathcal{V}}$, $\{\boldsymbol{\lambda}_i^{(0)}\}_{i\in\mathcal{V}}$ are set to zero.

To show that (2.6), (2.7), and (2.8) are equivalent to (2.9) and (2.10), we first consider (2.7) and observe that $(\mathbf{z}_i^j)^{(k)}$ is given by

$$(\mathbf{z}_i^j)^{(k)} = \arg\min_{\mathbf{z}_i^j}\Big\{-((\bar{\boldsymbol{\lambda}}_i^j)^{(k)} + (\tilde{\boldsymbol{\lambda}}_i^j)^{(k)})^{\mathrm{T}}\mathbf{z}_i^j + \frac{\rho}{2}\Big[\left\|\mathbf{x}_i^{(k)} - \mathbf{z}_i^j\right\|^2 + \left\|\mathbf{x}_j^{(k)} - \mathbf{z}_i^j\right\|^2\Big]\Big\}. \tag{2.11}$$

Problem (2.11) has a closed form solution that is given by

$$(\mathbf{z}_i^j)^{(k)} = \frac{1}{2\rho}((\bar{\boldsymbol{\lambda}}_i^j)^{(k)} + (\tilde{\boldsymbol{\lambda}}_i^j)^{(k)}) + \frac{1}{2}(\mathbf{x}_i^{(k)} + \mathbf{x}_j^{(k)}). \tag{2.12}$$

Using (2.12) and (2.8), it can be shown by induction that, if

$$(\bar{\boldsymbol{\lambda}}_i^j)^{(0)} = -(\tilde{\boldsymbol{\lambda}}_i^j)^{(0)},$$

then

$$(\bar{\boldsymbol{\lambda}}_i^j)^{(k)} = -(\tilde{\boldsymbol{\lambda}}_i^j)^{(k)}$$

for all $k \geq 0$, where $i \in \mathcal{V}$ and $j \in \mathcal{V}_i$. Therefore $(\mathbf{z}_i^j)^{(k)}$ is given by

$$(\mathbf{z}_i^j)^{(k)} = \frac{1}{2}(\mathbf{x}_i^{(k)} + \mathbf{x}_j^{(k)}), \quad j \in \mathcal{V}_i, \quad \forall i \in \mathcal{V}. \tag{2.13}$$

Algorithm 1 Distributed ADMM

```
At all agents $i \in \mathcal{V}$, initialize $\mathbf{x}_i^{(0)} = \mathbf{0}$, $\boldsymbol{\lambda}_i^{(0)} = \mathbf{0}$, and locally run:
for $k = 1, 2, \ldots, K$ do
    Share $\mathbf{x}_i^{(k-1)}$ with the neighbors in $\mathcal{V}_i$.
    Update the primal variables $\mathbf{x}_i^{(k)}$ via (2.9).
    Update the Lagrange multipliers $\boldsymbol{\lambda}_i^{(k)}$ via (2.10).
end for
```

By replacing (2.13) into the first equation of (2.8), we obtain

$$(\bar{\boldsymbol{\lambda}}_i^j)^{(k)} = (\bar{\boldsymbol{\lambda}}_i^j)^{(k-1)} + \rho(\mathbf{x}_i^{(k)} - \mathbf{x}_j^{(k)}), \quad j \in \mathcal{V}_i, \quad \forall i \in \mathcal{V}. \tag{2.14}$$

From (2.14), we have that, if $(\bar{\boldsymbol{\lambda}}_i^j)^{(0)} = -(\bar{\boldsymbol{\lambda}}_j^i)^{(0)}$, then $(\bar{\boldsymbol{\lambda}}_i^j)^{(k)} = -(\bar{\boldsymbol{\lambda}}_j^i)^{(k)}$ for all $k \geq 0$. From

$$(\bar{\boldsymbol{\lambda}}_i^j)^{(k)} = -(\tilde{\boldsymbol{\lambda}}_i^j)^{(k)}$$

and

$$(\bar{\boldsymbol{\lambda}}_i^j)^{(k)} = -(\tilde{\boldsymbol{\lambda}}_j^i)^{(k)},$$

we have

$$(\bar{\boldsymbol{\lambda}}_i^j)^{(k)} = (\tilde{\boldsymbol{\lambda}}_j^i)^{(k)}.$$

This implies that (2.6) is equivalent to (2.9) upon defining

$$\boldsymbol{\lambda}_i^{(k)} = 2 \sum_{j \in \mathcal{V}_i} (\bar{\boldsymbol{\lambda}}_i^j)^{(k)}.$$

The iterations (2.9) and (2.10) can be implemented in a fully-distributed manner as they only involve the parameters available within each agent's neighborhood. The distributed ADMM algorithm is summarized in Algorithm 1 where K denotes the number of ADMM iterations.

2.2.3 Evaluation Metrics

In this section, we introduce the two metrics that are used throughout the thesis to evaluate the performance of the proposed algorithms.

The former is the normalized error that is defined as

$$\text{normalized error} = \frac{\sum_{i=1}^{N} \left\| \mathbf{x}_i^{(k)} - \mathbf{x}^c \right\|^2}{\left\| \mathbf{x}^c \right\|^2} \tag{2.15}$$

where $\mathbf{x}^c$ is the centralized solution to problem (2.1) and $\mathbf{x}_i^{(k)}$ is the local estimate at iteration k.

The latter is the misalignment and is defined as

$$\text{misalignment} = \frac{\left\|\mathbf{x}^d(k) - \boldsymbol{\omega}\right\|^2}{\|\boldsymbol{\omega}\|^2} \tag{2.16}$$

where

$$\mathbf{x}^d(k) = \left[\mathbf{x}_1^{(k)\mathsf{T}}, \ldots, \mathbf{x}_N^{(k)\mathsf{T}}\right]^{\mathsf{T}},$$

$\mathbf{x}_i^{(k)}$ are the local estimates and $\boldsymbol{\omega} \in \mathbb{R}^P$ is drawn from multivariate normal distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and is related to the response vector $\mathbf{b}$ and the data matrix $\mathbf{A} \in \mathbb{R}^{M\times P}$ via

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\psi}$$

where $\boldsymbol{\psi} \in \mathbb{R}^M$ is drawn from multivariate normal distributions $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$ independently from $\boldsymbol{\omega}$.

From the definitions, it is clear that the two metrics slightly differ since the normalized error evaluates the algorithm's performance by measuring the error between the centralized solutions $\mathbf{x}^c$ and the local estimates while the misalignment measures the error between the vector generating the data and the local estimates.

2.3 Zeroth-Order Methods

In some real-world problems, obtaining first-order information is hard due to non-smooth objectives or even impossible due to the lack of the complete cost function. For example, in some non-smooth optimization problems, computation of subgradients might be hard to achieve for some objectives, or derivation of proximal operators might not be feasible. In bandit optimization [29], an adversary generates a sequence of loss functions and the goal is to minimize such sequence that is only available at some points. In addition, in simulation-based optimization, the objective is available only using repeated simulation [30], and in adversarial black-box machine learning models, only the function values are given [31]. This motivates the use of zeroth-order methods requiring only function values to approximate gradients. In the thesis, the goal of employing a zeroth-order method is twofold. First, we provide a distributed solution for non-smooth optimization problems when the first-order information is unavailable. Second, the stochasticity due to the use of the zeroth-order method endows the proposed D-ZOA algorithm with intrinsic privacy-preserving properties.

2.3.1 Two-Point Stochastic-Gradient Algorithm

In the thesis, we employ a zeroth-order method to solve the minimization problem in the ADMM primal update step (2.9). Therefore, we consider the following minimization problem

$$\mathbf{x}_i^{(k)} = \arg\min_{\mathbf{x}_i} \mathcal{F}_i(\mathbf{x}_i). \tag{2.17}$$

Since the objective function in (2.17) is assumed to be non-smooth, the corresponding minimization problem cannot be solved using any first-order method. To overcome this, we use a zeroth-order method as in [13]. We utilize the two-point stochastic-gradient algorithm that has been proposed in [3] for optimizing general non-smooth functions. More specifically, we use the stochastic mirror descent method with the proximal function $\frac{1}{2}\|\cdot\|$ and the gradient estimator at point $\mathbf{x}_i$ given by

$$\begin{aligned}\Gamma(\mathbf{x}_i, \boldsymbol{\lambda}_i^{(k-1)}, u_1, u_2, \mathbf{n}_1, \mathbf{n}_2) = u_2^{-1}[&\mathcal{F}_i(\mathbf{x}_i + u_1\mathbf{n}_1 \\ &+ u_2\mathbf{n}_2, \boldsymbol{\lambda}_i^{(k-1)}) - \mathcal{F}_i(\mathbf{x}_i + u_1\mathbf{n}_1, \boldsymbol{\lambda}_i^{(k-1)})]\mathbf{n}_2\end{aligned} \tag{2.18}$$

where $u_1 > 0$ and $u_2 > 0$ are smoothing constants and $\mathbf{n}_1$, $\mathbf{n}_2$ are independent zero-mean Gaussian random vectors with the covariance matrix $\mathbf{I}_P$, i.e., $\mathbf{n}_1, \mathbf{n}_2 \sim \mathcal{N}(\mathbf{0}_P, \mathbf{I}_P)$.

The two-point stochastic-gradient algorithm consists of two randomization steps where the second step is aimed at preventing the perturbation vector $\mathbf{n}_2$ from being close to a point of non-smoothness [3]. This algorithm entails an iterative procedure that consists of three steps at each iteration t. First, $S \in \mathbb{N}$ independent random vectors $\{\mathbf{n}_{1,t}^{s,i}\}_{s=1}^S$ and $\{\mathbf{n}_{2,t}^{s,i}\}_{s=1}^S$ are sampled from $\mathcal{N}(\mathbf{0}_P, \mathbf{I}_P)$. Second, an i-local stochastic gradient $\mathbf{g}_i^{(t)}$ is computed as

$$\mathbf{g}_i^{(t)} = \frac{1}{S}\sum_{s=1}^{S} \mathbf{g}_{s,i}^{(t)} \tag{2.19}$$

where

$$\mathbf{g}_{s,i}^{(t)} = \Gamma(\mathbf{y}_i^{(t-1)}, \boldsymbol{\lambda}_i^{(k-1)}, u_{1,t}, u_{2,t}, \mathbf{n}_{1,t}^{s,i}, \mathbf{n}_{2,t}^{s,i}),$$

$\mathbf{y}_i^{(t)}$ is the tth iterate of the two-point stochastic-gradient algorithm with the initial value $\mathbf{y}_i^{(0)} = \mathbf{0}$ and $\{u_{1,t}\}_{t=1}^{\infty}$ and $\{u_{2,t}\}_{t=1}^{\infty}$ are two non-increasing sequences of positive parameters such that $u_{2,t} \le u_{1,t}/2$. Finally, $\mathbf{y}_i^{(t)}$ is updated as

$$\mathbf{y}_i^{(t)} = \mathbf{y}_i^{(t-1)} - \alpha_t \mathbf{g}_i^{(t)} \tag{2.20}$$

where α_t is a time-varying step-size. The step-size is computed as

$$\alpha_t = \frac{\alpha_0 R}{L\sqrt{tP\log(2P)}}$$

where α_0 is an appropriate initial step-size and R is an upper bound on the distance between the minimizer $\mathbf{x}_i^*$ to (2.17) and the first iterate $\mathbf{y}_i^{(1)}$ as per [3].

We use multiple independent random samples $\{\mathbf{n}_{1,t}^{s,i}\}_{s=1}^{S}$ and $\{\mathbf{n}_{2,t}^{s,i}\}_{s=1}^{S}$ to obtain a more accurate estimate of the gradient $\mathbf{g}_i^{(t)}$ as remarked in [3].

2.4 Differential Privacy

In this section, we consider the privacy concerns associated with distributed learning with horizontal partitioning. Second, we present a perturbation mechanism that enables the ADMM to preserve (ϵ, δ)-differential privacy. This mechanism ensures privacy protection by perturbing the ADMM primal variable. We provide a description of this Gaussian privacy preserving mechanism because the intrinsic privacy preserving properties of the proposed D-ZOA algorithm are due to an inherent primal variable perturbation, which is brought about by the intrinsic randomness due to the use of a zeroth-order method in the inner loop. Subsequently, we define the l_2-norm sensitivity of the primal variable, which is needed to compute the covariance that the primal variable is required to have so that the privacy leakage of a single iteration of the ADMM is bounded at each agent.

2.4.1 Attack Model and Privacy Concerns

In Algorithm 1, the data stored at each agent, $\mathcal{X}_i$, is not shared with any other agent. However, the local estimates $\{\mathbf{x}_i^{(k)}\}_{i\in\mathcal{V}}$ are exchanged within the local neighborhoods. Therefore, the risk of privacy breach still exists as it has been shown by the model inversion attacks [70].

In the thesis, we consider the following attack model. We assume that the adversary is able to access the local estimates $\{\mathbf{x}_i^{(k)}\}_{i\in\mathcal{V}}$ that are exchanged throughout the intermediate ADMM iterations as well as the final output. The adversary can be either a honest-but-curious member of the network or an external eavesdropper. The adversary's goal is to infer sensitive data of one or more agents by sniffing the communicated information $\{\mathbf{x}_i^{(k)}\}_{i\in\mathcal{V}}$.

We will show that D-ZOA guarantees (ϵ, δ)-differential privacy as per the below definition since it is intrinsically resistant to such inference attacks.

Definition 1. A randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if for any two neighboring datasets $\mathcal{S}$ and $\mathcal{S}'$ differing in only one data sample and for any subset of outputs $\mathcal{O} \subseteq \text{range}(\mathcal{M})$, we have

$$\Pr[\mathcal{M}(\mathcal{S}) \in \mathcal{O}] \leq e^{\epsilon}\Pr[\mathcal{M}(\mathcal{S}') \in \mathcal{O}] + \delta. \tag{2.21}$$

This means the ratio of the probability distributions of $\mathcal{M}(\mathcal{S})$ and $\mathcal{M}(\mathcal{S}')$ is bounded by e^{ϵ}.

In Definition 1, ϵ and δ are privacy parameters indicating the level of privacy preservation ensured by a differentially private algorithm. A better privacy preservation is achieved with smaller ϵ or δ. On the other hand, low privacy guarantee corresponds to higher values of ϵ, i.e., close to 1. Therefore, it is reasonable to assume that $\epsilon \in (0, 1]$ as in [58, 62].

2.4.2 Primal Variable Perturbation Mechanism

In this section, we present a Gaussian primal variable perturbation mechanism that enables the ADMM to preserve (ϵ, δ)-differential privacy. This mechanism consists in adding Gaussian noise to the local primal variable updates that are exchanged throughout the learning process, i.e.,

$$\tilde{\mathbf{x}}_i^{(k)} = \mathbf{x}_i^{(k)} + \boldsymbol{\xi}_i^{(k)} \tag{2.22}$$

where $\boldsymbol{\xi}_i^{(k)} \in \mathbb{R}^P$ is a random variable representing the perturbation. Therefore, the steps of the distributed ADMM with primal variable perturbation (PVP) are expressed as

$$\mathbf{x}_i^{(k)} = \arg\min_{\mathbf{x}_i}\Big\{ f_i(\mathbf{x}_i; \mathcal{X}_i) + (\boldsymbol{\lambda}_i^{(k-1)})^{\mathrm{T}}\mathbf{x}_i + \rho \sum_{j \in \mathcal{V}_i} \left\| \mathbf{x}_i - \frac{\mathbf{x}_i^{(k-1)} + \mathbf{x}_j^{(k-1)}}{2} \right\|^2 \Big\} \tag{2.23}$$

$$\tilde{\mathbf{x}}_i^{(k)} = \mathbf{x}_i^{(k)} + \boldsymbol{\xi}_i^{(k)} \tag{2.24}$$

$$\boldsymbol{\lambda}_i^{(k)} = \boldsymbol{\lambda}_i^{(k-1)} + \rho \sum_{j \in \mathcal{V}_i} [\tilde{\mathbf{x}}_i^{(k)} - \tilde{\mathbf{x}}_j^{(k)}] \tag{2.25}$$

where all initial values $\{\mathbf{x}_i^{(0)}\}_{i \in \mathcal{V}}$, $\{\boldsymbol{\lambda}_i^{(0)}\}_{i \in \mathcal{V}}$ are set to zero. The distributed ADMM algorithm with PVP is summarized in Algorithm 2. Since we consider a Gaussian perturbation mechanism, $\boldsymbol{\xi}_i^{(k)}$ is distributed as $\boldsymbol{\xi}_i^{(k)} \sim \mathcal{N}(\mathbf{0}_P, \sigma_i^2 \mathbf{I}_P)$. The magnitude of the noise by which $\mathbf{x}_i^{(k)}$ has to be perturbed to preserve privacy is calibrated by the l_2-norm sensitivity that is defined as

Definition 2. The l_2-norm sensitivity of $\mathbf{x}_i^{(k)}$ is defined as

$$\Delta_{i,2} = \max_{\mathcal{S}_i, \mathcal{S}_i'} \left\| \mathbf{x}_{i,\mathcal{S}_i}^{(k)} - \mathbf{x}_{i,\mathcal{S}_i'}^{(k)} \right\| \tag{2.26}$$

Algorithm 2 Distributed ADMM with PVP

```
At all agents $i \in \mathcal{V}$, initialize $\mathbf{x}_i^{(0)} = \mathbf{0}$, $\boldsymbol{\lambda}_i^{(0)} = \mathbf{0}$, and locally run:
for $k = 1, 2, \ldots, K$ do
    Share $\mathbf{x}_i^{(k-1)}$ with the neighbors in $\mathcal{V}_i$.
    Update the primal variables $\mathbf{x}_i^{(k)}$ via (2.23).
    Perturb the primal variables via (2.24).
    Update the Lagrange multipliers $\boldsymbol{\lambda}_i^{(k)}$ via (2.25).
end for
```

where $\mathbf{x}_{i,\mathcal{S}_i}^{(k)}$ and $\mathbf{x}_{i,\mathcal{S}_i'}^{(k)}$ denote the local primal variables for two neighboring datasets $\mathcal{S}_i$ and $\mathcal{S}_i'$ differing in only one data sample, i.e., one row of $\mathbf{A}_i$ and the corresponding entry of $\mathbf{b}_i$.

Chapter 3

Distributed Optimization with Horizontal Partitioning

In this chapter, we present two contributions in the context of distributed learning with horizontal partitioning of data. The former is a consensus-based distributed total least-squares estimation algorithm using parametric semidefinite programming. The latter is a distributed algorithm for non-smooth optimization problems when the first-order information is unavailable.

3.1 Distributed Total Least-Squares Estimation Using Parametric Semidefinite Programming

In [12], we propose a new distributed algorithm to solve the total least-squares (TLS) problem when data are distributed over a multi-agent network. To develop the proposed algorithm, named distributed ADMM TLS (DA-TLS), we reformulate the TLS problem as a parametric semidefinite program and solve it using the alternating direction method of multipliers (ADMM). Unlike the existing consensus-based approaches to distributed TLS estimation, DA-TLS does not require careful tuning of any design parameter. Numerical experiments demonstrate that the DA-TLS converges to the centralized solution significantly faster than the existing consensus-based TLS algorithms.

3.1.1 Related work

The distributed TLS problem has previously been considered in [14, 21, 71–77]. The works in [14, 21] are based on the consensus strategy and rely on the dual-based subgradient method. Their relatively high computational complexity has partially motivated the works in [72–76]. While the approach of [72] is based on the average consensus strategy, the algorithms in [73–77] are based on diffusion strategies and, therefore, suffer from relatively slow convergence [22]. The convergence speed of the algorithm proposed in [14] greatly depends on the network topology and dimensionality of the data. Although these shortcomings are mitigated in [21], the convergence rate of algorithms in [14] and [21] highly depends on the choice of the step-size, whose optimal tuning requires the global knowledge of the data and network topology.

3.1.2 Contributions

In [12], we tackle the TLS problem by transforming the non-convex distributed TLS problem into a semidefinite program through a change of variable from a vector to a rank-one matrix and subsequent semidefinite relaxation (SDR). We solve the modified problem using the alternating direction method of multipliers (ADMM) and a generalization of the algorithm proposed in [78] for fractional programming. Since the optimal solution is rank-one, the relaxation is tight and does not incur any loss of optimality [79]. In addition, as the objective function in the modified problem is the sum of fractions of linear functions, the convergence of the proposed algorithm to the globally optimal solution is guaranteed.

The proposed algorithm, called distributed ADMM TLS (DA-TLS), is fully distributed in the sense that it requires the agents to share data only with their immediate neighbors at each iteration. Furthermore, the performance of DA-TLS is not sensitive to the tuning of its parameters. This makes DA-TLS more flexible and suitable for distributed deployment in comparison with the algorithms of [14, 21]. Simulation results show faster convergence of DA-TLS to the centralized solution at all agents in comparison with the existing algorithms.

3.1.3 System Model

We consider the system model described in Section 2.1 with horizontal partitioning of the observation matrix as presented in Section 2.1.1. In addition to the symbols introduced in Section 2.1.1, let us denote by $\tilde{\mathbf{A}} \in \mathbb{R}^{M\times P}$ and $\tilde{\mathbf{b}} \in \mathbb{R}^{M\times 1}$ the error in the observation matrix and the error in the response vector, respectively. The sought-after parameter vector $\mathbf{x} \in \mathbb{R}^{P\times 1}$ relates $\mathbf{A}$ and $\mathbf{b}$ through $(\mathbf{A} - \tilde{\mathbf{A}})\mathbf{x} = \mathbf{b} - \tilde{\mathbf{b}}$.

The TLS estimate of the unknown parameter vector $\mathbf{x}$ can be found by solving the constrained optimization problem

$$\begin{aligned} \min_{\mathbf{x},\tilde{\mathbf{A}},\tilde{\mathbf{b}}} \quad & \left\|\tilde{\mathbf{A}}\right\|_{\mathrm{F}} + \left\|\tilde{\mathbf{b}}\right\| \\ \text{s.t.} \quad & (\mathbf{A} - \tilde{\mathbf{A}})\mathbf{x} = \mathbf{b} - \tilde{\mathbf{b}}. \end{aligned} \tag{3.1}$$

When the entries of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{b}}$ are independent and identically distributed (i.i.d.), a centralized TLS solution $\mathbf{x}^c$ of (3.1) can be obtained as

$$\mathbf{x}^c = \frac{-1}{v_{P+1}}[v_1, v_2, \ldots, v_P]^{\mathrm{T}} \tag{3.2}$$

where $\mathbf{v} = [v_1, v_2, \ldots, v_{P+1}]^{\mathrm{T}}$ is the right singular vector corresponding to the smallest singular value of $[\mathbf{A}, \mathbf{b}]$ [80].

An equivalent but more practical solution can be obtained by minimizing the Rayleigh quotient cost function as [80]

$$\min_{\mathbf{x}} \frac{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2}{\|\mathbf{x}\|^2 + 1} \quad \text{or} \quad \min_{\mathbf{x}} \sum_{i=1}^{N} \frac{\|\mathbf{A}_i\mathbf{x} - \mathbf{b}_i\|^2}{\|\mathbf{x}\|^2 + 1}. \tag{3.3}$$

Since finding a centralized solution of (3.3) over a network may be inefficient, we propose a distributed algorithm for this purpose in the following section.

3.1.4 Distributed TLS

We first discuss the SDR technique that allows us to transform the TLS problem into a parametric semidefinite program, which we solve iteratively through two nested loops. Then, we describe the consensus-based reformulation of the resultant parametric semidefinite program that enables its distributed solution via the ADMM, which forms the inner loop. Finally, we describe the steps of the inner and outer loops of the algorithm.

Semidefinite Relaxation

Using the properties of the matrix trace operator, we rewrite the Rayleigh quotient cost function in (3.3) as

$$\sum_{i=1}^{N} \frac{\mathrm{tr}(\mathbf{x}\mathbf{x}^{\mathrm{T}}\mathbf{A}_i^{\mathrm{T}}\mathbf{A}_i) - 2\mathbf{b}_i^{\mathrm{T}}\mathbf{A}_i\mathbf{x} + \|\mathbf{b}_i\|^2}{\mathrm{tr}(\mathbf{x}\mathbf{x}^{\mathrm{T}}) + 1}. \tag{3.4}$$

Considering (3.4) and defining

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{x} \\ \mathbf{x}^T & 1 \end{bmatrix} \text{ and } \mathbf{C}_i = \begin{bmatrix} \mathbf{A}_i^T\mathbf{A}_i & -\mathbf{A}_i^T\mathbf{b}_i \\ -\mathbf{b}_i^T\mathbf{A}_i & \|\mathbf{b}_i\|^2 \end{bmatrix}, \tag{3.5}$$

(3.3) can be recast as

$$\begin{aligned} \min_{\mathbf{X} \succeq \mathbf{0}} \quad & \sum_{i=1}^{N} \frac{\mathrm{tr}(\mathbf{C}_i\mathbf{X})}{\mathrm{tr}(\mathbf{X})} \\ \text{s.t.} \quad & \mathrm{rank}(\mathbf{X}) = 1. \end{aligned} \tag{3.6}$$

Relaxing the rank constraint in (3.6) turns it into the following aggregate linear-fractional program

$$\min_{\mathbf{X}\succeq\mathbf{0}} \quad \sum_{i=1}^{N} \frac{\mathsf{tr}(\mathbf{C}_i\mathbf{X})}{\mathsf{tr}(\mathbf{X})}. \tag{3.7}$$

Both numerator and denominator of the summands in the objective function of (3.7) are linear functions of the matrix variable $\mathbf{X}$. Therefore, (3.7) can be converted to a parametric semidefinite program whose objective is in the subtractive form as per the following proposition.

Proposition 1. Let $\mathbf{X}^*$ denote the optimal solution to (3.7). Then, there exists a vector $\boldsymbol{\beta}^* = [\beta_1^*, \ldots, \beta_N^*]$ such that $\mathbf{X}^*$ is also the optimal solution of the following semidefinite program

$$\mathbf{X}^* = \arg\min_{\mathbf{X}\succeq\mathbf{0}} \sum_{i=1}^{N} \mathsf{tr}(\mathbf{C}_i\mathbf{X}) - \beta_i^*\mathsf{tr}(\mathbf{X}). \tag{3.8}$$

In addition, $\mathbf{X}^*$ also satisfies the following system of equations:

$$\mathsf{tr}(\mathbf{C}_i\mathbf{X}^*) - \beta_i^*\mathsf{tr}(\mathbf{X}^*) = 0, \; i = 1, 2, \ldots, N. \tag{3.9}$$

Proof. The Karush-Kuhn-Tucker (KKT) conditions of optimality [81] for problem (3.8) give the same solution set as the KKT conditions for the epigraph form of (3.7). Since the KKT conditions for both problems are sufficient for optimality, the two problems are equivalent. The system of equation (3.9) is due to the KKT conditions. □

In the next subsection, we describe a consensus-based reformulation of (3.8), which allows the application of the ADMM to solve (3.8) for any given $\boldsymbol{\beta}^*$.

Building Consensus

In order to tackle (3.8) in a distributed fashion, we introduce $\mathcal{P} := \{\mathbf{X}_i\}_{i=1}^N$ representing the local copies of $\mathbf{X}$ at the agents. Therefore, we rewrite (3.8) in the following equivalent form

$$\begin{aligned} \min_{\{\mathbf{X}_i\succeq\mathbf{0}\}} \quad & \sum_{i=1}^{N} \mathsf{tr}(\mathbf{C}_i\mathbf{X}_i) - \beta_i^*\mathsf{tr}(\mathbf{X}_i) \\ \text{s.t.} \quad & \mathbf{X}_i = \mathbf{X}_j, \quad j \in \mathcal{V}_i, \quad i \in \mathcal{V}. \end{aligned} \tag{3.10}$$

The equality constraints enforce consensus over $\mathbf{X}_i$, $i = 1, \ldots, N$, across each agent's neighborhood $\mathcal{V}_i$.

To solve (3.10) in a distributed fashion, we employ the ADMM [7]. Hence, we introduce the auxiliary local variables $\mathcal{A} := \{\mathbf{Z}_i^j\}_{j\in\mathcal{V}_i}$ and rewrite (3.10) as

$$\begin{aligned} \min_{\{\mathbf{X}_i\succeq\mathbf{0}\}} \quad & \sum_{i=1}^{N} \mathsf{tr}(\mathbf{C}_i\mathbf{X}_i) - \beta_i^*\mathsf{tr}(\mathbf{X}_i) \\ \text{s.t.} \quad & \mathbf{X}_i = \mathbf{Z}_i^j, \mathbf{X}_j = \mathbf{Z}_i^j, \quad j\in\mathcal{V}_i, \quad i\in\mathcal{V}. \end{aligned} \tag{3.11}$$

Using the auxiliary variables $\mathcal{A}$, we obtain an equivalent alternative representation of the constraints in (3.10). These variables are only used to derive the local recursions and are eventually eliminated. By associating the Lagrange multipliers $\mathcal{D} := \{\{\mathbf{\Gamma}_i^j\}_{j\in\mathcal{V}_i}, \{\mathbf{\Lambda}_i^j\}_{j\in\mathcal{V}_i}\}_{i=1}^N$ with the constraints in (3.11), we get the following augmented Lagrangian function:

$$\begin{aligned} \mathcal{L}_\rho(\mathcal{P},\mathcal{A},\mathcal{D}) &= \sum_{i=1}^{N} \mathsf{tr}\left(\mathbf{C}_i\mathbf{X}_i\right) - \beta_i^*\mathsf{tr}(\mathbf{X}_i) \\ &+ \sum_{i=1}^{N}\sum_{j\in\mathcal{V}_i} \mathsf{tr}\left(\left(\mathbf{\Lambda}_i^j\right)^{\mathrm{T}}\left(\mathbf{X}_i - \mathbf{Z}_i^j\right) + \left(\mathbf{\Gamma}_i^j\right)^{\mathrm{T}}\left(\mathbf{X}_j - \mathbf{Z}_i^j\right)\right) \\ &+ \frac{\rho}{2}\sum_{i=1}^{N}\sum_{j\in\mathcal{V}_i}\left(\left\|\mathbf{X}_i - \mathbf{Z}_i^j\right\|_{\mathrm{F}}^2 + \left\|\mathbf{X}_j - \mathbf{Z}_i^j\right\|_{\mathrm{F}}^2\right), \end{aligned} \tag{3.12}$$

where the constant $\rho > 0$ is a penalty parameter.

Obtaining the solution through the ADMM entails an iterative process consisting of the following steps at each iteration: 1) $\mathcal{L}_\rho$ is minimized with respect to $\mathcal{P}$; 2) $\mathcal{L}_\rho$ is minimized with respect to $\mathcal{A}$; and, 3) the Lagrange multipliers $\mathcal{D}$ are updated through gradient-ascent [7].

Thanks to the reformulation of (3.8) as (3.11), the Lagrangian function (3.12) can be decoupled with respect to variables in $\mathcal{P}$ and $\mathcal{A}$ as well as across the network agents $\mathcal{V}$. It can be shown that, in the ADMM steps, the auxiliary variables $\mathcal{A}$ and the Lagrange multipliers $\{\mathbf{\Gamma}_i^j\}_{j\in\mathcal{V}_i}$ are eliminated. Hence, we end up with the following iterative updates at the ith agent

$$\mathbf{X}_i^{(k+1)} = \arg\min_{\mathbf{X}_i\succeq\mathbf{0}} \mathcal{L}_\rho(\mathbf{X}_i, \mathbf{\Lambda}_i^{(k)}) \tag{3.13}$$

$$\mathbf{\Lambda}_i^{(k+1)} = \mathbf{\Lambda}_i^{(k)} + \rho\sum_{j\in\mathcal{V}_i}[\mathbf{X}_i^{(k+1)} - \mathbf{X}_j^{(k+1)}], \tag{3.14}$$

where $\mathbf{\Lambda}_i^{(k)} = 2\sum_{j\in\mathcal{V}_i}(\mathbf{\Lambda}_i^j)^{(k)}$ and superscript (k) denotes the iteration index.

The constrained minimization problem in (3.13) can be expressed as the following semidefinite least-squares problem

$$\min_{\mathbf{X}_i \succeq \mathbf{0}} \mathrm{tr}[\mathbf{X}_i^{\mathrm{T}}(\mathbf{X}_i - 2\mathbf{G}_i^{(k)})], \tag{3.15}$$

where

$$\mathbf{G}_i^{(k)} = \frac{1}{2\rho|\mathcal{V}_i|}\Big(\rho|\mathcal{V}_i|\mathbf{X}_i^{(k)} + \rho\sum_{j\in\mathcal{V}_i}\mathbf{X}_j^{(k)} - \mathbf{C}_i + \beta_i\mathbf{I}_{P+1} - \mathbf{\Lambda}_i^{(k)}\Big). \tag{3.16}$$

The solution of (3.15) is given by

$$\mathbf{X}_i^{(k+1)} = \mathbf{U}^{(k)}\max\left(\mathbf{\Sigma}^{(k)}, \mathbf{0}\right)(\mathbf{U}^{(k)})^{\mathrm{T}}, \tag{3.17}$$

where $\mathbf{U}^{(k)}$ and $\mathbf{\Sigma}^{(k)}$ are the orthogonal and diagonal matrices coming from the eigen-decomposition (EVD) $\mathbf{G}^{(k)} = \mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}(\mathbf{U}^{(k)})^{\mathrm{T}}$ and $\max(\mathbf{\Sigma}^{(k)}, \mathbf{0})$ denotes the diagonal matrix whose entries are the maxima of the diagonal entries of $\mathbf{\Sigma}^{(k)}$, i.e., the eigenvalues of $\mathbf{G}_i^{(k)}$, and zero. Note that the most computationally intensive operation is the EVD.

Algorithm

The DA-TLS algorithm consists of two loops. In the inner loop, the solution of (3.8) is obtained using the ADMM for a given $\boldsymbol{\beta}^*$. In the outer loop, we use a single iteration of the Newton's method [82] to find the solution of (3.9), i.e.,

$$\beta_i^{(l+1)} = \beta_i^{(l)} - [\mathrm{tr}(\mathbf{X})]^{-1}[\beta_i^{(l)}\mathrm{tr}(\mathbf{X}) - \mathrm{tr}(\mathbf{C}_i\mathbf{X})].$$

The proposed algorithm is summarized in Algorithm 3.

After estimating $\mathbf{X}_i$, the vector estimate $\mathbf{x}_i$ is found as follows. Let $\breve{\mathbf{X}}_i = \mathbf{X}_i/\omega_i$ where ω_i is the $(P+1),(P+1)$ entry of $\mathbf{X}_i$. Then, $\mathbf{x}_i$ is the eigenvector corresponding to the largest eigenvalue of the $P \times P$ upper-left submatrix of $\breve{\mathbf{X}}_i$.

Using the results in [79, 83], it can be observed that the solution of (3.7) and consequently (3.8) is rank-one. Hence, optimizing with respect to the matrix variable $\mathbf{X}$ and relaxing the rank constraint do not lead to any loss of optimality [79]. Therefore, the solutions to (3.3) and (3.10) coincide.

Convergence Analysis

Convergence of the proposed DA-TLS algorithm to the global centralized solution can be proven by checking that both inner and outer loops converge. The convergence of the inner loop can be verified following [84, Proposition 3], i.e., for all $i \in \mathcal{V}$, the iterates $\{\mathbf{X}_i^{(k)}\}$, $\{\mathbf{\Lambda}_i^{(k)}\}$ produced by (3.13) and (3.14) are convergent

Algorithm 3 DA-TLS

```
All agents $i \in \mathcal{V}$ initialize $\beta_1^{(1)} = \text{tr}(\mathbf{C}_i)/(P+1)$ and locally run
for $l = 1, 2, \dots$ do
    Initialize $\mathbf{X}_i^{(0)} = \mathbf{0}_{(P+1)\times(P+1)}$ and $\mathbf{\Lambda}_i^{(0)} = \mathbf{0}_{(P+1)\times(P+1)}$
    for $k = 1, 2, \dots$ do
        Receive $\mathbf{X}_i^{(k)}$ from neighbors in $\mathcal{V}_i$
        Update $\mathbf{\Lambda}_i^{(k+1)}$ as in (3.14)
        Compute $\mathbf{G}_i^{(k)}$ as in (3.16)
        Compute EVD of $\mathbf{G}_i^{(k)} = \mathbf{U}^{(k)}\mathbf{\Sigma}^{(k)}(\mathbf{U}^{(k)})^{\mathrm{T}}$
        Update $\mathbf{X}_i^{(k+1)} = \mathbf{U}^{(k)} \max\left(\mathbf{\Sigma}^{(k)}, \mathbf{0}\right)(\mathbf{U}^{(k)})^{\mathrm{T}}$
    end for
    Update $\beta_i^{(l+1)} = \frac{\text{tr}\left(\mathbf{C}_i\mathbf{X}_i^{(k+1)}\right)}{\text{tr}\left(\mathbf{X}_i^{(k+1)}\right)}$
end for
```

and $\mathbf{X}_i^{(k)} \to \mathbf{X}^*$ as $k \to \infty$. Moreover, the convergence of the outer loop follows setting $\bar{\mathbf{C}} = \sum_{i=1}^{N} \mathbf{C}_i$ and $\bar{\beta}^* = \sum_{i=1}^{N} \beta_i^*$ and observing that the optimization in (3.8) is equivalent to

$$\min_{\mathbf{X}\succeq\mathbf{0}} \text{tr}(\bar{\mathbf{C}}\mathbf{X}) - \bar{\beta}^*\text{tr}(\mathbf{X}). \tag{3.18}$$

The objective function in (3.18) is the sum of two trace functions that are linear, so the objective itself is linear and (3.18) is a standard semidefinite program with a unique solution in the variable $\bar{\beta}^*$. Furthermore, since linear functions are both convex and concave, the requirements for the algorithm in [78] for fractional programming are satisfied and, therefore, convergence of the DA-TLS algorithm is guaranteed.

3.1.5 Simulations

The simulated network is connected with a random topology and consists of $N = 20$ agents where each agent is linked to three other agents on average. We average results over 100 independent trials. In each trial, the scenario is generated according to the same procedure as described in the simulation sections of [14, 21]. The parameters are also chosen as in [14, 21] in order to ensure a fair comparison with the benchmark algorithms. For each agent $i \in \mathcal{V}$, we create a $2P \times P$ local observation matrix $\mathbf{A}_i$ whose entries are drawn from a standard normal distribution. The entries of the parameter vector $\mathbf{x}$ are also drawn from a standard normal distribution. The entries of the error matrix $\tilde{\mathbf{A}}$ and error vector $\tilde{\mathbf{b}}$ are i.i.d. zero-mean Gaussian with variance 0.25. To evaluate the performance of the proposed algorithm, we use the normalized error between the centralized TLS solution $\mathbf{x}^c$ as per (3.2) and the local estimates that is defined as

$$\frac{\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{x}^c\|^2}{\|\mathbf{x}^c\|^2}$$

where $\mathbf{x}_i$ denotes the local estimate at agent i.

In Figs. 3.1-3.2, we plot the normalized error versus the total number of iterations, which is given by the product between the number of iterations of the inner and the outer loops. The former is set to 80 for Fig. 3.1 and 40 for Fig. 3.2, while the latter is set to 5 for both the plots.
Fig. 3.1 shows that, for $P = 9$, DA-TLS with $\rho = 2$ and $\rho = 3$ converges significantly faster than the existing approaches, i.e., the distributed TLS (D-TLS) algorithm of [14] and the inverse-power-iteration-based distributed TLS (IPI-D-TLS) algorithm of [21]. Fig. 3.2 shows the superiority of DA-TLS with $\rho = 1$ over IPI-D-TLS with $\mu = 1$ for two different values of P.

3.1.6 Conclusion

In [12], we developed a new distributed algorithm for solving the TLS problem. We recast the original optimization problem into an equivalent linear-fractional program. Then, employing semidefinite relaxation, we transformed the resultant problem into a parametric semidefinite program whose structure is suitable for distributed treatment via ADMM. Simulation results showed that the proposed algorithm converges faster than the existing alternative algorithms while being less sensitive to tuning of the parameters involved in the algorithm.

3.2 Distributed Learning with Non-Smooth Objective Functions

In [13], we develop a new distributed algorithm to solve a learning problem with non-smooth objective functions when data are distributed over a multi-agent network. We employ a zeroth-order method to minimize the associated augmented Lagrangian in the primal domain using the alternating direction method of multipliers (ADMM) to develop the proposed algorithm, named distributed zeroth-order based ADMM (D-ZOA). Unlike most existing algorithms for non-smooth optimization, which rely on calculating subgradients or proximal operators, D-ZOA only requires function values to approximate gradients of the objective function. Convergence of D-ZOA to the centralized solution is confirmed via theoretical analysis and simulation results.

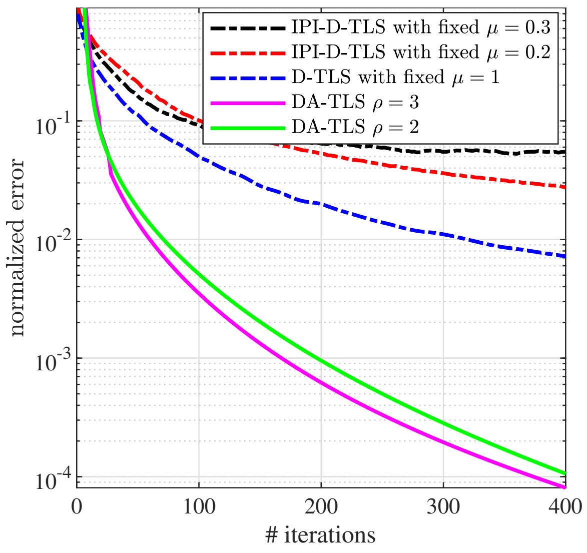


Figure 3.1: Normalized error of the DA-TLS, D-TLS, and IPI-D-TLS algorithms with two values of penalty parameter ($\rho = 2$ and $\rho = 3$) for DA-TLS and two values of the step-size ($\mu = 0.2$ and $\mu = 0.3$) for IPI-D-TLS.

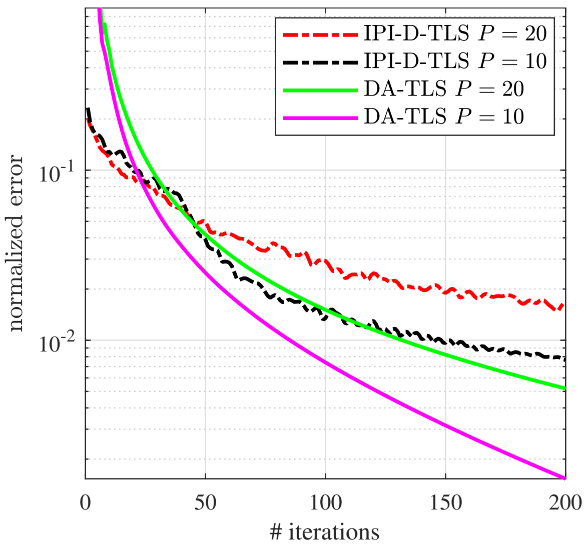


Figure 3.2: Normalized error for different values of P. For DA-TLS, we set $\rho = 1$ and, for IPI-D-TLS, we set $\mu = 1$.

3.2.1 Related Work

There have been several works developing algorithms for solving distributed convex optimization problems over ad-hoc networks. However, many existing algorithms only offer solutions for problems with smooth objective functions, see, e.g., [22–24]. Distributed optimization problems with non-smooth objectives have been considered in [5, 8–11, 85–88]. The approaches taken in [9, 10, 85] are based on subgradient methods. The works of [86, 87] are based on dual decomposition techniques while the algorithms in [5, 11] are developed using soft-thresholding operations. However, all the aforementioned algorithms require either the computation of subgradients, which might be hard to achieve for some objectives, or derivation of proximal operators, which might not be feasible in some scenarios.

Moreover, as remarked in the introductory chapter, there are some real-world problems where obtaining first-order information is impossible due to the lack of any complete objective function, e.g., in bandit optimization [29], in simulation-based optimization [30], or in adversarial black-box machine learning [31]. This motivates the use of zeroth-order methods, which only use the values of the objective functions to approximate their gradients [4, 32, 33]. This motivates the use of zeroth-order methods requiring only function values to approximate gradients.

The works in [8, 88] are based on zeroth-order methods within the distributed optimization setting. While the approach of [88] relies on approximate projections for dealing with constraints, the algorithm ZONE-S proposed in [8] is based on a primal-dual approach and deals with non-convex objectives. However, ZONE-S addresses only consensus problems with a non-smooth regularization that is handled by a central collector making the algorithm not fully distributed.

3.2.2 Contributions

In [13], we develop a fully-distributed algorithm to solve an optimization problem with a non-smooth convex objective function over an ad-hoc network. We utilize the alternating direction method of multipliers (ADMM) for distributed optimization. Furthermore, we employ the zeroth-order method called the two-point stochastic gradient algorithm [3] that is suitable for non-smooth objectives to obtain an approximate minimizer of the augmented Lagrangian in the ADMM's primal update step. The proposed algorithm, called distributed zeroth-order based ADMM (D-ZOA), is fully distributed in the sense that each agent in the network communicates only with its neighbors and no central coordinator is necessary. Furthermore, D-ZOA does not compute any subgradient and only requires the objective function values to approximate the gradient of the augmented Lagrangian. The simulations show that D-ZOA is competitive even on a problem that can be easily solved with a subgradient-based algorithm. Furthermore, the experiments show the usefulness of D-ZOA on a problem where calculating any subgradient is impractical. Convergence of D-ZOA to the centralized solution at all agents is verified through theoretical analysis and simulation results.

3.2.3 Non-Smooth Distributed Learning

We consider the system model described in Section 2.1 with horizontal partitioning of data as presented in Section 2.1.1. Accordingly, we consider the minimization problem in (2.1) where the local cost functions f_i for $i \in \mathcal{V}$ are closed and convex but *non-smooth*.

We first present the proposed D-ZOA algorithm that consists of two nested loops: an ADMM-based outer loop and the zeroth-order two-point stochastic gradient algorithm that constitutes the inner loop and solves the minimization problem in the ADMM primal update step. Next, we discuss the related computational complexity. Finally, we establish the convergence of D-ZOA theoretically.

Algorithm

To solve (2.1) in a fully-distributed fashion, we employ the distributed ADMM algorithm presented in Section 2.2.2. Since the objective function in (2.1) is non-smooth, the objective function of the minimization problem in the ADMM primal update step (2.9) is also non-smooth, which makes it hard to obtain a solution using first-order information. To solve this problem, we employ the zeroth-order two-point stochastic gradient algorithm described in Section 2.3.1. Therefore, the function $\mathcal{F}_i(\cdot)$ in (2.17) is defined as

$$\mathcal{F}_i(\mathbf{x}_i) := f_i(\mathbf{x}_i; \mathcal{X}_i) + h_i(\mathbf{x}_i) \tag{3.19}$$

where

$$h_i(\mathbf{x}_i) := (\boldsymbol{\lambda}_i^{(k-1)})^T \mathbf{x}_i + \rho \sum_{j \in \mathcal{V}_i} \left\| \mathbf{x}_i - \frac{\mathbf{x}_i^{(k-1)} + \mathbf{x}_j^{(k-1)}}{2} \right\|^2 . \tag{3.20}$$

In order to solve the minimization problem in (2.9) utilizing a zeroth-order method, we assume that $\mathcal{F}_i(\cdot)$, for all $i \in \mathcal{V}$, is Lipschitz-continuous with the Lipschitz constant L. This assumption is common for zeroth-order optimization, see, e.g., [3, 8]. The resulting algorithm to solve (2.1) when the objective is non-smooth or first-order information is unavailable consists of two nested loops: an outer loop that is based on the ADMM and an inner loop that is based on the two-point stochastic gradient algorithm. Note that no communication among agents is involved throughout the inner loop. The proposed algorithm, D-ZOA, is summar-

Algorithm 4 D-ZOA

```
At all agents $i \in \mathcal{V}$, initialize $\mathbf{x}_i^{(0)} = \mathbf{0}$, $\boldsymbol{\lambda}_i^{(0)} = \mathbf{0}$, and locally run
for $k = 1, 2, \ldots, K$ do
    Share $\mathbf{x}_i^{(k-1)}$ with the neighbors in $\mathcal{V}_i$
    Update $\boldsymbol{\lambda}_i^{(k)}$ as in (2.10)
    Initialize $\mathbf{y}_i^{(0)} = \mathbf{0}$
    for $t = 1, 2, \ldots, T$ do
        Draw independent $\mathbf{n}_1, \mathbf{n}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_P)$
        Set $u_{1,t} = u_{1,1}/t$, $u_{2,t} = u_{1,1}/t^2$ and compute $\mathbf{g}_i^{(t)}$ as in (2.19)
        Update $\mathbf{y}_i^{(t)}$ as in (2.20)
    end for
    Update $\mathbf{x}_i^{(k)} = \mathbf{y}_i^{(T)}$
end for
```

ized in Algorithm 4 where K and T denote the number of iterations of the ADMM and the two-point stochastic gradient, respectively.

Computational Complexity

Solving D-ZOA's inner loop, i.e., the minimization in (2.9), requires multiple evaluations of the function $\mathcal{F}_i(\cdot)$. The computational requirement at each agent and each ADMM outer loop iteration depends on the local objective function f_i. Let us indicate the number of computations required by D-ZOA to carry out one iteration of the inner loop at agent i and the number of iterations of the inner loop by m_i and T, respectively. Hence, the total number of computations required by D-ZOA at agent i and each ADMM outer loop iteration is $\mathcal{O}(Tm_i)$. However, the cost of transmission/communication among the neighboring agents does not depend on T or m_i since the inner loop does not require any communication among agents.

Convergence Analysis

The convergence of D-ZOA to a near-optimal solution is established by corroborating that both inner and outer loops of the algorithm converge.

The convergence of the inner loop can be verified following [3, Theorem 2], i.e., it can be shown that, if $\mathcal{F}_i(\cdot)$ is Lipschitz-continuous with the Lipschitz constant L, there exists a constant c such that, for each T representing a fixed number of inner-loop iterations, the following inequality holds:

$$\mathbb{E}[\mathcal{F}_i(\hat{\mathbf{y}}_i^{(T)}) - \mathcal{F}_i(\mathbf{x}_i^*)] \leq c\frac{RL\sqrt{P}}{\sqrt{T}}\Big(\max\{\alpha_0, \alpha_0^{-1}\}\sqrt{\log(2P)} + \frac{u_1\log(2T)}{\sqrt{T}}\Big) \tag{3.21}$$

where $\mathbf{x}_i^*$ is the minimizer to (2.17) and

$$\hat{\mathbf{y}}_i^{(T)} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{y}_i^{(t)}.$$

The inequality in (3.21) implies that the two-point stochastic gradient algorithm constituting the inner loop converges at a rate of $\mathcal{O}(\sqrt{P/T})$. In [3], it is shown that $c = 0.5$ is suitable when $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ are sampled from a normal distribution. The function $\mathcal{F}_i(\cdot)$ is the sum of $f_i(\cdot)$, which is assumed to be closed and convex, and $h_i(\cdot)$ that is also both closed and convex since it is a positive definite quadratic function. Hence, the function $\mathcal{F}_i(\cdot)$ is closed and convex in addition to being Lipschitz-continuous [81]. Therefore the convergence result in (3.21) follow from [3].

The convergence of the outer loop can be proven by verifying the convergence of a fully-distributed ADMM with perturbed primal updates. Therefore, $\mathbf{x}_i^{(k)}$ can be written as

$$\mathbf{x}_i^{(k)} = \breve{\mathbf{x}}_i^{(k)} + \boldsymbol{\xi}_i^{(k)} \tag{3.22}$$

where $\breve{\mathbf{x}}_i^{(k)} \in \mathbb{R}^P$ is the exact ADMM primal update and $\boldsymbol{\xi}_i^{(k)} \in \mathbb{R}^P$ is a random variable representing the perturbation of $\breve{\mathbf{x}}_i^{(k)}$. Similar to [58], we assume the perturbation to have zero expectation, i.e., $\mathbb{E}[\boldsymbol{\xi}_i^{(k)}] = \mathbf{0}$, $\forall i \in \mathcal{V}$ and for all the ADMM iterations k, and have finite covariance matrix, i.e., $\text{cov}[\boldsymbol{\xi}_i^{(k)}]_{i,j} < \infty$, $\forall i \in \mathcal{V}$, $\forall i,j = 1,\dots,P$ and for all the ADMM iterations k. To present the convergence result, we rewrite (2.4) in the matrix form. By defining $\mathbf{w} \in \mathbb{R}^{NP}$ concatenating all $\mathbf{x}_i$ and $\mathbf{z} \in \mathbb{R}^{2EP}$ concatenating all $\mathbf{z}_i^j$, (2.4) can be written as

$$\begin{aligned} \min_{\mathbf{w},\mathbf{z}} \quad & f(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{C}\mathbf{w} + \mathbf{D}\mathbf{z} = \mathbf{0} \end{aligned} \tag{3.23}$$

where

$$f(\mathbf{w}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i),$$

$\mathbf{C} = [\mathbf{C}_1^\mathsf{T}, \mathbf{C}_2^\mathsf{T}]^\mathsf{T}$, and $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{R}^{2EP\times NP}$ are both composed of $2E \times N$ blocks of $P \times P$ matrices. If $(i,j) \in \mathcal{E}$ and $\mathbf{z}_i^j$ is the qth block of $\mathbf{z}$, then the (q,i)th block of $\mathbf{C}_1$ and the (q, j)th block of $\mathbf{C}_2$ are the identity matrix $\mathbf{I}_P$. Otherwise, the corresponding blocks are $\mathbf{0}_{P\times P}$. Furthermore, we have

$$\mathbf{D} = [-\mathbf{I}_{2EP}, -\mathbf{I}_{2EP}]^{\mathsf{T}}.$$

To facilitate the representation, we also define the following matrices

$$\begin{aligned}
\mathbf{M}_+ &= \mathbf{C}_1^{\mathsf{T}} + \mathbf{C}_2^{\mathsf{T}}\\
\mathbf{M}_- &= \mathbf{C}_1^{\mathsf{T}} - \mathbf{C}_2^{\mathsf{T}}\\
\mathbf{L}_+ &= 0.5\mathbf{M}_+\mathbf{M}_+^{\mathsf{T}}\\
\mathbf{L}_- &= 0.5\mathbf{M}_-\mathbf{M}_-^{\mathsf{T}}\\
\mathbf{H} &= 0.5(\mathbf{L}_+ + \mathbf{L}_-)\\
\mathbf{Q} &= \sqrt{0.5\mathbf{L}_-}.
\end{aligned}$$

We construct the auxiliary sequence

$$\mathbf{r}^{(k)} = \sum_{s=0}^{k} \mathbf{Q}\mathbf{w}^{(s)}$$

and define the auxiliary vector $\mathbf{q}^{(k)}$ and the auxiliary matrix $\mathbf{G}$ as

$$\mathbf{q}^{(k)} = \begin{bmatrix} \mathbf{r}^{(k)} \\ \mathbf{w}^{(k)} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho\mathbf{I}_P & \mathbf{0}_{P\times P} \\ \mathbf{0}_{P\times P} & \rho\frac{\mathbf{L}_+}{2} \end{bmatrix}. \tag{3.24}$$

The convergence results of [63], [58], and [64] can now be adapted to D-ZOA as per the following theorem that also provides an explicit privacy-accuracy trade-off.

Theorem 3.1. For any $K > 0$, we have

$$\mathbb{E}[f(\hat{\mathbf{w}}^{(K)}) - f(\mathbf{w}^*)] \le \frac{\left\|\mathbf{q}^{(0)} - \mathbf{q}\right\|_{\mathbf{G}}^2}{K} + \frac{\rho\lambda_{\text{max}}^2(\mathbf{L}_+)\sum_{k=0}^{K-1} \text{tr}\left(\text{cov}[\boldsymbol{\xi}^{(k)}]\right)}{2K\lambda_{\text{min}}(\mathbf{L}_-)} \tag{3.25}$$

where $\mathbf{q} = [\mathbf{r}^{\mathsf{T}}, (\mathbf{w}^*)^{\mathsf{T}}]^{\mathsf{T}}$, $\mathbf{w}^*$ is the optimal solution of (5.52), $\boldsymbol{\xi}^{(k)} \in \mathbb{R}^{NP}$ is the vector concatenating all $\boldsymbol{\xi}_i^{(k)} \in \mathbb{R}^P$, and

$$\hat{\mathbf{w}}^{(K)} = \frac{1}{K}\sum_{k=1}^{K} \breve{\mathbf{w}}^{(k)}.$$

Proof. Since $\mathbb{E}[\boldsymbol{\xi}_i^{(k)}] = \mathbf{0}$ and $\text{cov}[\boldsymbol{\xi}_i^{(k)}]_{i,j} < \infty$, $\forall i \in \mathcal{V}$, $\forall i, j = 1, \ldots, P$ and for all the ADMM iterations k, proof follows from [2, Theorem 3]. $\square$

3.2.4 Simulations

The D-ZOA algorithm is tested on a multi-agent network with a random topology, where each agent is linked to three other agents on average. To benchmark D-ZOA with existing solutions, we consider a distributed version of the generalized lasso [5] that can be solved with subgradient methods [9]. Furthermore, we consider a distributed version of the reduced-rank regression (RRR) problem where the objective function is least squares with nuclear norm regularization [7]. Nuclear norm is a non-smooth function that is used as a convex surrogate for the rank. Calculating any subgradient of the nuclear norm function is impractical. RRR has applications in robust PCA [26], low-rank matrix decomposition [27], matrix completion [28], etc.

The network-wide observations are represented as an observation matrix $\mathbf{A} \in \mathbb{R}^{M \times P}$ and a response matrix $\mathbf{B} \in \mathbb{R}^{M \times S}$, where M is the number of data samples and P is the number of features in each sample. The matrix $\mathbf{A}$ consists of N submatrices $\mathbf{A}_i$, i.e.,

$$\mathbf{A} = [\mathbf{A}_1^\mathsf{T}, \mathbf{A}_2^\mathsf{T}, \ldots, \mathbf{A}_N^\mathsf{T}]^\mathsf{T},$$

and the matrix $\mathbf{B}$ of N submatrices $\mathbf{B}_i$, i.e., $\mathbf{B} = \left[\mathbf{B}_1^\mathsf{T}, \mathbf{B}_2^\mathsf{T}, \ldots, \mathbf{B}_N^\mathsf{T}\right]^\mathsf{T}$, as the data are distributed among the agents and each agent i holds its respective $\mathbf{A}_i \in \mathbb{R}^{M_i \times P}$ and $\mathbf{B}_i \in \mathbb{R}^{M_i \times S}$ where $\sum_{i=1}^{N} M_i = M$. The parameter matrix that establishes a linear regression between $\mathbf{A}$ and $\mathbf{B}$ is $\mathbf{X} \in \mathbb{R}^{P \times S}$. In the generalized lasso, $S = 1$ and, hence, $\mathbf{B}$ is the vector $\mathbf{b} \in \mathbb{R}^M$ and $\mathbf{X}$ becomes $\mathbf{x} \in \mathbb{R}^P$ as in Section 2.1.1. In the centralized approach, a generalized lasso estimate of $\mathbf{x}$ is given by

$$\mathbf{x}^c = \arg\min_{\mathbf{x}} \{ \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \eta \, \|\mathbf{F}\mathbf{x}\|_1 \} \tag{3.26}$$

where $\eta > 0$ is a regularization parameter and $\mathbf{F}$ is an arbitrary matrix. An RRR estimate of $\mathbf{X}$ is also given by

$$\mathbf{X}^c = \arg\min_{\mathbf{X}} \{ \|\mathbf{A}\mathbf{X} - \mathbf{B}\|^2 + \eta_* \, \|\mathbf{X}\|_* \} \tag{3.27}$$

where $\eta_* > 0$ is a rank-controlling parameter. In the distributed setting, we solve problem (2.3) with

$$f_i(\mathbf{x}_i; \mathbf{A}_i, \mathbf{b}_i) = \|\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i\|^2 + \frac{\eta}{N} \|\mathbf{F}\mathbf{x}_i\|_1 \tag{3.28}$$

for the generalized lasso case and with

$$f_i(\mathbf{X}_i; \mathbf{A}_i, \mathbf{B}_i) = \|\mathbf{A}_i \mathbf{X}_i - \mathbf{B}_i\|^2 + \frac{\eta}{N} \|\mathbf{X}_i\|_* \tag{3.29}$$

for the RRR case. For each agent $i \in \mathcal{V}$, we create a $10P \times P$ local observation matrix $\mathbf{A}_i$ whose entries are independent identically distributed zero-mean unit-variance Gaussian random variables. The response vector $\mathbf{b}$ is obtained as

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\phi}$$

where $\boldsymbol{\omega} \in \mathbb{R}^P$ and $\boldsymbol{\phi} \in \mathbb{R}^M$ are chosen as random vector with distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$. The response matrix $\mathbf{H}$ is obtained as

$$\mathbf{B} = \mathbf{A}\boldsymbol{\Omega} + \boldsymbol{\Phi}$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{P\times S}$ and $\boldsymbol{\Phi} \in \mathbb{R}^{M\times S}$ are random matrices with matrix normal distributions $\mathcal{MN}(\mathbf{0}_{P\times S}, \mathbf{I}_P, \mathbf{I}_S)$ and $\mathcal{MN}(\mathbf{0}_{M\times S}, 0.1\mathbf{I}_M, 0.1\mathbf{I}_S)$, respectively. The regularization parameters η and η_* are set to

$$\begin{aligned}\eta &= 0.01 \left\|\mathbf{D}^{\mathsf{T}}\mathbf{b}\right\|_\infty \\ \eta_* &= 0.01 \left\|(\mathbf{I}_S \otimes \mathbf{D})^{\mathsf{T}}\text{vec}(\mathbf{H})\right\|_\infty\end{aligned}$$

as in [5]. The number of iterations of the ADMM outer loop is set to 200. For the inner loop, the number of iterations is set to 1000, the smoothing constant $u_{1,1}$ is set to 1 and the convergence in mean is achieved by averaging the outputs of 10 inner loops. Performance of D-ZOA is evaluated using the normalized error between the centralized solutions $\mathbf{x}^c$ as per (4.19) or $\mathbf{X}^c$ as per (3.27) and the local estimates. It is defined as

$$\frac{\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{x}^c\|^2}{\|\mathbf{x}^c\|^2}$$

for generalized lasso and as

$$\frac{\sum_{i=1}^{N} \|\mathbf{X}_i - \mathbf{X}^c\|_F^2}{\|\mathbf{X}^c\|_F^2}$$

for RRR, where $\mathbf{x}_i$ and $\mathbf{X}_i$ denote the local estimates at agent i. The centralized solutions $\mathbf{x}^c$ and $\mathbf{X}^c$ are computed using the convex optimization toolbox CVX [89]. Results are obtained by averaging over 100 independent trials.

Figs. 3.3-3.4 show the performance of D-ZOA for the generalized lasso and the RRR scenarios, respectively. In Fig. 3.3, we plot the normalized error versus the outer loop iteration index for D-ZOA and a subgradient-based distributed algorithm, called D-SG and proposed in [9]. We observe that, for $P = 10$ and $\rho = 3$, D-ZOA has similar performance to D-SG both when the network consists of 15 and 30 agents. Fig. 3.4 shows that D-ZOA converges to the centralized

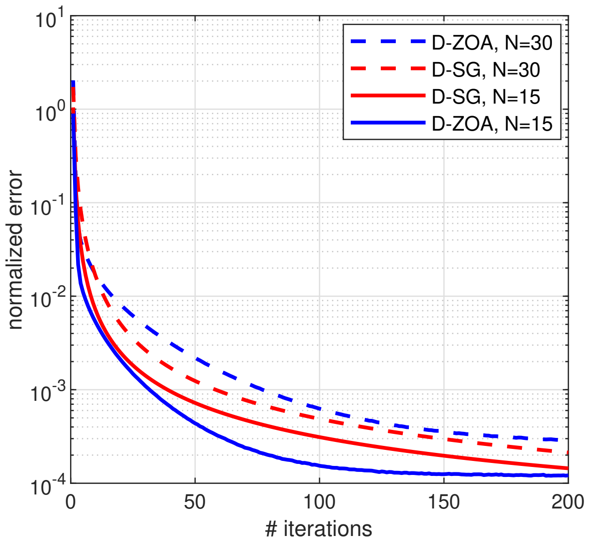


Figure 3.3: Normalized error of D-ZOA and D-SG for generalized lasso with $P = 10$, $\rho = 3$ and two different values of N.

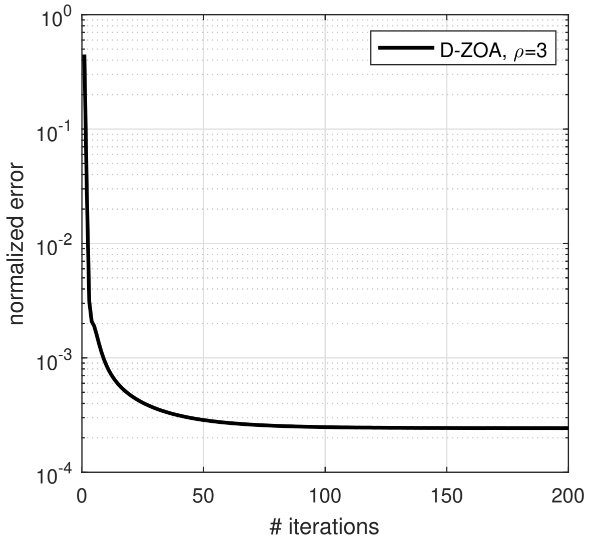


Figure 3.4: Normalized error of D-ZOA for RRR with $P = 5$, $S = 4$, $\rho = 3$ and $N = 10$.

solution of the considered RRR problem for $P = 5$, $S = 4$, $N = 10$ and $\rho = 3$. The parameters related to the inner loop have been chosen to get the convergence according to [3].The other parameters have been tuned in order to get the best performance compared to the benchmark algorithms while benefiting from the convergence properties of the ADMM illustrated in [5].

3.2.5 Conclusion

In [13], we developed a new consensus-based algorithm for solving a distributed optimization problem with a non-smooth convex objective. We recast the original problem into an equivalent constrained optimization problem whose structure is suitable for distributed implementation via ADMM. We employed a zeroth-order method, known as the two-point stochastic gradient algorithm, to minimize the augmented Lagrangian in the primal update step. Compared to existing algorithms for non-smooth optimization, D-ZOA is fully-distributed and does not require the computation of subgradients, nor proximal operators which may be difficult to derive in some scenarios. D-ZOA only requires the computation of objective function values. The convergence of D-ZOA to the centralized solution was verified through theoretical analysis and simulations.

Chapter 4

Privacy-Preserved Distributed Learning with Zeroth-Order Optimization

In this chapter, we present our contribution developed in [2] where we propose a privacy-preserving distributed algorithm to minimize a regularized empirical risk function when the first-order information is not available and data is distributed over a multi-agent network. We employ a zeroth-order method to minimize the associated augmented Lagrangian function in the primal domain using the alternating direction method of multipliers (ADMM). We show that the D-ZOA algorithm presented in Section 3.2 has intrinsic privacy-preserving properties. Most existing privacy-preserving distributed optimization/estimation algorithms exploit some perturbation mechanism to preserve privacy, which comes at the cost of reduced accuracy. Contrarily, by analyzing the inherent randomness due to the use of a zeroth-order method, we show that D-ZOA is intrinsically endowed with $(\epsilon, \delta)-$differential privacy. In addition, we employ the moments accountant method to show that the total privacy leakage of D-ZOA grows sublinearly with the number of ADMM iterations. D-ZOA outperforms the existing differentially-private approaches in terms of accuracy while yielding similar privacy guarantee. We prove that D-ZOA reaches a neighborhood of the optimal solution whose size depends on the privacy parameter. The convergence analysis also reveals a practically important trade-off between privacy and accuracy. Simulation results verify the desirable privacy-preserving properties of D-ZOA and its superiority over the state-of-the-art algorithms as well as its network-wide convergence.

4.1 Related Work

There have been several works developing privacy-preserving algorithms for distributed convex optimization [40, 41, 56–61, 90–92]. The work in [41] proposes two differentially private distributed algorithms that are based on the alternating direction method of multipliers (ADMM). The algorithms in [41] are obtained by perturbing the dual and the primal variable, respectively. However, in both algorithms, the privacy leakage of an agent is bounded only at a single iteration and an adversary might exploit knowledge available from all iterations to infer sensitive information. This shortcoming is mitigated in [56–59]. The works in [56, 57] develop ADMM-based differentially private algorithms with improved accuracy. The work in [58] employs the ADMM to develop a distributed algorithm where the primal variable is perturbed by adding a Gaussian noise with diminishing variance to ensure zero-concentrated differential privacy enabling higher accuracy compared to the common (ϵ, δ)-differential privacy. The work in [59] develops a stochastic ADMM-based distributed algorithm that further enhances the accuracy while ensuring differential privacy. The authors of [90–92] propose differentially-private distributed algorithms that utilize the projected-gradient-descent method for handling constraints. The differentially private distributed algorithm proposed in [60] is based on perturbing the local objective functions. However, the algorithms in [41, 56–60, 90–92] offer distributed solutions only for problems with smooth objective functions.

The work in [40] addresses problems with non-smooth objective functions by employing a first-order approximation of the augmented Lagrangian with a scalar l_2-norm proximity operator. However, this algorithm is not fully distributed since it requires a central coordinator to average all the perturbed primal variable updates over the network at every iteration. All the above-mentioned algorithms in [40, 41, 56–60, 90–92] require some modifications through deliberately perturbing either the local estimates or the objective functions. This compromises the performance of the algorithm by degrading its accuracy especially when large amount of noise is required to provide high privacy levels. The work in [61] considers privacy-preserving properties that are intrinsic, i.e., they do not require any change in the algorithm but are associated with the algorithm's inherent properties. However, the approach taken in [61] considers a privacy metric based on the topology of the communication graph. Therefore, none of the existing algorithms are able to offer fully-distributed solutions that are intrinsically capable of ensuring differential privacy.

4.2 Contributions

In [2], we employ the D-ZOA algorithm presented in Section 3.2 to solve a class of regularized empirical risk minimization (ERM) problems when first-order information is unavailable or hard to obtain. Furthermore, we show that D-ZOA has intrinsic privacy-preserving properties. To substantiate this novel finding, we model the primal variable at each agent as the sum of an exact (unperturbed) value and a random perturbation. This enables us to address the challenging problem of approximating the distribution of the primal variable and verify that the stochasticity inherent to the employed zeroth-order method can adequately make D-ZOA differentially private. To this end, we find a suitable approximation for the probability distribution of the primal variable. Subsequently, we show that the inherent randomness in D-ZOA enables it to preserve (ϵ, δ)-differential privacy. Utilizing the moments accountant method [93], we also show that the total privacy leakage over all iterations grows sublinearly with the number of ADMM iterations. This is particularly important as we observe that, with any similar level of privacy, the optimization accuracy of D-ZOA is higher compared to the existing privacy-preserving approaches, which perturb the variables exchanged among the network agents by adding noise.

We prove that D-ZOA reaches a neighborhood of the optimal solution, i.e., a near-optimal solution, and the size of the neighborhood is determined by the privacy parameter. This gives an explicit privacy-accuracy trade-off, where a stronger privacy guarantee corresponds to a lower accuracy. Through numerical simulations, we show that D-ZOA is competitive with the state-of-the-art zeroth-order-based optimization algorithms even though they are designed for centralized processing. We also verify numerically that the entries of the zeroth-order stochastic gradient are normally distributed by illustrating the associated histograms and (quantile-quantile) QQ plots. Simulation results also demonstrate that, with any given level of required privacy guarantee, D-ZOA outperforms existing privacy-preserving algorithms in terms of accuracy. To the best of our knowledge, this is the first work on distributed non-smooth optimization that is capable of exploiting the inherent randomness due to the use of a zeroth-order method and enjoy the ensuing intrinsic (ϵ, δ)-differential privacy.

4.3 System Model

We consider the system model described in Section 2.1 with horizontal partitioning of the observation matrix as presented in Section 2.1.1. In addition to the symbols introduced in Section 2.1.1, each agent $i \in \mathcal{V}$ has a private dataset $\mathcal{S}_i$ that is defined as

$$\mathcal{S}_i = \Big\{(\mathbf{A}_i, \mathbf{b}_i) : \mathbf{A}_i = [\mathbf{a}_{i,1}, \mathbf{a}_{i,2}, \ldots, \mathbf{a}_{i,M_i}]^\mathsf{T} \in \mathbb{R}^{M_i \times P}, \\ \mathbf{b}_i = [b_{i,1}, b_{i,2}, \ldots, b_{i,M_i}]^\mathsf{T} \in \mathbb{R}^{M_i}\Big\}.$$

We consider the problem of estimating a parameter of interest $\mathbf{x} \in \mathbb{R}^P$ that relates the value of an output measurement stored in the response vector $\mathbf{b}_i$ to input measurements collected in the corresponding row of the local matrix $\mathbf{A}_i$. The associated supervised learning problem can be cast as a regularized ERM expressed by

$$\min_{\mathbf{x}} \sum_{i=1}^{N} \frac{1}{M_i} \sum_{j=1}^{M_i} \ell(\mathbf{a}_{i,j}, b_{i,j}; \mathbf{x}) + \eta R(\mathbf{x}) \tag{4.1}$$

where $\ell : \mathbb{R}^P \to \mathbb{R}$ is the loss function, $R : \mathbb{R}^P \to \mathbb{R}$ is the regularizer function, and $\eta > 0$ is the regularization parameter. The ERM problem pertains to several applications in machine learning, e.g., linear regression [11], support vector machine [69], and logistic regression [40, 58]. We assume that the loss function $\ell(\cdot)$ and the regularizer function $R(\cdot)$ are both closed and convex but at least one of them is *non-smooth*. Let us denote the optimal solution of (4.1) by $\mathbf{x}^c$. A distributed solution to the minimization problem in (4.1) is found by employing the D-ZOA algorithm presented in (3.2.3) with

$$f_i(\mathbf{x}_i; \mathcal{X}_i) = \frac{1}{M_i} \sum_{j=1}^{M_i} \ell(\mathbf{a}_{i,j}, b_{i,j}; \mathbf{x}) + \eta R(\mathbf{x}). \tag{4.2}$$

4.4 Intrinsic Differential Privacy Guarantee

In this section, we consider the privacy concerns associated with distributed learning and reveal that the inherent randomness due to the use of a zeroth-order method is sufficient for the proposed D-ZOA algorithm to preserve (ϵ, δ)-differential privacy. First, we propose our solution to the challenging problem of characterizing the randomness inherent to the algorithm. Subsequently, we assess the l_2-norm sensitivity of the primal variable and compute the covariance that the primal variable is required to have so that the privacy leakage of a single iteration of D-ZOA is bounded at each agent. Finally, we prove that the total privacy leakage over all iterations grows sublinearly with the number of ADMM iterations.

4.4.1 Primal Variable Distribution

Due to the stochasticity inherent to the zeroth-order method, its employment for the ADMM primal update produces a perturbed estimate. Therefore, the solution in the primal update step in (2.9) at agent i using D-ZOA can be modeled as

$$\mathbf{x}_i^{(k)} = \breve{\mathbf{x}}_i^{(k)} + \boldsymbol{\xi}_i^{(k)} \tag{4.3}$$

where $\breve{\mathbf{x}}_i^{(k)} \in \mathbb{R}^P$ is the exact ADMM primal update and $\boldsymbol{\xi}_i^{(k)} \in \mathbb{R}^P$ is a random variable representing the perturbation. As in [6], the optimality condition for $\breve{\mathbf{x}}_i^{(k)}$ is given by

$$\mathbf{0} \in \partial f_k(\breve{\mathbf{x}}_i^{(k)}) + \boldsymbol{\lambda}_i^{(k-1)} + 2\rho|\mathcal{V}_i|\breve{\mathbf{x}}_i^{(k)} - \rho|\mathcal{V}_i|\mathbf{x}_i^{(k)} - \rho \sum_{j\in\mathcal{V}_i} \mathbf{x}_j^{(k)}. \tag{4.4}$$

Hence, for any subgradient $f_i'(\breve{\mathbf{x}}_i^{(k)}) \in \partial f_i(\breve{\mathbf{x}}_i^{(k)})$, we have

$$f_k'(\breve{\mathbf{x}}_i^{(k)}) = -\boldsymbol{\lambda}_i^{(k-1)} - 2\rho|\mathcal{V}_i|\breve{\mathbf{x}}_i^{(k)} + \rho|\mathcal{V}_i|\mathbf{x}_i^{(k)} + \rho \sum_{j\in\mathcal{V}_i} \mathbf{x}_j^{(k)}. \tag{4.5}$$

The model (4.3) represents an implicit primal variable perturbation that can be contrasted with the explicit primal variable perturbation used in [41, 58]. Using (4.3) and the primal update equation in (4.5), the ADMM primal update step in (2.9) can be expressed as

$$\breve{\mathbf{x}}_i^{(k)} = -\frac{1}{2\rho|\mathcal{V}_i|} f_k'(\breve{\mathbf{x}}_i^{(k)}) + \frac{1}{2|\mathcal{V}_i|}\Big(|\mathcal{V}_i|\mathbf{x}_i^{(k)} + \sum_{j\in\mathcal{V}_i} \mathbf{x}_j^{(k)}\Big) - \frac{1}{2\rho|\mathcal{V}_i|}\boldsymbol{\lambda}_i^{(k-1)} \tag{4.6}$$

$$\mathbf{x}_i^{(k)} = \breve{\mathbf{x}}_i^{(k)} + \boldsymbol{\xi}_i^{(k)} \tag{4.7}$$

where $\breve{\mathbf{x}}_i^{(k)}$ is the local exact primal update at agent i and iteration k and $\boldsymbol{\xi}_i^{(k)}$ is the local perturbation of $\breve{\mathbf{x}}_i^{(k)}$ at agent i and iteration k.

To prove that the inherent randomness due to employing a zeroth-order method makes D-ZOA differentially private, we need the knowledge of the probability distribution of the primal variable $\mathbf{x}_i^{(k)}$. To approximate the probability distribution, in view of (2.20) and the fact that $\mathbf{x}_i^{(0)} = \mathbf{0}$, we unfold $\mathbf{x}_i^{(k)}$ as $\mathbf{x}_i^{(k)} = -\sum_{t=1}^{T} \alpha_t \mathbf{g}_i^{(t)}$. The stochastic gradient $\mathbf{g}_i^{(t)}$ is the average of J independent random samples $\mathbf{g}_{j,i}^{(t)}$ that are functions of the random values $\{\boldsymbol{\nu}_{1,t}^{j,i}\}_{j=1}^{J}$ and $\{\boldsymbol{\nu}_{2,t}^{j,i}\}_{j=1}^{J}$ drawn from the same normal distribution. Therefore, it is realistic to assume that $\{\mathbf{g}_{j,i}^{(t)}\}_{j=1}^{J}$ are independent and identically normally distributed with a common mean $\boldsymbol{\mu}_i^{(t)}$ and finite covariance matrix $\boldsymbol{\Psi}_i^{(t)}$, i.e., $\mathbf{g}_{j,i}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_i^{(t)}, \Psi_i^{(t)})$. Thus, the probability distribution of $\mathbf{x}_i^{(k)}$ is given by the following lemma.

Lemma 1. *If $\{\mathbf{g}_{j,i}^{(t)}\}_{j=1}^{J}$ are independent and identically distributed (i.i.d.) with $\mathbf{g}_{j,i}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_i^{(t)}, \Psi_i^{(t)})$, then $\mathbf{x}_i^{(k)}$ is distributed as*

$$\mathbf{x}_i^{(k)} \sim \mathcal{N}\Big(\check{\mathbf{x}}_i^{(k)}, \frac{1}{J}\sum_{t=1}^{T} \alpha_t^2 \Psi_i^{(t)}\Big). \tag{4.8}$$

Proof. See proof of [2, Lemma 1].

In the next subsection, we find an explicit expression for the covariance of $\mathbf{x}_i^{(k)}$.

The assumption of normal distribution for $\mathbf{g}_k^{(t)}$ is a natural one as $\mathbf{g}_k^{(t)}$ is the average of stochastic variable vectors $\{\mathbf{g}_{j,k}^{(t)}\}_{j=1}^{J}$, which are themselves functions of normally-distributed random variable vectors $\{\boldsymbol{\nu}_{1,t}^{j,k}\}_{j=1}^{J}$ and $\{\boldsymbol{\nu}_{2,t}^{j,k}\}_{j=1}^{J}$. We provide some numerical experiments to explicitly verify this assumption in Section 4.6.

The assumption is necessary to make the problem of deriving theoretical differential privacy guarantees for the proposed algorithm tractable. Note that our analysis based on this and other assumptions does not result in any deterministic guarantee but yields a probabilistic statement for privacy guarantee by setting a bound on a ratio of probabilities relevant in the concept of differential privacy. Therefore, we do not require perfectly accurate evaluations of the parameters, variables, or statistical models involved in the analysis. Nonetheless, we are cognizant that the reliability of the results highly depends on the accuracy of the underlying assumptions and approximations. Our simulation results in Section 4.6 implicitly corroborate the veracity of our assumptions.

4.4.2 Covariance of the Primal Variable

In this subsection, we derive an explicit expression for the covariance of the primal variable $\mathbf{x}_i^{(k)}$. This is needed to show that the privacy leakage of any iteration of D-ZOA is bounded at all agents.

To make the problem more tractable, we assume that the entries of the random vector $\mathbf{x}_i^{(k)}$ are independent of each other and have the same variance [40, 41, 62]. Let us denote the variance of every entry of $\boldsymbol{\xi}_i^{(k)}$ by σ_i^2. Therefore, in view of Lemma 1, we have

$$\sigma_i^2 = \frac{1}{JP}\sum_{t=1}^{T} \alpha_t^2 \mathrm{tr}\left(\boldsymbol{\Psi}_i^{(t)}\right),$$

which can be computed as per the following lemma.

Lemma 2. *There exists a constant c such that*

$$\sigma_i^2 = \frac{c\alpha_0^2 R^2}{JP\log(2P)}\Big(s_1(1+\log P)+s_2\Big) - \frac{4\left\|\mathbf{x}^c\right\|^2}{TJP}. \tag{4.9}$$

where $s_1 = \sum_{t=1}^{T} t^{-1}$, $s_2 = \sum_{t=1}^{T} t^{-1.5}$, *and* $\mathbf{x}^c$ *is the optimal solution.*

Proof. See proof of [2, Lemma 2].

In [3], it is shown that $c = 0.5$ is suitable when $\mathbf{n}_1$ and $\mathbf{n}_2$ are sampled from a multivariate normal distribution, i.e., $\mathbf{n}_1, \mathbf{n}_2 \sim \mathcal{N}(\mathbf{0}_P, \mathbf{I}_P)$. Note that s_1 and s_2 grow slowly with T. Hence, even for a very large T, s_1 and s_2 have reasonable values. For example, with $T = 2.5 \times 10^8$, we have $\sum_{t=1}^{T} t^{-1} < 20$. A large T will increase the computational complexity according to the discussions of Section 3.2.3.

4.4.3 l$_2$-Norm Sensitivity

In this subsection, we estimate the l_2-norm sensitivity of $\breve{\mathbf{x}}_i^{(k)}$. The l_2-norm sensitivity calibrates the magnitude of the noise by which $\breve{\mathbf{x}}_i^{(k)}$ has to be perturbed to preserve privacy. Unlike the existing privacy-preserving methods where the noise is added to the output of the algorithm [40, 41, 58, 62, 94, 95], in D-ZOA, the noise is inherent.

We introduce the following assumption that is widely used in the literature, see, e.g., [40, 41, 62].

Assumption 1: There exists a constant c_1 such that $\|\ell'(\cdot)\| \leq c_1$ where $\ell(\cdot)$ is the loss function defined in Section 4.3.

The l_2-norm sensitivity of $\breve{\mathbf{x}}_i^{(k)}$ is an upper bound on $\Delta_{i,2}$ defined in (2.26) and is computed as in the following lemma.

Lemma 3. *Under Assumption 1, the l_2-norm sensitivity of $\breve{\mathbf{x}}_i^{(k)}$ is given by*

$$\Delta_{i,2} = \frac{c_1}{\rho|\mathcal{V}_i|M_i}. \tag{4.10}$$

Proof. See proof of [2, Lemma 3].

4.4.4 Intrinsic (ϵ, δ)-Differential Privacy Guarantee

In this subsection, we reveal that the immanent stochasticity imparted by the embedded zeroth-order method makes D-ZOA (ϵ, δ)-differentially private. We provide an expression relating the primal variable variance, σ_i, to the privacy parameters ϵ and δ as well as an expression for ϵ relating it to the relevant algorithmic parameters.

We first prove that Algorithm 4 is (ϵ, δ)-differentially private at each iteration providing a relationship between σ_i and ϵ, δ.

Theorem 4.1. Let $\epsilon \in (0, 1]$ and

$$\sigma_i = \frac{c_1\sqrt{2.1\log(1.25/\delta)}}{\rho|\mathcal{V}_i|M_i\epsilon}. \tag{4.11}$$

Under Assumption 1, at each iteration of D-ZOA, (ϵ, δ)-differential privacy is guaranteed. Specifically, for any neighboring datasets $\mathcal{S}_i$ and $\mathcal{S}_i'$ and any output $\mathbf{x}_i^{(k)}$, the following inequality holds:

$$\text{Pr}[\mathbf{x}_{i,\mathcal{S}_i}^{(k)}] \leq e^{\epsilon}\text{Pr}[\mathbf{x}_{i,\mathcal{S}_i'}^{(k)}] + \delta. \tag{4.12}$$

Proof. See proof of [2, Theorem 1].

Theorem 1 shows that the primal variable variance is inversely proportional to the privacy parameter ϵ. This implies that a higher variance leads to a smaller ϵ and higher privacy guarantee. A smaller ϵ means that the ratio of the probability distributions of $\mathbf{x}_{i,\mathcal{S}_i}^{(k)}$ and $\mathbf{x}_{i,\mathcal{S}_i'}^{(k)}$ is smaller and consequently less information is available to any sniffing/spoofing adversary through $\mathbf{x}_i$ hence the improved privacy [41].

We then introduce the following corollary.

Corollary 1. *If* $\{\mathbf{g}_{j,i}^{(t)}\}_{j=1}^{J}$ *are i.i.d. with* $\mathbf{g}_{j,i}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_i^{(t)}, \Psi_i^{(t)})$*, and Assumption 1 holds, we have*

$$\epsilon = \frac{c_1}{\rho|\mathcal{V}_i|M_i}\sqrt{\frac{2.1JP\log(\frac{1.25}{\delta})}{\frac{cR^2\alpha_0^2}{\log(2P)}\left(s_1(1+\log P)+s_2\right) - \frac{4\|\mathbf{x}^c\|^2}{T}}}. \tag{4.13}$$

Proof. The proof follows from equating the expressions for σ_i in Lemma 2, (4.9), and Theorem 1, (4.11), and solving for ϵ. □

The equation (4.13) shows how the intrinsic privacy preserving property of D-ZOA is affected by various involved parameters. For example, a smaller J results in a smaller ϵ. This is consistent with the fact that a smaller J leads to a higher variance, which yields a higher privacy guarantee due to the inherent randomness brought about by using a zeroth-order method in the inner loop.

The denominator of the second factor in (4.13) is required to be positive for the factor to be real. We ensure this by setting

$$T > \frac{4\left\|\mathbf{x}^c\right\|^2 \log(2P)}{cR^2\alpha_0^2(s_1(1+\log(P))+s_2)}. \tag{4.14}$$

4.4.5 Total Privacy Leakage

In this subsection, we consider the total privacy leakage of the proposed D-ZOA algorithm. Since D-ZOA is a K-fold adaptive algorithm, we utilize the results of [93] together with the moments accountant method to evaluate its total privacy leakage. The main result is summarized in the following theorem.

Theorem 4.2. Let $\epsilon \in (0, 1]$ and

$$\sigma_i = \frac{c_1\sqrt{2.1\log(1.25/\delta)}}{\rho|\mathcal{V}_i|M_i\epsilon}. \tag{4.15}$$

Under Assumption 1, Algorithm 4 guarantees $(\bar{\epsilon}, \delta)$-differential privacy where

$$\bar{\epsilon} = \epsilon\sqrt{\frac{K\log(1/\delta)}{1.05\log(1.25/\delta)}}. \tag{4.16}$$

Proof. The proof is obtained by using the log moments of the privacy loss and their linear composability in the same way as in [40, Theorem 2]. □

4.5 Convergence Analysis and Privacy-Accuracy Trade-off

We establish the convergence of D-ZOA to a near-optimal solution by corroborating that both inner and outer loops of the algorithm converge as in Section 3.2.3. By employing the covariance of the primal variable as in Section 4.4.2, the convergence result in Theorem 3.1 can be extended as per the following theorem that also provides an explicit privacy-accuracy trade-off.

Theorem 4.3. For any $K > 0$, we have

$$\mathbb{E}[f(\hat{\mathbf{w}}^{(K)}) - f(\mathbf{w}^*)] \leq \frac{\left\|\mathbf{q}^{(0)} - \mathbf{q}\right\|_{\mathbf{G}}^2}{K} + \frac{2.1c_1^2 P\log(1.25/\delta)\lambda_{\max}^2(\mathbf{L}_+)}{2\rho|\mathcal{V}_i|^2 M_i^2\epsilon^2\lambda_{\min}(\mathbf{L}_-)} \tag{4.17}$$

where $\mathbf{q} = [\mathbf{r}^\mathsf{T}, (\mathbf{w}^*)^\mathsf{T}]^\mathsf{T}$ and

$$\hat{\mathbf{w}}^{(K)} = \frac{1}{K} \sum_{k=1}^{K} \breve{\mathbf{w}}^{(k)}.$$

Proof. See proof of [2, Theorem 3]. □

Theorem 3 shows that D-ZOA reaches a neighborhood of the optimal (centralized) solution with the size of the neighborhood determined by the privacy-parameter ϵ. This discloses a privacy-accuracy trade-off offered by D-ZOA. When the privacy guarantee is stronger (smaller ϵ and δ), the accuracy is lower.

Note that we do not need to solve the minimization problem in the ADMM primal update step of the outer loop with high accuracy [6, 40]. We perform the ADMM primal update step in the outer loop after obtaining an inexact solution to (2.9). Therefore, we select the number of inner loop iterations T as the minimum number of iterations satisfying (4.14) and entailing an accuracy that is sufficient to ensure convergence of ADMM according to [3]. In fact, T should be chosen as low as possible within the above-mentioned constraints to minimize the computational complexity according to the discussions of Section 3.2.3.

4.6 Simulations

In this section, we present some simulation examples to evaluate the performance and the privacy-accuracy trade-off of the proposed D-ZOA algorithm. We first benchmark D-ZOA against two popular existing algorithms for zeroth-order-based optimization, which have originally been designed for centralized settings, i.e., those proposed in [4] and [3] and called zeroth-order online ADMM (ZOO-ADMM) and optimal-rate zeroth-order (OR-ZO) algorithm, respectively. Then, we illustrate two sets of example histograms and QQ plots to verify that the entries of the gradient vector $\mathbf{g}_k^{(t)}$ are normally distributed. Next, we benchmark D-ZOA against some existing baseline differentially-private algorithms: the ADMM algorithm with primal variable perturbation (PVP) proposed in [40, 41], the ADMM with dual variable perturbation proposed in [41], the ADMM-based algorithm DP-ADMM proposed in [40], and the distributed subgradient algorithm (DPSGD) proposed in [9] that is customized to include differential privacy. As for the applications, we consider a distributed version of the empirical risk minimization problem with an ℓ_1-norm regularization (lasso penalty) and an ℓ_2-norm regularization (ridge penalty) [5].

In the centralized approach, a lasso estimate of $\mathbf{x}$ is given by

$$\mathbf{x}^c = \arg\min_{\mathbf{x}} \{ \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \eta \|\mathbf{x}\|_1 \} \tag{4.18}$$

while a ridge estimate of $\mathbf{x}$ is given by

$$\mathbf{x}^c = \arg\min_{\mathbf{x}} \{ \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \eta \|\mathbf{x}\|^2 \}. \tag{4.19}$$

In the distributed setting, we solve problem (4.1) with

$$\sum_{j=1}^{M_i} \ell(\mathbf{a}_{i,j}, b_{i,j}; \mathbf{x}_i) = \|\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i\|^2 \tag{4.20}$$

and $R(\mathbf{x}_i) = \|\mathbf{x}_i\|_1$ for the lasso penalty. For the ridge penalty, we have $R(\mathbf{x}_i) = \|\mathbf{x}_i\|^2$.

We assess the performance of the D-ZOA algorithm over a network of $N = 5$ agents with edge set $\mathcal{E} = \{e_{12}, e_{14}, e_{23}, e_{34}, e_{45}\}$. The number of samples at each agent is set to $M_i = 20 \; \forall i \in \mathcal{V}$ and the total number of samples is $M = 100$. The number of features in each sample is $P = 10$. For each agent $i \in \mathcal{V}$, we create a $2P \times P$ local observation matrix $\mathbf{A}_i$ whose entries are i.i.d. zero-mean unit-variance Gaussian random variables. The response vector $\mathbf{b}$ is synthesized as

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\phi}$$

where $\boldsymbol{\omega} \in \mathbb{R}^P$ and $\boldsymbol{\phi} \in \mathbb{R}^M$ are random vectors with distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$, respectively. The data are preprocessed by normalizing the columns of $\mathbf{A}$ to guarantee that the maximum value of each column is 1 and by normalizing the rows to enforce their l_2-norm to be less than 1 as in [40]. This is motivated by the need for homogeneous scaling of the features. Therefore, we have $c_1 = 1$. The regularization parameter is set to $\eta = 1$ and the penalty parameter is set to $\rho = 4$. The number of iterations of the ADMM outer loop is set to 200. For the inner loop, the number of iterations is set to 100 and the smoothing constant u_1 to 1. We set $\alpha_0 = 0.54$ according to [96] and calculate J from equation (4.13) by fixing ϵ, solving for J and rounding the solution to the nearest integer. With the current choice of parameters in the simulation section, the right hand side in equation (4.14) attains the value 5.85, so the inequality in (4.14) clearly holds since the left hand side is given by $T = 100$. Performance of D-ZOA is evaluated using the normalized error between the centralized solutions $\mathbf{x}^c$ as per (4.19) and the local estimates. It is defined as

$$\frac{\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{x}^c\|^2}{\|\mathbf{x}^c\|^2}$$

where $\mathbf{x}_i$ denotes the local estimate at agent i. The centralized solution $\mathbf{x}^c$ is computed using the convex optimization toolbox CVX [89]. Results are obtained by averaging over 100 independent trials. The parameters related to the inner loop have been chosen to get the convergence according to [3].The other parameters have been chosen in order to achieve the best performance in terms of accuracy and convergence rate in comparison with the benchmark algorithms, i.e., DP-ADMM, DPSGD, PVP, DVP, OR-ZO, ZOO-ADMM while benefiting from the convergence properties of the ADMM illustrated in [5] with respect to, e.g., to the number of outer loop iterations or the choice of the regularization parameter ρ.

In Fig. 4.1, we compare the performance of the proposed D-ZOA algorithm with that of two existing zeroth-order-based algorithms, i.e., those proposed in [4] and [3]. The simulation results show that the steady-state normalized error of D-ZOA is comparable to those of these algorithms, even though they are designed for centralized processing. The centralized algorithms converge faster than D-ZOA since, unlike our fully-distributed D-ZOA, they have all data concentrated at a central processing hub and do not rely on diffusing information across the network by sharing the local estimates within each agent's neighborhood. Note that the notion of iteration is essentially different for each algorithm whose learning curve is shown in Fig. 4.1. Thus, we provide the learning curve plots in Fig. 4.1 only to examine how D-ZOA performs in comparison with the existing zeroth-order optimization algorithms notwithstanding the underlying fundamental differences.

In Figs. 4.2-4.5, we provide two sets of histograms and QQ plots for an arbitrary entry of the stochastic gradient vector $\mathbf{g}_k^{(t)}$, i.e., the one corresponding to agent 2, and different inner and outer loop iterations, i.e., $t = 100$, $t = 50$, and $m = 50$, $m = 150$. The plots help us verify that the entries of $\mathbf{g}_k^{(t)}$ are normally distributed hence attest to the validity of our related assumption in Section 4.4.1.

We first benchmark our D-ZOA in the case of the ERM with the lasso penalty. Since PVP and DVP cannot be employed when the objective function is non-smooth, we benchmark our algorithm only with DP-ADMM and DPSGD similar to [40]. In Fig. 4.6 and 4.7, we plot the normalized error versus the outer loop iteration index for D-ZOA, DP-ADMM, and DPSGD. The plots show that all algorithms converge for two different values of ϵ and δ. In all plots, accuracy improves as ϵ increases. This is consistent with both Theorem 3 and [40, Theorem 3]. The hyper-parameters in DP-ADMM and DPSGD are tuned to achieve the best accuracy and convergence rate. However, D-ZOA has higher accuracy than DP-ADMM and DPSGD.

In Fig. 4.8 and 4.9, we illustrate the privacy-accuracy trade-off for D-ZOA, DP-ADMM, and DPSGD. The figures show that D-ZOA, DP-ADMM, and DPSGD

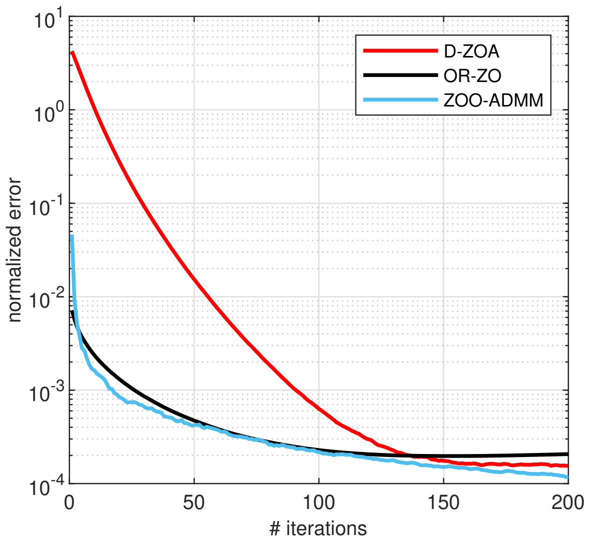


Figure 4.1: The normalized errors of D-ZOA, OR-ZO [3], and ZOO-ADMM [4] versus the iteration number.

achieve higher accuracy with larger ϵ and δ. In Fig. 4.8, we show the normalized error versus the privacy parameter $\bar{\epsilon}$ as given in (4.16) for $\delta = 10^{-6}$ and $\delta = 10^{-3}$. We observe that D-ZOA outperforms both DP-ADMM and DPSGD in terms of accuracy likely due to its intrinsic privacy-preserving properties. Fig. 4.9 also attests to the superiority of D-ZOA over DP-ADMM and DPSGD when $\epsilon = 0.15$ and $\epsilon = 0.95$ and δ varies between 10^{-6} and 10^{-2}.

We also evaluate the performance of the D-ZOA algorithm in comparison with the considered benchmark algorithms for the ERM with the ridge penalty. In Fig. 4.10 and 4.11, we plot the normalized error versus the outer loop iteration index for D-ZOA, DP-ADMM, DPSGD, PVP, and DVP. The plots show that D-ZOA outperforms all other considered algorithms in terms of accuracy for different values of ϵ.

In Fig. 4.12 and 4.13, we demonstrate the privacy-accuracy trade-off for D-ZOA, DP-ADMM, DPSGD, PVP, and DVP. As expected, smaller values of the privacy parameters ϵ and δ lead to lower accuracy. However, D-ZOA outperforms all the other approaches in terms of accuracy due to its intrinsic privacy-preserving properties.

In the considered applications of ERM with lasso and ridge penalty, we make the following observations regarding the complexity-accuracy trade-off of D-ZOA

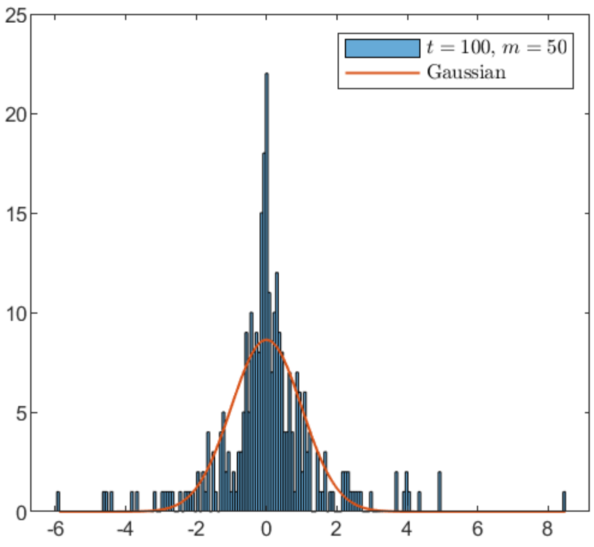


Figure 4.2: The histogram of $\mathbf{g}_k^{(t)}$ at agent 2, the inner loop iteration $t = 100$, and the outer loop iteration $m = 50$.

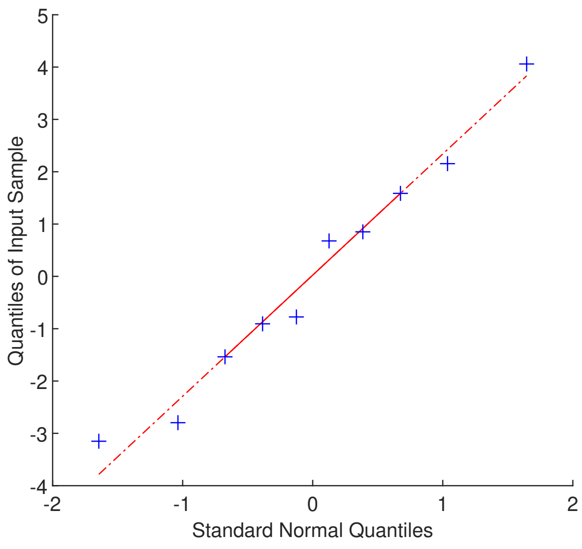


Figure 4.3: The QQ plot of $\mathbf{g}_k^{(t)}$ at agent 2, the inner loop iteration $t = 100$, and the outer loop iteration $m = 50$.

and the baseline algorithms. D-ZOA outperforms all the baseline algorithms in

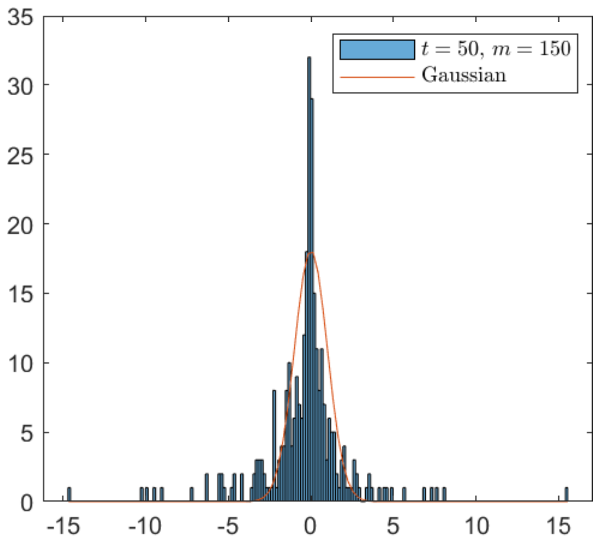


Figure 4.4: The histogram of $\mathbf{g}_k^{(t)}$ at agent 2, the inner loop iteration $t = 50$, and the outer loop iteration $m = 150$.

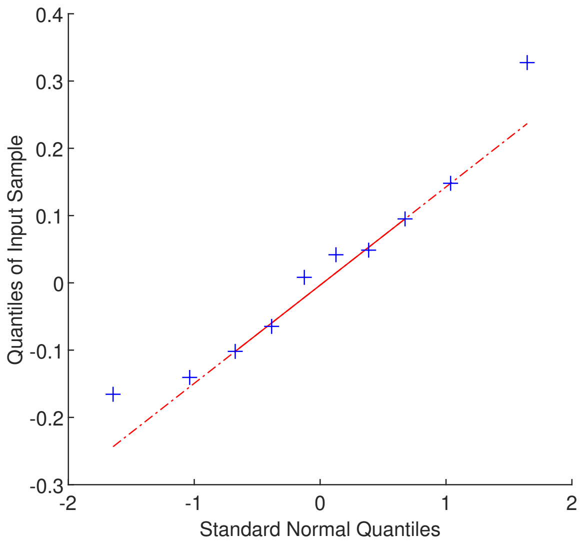


Figure 4.5: The QQ plot of $\mathbf{g}_k^{(t)}$ at agent 2, the inner loop iteration $t = 50$, and the outer loop iteration $m = 150$.

terms of accuracy by roughly two orders of magnitude. However, D-ZOA has a

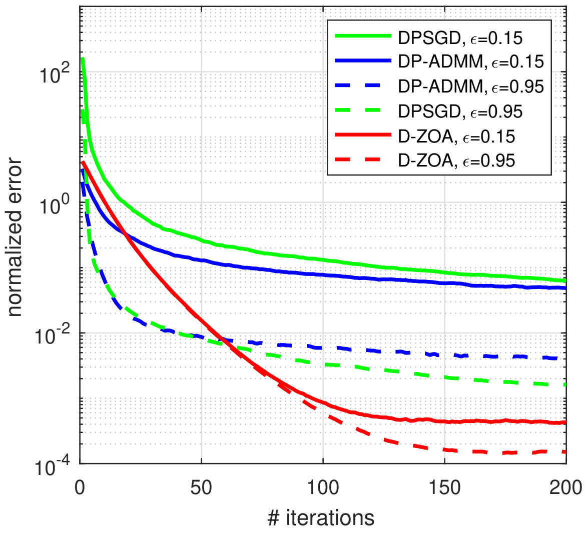


Figure 4.6: Normalized error of DPSGD, DP-ADMM, and D-ZOA for two values of ϵ and $\delta = 10^{-3}$ for ERM with ℓ_1-norm regularization.

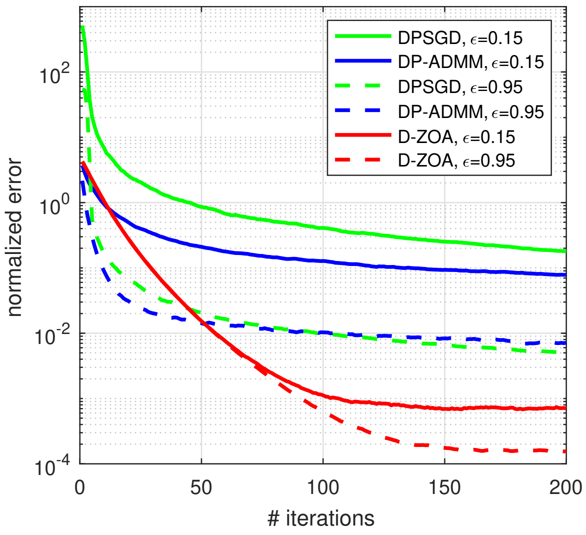


Figure 4.7: Normalized error of DPSGD, DP-ADMM, and D-ZOA for two values of ϵ and $\delta = 10^{-6}$ for ERM with ℓ_1-norm regularization.

relatively high computational complexity due to its inner loop that is run at every agent i and every ADMM iteration. Since the number of arithmetic operations

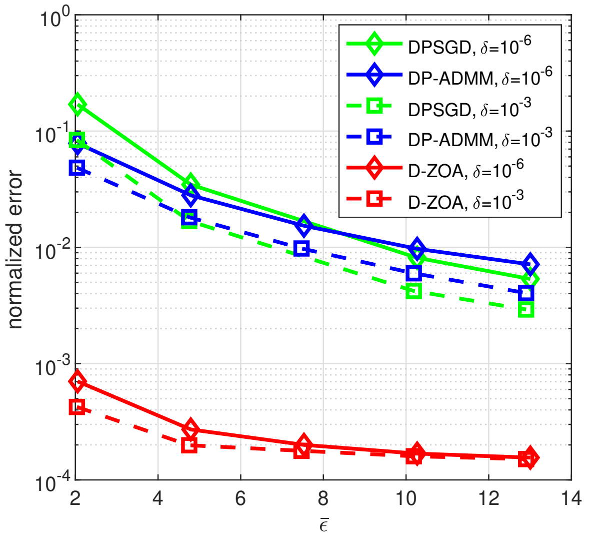


Figure 4.8: Privacy-accuracy trade-off of DPSGD, DP-ADMM, and D-ZOA for ERM with ℓ_1-norm regularization for $\delta = 10^{-6}$ and $\delta = 10^{-3}$.

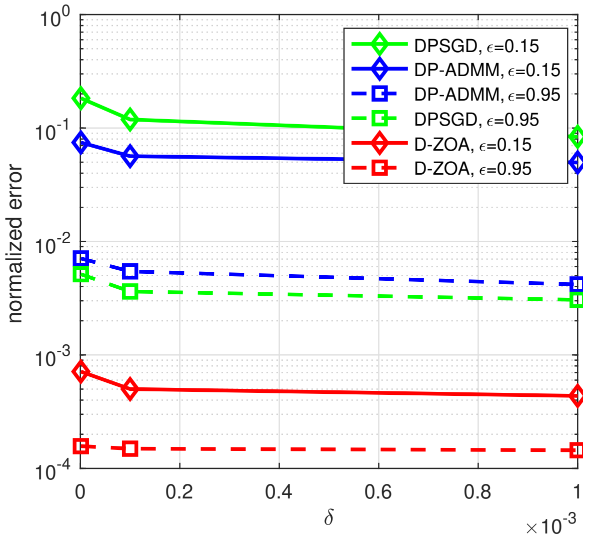


Figure 4.9: Privacy-accuracy trade-off of DPSGD, DP-ADMM, and D-ZOA for ERM with ℓ_1-norm regularization for $\epsilon = 0.15$ and $\epsilon = 0.95$.

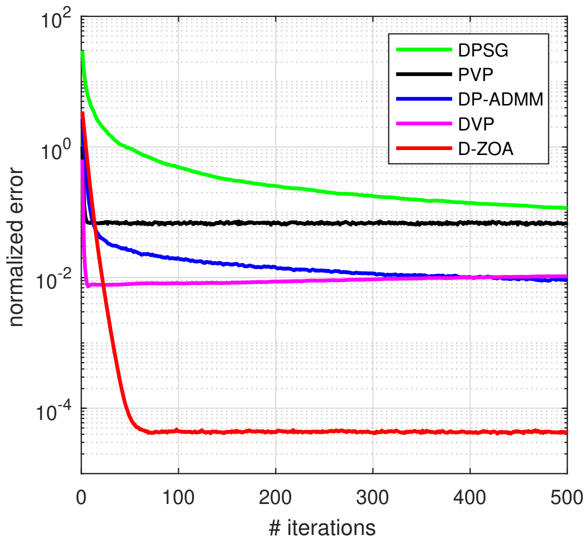


Figure 4.10: Normalized error of DPSGD, PVP, DP-ADMM, DVP, and D-ZOA for $\epsilon = 0.40$ and $\delta = 10^{-3}$ for ERM with ℓ_2-norm regularization.

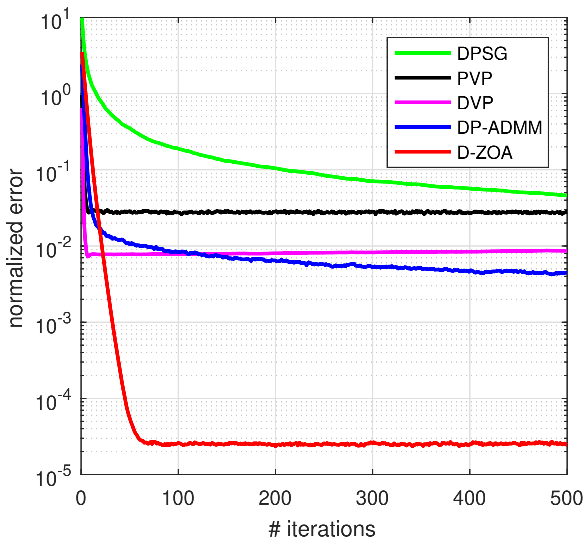


Figure 4.11: Normalized error of DPSGD, PVP, DP-ADMM, DVP, and D-ZOA for $\epsilon = 0.80$ and $\delta = 10^{-3}$ for ERM with ℓ_2-norm regularization.

required to evaluate the objective function is $\mathcal{O}(PM_i)$, calculation of (2.19) needs $\mathcal{O}(JPM_i)$ operations and, therefore, D-ZOA requires $\mathcal{O}(TJPM_i)$ operations to

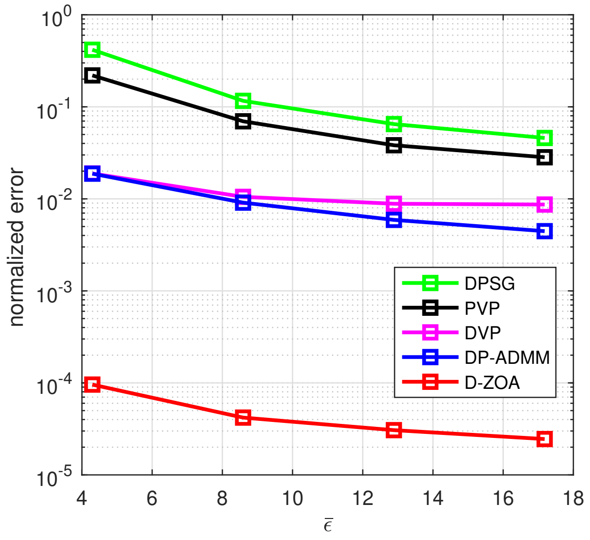


Figure 4.12: Privacy-accuracy trade-off of DPSGD, PVP, DP-ADMM, DVP, and D-ZOA for ERM with ℓ_2-norm regularization for $\delta = 10^{-3}$.

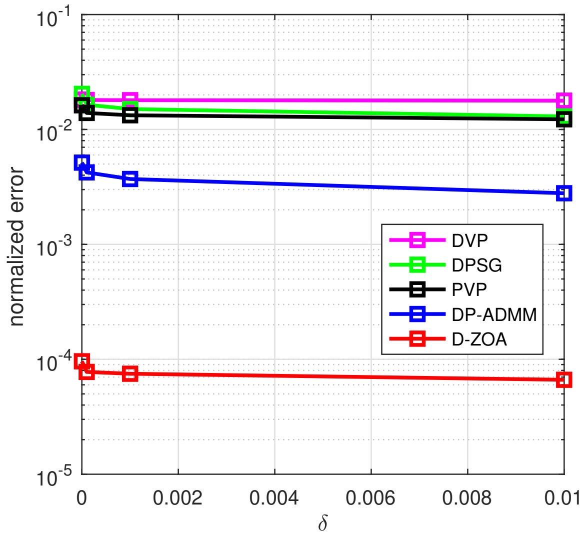


Figure 4.13: Privacy-accuracy trade-off of DPSGD, PVP, DP-ADMM, DVP, and D-ZOA for ERM with ℓ_2-norm regularization for $\epsilon = 0.95$.

perform (2.9). The baseline algorithms have the following computational complexities: $\mathcal{O}(P^2 M_i)$ for DP-ADMM and DPSGD, and $\mathcal{O}(P^2 M_i + P^3)$ for PVP and DVP. In order to be able to have a quicker and easier comparison between the above-mentioned complexities, we recall that, in the simulations, the values $J = 30, T = 100$ were used.

4.7 Conclusion

In [2], we showed that the D-ZOA algorithm proposed in [13] and described in Section 3.2 is intrinsically capable of ensuring differential privacy. We proved that the inherent randomness due to employing the zeroth-order method can adequately make the D-ZOA algorithm intrinsically privacy-preserving. In addition, we used the moments accountant method to show that the total privacy leakage of D-ZOA grows sublinearly with the number of ADMM iterations. We verified the convergence of D-ZOA to a near-optimal solution as well as studying its privacy-preserving properties through both theoretical analysis and numerical simulations.

Chapter 5

Distributed Optimization with Feature Partitioning

In this chapter, we present three contributions in the context of distributed learning with feature partitioning of data. The first contribution deals with a fully-distributed algorithm to solve the ridge regression problem with feature partitioning of data. In the second contribution, we propose an algorithm to solve feature-distributed learning problems with an ℓ_2-norm-square cost function and non-smooth regularizer functions. In the third contribution, we extend the two previously proposed algorithms by developing a fully-distributed algorithm for solving learning problems when the data is distributed among agents in feature partitions and computing the conjugate of the possibly non-smooth cost or regularizer functions is challenging or unfeasible.

5.1 Distributed Ridge Regression

In [22], we develop a new distributed algorithm to solve the ridge regression problem with feature partitioning of the observation matrix. The proposed algorithm, named D-Ridge, is based on the alternating direction method of multipliers (ADMM) and estimates the parameters when the observation matrix is distributed among different agents with feature (or vertical) partitioning. We formulate the associated ridge regression problem as a distributed convex optimization problem and utilize the ADMM to obtain an iterative solution. Numerical results demonstrate that D-Ridge converges faster than its diffusion-based contender does.

5.1.1 Related Work

The regression problem with feature partitioning has previously been considered in [35, 37, 38, 42]. However, works of [37, 38] assume a proper coloring scheme of the network and cannot be extended to a general graph labeling. The algorithm proposed in [42] is not truly distributed since its consensus constraints involve the entire network instead of each agent's local neighborhood. The algorithm in [35] is fully distributed and based on the diffusion strategy [97]. However, as we will show later on, it converges relatively slowly.

5.1.2 Contributions

In [22], we solve the ridge regression problem with feature partitioning of the observation matrix in a distributed fashion using the alternating direction method of multipliers (ADMM). The proposed algorithm, called D-Ridge, is fully distributed and requires communications only among neighboring agents. It also converges faster than the diffusion-based algorithm of [35] and has a per-iteration per-agent computational complexity order that is linear in the sample size. In addition, D-Ridge does not require the agents to share their local data or dual variables with the other agents but only the primal variables, which are the estimate solutions of the corresponding local optimization subproblems. Hence, D-Ridge respects the possible data privacy of the agents. We verify the convergence of D-Ridge to the centralized solution at all agents through both theoretical analysis and simulations. Our experiments with a variety of network topologies show that D-Ridge outperforms its diffusion-based contender in terms of convergence rate.

5.1.3 System Model

We consider the system model described in Section 2.1 with feature partitioning of data as presented in Section 2.1.2. In the centralized approach, a ridge regression estimator of $\mathbf{x}$ is given by

$$\hat{\mathbf{x}}^c = \arg\min_{\mathbf{x}}\{\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2 + \eta\,\|\mathbf{x}\|^2\} \tag{5.1}$$

where $\eta > 0$ is the regularization parameter. From the normal equation associated with (5.1), the centralized estimate is given by

$$\hat{\mathbf{x}}^c = \mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}} + \eta\mathbf{I}_M)^{-1}\mathbf{b}. \tag{5.2}$$

Since computing a centralized solution of (5.1) over a network may be inefficient, we propose a distributed algorithm for this purpose in the following section.

5.1.4 Distributed Ridge Regression via ADMM

We first discuss the reformulation of the ridge regression problem whose solution allows us to find a distributed solution to (5.1) via the ADMM. Then, we describe the construction steps and main properties of the proposed algorithm for solving the resulting constrained minimization problem. Finally, we establish the global convergence of D-Ridge theoretically.

Reformulation of Ridge Regression

Let us define a vector $\mathbf{f}^o \in \mathbb{R}^{M\times 1}$ as

$$\mathbf{f}^o = (\mathbf{A}\mathbf{A}^{\mathrm{T}} + \eta\mathbf{I}_M)^{-1}\mathbf{b}.$$

From (5.2), the part of $\mathbf{x}^c$ corresponding to agent i can be calculated as

$$\hat{\mathbf{x}}_i^c = \mathbf{A}_i^{\mathrm{T}}\mathbf{f}^o. \tag{5.3}$$

For computing $\mathbf{f}^o$ at all agents using only in-network processing of the locally available data, we propose a consensus-based distributed algorithm. Note that $\mathbf{f}^o$ is the unique minimizer of the quadratic global cost function $\mathcal{J}(\mathbf{f})$ defined as

$$\mathcal{J}(\mathbf{f}) = \frac{1}{2}\mathbf{f}^{\mathrm{T}}(\mathbf{A}\mathbf{A}^{\mathrm{T}} + \eta\mathbf{I}_M)\mathbf{f} - \mathbf{f}^{\mathrm{T}}\mathbf{b}. \tag{5.4}$$

Since $\mathbf{A}\mathbf{A}^{\mathrm{T}} = \sum_{i=1}^{N}\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}}$, $\mathbf{f}^o$ is given by

$$\mathbf{f}^o = \arg\min_{\mathbf{f}} \sum_{i=1}^{N} \mathcal{J}_i(\mathbf{f}) \tag{5.5}$$

where

$$\mathcal{J}_i(\mathbf{f}) = \frac{1}{2}\mathbf{f}^{\mathrm{T}}\Big(\mathbf{A}_i\mathbf{A}_i^{\mathrm{T}} + \frac{\eta}{N}\mathbf{I}_M\Big)\mathbf{f} - \frac{\delta_i}{B}\mathbf{f}^{\mathrm{T}}\mathbf{b}, \tag{5.6}$$

$B \in \mathbb{N}$ is the number of agents having access to $\mathbf{b}$, and $\delta_i = 1$ if $\mathbf{b}$ is available at agent i and $\delta_i = 0$ otherwise.

Solving (5.5) is equivalent to solving problem (2.1) with horizontal partitioning. Minimizing the objective in (5.5) through ADMM entails an iterative process that is described in the next section.

Algorithm Description

To solve problem (5.5) in a fully-distributed fashion, we employ the ADMM as described in Section 2.2. The resulting D-Ridge algorithm consists of two steps at each iteration: the primal update and the dual update steps. Hence, the D-Ridge algorithm consists of the following iterative updates that are carried out locally at

Algorithm 5 D-Ridge

```
At all agents $i \in \mathcal{V}$, initialize $\mathbf{f}_i^{(0)}$, $\boldsymbol{\lambda}_i^{(0)}$ to zero vectors, and run locally
**for** $k = 0, 1, \ldots, K$ **do**
  Receive $\mathbf{f}_i^{(k)}$ from neighbors in $\mathcal{V}_i$.
  Update $\boldsymbol{\lambda}_i^{(k)}$ as in (5.7).
  Update $\mathbf{f}_i^{(k+1)}$ as in (5.8).
**end for**
Estimate $\hat{\mathbf{x}}_i = \mathbf{A}_i^{\mathrm{T}} \mathbf{f}_i^{(K+1)}$.
```

every agent:

$$\boldsymbol{\lambda}_i^{(k)} = \boldsymbol{\lambda}_i^{(k-1)} + \rho \sum_{j \in \mathcal{V}_i} [\mathbf{f}_i^{(k)} - \mathbf{f}_j^{(k)}] \tag{5.7}$$

$$\begin{aligned}
\mathbf{f}_i^{(k+1)} &= \arg\min_{\{\mathbf{f}_i\}} \Big\{ \frac{1}{2} \mathbf{f}_i^{\mathrm{T}} \Big(\mathbf{A}_i \mathbf{A}_i^{\mathrm{T}} + \frac{\eta}{N} \mathbf{I}_M \Big) \mathbf{f}_i - \frac{\delta_i}{B} \mathbf{f}_i^{\mathrm{T}} \mathbf{b} \\
&\quad + (\boldsymbol{\lambda}_i^{(k)})^{\mathrm{T}} \mathbf{f}_i + \rho \sum_{j \in \mathcal{V}_i} \left\| \mathbf{f}_i - \frac{\mathbf{f}_i^{(k)} + \mathbf{f}_j^{(k)}}{2} \right\|^2 \Big\} \\
&= \Big[\mathbf{A}_i \mathbf{A}_i^{\mathrm{T}} + \Big(\frac{\eta}{N} + 2\rho |\mathcal{V}_i| \Big) \mathbf{I}_M \Big]^{-1} \\
&\quad \Big(\frac{\delta_i}{B} \mathbf{b} - \boldsymbol{\lambda}_i^{(k)} + \rho |\mathcal{V}_i| \mathbf{f}_i^{(k)} + \rho \sum_{j \in \mathcal{V}_i} \mathbf{f}_j^{(k)} \Big)
\end{aligned} \tag{5.8}$$

where superscript (k) is the iteration index and all initial values $\{\mathbf{f}_i^{(0)}\}_{i \in \mathcal{V}}$, $\{\boldsymbol{\mu}_i^{(0)}\}_{i \in \mathcal{V}}$ are set to zero. The proposed approach is summarized in Algorithm 5.

Note that $\mathbf{f}_i^{(k)}$ is the only vector that is shared between the agents at every iteration. The computation of (5.8) has a per-iteration per-agent complexity of $\mathcal{O}(M^2)$. It involves the inversion of the $M \times M$ matrix $\mathbf{A}_i \mathbf{A}_i^{\mathrm{T}} + \Big(\frac{\eta}{N} + 2\rho |\mathcal{V}_i| \Big) \mathbf{I}_M$ that may be computationally demanding for $M \gg P_i$. However, this operation can be carried out off-line before running the algorithm. Hence, we achieve a per-iteration per-agent computational complexity of $\mathcal{O}(2MP_i + P_i^2)$.

In the next subsection, we show that D-Ridge generates sequences of local iterates $\mathbf{f}_i^{(k)}$, $i = 1, \ldots, N$, that, at each agent i, converge to the global centralized solution $\mathbf{x}^c$ as $k \to \infty$.

Convergence Analysis

Convergence of the proposed algorithm is established by verifying that both conditions for the ADMM to converge are fulfilled, namely, for each agent $i \in \mathcal{V}$, the cost function $\mathcal{J}_i(\mathbf{f})$ is strongly convex and its gradient $\nabla_{\mathbf{f}}\mathcal{J}_i(\mathbf{f})$ is Lipschitz continuous [98].
The function $\mathcal{J}_i(\mathbf{f})$ is strongly convex since it is twice continuously differentiable and has a positive-definite Hessian matrix:

$$\nabla^2_{\mathbf{f}}\mathcal{J}_i(\mathbf{f}) = \mathbf{A}_i\mathbf{A}_i^{\mathrm{T}} + \frac{\eta}{N}\mathbf{I}_M \succ 0.$$

Moreover, $\nabla_{\mathbf{f}}\mathcal{J}_i(\mathbf{f})$ is a linear function of $\mathbf{f}$. Therefore, it is Lipschitz continuous [99] with a Lipschitz constant being the operator norm of $\nabla^2_{\mathbf{f}}\mathcal{J}_i(\mathbf{f})$.

5.1.5 Simulations

The D-Ridge algorithm is tested here on a network of $N = 10$ agents with the topology as shown in Fig. 5.1. Each agent holds the data for two features. Therefore, $P_i = 2$, $i = 1, ..., N$, and $P = 20$. The observation data matrix $\mathbf{A}$ has $M = 50$ regressor vectors with independent zero-mean multivariate Gaussian distribution as its rows. We calculate the response vector $\mathbf{b}$ as

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\psi}$$

where $\boldsymbol{\omega} \in \mathbb{R}^P$ and $\boldsymbol{\psi} \in \mathbb{R}^M$ are independently drawn from the multivariate normal distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$, respectively. We evaluate the performance of the proposed algorithm using the misalignment metric that is defined as

$$\frac{\left\|\mathbf{x}^d(k) - \boldsymbol{\omega}\right\|^2}{\left\|\boldsymbol{\omega}\right\|^2}$$

where

$$\mathbf{x}^d(k) = \left[\mathbf{x}_1^{(k)\mathsf{T}}, \ldots, \mathbf{x}_N^{(k)\mathsf{T}}\right]^{\mathsf{T}}$$

and $\mathbf{x}_i^{(k)} = \mathbf{A}_i^{\mathrm{T}}\mathbf{f}_i^{(k)}\ \forall i \in \mathcal{V}$ denotes the local estimate at agent i. The penalty parameter is set to $\rho = 4$ and, as in [35], the regularization parameter is set to $\eta = 10^{-3}$. The parameters are also chosen as in [35] in order to ensure a fair comparison with the benchmark algorithms.
In Figs. 2-4, we plot the misalignment versus the iteration index for D-Ridge and the diffusion-based algorithm of [35] with different values of the step-size μ.

The results in Figs. 5.2-5.4 are obtained by averaging over 100 independent trials. The number of agents having access to $\mathbf{b}$, i.e., B affects the convergence speed of

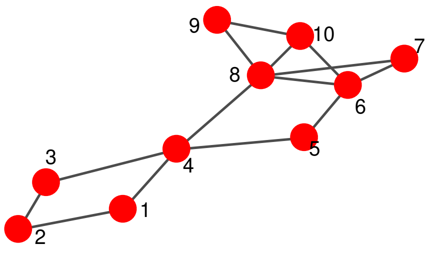


Figure 5.1: Topology of the considered multi-agent network.

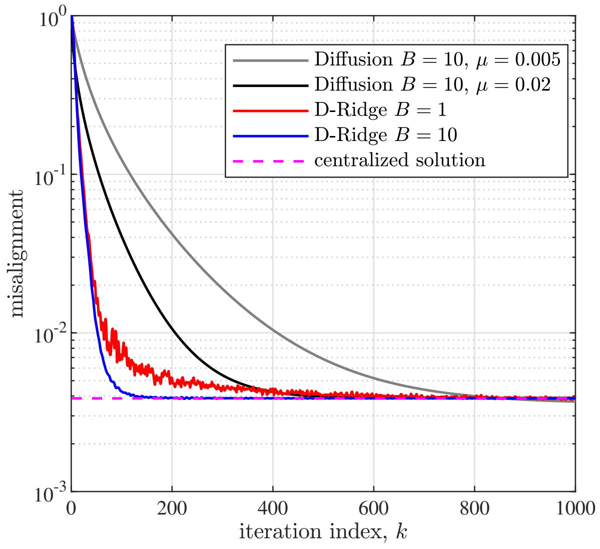


Figure 5.2: The misalignment of D-Ridge and the diffusion-based algorithm with different values of the step-size μ when one or all agents have access to $\mathbf{b}$.

D-Ridge, while it does not have any significant effect on the performance of the diffusion-based algorithm [35]. In Figs. 5.2-5.4, the regression vector is placed in the agent i with the greatest $|\mathcal{V}_i|$ if $B = 1$, while it is randomly placed over the network if $B > 1$.

Fig. 5.2 shows that D-Ridge converges significantly faster than the diffusion-based algorithm, especially when all agents have access to $\mathbf{b}$, i.e., $B = 10$. Fig. 5.3 shows that the D-Ridge algorithm converges faster as the number of agents that have access to $\mathbf{b}$ increases. Fig. 5.4 shows that D-Ridge converges faster than the diffusion-based algorithm irrespective of the network topology. The performance of the algorithm with the topology shown in Fig. 5.1 is compared to a linear topology with the same number of agents where the agents are connected one after the other, hence $|\mathcal{V}_i| = 3$ for $1 < i < N$ and $|\mathcal{V}_i| = 2$ for $i = 1$ and $i = N$.

5.1.6 Conclusion

In[22], we developed a new consensus-based algorithm for distributed solution of the ridge regression problem with feature partitioning of the observation matrix. To this end, we recast the ridge regression problem into an equivalent constrained separable form, whose structure is suitable for distributed implementation through ADMM. In the proposed algorithm, D-Ridge, the agents exchange messages only within their neighborhoods. Simulation results showed that the sequences of local iterates generated by D-Ridge converge to the centralized solution faster than the diffusion-based algorithm does.

5.2 Distributed Optimization with Non-Smooth Regularizers

In [39], we develop a new algorithm for distributed learning with non-smooth regularizers and feature partitioning. To this end, we transform the underlying optimization problem into a suitable dual form and solve it using the alternating direction method of multipliers. The proposed algorithm is fully-distributed and does not require the conjugate function of any non-smooth regularizer function, which may be unfeasible or computationally inefficient to acquire. Numerical experiments demonstrate the effectiveness of the proposed algorithm.

5.2.1 Related Work

There have been several attempts to solve learning problems with feature partitioning of data, e.g., in [5, 15, 22, 35–38, 47, 50–55, 65]. However, the algorithms in [36, 37] can only be used to solve the basis pursuit and lasso problems, respectively, while the work in [38] is based on assuming an appropriate coloring scheme of the network and cannot be extended to a general graph labeling. The algorithms developed in [15, 35, 47] are based on the diffusion strategy. In contrast, the approaches in [5, 22] are based on the consensus strategy. However, [5] is not fully distributed since the consensus constraints are imposed globally across the entire network rather than being applied locally within each agent's neighborhood. Although the algorithm in [22] is fully distributed, it assumes a specific structure for the objective function and is only suitable for ridge regression. The works of [52–55] consider distributed agent-specific estimation. However, the objective functions considered in these works are smooth. The authors of [50] propose a coordinate-descent-based algorithm with an inexact update to reduce communic-

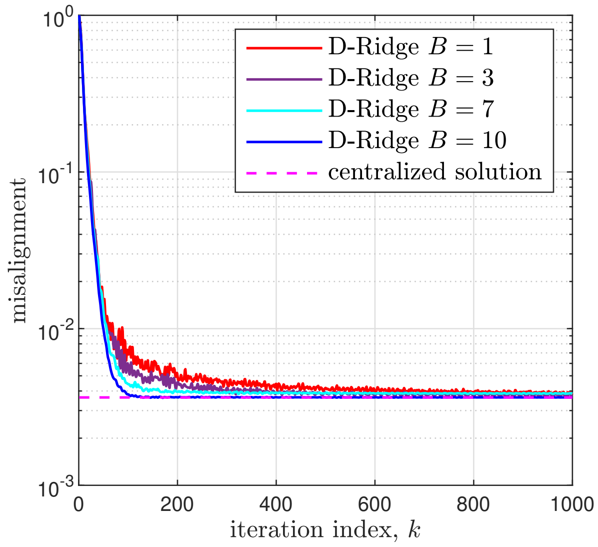


Figure 5.3: The misalignment of D-Ridge for different values of B.

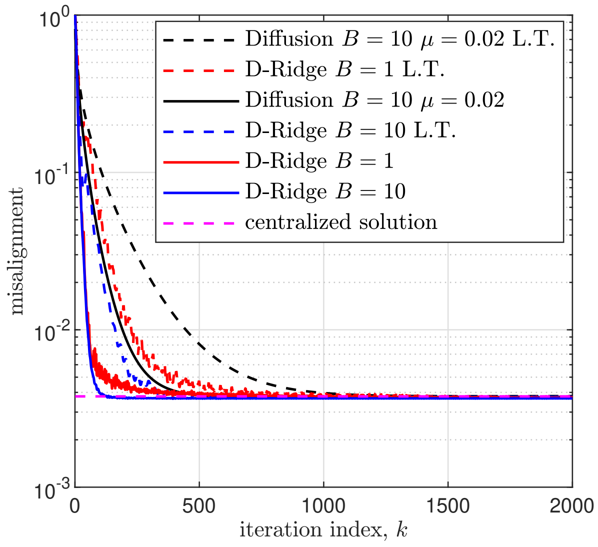


Figure 5.4: The misalignment of D-Ridge and the diffusion-based algorithm for the considered network topology and for the linear topology (L.T.).

ation costs for feature-partitioned distributed learning. In [51], an asynchronous stochastic gradient-descent algorithm was developed for distributed learning with feature partitioning of data. However, none of the above-mentioned algorithms consider distributed problems with general non-smooth regularization and arbitrary graphs.

5.2.2 Contributions

In [39], we develop a new fully-distributed algorithm for distributed learning with non-smooth regularizers and feature partitioning of data. We consider a general regularized learning problem whose cost function cannot be written as the sum of the local agent-specific cost functions, i.e., it is not separable. To achieve separability, we formulate the dual problem associated with the underlying convex optimization problem and exploit its favorable structure that, unlike the original problem, allows us to solve it by utilizing the alternating direction method of multipliers (ADMM). By utilizing the dual of the optimization problem associated with the ADMM primal variable update step, we devise a new strategy that does not require any conjugate function of the non-smooth regularizers, which may be infeasible or hard to obtain in some scenarios. The proposed algorithm is fully-distributed as every agent communicates only with its neighboring agents and no central coordinator is needed. Our simulation results show that the proposed algorithm converges in various scenarios.

5.2.3 System Model

We consider the system model described in Section 2.1 with feature partitioning of data as presented in Section 2.1.2. We consider a regularized learning problem of form

$$\min_{\{\mathbf{x}_i\}} \quad g\left(\textstyle\sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_i - \mathbf{b}\right) + \sum_{i=1}^{N} r_i(\mathbf{x}_i) \tag{5.9}$$

where $g(\cdot)$ is the global cost function and $r_i(\cdot)$, $i = 1, \ldots, N$, are the agent-specific regularizer functions. The learning problem (5.29) pertains to several applications in machine learning, e.g., regression over distributed features [5], clustering in graphs [100], smart grid control [101], dictionary learning [46], and network utility maximization [102]. In this work, we consider learning problems where functions $r_i(\cdot)$, $i = 1, \ldots, N$, are convex, proper, and lower semi-continuous but not necessarily smooth and $g(\cdot) = \|\cdot\|^2$. In the next section, we solve (5.29) in a distributed manner, where each agent communicates only with its neighbors.

5.2.4 Distributed Algorithm for Learning with Feature Partitioning

First, we present the reformulation of the considered non-separable problem into a dual form that is separable and can be solved in a fully-distributed fashion via the ADMM. Then, we describe the new strategy that allows us to employ the ADMM without computing any conjugate function of the non-smooth regularizers explicitly.

Distributed ADMM for the Dual Problem

To develop a distributed solution, we introduce the auxiliary variables $\{\mathbf{z}_i\}_{i=1}^N$ and recast (5.29) as

$$\begin{aligned} \min_{\{\mathbf{x}_i,\mathbf{z}_i\}} \quad & g\left(\textstyle\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}\right) + \sum_{i=1}^N r_i(\mathbf{x}_i) \\ \text{s. t.} \quad & \mathbf{A}_i \mathbf{x}_i = \mathbf{z}_i, \quad i = 1, \ldots, N. \end{aligned} \tag{5.10}$$

The objective function in (5.30) is not separable among the agents. Therefore, we consider the dual problem of (5.30). For this purpose, we associate the Lagrange multipliers $\{\boldsymbol{\mu}_i\}_{i=1}^N$ with the equality constraints in (5.30) and form the Lagrangian function $\mathcal{L}(\{\mathbf{x}_i\}, \{\mathbf{z}_i\}, \{\boldsymbol{\mu}_i\})$. The dual function for problem (5.30) is given by

$$\begin{aligned} d(\{\boldsymbol{\mu}_i\}) &= \inf_{\{\mathbf{x}_i,\mathbf{z}_i\}} \mathcal{L}(\{\mathbf{x}_i\}, \{\mathbf{z}_i\}, \{\boldsymbol{\mu}_i\}) \\ &= -\sum_{i=1}^N r_i^*(-\mathbf{A}_i^\mathsf{T} \boldsymbol{\mu}_i) + \inf_{\mathbf{z}_i} \Big\{ g(\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}) - \sum_{i=1}^N \boldsymbol{\mu}_i^\mathsf{T} \mathbf{z}_i \Big\} \end{aligned} \tag{5.11}$$

where r_i^* is the conjugate function of r defined as

$$r_i^*(\mathbf{y}) = \sup_{\mathbf{x}} \ \mathbf{y}^T \mathbf{x} - r_i(\mathbf{x}).$$

Next, for the second infimum in (5.32), introducing

$$\mathbf{z} = \sum_{i=1}^N \mathbf{z}_i$$

and its corresponding dual variable $\boldsymbol{\lambda}$, and using the duality theory, an alternate form of the dual function (5.32) is obtained as

$$\tilde{d}(\{\boldsymbol{\mu}_i\}, \boldsymbol{\lambda}) = \begin{cases} -\tilde{g}^*(\boldsymbol{\lambda}) - \sum_{i=1}^N r_i^*(-\mathbf{A}_i^\mathsf{T} \boldsymbol{\mu}_i), & \boldsymbol{\lambda} = \boldsymbol{\mu}_i, \forall i \in \mathcal{V} \\ -\infty, & \text{otherwise} \end{cases} \tag{5.12}$$

where

$$\tilde{g}^*(\boldsymbol{\lambda}) = g^*(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^\mathsf{T} \mathbf{b}.$$

Eliminating the redundant variable $\boldsymbol{\lambda}$, the dual problem for (5.30) can be expressed as

$$\begin{aligned} \max_{\{\boldsymbol{\mu}_i\}} \quad & -\frac{1}{N}\sum_{i=1}^{N} \tilde{g}^*(\boldsymbol{\mu}_i) - \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ \text{s. t.} \quad & \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \cdots = \boldsymbol{\mu}_N. \end{aligned} \tag{5.13}$$

To solve (5.13) in a distributed fashion, we employ the ADMM [7]. First, we recast (5.13) as a constrained minimization problem by imposing consensus constraints across each agent's neighborhood $\mathcal{V}_i$ as follows

$$\begin{aligned} \min_{\{\boldsymbol{\mu}_i\}} \quad & \frac{1}{N}\sum_{i=1}^{N} \tilde{g}^*(\boldsymbol{\mu}_i) + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ \text{s. t.} \quad & \boldsymbol{\mu}_i = \boldsymbol{\mu}_j, \quad j \in \mathcal{V}_i,\ i = 1, \ldots, N. \end{aligned} \tag{5.14}$$

In (5.14), we recast the consensus constraints throughout the whole network into localized constraints across each agent's neighborhood. The previous problem can be rewritten as the following constrained optimization problem

$$\begin{aligned} \min_{\{\boldsymbol{\mu}_i\},\{\mathbf{u}_i^j\}} \quad & \frac{1}{N}\sum_{i=1}^{N} \tilde{g}^*(\boldsymbol{\mu}_i) + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ \text{s. t.} \quad & \boldsymbol{\mu}_i = \mathbf{u}_i^j, \quad \boldsymbol{\mu}_j = \mathbf{u}_i^j,\ j \in \mathcal{V}_i,\ i = 1, \ldots, N. \end{aligned} \tag{5.15}$$

To facilitate a fully-distributed solution, we decouple the constraints in (5.13) by introducing the auxiliary variables $\{\mathbf{u}_i^j\}_{j\in\mathcal{V}_i}$. Then, we generate the relevant augmented Lagrangian function by associating the Lagrange multipliers $\{\bar{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$, $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ with the consensus constraints. In [7], it is shown that, by setting

$$\mathbf{v}_i^{(k)} = 2\sum_{j\in\mathcal{V}_i} (\bar{\mathbf{v}}_i^j)^{(k)},$$

the Lagrange multipliers $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ and the auxiliary variables $\{\mathbf{u}_i^j\}_{j\in\mathcal{V}_i}$ are eliminated through the steps illustrated in Section 2.2.2 and the ADMM reduces to an iterative procedure with two steps at each iteration as

$$\begin{aligned}\boldsymbol{\mu}_i^{(k)} =& \arg\min_{\boldsymbol{\mu}_i} \Big\{ \frac{1}{N} g^*(\boldsymbol{\mu}_i) + \frac{1}{N}\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b} + r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ &+ \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{v}_i^{(k-1)} + \rho \sum_{j\in\mathcal{V}_i} \Big\| \boldsymbol{\mu}_i - \frac{\boldsymbol{\mu}_i^{(k-1)} + \boldsymbol{\mu}_j^{(k-1)}}{2} \Big\|^2 \Big\}, \end{aligned} \tag{5.16}$$

$$\mathbf{v}_i^{(k)} = \mathbf{v}_i^{(k-1)} + \rho \sum_{j\in\mathcal{V}_i} (\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}). \tag{5.17}$$

where $\rho > 0$ is the penalty parameter.

Since $r_i(\cdot)$, $i = 1, \dots, N$, are non-smooth, the minimization problem in (5.36) can be solved by employing appropriate subgradients or proximal operators [103, 104]. However, computing the conjugate function of the regularizers in (5.36) may be hard. To overcome this challenge, in the next subsection, we describe a new procedure that does not require the explicit calculation of any conjugate function.

ADMM with no Conjugate Function

In order to solve the problem in (5.36), we need to calculate the conjugate function of r_i^*. This can be difficult, especially for non-smooth functions. We exploit the Fenchel-Moreau theorem to eliminate the computation of conjugate function.

To that end, the problem in (5.36) can be restated as

$$\begin{aligned} \min_{\{\boldsymbol{\mu}_i, \boldsymbol{\nu}_i\}} \quad & \frac{g^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{c}_i^{(k-1)} + \bar{\rho}_i \|\boldsymbol{\mu}_i\|_2^2 \\ \text{s. t.} \quad & \mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i = \mathbf{0} \end{aligned} \tag{5.18}$$

where

$$\mathbf{c}_i^{(k-1)} = \mathbf{v}_i^{(k-1)} - \rho|\mathcal{V}_i|\boldsymbol{\mu}_i^{(k-1)} - \rho \sum_{j\in\mathcal{V}_i} \boldsymbol{\mu}_j^{(k-1)}$$

and $\bar{\rho}_i = \rho|\mathcal{V}_i|$. The Lagrangian function for (5.18) is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \boldsymbol{\theta}_i) =& \frac{g^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{c}_i^{(k-1)} \\ &+ \bar{\rho}_i \|\boldsymbol{\mu}_i\|_2^2 + \boldsymbol{\theta}_i^{\mathsf{T}}(\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i) \end{aligned} \tag{5.19}$$

where $\boldsymbol{\theta}_i$ is the Lagrange multiplier vector associated with the constraint in (5.18).

Hence, the dual function for the objective in (5.18) can be expressed as

$$\begin{aligned}\delta(\boldsymbol{\theta}_i) &= \inf_{\{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i\}} \mathcal{L}(\boldsymbol{\mu}_i,\boldsymbol{\nu}_i,\boldsymbol{\theta}_i)\\ &= \inf_{\boldsymbol{\nu}_i}\{r_i^*(\boldsymbol{\nu}_i)+\boldsymbol{\theta}_i^{\mathsf{T}}\boldsymbol{\nu}_i\}\\ &+\inf_{\boldsymbol{\mu}_i}\left\{\frac{g^*(\boldsymbol{\mu}_i)+\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N}+(\mathbf{c}_i^{(k-1)}+\mathbf{A}_i\boldsymbol{\theta}_i)^{\mathsf{T}}\boldsymbol{\mu}_i+\bar{\rho}_i\left\|\boldsymbol{\mu}_i\right\|^2\right\}\\ &= -\,r_i^{**}(-\boldsymbol{\theta}_i)\\ &+\inf_{\boldsymbol{\mu}_i}\left\{\frac{g^*(\boldsymbol{\mu}_i)}{N}+\left(\mathbf{c}_i^{(k-1)}+\mathbf{A}_i\boldsymbol{\theta}_i+\frac{\mathbf{b}}{N}\right)^{\mathsf{T}}\boldsymbol{\mu}_i+\bar{\rho}_i\left\|\boldsymbol{\mu}_i\right\|^2\right\}\end{aligned} \tag{5.20}$$

where the last equality follows from the definition of conjugate function.

For $g(\cdot)=\|\cdot\|^2$, the conjugate function is given by $g^*(\boldsymbol{\mu}_i)=\|\boldsymbol{\mu}_i\|^2/4$. Thus, the optimal value of second infimum of the dual function in (5.20) is

$$\frac{-1}{4\rho|\mathcal{V}_i|+\frac{1}{N}}\left\|\mathbf{A}_i\boldsymbol{\theta}_i+\mathbf{c}_i^{(k-1)}+\frac{\mathbf{b}}{N}\right\|^2$$

and the infimum is attained at the optimal point

$$\boldsymbol{\mu}_i^o=\frac{-2}{4\rho|\mathcal{V}_i|+\frac{1}{N}}\left(\mathbf{A}_i\boldsymbol{\theta}_i^o+\mathbf{c}_i^{(k-1)}+\frac{\mathbf{b}}{N}\right) \tag{5.21}$$

where $\boldsymbol{\theta}_i^o=\arg\max_{\boldsymbol{\theta}_i}\delta(\boldsymbol{\theta}_i)$. Since $r_i(\cdot)$ is convex, proper, and lower semi-continuous, we have $r_i^{**}=r_i$ due to the Fenchel-Moreau theorem [105]. Therefore, the dual function is given by

$$\delta(\boldsymbol{\theta}_i)=-r_i(-\boldsymbol{\theta}_i)-\frac{1}{4\rho|\mathcal{V}_i|+\frac{1}{N}}\left\|\mathbf{A}_i\boldsymbol{\theta}_i+\mathbf{c}_i^{(k-1)}+\frac{\mathbf{b}}{N}\right\|^2. \tag{5.22}$$

Using (5.21) and (5.22), the ADMM steps in (5.36) and (5.37) can be equivalently expressed as

$$\boldsymbol{\theta}_i^{(k)}=\arg\min_{\boldsymbol{\theta}_i}\left\{r_i(-\boldsymbol{\theta}_i)+\frac{1}{4\rho|\mathcal{V}_i|+\frac{1}{N}}\left\|\mathbf{A}_i\boldsymbol{\theta}_i+\mathbf{c}_i^{(k-1)}+\frac{\mathbf{b}}{N}\right\|^2\right\} \tag{5.23}$$

$$\boldsymbol{\mu}_i^{(k)}=\frac{-2}{4\rho|\mathcal{V}_i|+\frac{1}{N}}\left(\mathbf{A}_i\boldsymbol{\theta}_i^{(k)}+\mathbf{c}_i^{(k-1)}+\frac{\mathbf{b}}{N}\right) \tag{5.24}$$

$$\mathbf{v}_i^{(k)}=\mathbf{v}_i^{(k-1)}+\rho\sum_{j\in\mathcal{V}_i}(\boldsymbol{\mu}_i^{(k)}-\boldsymbol{\mu}_j^{(k)}) \tag{5.25}$$

$$\mathbf{c}_i^{(k)}=\mathbf{v}_i^{(k)}-\rho|\mathcal{V}_i|\boldsymbol{\mu}_i^{(k)}-\rho\sum_{j\in\mathcal{V}_i}\boldsymbol{\mu}_j^{(k)}. \tag{5.26}$$

Algorithm 6 Proposed algorithm for feature-partitioned distributed learning

```
At all agents $i \in \mathcal{V}$, initialize $\boldsymbol{\mu}_i^{(0)} = \mathbf{0}$, $\mathbf{v}_i^{(0)} = \mathbf{0}$, and locally run:
for $k = 1, 2, \ldots, K$ do
    Update $\boldsymbol{\theta}_i^{(k)}$ via (5.23).
    Update the dual variables $\boldsymbol{\mu}_i^{(k)}$ via (5.24).
    Share $\boldsymbol{\mu}_i^{(k)}$ with the neighbors in $\mathcal{V}_i$.
    Update the Lagrange multipliers $\mathbf{v}_i^{(k)}$ via (5.25).
    Update the auxiliary variables $\mathbf{c}_i^{(k)}$ via (5.26).
end for
```

The proposed algorithm is summarized in Algorithm 6. Note that the minimization problem in (5.23) can be solved using standard optimization techniques, or alternatively, subgradient-based algorithms [6]. Regardless of the technique used to solve (5.23), the proposed algorithm converges according to [6, Section 3.6.2]. Convergence of Algorithm 6 follows from [11, Proposition 2] and [106]. Moreover, due to the strong duality theorem, we have $\boldsymbol{\theta}_i^o = \mathbf{x}_i^o$, i.e., the optimal dual variable $\boldsymbol{\theta}_i^o$ at agent i is the optimal estimate $\mathbf{x}_i^o$ [81].

5.2.5 Simulations

To illustrate the performance of the proposed algorithm, we consider the elastic-net regression problem [107] and benchmark the proposed algorithm against a broadcast-based algorithm for learning with distributed features [5]. The only existing work considering non-smooth distributed learning with feature partitioning over general graphs is [5]. Therefore, we compared our algorithm only with this algorithm to provide a comparison that is as fair as possible. In a centralized setting, the optimal solution $\mathbf{x}^c$ is obtained as

$$\mathbf{x}^c = \arg\min_{\mathbf{x}} \ \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \eta_1 \|\mathbf{x}\|_1 + \eta_2 \|\mathbf{x}\|_2^2 \tag{5.27}$$

where

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]$$
$$\mathbf{x} = \left[\mathbf{x}_1^{\mathsf{T}}, \mathbf{x}_2^{\mathsf{T}}, \ldots, \mathbf{x}_N^{\mathsf{T}}\right]^{\mathsf{T}},$$

and $\eta_1 \in \mathbb{R}^+$ and $\eta_2 \in \mathbb{R}^+$ are the regularization parameters. In the distributed setting, we solve the problem (5.29) with

$$g(\mathbf{x}_i) = \left\| \sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_i - \mathbf{b} \right\|^2,$$
$$r_i(\mathbf{x}_i) = \eta_1 \|\mathbf{x}_i\|_1 + \eta_2 \|\mathbf{x}_i\|^2.$$

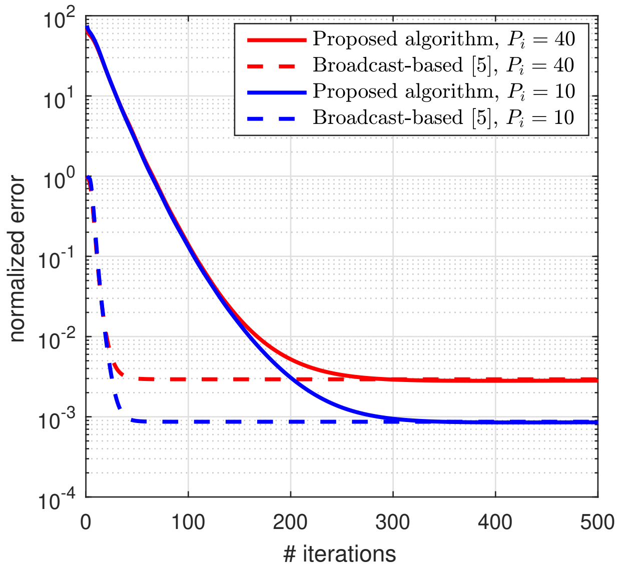


Figure 5.5: Normalized error of the proposed algorithm and the broadcast-based algorithm of [5] with $N = 20$ agents and different values of P_i.

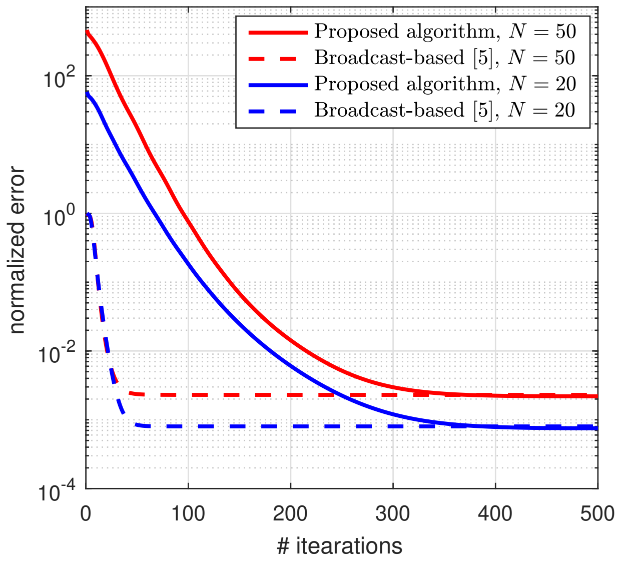


Figure 5.6: Normalized error of the proposed algorithm and the broadcast-based algorithm of [5] with $P_i = 10$ and different values of N.

We test the proposed algorithm on a multi-agent network with a random topology, where each agent links to three other agents on average. For each agent $i \in \mathcal{V}$, we create an $M \times P_i$ local observation matrix $\mathbf{A}_i$ whose entries are independent identically distributed Gaussian random variables with zero mean and unit variance. The response vector $\mathbf{b}$ is obtained as

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\phi}$$

where $\boldsymbol{\omega} \in \mathbb{R}^P$, $P = \sum_{i=1}^{N} P_i$, and $\boldsymbol{\phi} \in \mathbb{R}^M$ are drawn from the distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$, respectively. The regularization parameters are set to $\eta_1 = \eta_2 = 1$ and penalty parameter to $\rho = 1$. The performance of the proposed algorithm is evaluated using the normalized error between the centralized solution $\mathbf{x}^c$ as per (5.27) and the solution from Algorithm 6 at iteration k denoted by

$$\mathbf{x}^d(k) = \left[(\mathbf{x}_1^{(k)})^\mathsf{T}, \ldots, (\mathbf{x}_N^{(k)})^\mathsf{T}\right]^\mathsf{T}.$$

The normalized error is defined as

$$\frac{\left\|\mathbf{x}^d(k) - \mathbf{x}^c\right\|^2}{\left\|\mathbf{x}^c\right\|^2}.$$

The centralized solution $\mathbf{x}^c$ is computed using the optimization toolbox CVX [89]. Results are obtained by averaging over 100 independent trials. The parameters have been tuned to achieve the best performance in terms of accuracy and convergence rate while benefiting from the convergence properties of the ADMM illustrated in [5].

Fig. 5.5 shows that, for $N = 20$ agents, the proposed algorithm converges when the number of parameters at the ith agent is $P_i = 10$ and $P_i = 40$, $\forall i \in \mathcal{V}$. Fig. 5.6 shows that the proposed algorithm converges when $P_i = 10$ and the network consists of 20 or 50 agents. The faster convergence of the broadcast-based algorithm of [5] is due to its centralized processing.

5.2.6 Conclusion

In [39], we developed a fully-distributed algorithm for learning with non-smooth regularization functions under distributed features. We reformulated the underlying problem into an equivalent dual form and used the ADMM to solve it in a distributed fashion without using any conjugate function. To the best of our knowledge, the proposed algorithm is the first of its kind that solves the feature-distributed learning problems with non-smooth regularizer functions over arbitrary graphs while not relying on any conjugate function. We verified the convergence of the proposed algorithm at all agents via simulation results.

5.3 Decentralized Optimization with Distributed Features and Non-Smooth Objective Functions

In [1], we develop a new consensus-based distributed algorithm for solving learning problems with feature partitioning and non-smooth convex objective functions. Such learning problems are not separable, i.e., the associated objective functions cannot be directly written as a summation of agent-specific objective functions. To overcome this challenge, we redefine the underlying optimization problem as a dual convex problem whose structure is suitable for distributed optimization using the alternating direction method of multipliers (ADMM). Next, we propose a new method to solve the minimization problem associated with the ADMM update step that does not rely on any conjugate function. Calculating the relevant conjugate functions may be hard or even unfeasible, especially when the objective function is non-smooth. To obviate computing any conjugate function, we solve the optimization problem associated with each ADMM iteration in the dual domain utilizing the block coordinate descent algorithm. Unlike the existing related algorithms, the proposed algorithm is fully distributed and does away with the conjugate of the objective function. We prove theoretically that the proposed algorithm attains the optimal centralized solution. We also confirm its network-wide convergence via simulations.

5.3.1 Related Work

Learning problems with feature partitioning of data have been considered in [5, 15, 22, 35–39, 42–51]. The algorithms proposed in [36, 37] solve the basis pursuit and lasso problems, respectively. The work of [38] assumes an appropriate coloring scheme of the network and cannot be extended to a general graph labeling.

The algorithms proposed in [5, 42–45] are not fully distributed since their consensus constraints involve the entire network instead of each agent's local neighborhood. Furthermore, the algorithms proposed in [42, 44] only solve the ridge regression problem, while the works of [43, 45] assume the cost function to be convex and smooth with Lipschitz-continuous gradient. Both algorithms proposed in [43, 45] can only be used for minimization problems with ℓ_2-norm regularization (ridge penalty) and rely on the computation of the conjugate of the cost function.

The algorithms in [15, 35, 46, 47] are based on the diffusion strategy, which is suitable when stochastic gradients are available. Furthermore, the work in [15] assumes that the cost function is convex and smooth with Lipschitz-continuous gradient. The algorithm developed in [46] relies on the calculation of the relevant conjugate functions. In addition, it assumes that the cost function is convex and smooth, and the regularization functions are strongly convex. The work of [47] also assumes that the regularizer functions are smooth and strongly convex. Moreover, it relies on the computation of conjugate functions similar to the algorithms proposed in [48, 49]. The diffusion-based algorithm proposed in [35] only solves the ridge regression problem. The consensus-based algorithm of [22] is also designed for ridge regression with feature partitioning. It outperforms the algorithm proposed in [35] in terms of convergence speed.

The algorithm proposed in [39] is designed for an ℓ_2-norm-square cost function and hence cannot be extended to general objective functions. The works of [52–55] consider distributed agent-specific parameter estimation problem. However, the objective functions considered in these works are smooth. The authors of [50] propose a distributed coordinate-descent algorithm to reduce the communication cost in distributed learning with feature partitioning. However, the cost function in [50] is assumed to be strongly convex and smooth. The work of [51] considers an asynchronous stochastic gradient-descent algorithm for learning with distributed features. However, the objective function in [51] is assumed to be smooth.

None of the above-mentioned existing algorithms for distributed learning with feature partitioning is designed for optimizing generic non-smooth objective functions over arbitrary graphs without using or computing any conjugate function.

5.3.2 Contributions

In [1], we develop a new fully-distributed algorithm for solving learning problems when the data is distributed among agents in feature partitions and computing the conjugate of the possibly non-smooth cost or regularizer functions is challenging or unfeasible. We consider a general regularized non-smooth learning problem whose cost function cannot be written as the sum of local agent-specific cost functions, i.e., it is not separable as in (5.29) ahead.

To tackle the problem, we articulate the associated dual optimization problem and utilize the alternating direction method of multipliers (ADMM) to solve it as, unlike the original problem, its structure is suitable for distributed treatment via the ADMM. We then consider the dual of the optimization problem associated with the ADMM update step and solve it via the block coordinate-descent (BCD) algorithm. In that manner, we devise an approach that enables us to avoid the explicit computation of any conjugate function, which may be hard or infeasible for some objective functions. The proposed algorithm is fully distributed, i.e., it only relies on single-hop communications among neighboring agents and does not need any central coordinator or processing hub. We demonstrate that the proposed algorithm approaches the optimal centralized solution at all agents. Our experiments show that the proposed algorithm converges to the optimal solution in various scenarios and is competitive with the relevant existing algorithms even when dealing with problems that, unlike its contenders, it is not tailored for.

5.3.3 System Model

We consider the system model described in Section 2.1 with feature partitioning of data as presented in Section 2.1.2. We consider a regularized learning problem consisting in minimizing a global cost function $g(\cdot)$ that is a function of the error $\mathbf{Ax}-\mathbf{b}$ and is added by a regularization function $r(\cdot)$. In the centralized approach, the optimal solution is given by

$$\mathbf{x}^o = \arg\min_{\mathbf{x}} \Big\{ g\left(\mathbf{Ax}-\mathbf{b}\right) + r(\mathbf{x}) \Big\}. \tag{5.28}$$

Considering feature partitioning of the data, $\mathbf{Ax}$ can be written as

$$\mathbf{Ax} = \sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_i$$

and assuming that the regularizer function $r(\cdot)$ can be written as a sum of agent-specific regularizer functions as

$$r(\mathbf{x}) = \sum_{i=1}^{N} r_i(\mathbf{x}_i),$$

the regularized learning problem (5.28) is of the following form

$$\min_{\{\mathbf{x}_i\}} \quad g\left(\textstyle\sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_i - \mathbf{b}\right) + \sum_{i=1}^{N} r_i(\mathbf{x}_i). \tag{5.29}$$

The learning problem (5.29) pertains to several applications in machine learning, e.g., regression over distributed features [5], clustering in graphs [100], smart grid control [101], dictionary learning [46], and network utility maximization [102]. Similar to most existing works, e.g., [5, 15, 108], we consider learning problems where functions $g(\cdot)$ and $r_i(\cdot)$, $i = 1, \dots, N$, are convex, proper, and lower semi-continuous. However, in this work, the objective functions are not necessarily smooth or their conjugate functions known. Therefore, we propose a novel algorithm that solves (5.29) in a fully distributed fashion wherein each agent communicates only with its neighbors without requiring the computation of any conjugate function. In the next section, we describe our proposed algorithm.

5.3.4 Algorithm

We first present the reformulation of the considered non-separable problem into a dual form that can be solved in a fully-distributed fashion via the ADMM. Subsequently, we describe a new approach to perform the ADMM primal update step without explicitly computing any conjugate function of the cost or regularizer functions.

Distributed ADMM for the Dual Problem

To develop a distributed solution, we introduce the auxiliary variables $\{\mathbf{z}_i\}_{i=1}^N$ and recast (5.29) as

$$\begin{aligned} \min_{\{\mathbf{x}_i,\mathbf{z}_i\}} \quad & g\left(\textstyle\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}\right) + \sum_{i=1}^N r_i(\mathbf{x}_i) \\ \text{s. t.} \quad & \mathbf{A}_i\mathbf{x}_i = \mathbf{z}_i, \quad i = 1, \ldots, N. \end{aligned} \tag{5.30}$$

The cost function $g(\cdot)$ in (5.30) is not separable among the agents. We consider the dual problem of (5.30) and exploit its separability property, which is lacking in the primal domain, to solve it by employing the ADMM. For this purpose, we associate the Lagrange multipliers $\{\boldsymbol{\mu}_i\}_{i=1}^N$ with the equality constraints in (5.30) and state the related Lagrangian function as

$$\begin{aligned} & \mathcal{L}(\{\mathbf{x}_i\}, \{\mathbf{z}_i\}, \{\boldsymbol{\mu}_i\}) \\ =& g\left(\textstyle\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}\right) + \sum_{i=1}^N r_i(\mathbf{x}_i) + \sum_{i=1}^N \boldsymbol{\mu}_i^{\mathsf{T}}(\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i) \\ =& \sum_{i=1}^N \left(r_i(\mathbf{x}_i) + (\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i)^{\mathsf{T}}\mathbf{x}_i\right) \\ & + g\left(\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}\right) - \sum_{i=1}^N \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{z}_i. \end{aligned} \tag{5.31}$$

The dual function for problem (5.30) can be computed as

$$\begin{aligned} d(\{\boldsymbol{\mu}_i\}) =& \inf_{\{\mathbf{x}_i,\mathbf{z}_i\}} \mathcal{L}(\{\mathbf{x}_i\}, \{\mathbf{z}_i\}, \{\boldsymbol{\mu}_i\}) \\ =& -\sum_{i=1}^N r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) + \inf_{\mathbf{z}_i} g(\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}) - \sum_{i=1}^N \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{z}_i \end{aligned} \tag{5.32}$$

where r_i^* is the conjugate function of r defined as

$$r_i^*(\mathbf{y}) = \sup_{\mathbf{x}} \ \mathbf{y}^T\mathbf{x} - r_i(\mathbf{x}).$$

Introducing auxiliary variable $\mathbf{z}$ that is defined as

$$\mathbf{z} = \sum_{i=1}^{N} \mathbf{z}_i$$

and using the duality theory, an alternate form of the dual function (5.32) is given by

$$\tilde{d}(\{\boldsymbol{\mu}_i\}, \boldsymbol{\lambda}) = -g^*(\boldsymbol{\lambda}) - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b} - \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \tag{5.33}$$

when $\boldsymbol{\lambda} = \boldsymbol{\mu}_i \ \forall i \in \mathcal{V}$ with $\boldsymbol{\lambda}$ being the dual variable corresponding to $\mathbf{z} = \sum_{i=1}^{N} \mathbf{z}_i$. Otherwise, we have $\tilde{d}(\{\boldsymbol{\mu}_i\}, \boldsymbol{\lambda}) = -\infty$.

By eliminating $\boldsymbol{\lambda}$, the dual problem for (5.30) can be expressed as

$$\begin{aligned} \min_{\{\boldsymbol{\mu}_i\}} \quad & \frac{1}{N}\sum_{i=1}^{N}\Big(g^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}\Big) + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ \text{s. t.} \quad & \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \cdots = \boldsymbol{\mu}_N. \end{aligned} \tag{5.34}$$

To facilitate a fully-distributed solution, we decouple the constraints in (5.34) as

$$\boldsymbol{\mu}_i = \mathbf{u}_i^j, \quad \boldsymbol{\mu}_j = \mathbf{u}_i^j,\ j \in \mathcal{V}_i,\ i = 1, \ldots, N \tag{5.35}$$

where $\{\mathbf{u}_i^j\}_{i\in\mathcal{V}, j\in\mathcal{V}_i}$ are auxiliary variables that will eventually be eliminated. For further details on the reformulation of consensus throughout the whole network into localized consensus in the neighborhoods, the reader can refer to the reformulation of (5.13) into (5.15) via (5.14) in 5.2.4. Regarding the elimination for auxiliary variables, the reader can refer to the steps illustrated in Section 2.2.2.

We generate a new augmented Lagrangian function by associating the new Lagrange multipliers $\{\bar{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ and $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ with the consensus constraints in (5.35). By using the Karush-Kuhn-Tucker conditions of optimality for (5.35) and setting

$$\mathbf{v}_i^{(k)} = 2\sum_{j\in\mathcal{V}_i} (\bar{\mathbf{v}}_i^j)^{(k)},$$

it can be shown that the Lagrange multipliers $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ and the auxiliary variables $\{\mathbf{u}_i^j\}_{j\in\mathcal{V}_i}$ are eliminated [7]. Hence, the ADMM to solve (5.34) reduces to the following iterative updates at the ith agent

$$\begin{aligned}\boldsymbol{\mu}_i^{(k)} = \arg\min_{\boldsymbol{\mu}_i} \Big\{ & \frac{1}{N} g^*(\boldsymbol{\mu}_i) + \frac{1}{N}\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b} + r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ & + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{v}_i^{(k-1)} + \rho \sum_{j\in\mathcal{V}_i} \left\| \boldsymbol{\mu}_i - \frac{\boldsymbol{\mu}_i^{(k-1)} + \boldsymbol{\mu}_j^{(k-1)}}{2} \right\|^2 \Big\} \end{aligned} \tag{5.36}$$

$$\mathbf{v}_i^{(k)} = \mathbf{v}_i^{(k-1)} + \rho \sum_{j\in\mathcal{V}_i} (\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}) \tag{5.37}$$

where $\rho > 0$ is the penalty parameter and k is the iteration index.

The objective function in (5.36) may be non-smooth as the global cost function $g(\cdot)$ or the agent-specific regularizer functions $r_i(\cdot)$, $i = 1, \ldots, N$, and consequently their conjugate functions may be non-smooth. Thus, the minimization problem in (5.36) can be solved by employing suitable subgradient methods or proximal operators [103, 104]. However, computing the conjugate functions of the cost or the regularizer functions in (5.36) may be hard or even unfeasible. To overcome this challenge, in the next subsection, we describe a new approach that does not require the explicit calculation of any conjugate function.

ADMM without Conjugate Function

We rewrite the minimization problem in the ADMM primal update (5.36) as

$$\begin{aligned} \boldsymbol{\mu}_i^{(k)} = \arg \min_{\{\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \boldsymbol{\alpha}_i\}} & \Big\{ \frac{g^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{c}_i^{(k-1)} + \bar{\rho}_i \|\boldsymbol{\alpha}_i\|^2 \Big\} \\ \text{s.t.} \quad & \mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i = \mathbf{0} \\ & \boldsymbol{\mu}_i = \boldsymbol{\alpha}_i \end{aligned} \tag{5.38}$$

where $\bar{\rho}_i = \rho|\mathcal{V}_i|$ and

$$\mathbf{c}_i^{(k-1)} = \mathbf{v}_i^{(k-1)} - \rho|\mathcal{V}_i|\boldsymbol{\mu}_i^{(k-1)} - \rho \sum_{j\in\mathcal{V}_i} \boldsymbol{\mu}_j^{(k-1)}. \tag{5.39}$$

The Lagrangian function related to (5.38) is stated as

$$\begin{aligned} \mathcal{L}_k(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \boldsymbol{\alpha}_i, \boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}) = & \frac{g^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{c}_i^{(k-1)} \\ & + \bar{\rho}_i \|\boldsymbol{\alpha}_i\|^2 + (\boldsymbol{\theta}_i^{(k)})^{\mathsf{T}}(\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i) \\ & + (\boldsymbol{\beta}_i^{(k)})^{\mathsf{T}}(\boldsymbol{\mu}_i - \boldsymbol{\alpha}_i) \end{aligned} \tag{5.40}$$

where $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$ are the Lagrange multipliers associated with the first and the second constraints in (5.38), respectively, at iteration k.

Motivated by the close connection between a function and its double conjugate (conjugate of conjugate), we express the dual for the objective in (5.38) as

$$\begin{aligned}\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}) &= \inf_{\{\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \boldsymbol{\alpha}_i\}} \mathcal{L}_k(\boldsymbol{\mu}_i, \boldsymbol{\nu}_i, \boldsymbol{\alpha}_i, \boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}) \\ &= \inf_{\boldsymbol{\nu}_i} \{ r_i^*(\boldsymbol{\nu}_i) + (\boldsymbol{\theta}_i^{(k)})^\mathsf{T} \boldsymbol{\nu}_i \} + \inf_{\boldsymbol{\alpha}_i} \{ \bar{\rho}_i \left\| \boldsymbol{\alpha}_i \right\|^2 - (\boldsymbol{\beta}_i^{(k)})^\mathsf{T} \boldsymbol{\alpha}_i \} \\ &+ \inf_{\boldsymbol{\mu}_i} \Big\{ \frac{g^*(\boldsymbol{\mu}_i)}{N} + \left(\mathbf{c}_i^{(k-1)} + \mathbf{A}_i \boldsymbol{\theta}_i^{(k)} + \frac{\mathbf{b}}{N} + \boldsymbol{\beta}_i^{(k)} \right)^\mathsf{T} \boldsymbol{\mu}_i \Big\}.\end{aligned} \tag{5.41}$$

By employing the definition of conjugate function, the first infimum in (5.41) is equal to $-r_i^{**}(-\boldsymbol{\theta}_i^{(k)})$. The second infimum in (5.41) can be easily obtained by noting that the function

$$l_k(\boldsymbol{\alpha}_i) \coloneqq \bar{\rho}_i \left\| \boldsymbol{\alpha}_i \right\|^2 - (\boldsymbol{\beta}_i^{(k)})^\mathsf{T} \boldsymbol{\alpha}_i$$

is quadratic in $\boldsymbol{\alpha}_i$. Hence, this infimum can be calculated by computing the gradient of $l_k(\cdot)$ and equating it to zero, i.e.,

$$\bar{\rho}_i \left\| \boldsymbol{\alpha}_i \right\|^2 - (\boldsymbol{\beta}_i^{(k)})^\mathsf{T} \boldsymbol{\alpha}_i = 0.$$

Solving this equation for $\boldsymbol{\alpha}_i$ gives

$$\boldsymbol{\alpha}_i^o = \frac{\boldsymbol{\beta}_i^{(k)}}{2\bar{\rho}_i}. \tag{5.42}$$

This implies that the second infimum in (5.41) is attained at the optimal value $\boldsymbol{\alpha}_i^o$, which in turn means that it is equal to

$$l_k(\boldsymbol{\alpha}_i^o) = -\frac{\left\| \boldsymbol{\beta}_i^{(k)} \right\|^2}{4\bar{\rho}_i}.$$

In view of the definition and properties of the conjugate function [81], the third infimum in (5.41) is given by

$$-Ng^{**}\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i \boldsymbol{\theta}_i^{(k)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)} \right).$$

Therefore, we have

$$\begin{aligned}\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}) = &- r_i^{**}(-\boldsymbol{\theta}_i^{(k)}) - \frac{\left\| \boldsymbol{\beta}_i^{(k)} \right\|^2}{4\bar{\rho}_i} \\ &- Ng^{**}\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i \boldsymbol{\theta}_i^{(k)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)} \right).\end{aligned} \tag{5.43}$$

Since $g(\cdot)$ and $r_i(\cdot)$ are convex, proper, and lower semi-continuous, we know $g^{**} = g$ and $r_i^{**} = r_i$ due to the Fenchel Moreau Theorem [109]. Therefore, we have

$$\begin{aligned}\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}) = &- r_i(-\boldsymbol{\theta}_i^{(k)}) - \frac{\left\|\boldsymbol{\beta}_i^{(k)}\right\|^2}{4\bar{\rho}_i} \\ &- Ng\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(k)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)}\right).\end{aligned} \tag{5.44}$$

To find the optimal $(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$, we need to maximize $\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$ or, equivalently, to minimize $-\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$. Since this is a function of two variables $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$, we employ the block coordinate descent algorithm (BCD) to minimize $-\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$ and find the optimal values for $(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$. The BCD steps are obtained by alternatively minimizing $-\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$ with respect to $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$ as follows

$$\begin{aligned}\boldsymbol{\theta}_i^{(k,t)} = \arg\min_{\boldsymbol{\theta}_i^{(k)}} &\Big\{ r_i(-\boldsymbol{\theta}_i^{(k)}) \\ &+ Ng\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(k,t)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k,t-1)}\right)\Big\}\end{aligned} \tag{5.45}$$

$$\begin{aligned}\boldsymbol{\beta}_i^{(k,t)} = \arg\min_{\boldsymbol{\beta}_i^{(k)}} &\Big\{ \frac{1}{4\bar{\rho}_i}\left\|\boldsymbol{\beta}_i^{(k)}\right\|^2 \\ &+ Ng\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(t)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)}\right)\Big\}\end{aligned} \tag{5.46}$$

where t is the BCD iteration index. If we assume that the BCD algorithm converges after T iterations. The optimal values of $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$ can be denoted by $\boldsymbol{\theta}_i^{(k,T)}$ and $\boldsymbol{\beta}_i^{(k,T)}$, respectively.

To update the Lagrange multipliers $\boldsymbol{\mu}_i^{(k)}$ we employ the complementary slackness conditions, i.e.,

$$\boldsymbol{\beta}_i^{(k,T)}(\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\alpha}_i^o) = \mathbf{0} \quad \forall i \in \mathcal{V}.$$

Since $\boldsymbol{\beta}_i^{(k,T)} \neq \mathbf{0}$, $\forall i \in \mathcal{V}$, we have

$$\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\alpha}_i^o = \mathbf{0} \quad \forall i \in \mathcal{V}.$$

Using (5.42), we can update $\boldsymbol{\mu}_i^{(k)}$ as

$$\boldsymbol{\mu}_i^{(k)} = \frac{\boldsymbol{\beta}_i^{(k,T)}}{2\bar{\rho}_i}. \tag{5.47}$$

Algorithm 7 The proposed algorithm for feature-partitioned distributed learning with unknown conjugate functions

```
At all agents $i \in \mathcal{V}$, initialize $\boldsymbol{\mu}_i^{(0)} = \mathbf{0}$, $\mathbf{v}_i^{(0)} = \mathbf{0}$, and locally run:
for $k = 1, 2, \ldots, K$ do
    Run BCD loop
    for $t = 1, 2, \ldots, T$ do
        Update $\boldsymbol{\theta}_i^{(k,t)}$ via (5.45).
        Update $\boldsymbol{\beta}_i^{(k,t)}$ via (5.46).
    end for
    Update the dual variables $\boldsymbol{\mu}_i^{(k)} = \boldsymbol{\beta}_i^{(k,T)}/(2\bar{\rho}_i)$.
    Share $\boldsymbol{\mu}_i^{(k)}$ with the neighbors in $\mathcal{V}_i$.
    Update the Lagrange multipliers $\mathbf{v}_i^{(k)}$ via (5.49).
    Update the auxiliary variables $\mathbf{c}_i^{(k)}$ via (5.50).
end for
```

Collating the expressions in (5.47), (5.39), (5.37), the ADMM steps in (5.36) and (5.37) can be equivalently expressed as

$$\boldsymbol{\mu}_i^{(k)} = \frac{\boldsymbol{\beta}_i^{(k,T)}}{2\bar{\rho}_i} \tag{5.48}$$

$$\mathbf{v}_i^{(k)} = \mathbf{v}_i^{(k-1)} + \rho \sum_{j \in \mathcal{V}_i} (\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}) \tag{5.49}$$

$$\mathbf{c}_i^{(k-1)} = \mathbf{v}_i^{(k-1)} - \rho |\mathcal{V}_i| \boldsymbol{\mu}_i^{(k-1)} - \rho \sum_{j \in \mathcal{V}_i} \boldsymbol{\mu}_j^{(k-1)} \tag{5.50}$$

where k is the ADMM iteration index. We summarize the proposed algorithm in Algorithm 7.

Assuming that the ADMM outer loop converges after K iterations, we denote the optimal dual variable $\boldsymbol{\theta}_i^{(K,T)}$ by $\boldsymbol{\theta}_i^o$. The estimate $\boldsymbol{\theta}_i^o$ at agent i is indeed the optimal solution to the original problem (5.29), i.e., $\mathbf{x}_i^o$, as per the following theorem.

Theorem 5.1. For all agents $i \in \mathcal{V}$, the optimal dual variable $\boldsymbol{\theta}_i^o$ at agent i is equal to the optimal estimate $\mathbf{x}_i^o$, i.e., $\boldsymbol{\theta}_i^o = \mathbf{x}_i^o$.

Proof. Since the optimization problem in (5.30) has a convex objective and is feasible, the Slater's condition is satisfied. Therefore, due to the Slater's theorem, strong duality holds and $\boldsymbol{\theta}_i^o = \mathbf{x}_i^o$, $\forall i \in \mathcal{V}$ [81]. □

5.3.5 Convergence Analysis

The convergence of the proposed algorithm can be proven by corroborating that both the inner-loop BCD and outer-loop ADMM iterations converge. First, the convergence of the inner loop can be verified from results in [6] since all the assumptions required for the convergence are satisfied, i.e., the function $\delta(\cdot)$ is convex and the feasible sets $\mathbb{R}^M$ and $\mathbb{R}^{P_i}$, $\forall i \in \mathcal{V}$, are all convex. Assuming that the optimal solution $\boldsymbol{\beta}_i^o$ of the inner-loop BCD algorithm is attained for each $i \in \mathcal{V}$, the dual variable $\boldsymbol{\mu}_i^{(k)}$ in the outer loop is updated accordingly.

Next, we prove that the estimates produced by the fully-distributed ADMM outer loop, i.e., (5.36) and (5.37), approach the optimal centralized solution at all agents. To present the convergence result, we rewrite the constraints in (5.34) as follows

$$\boldsymbol{\mu}_i = \bar{\mathbf{u}}_i^j, \quad \boldsymbol{\mu}_j = \breve{\mathbf{u}}_i^j, \quad \bar{\mathbf{u}}_i^j = \breve{\mathbf{u}}_i^j, \quad j \in \mathcal{V}_i,\ i = 1, \ldots, N. \tag{5.51}$$

Note that the constraints $\mathbf{u} \in \mathcal{C}_{\mathbf{u}} := \{\mathbf{u} : \bar{\mathbf{u}}_i^j = \breve{\mathbf{u}}_i^j,\ i \in \mathcal{V},\ j \in \mathcal{N}_i\}$ are not dualized and are introduced only to present the convergence result. Let us define the following vectors

$$\begin{aligned}
\boldsymbol{\mu} =& [\boldsymbol{\mu}_1^\mathsf{T}, \ldots, \boldsymbol{\mu}_N^\mathsf{T}]^\mathsf{T} \\
\mathbf{u} =& [(\mathbf{u}_1^{a_1(1)})^\mathsf{T}, \ldots, (\mathbf{u}_1^{a_N(|\mathcal{V}_N|)})^\mathsf{T}, \\
& \ldots, (\mathbf{u}_N^{a_N(1)})^\mathsf{T}, \ldots, (\mathbf{u}_N^{a_N(|\mathcal{V}_N|)})]^\mathsf{T}
\end{aligned}$$

where $a_i(j)$ is the index of the jth neighbor of agent i.

The problem (5.35) with the constraints in (5.51) can be written as

$$\begin{aligned}
\min_{\boldsymbol{\mu}, \mathbf{u}} \quad & G_1(\boldsymbol{\mu}) + G_2(\mathbf{u}) \\
\text{s.t.} \quad & \boldsymbol{\mu} \in \mathcal{C}_1,\ \mathbf{u} \in \mathcal{C}_2,\ \mathbf{C}\boldsymbol{\mu} = \mathbf{u}
\end{aligned} \tag{5.52}$$

where $\mathbf{C} = [\mathbf{C}_1^\mathsf{T}, \mathbf{C}_2^\mathsf{T}]^\mathsf{T}$, $G_2(\mathbf{u}) = 0$, $\mathcal{C}_1 := \mathbb{R}^M$, $\mathcal{C}_2 := \mathcal{C}_{\mathbf{u}}$,

$$\mathbf{C}_1 = \begin{bmatrix} \mathbf{C}_{11} \\ \vdots \\ \mathbf{C}_{1N} \end{bmatrix}, \quad \mathbf{C}_{1i} := (\mathbf{1}_{|\mathcal{N}_i|}\mathbf{e}_i^\mathsf{T}) \otimes \mathbf{I}_M,\ i \in \mathcal{V}$$

$$\mathbf{C}_2 = \begin{bmatrix} \mathbf{C}_{21} \\ \vdots \\ \mathbf{C}_{2N} \end{bmatrix}, \quad \mathbf{C}_{2i} := \begin{bmatrix} \mathbf{e}_{i_i(1)}^\mathsf{T} \\ \vdots \\ \mathbf{e}_{i_i(|\mathcal{N}_i|)}^\mathsf{T} \end{bmatrix} \otimes \mathbf{I}_M,\ i \in \mathcal{V}$$

$$G_1(\boldsymbol{\mu}) = \frac{1}{N}\sum_{i=1}^{N}\Big(g^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{b}\Big) + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i),$$

and $\mathbf{e}_i$ is the ith vector of the canonical basis of $\mathbb{R}^M$. The convergence result relies on the following lemma.

Lemma 4. *If $\mathcal{G}$ is a connected graph, then the local optimal solution $\boldsymbol{\mu}_i^o$ at agent i is equal to the optimal centralized solution of* (5.35)*, i.e., $\boldsymbol{\mu}_i^o = \boldsymbol{\mu}^o$, $\forall i \in \mathcal{V}$, where*

$$\boldsymbol{\mu}^o = \arg\min_{\boldsymbol{\mu}} \Big\{ g^*(\boldsymbol{\mu}) + \boldsymbol{\mu}^\mathsf{T}\mathbf{b} + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}) \Big\}.$$

Proof. Let i and i' be arbitrary agents in $\mathcal{G}$ and $p(i,i') \,:\, i, i_1, i_2, \ldots, i_n, i'$ an arbitrary path on $\mathcal{G}$ that connects i and i'. Since the adjacent agents in $p(i,i')$ are neighbors, we have

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i_1} = \boldsymbol{\mu}_{i_2} = \ldots = \boldsymbol{\mu}_{i_n} = \boldsymbol{\mu}_{i'},$$

which imply

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i'}.$$

Since $\mathcal{G}$ is connected and the path is arbitrary, the local constraints $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i'}$ can be removed and replaced by the common constraint $\boldsymbol{\mu}_i = \boldsymbol{\mu}$. Hence, $\boldsymbol{\mu}_i^o = \boldsymbol{\mu}^o$ $\forall i \in \mathcal{V}$ where

$$\boldsymbol{\mu}^o = \arg\min_{\boldsymbol{\mu}} \Big\{ g^*(\boldsymbol{\mu}) + \boldsymbol{\mu}^\mathsf{T}\mathbf{b} + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}) \Big\}.$$

□

We can now prove the convergence of the proposed algorithm as per the following theorem.

Theorem 5.2. If $\mathcal{G}$ is a connected graph, then the proposed algorithm converges to the optimal centralized solution, i.e.,

$$\lim_{k\to\infty} \boldsymbol{\mu}_i^{(k)} = \boldsymbol{\mu}^o, \forall i \in \mathcal{V}. \tag{5.53}$$

Proof. Thanks to Lemma 1, we only need to prove that

$$\lim_{k\to\infty} \boldsymbol{\mu}_i^{(k)} = \boldsymbol{\mu}_i^o.$$

For this purpose, we observe that (5.52) is in the same form as [103, eq. 4.77, p. 255]. Furthermore, the following assumptions are satisfied:

- $G_1(\cdot)$ and $G_2(\cdot)$ are convex functions;
- $\mathcal{C}_1$ and $\mathcal{C}_2$ are nonempty polyhedral sets;
- $\mathbf{C}$ is full column rank, hence, $\mathbf{C}^\mathsf{T}\mathbf{C}$ is invertible.

Therefore, due to [103, Proposition 4.2, p. 256], we have

$$\lim_{k\to\infty} \boldsymbol{\mu}_i^{(k)} = \boldsymbol{\mu}^o, \forall i \in \mathcal{V}.$$

□

5.3.6 Simulations

In this section, we present some simulation results to evaluate the performance of the proposed algorithm. We first assess the proposed algorithm considering a distributed elastic-net regression problem with different numbers of local features, samples, and agents as well as different network topologies. Subsequently, we benchmark the proposed algorithm against the most relevant existing algorithms considering distributed ridge and lasso regression problems. The parameters are tuned to achieve the best performance in terms of accuracy and convergence rate while providing a comparison as fair as possible with the benchmark algorithms.

Distributed Elastic-Net Regression

To evaluate the performance of the proposed algorithm in different scenarios, we consider the elastic-net regression problem. The calculation of the conjugate function for the objective function corresponding to this problem is practically infeasible. In the distributed setting, we solve the elastic-net regression problem by considering

$$\begin{aligned} g(\mathbf{x}_i) &= \left\| \sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_i - \mathbf{b} \right\|^2 \\ r_i(\mathbf{x}_i) &= \eta_1 \|\mathbf{x}_i\|_1 + \eta_2 \|\mathbf{x}_i\|^2 \end{aligned} \tag{5.54}$$

where $\eta_1 \in \mathbb{R}^+$ and $\eta_2 \in \mathbb{R}^+$ are the regularization parameters. We calculate the response vector $\mathbf{b}$ as

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\psi} \tag{5.55}$$

where $\boldsymbol{\omega} \in \mathbb{R}^P$ and $\boldsymbol{\psi} \in \mathbb{R}^M$ are independently drawn from the multivariate normal distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$, respectively. We set the regularization parameters to $\eta_1 = 1$, $\eta_2 = 1$ and the penalty parameter to $\rho = 2$. We use two iterations in the inner-loop BCD algorithm. We obtain the results by averaging over 100 independent trials while considering a multi-agent network with

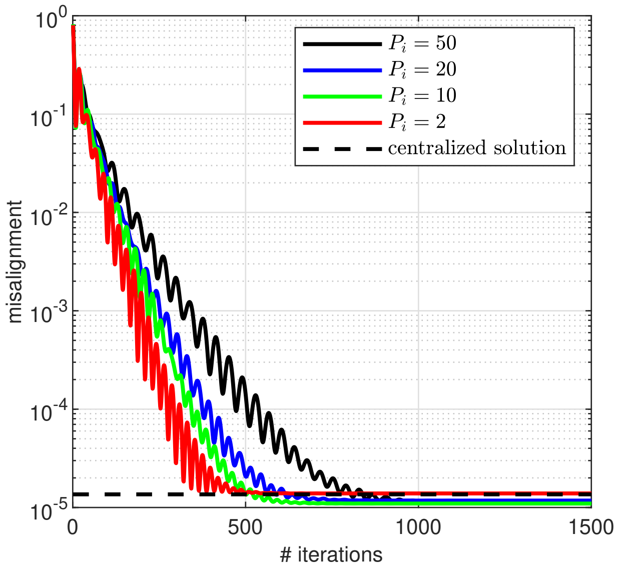


Figure 5.7: The misalignment of the proposed algorithm solving the distributed elastic-net regression problem with different values of P_i

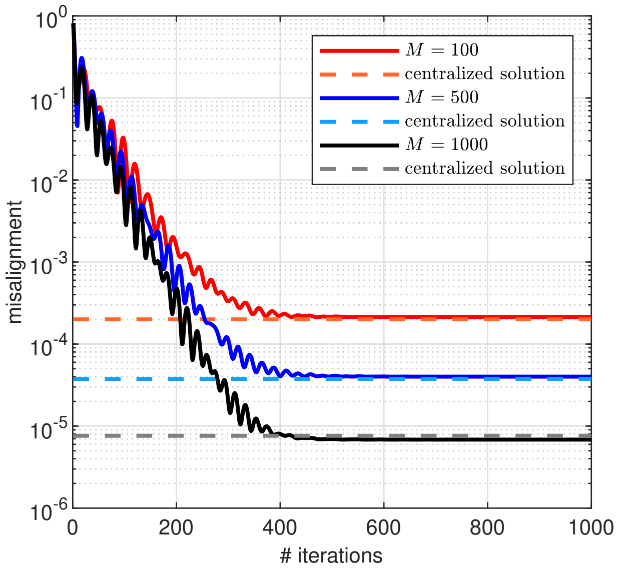


Figure 5.8: The misalignment of the proposed algorithm solving the distributed elastic-net regression problem with different values of M.

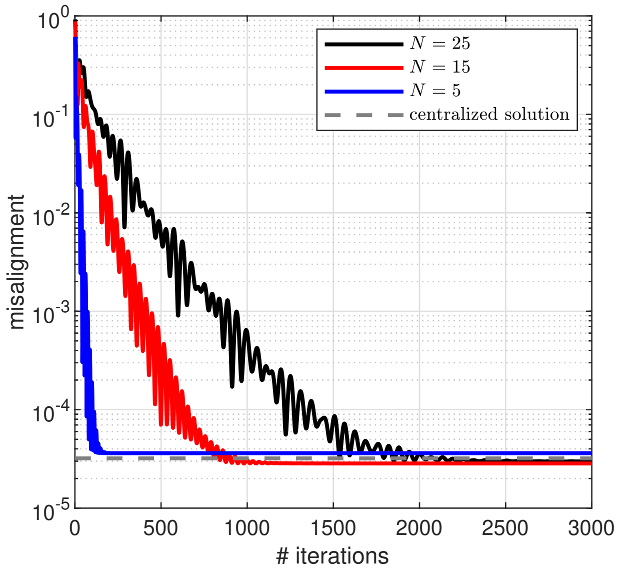


Figure 5.9: The misalignment of the proposed algorithm solving the distributed elastic-net regression problem with different values of N.

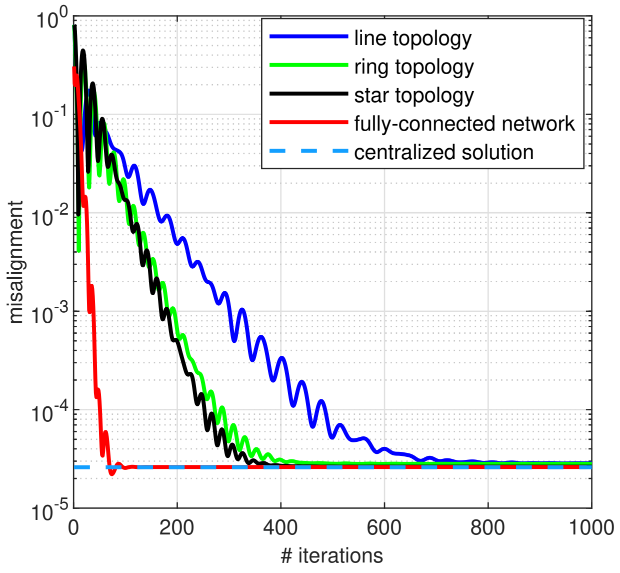


Figure 5.10: The misalignment of the proposed algorithm solving the distributed elastic-net regression problem with different topologies.

a random topology where each agent links to three other agents on average. We evaluate the performance of the proposed algorithm using the misalignment metric that is defined as

$$\frac{\left\|\mathbf{x}^d(k)-\boldsymbol{\omega}\right\|^2}{\left\|\boldsymbol{\omega}\right\|^2}$$

where

$$\mathbf{x}^d(k)=\left[\mathbf{x}_1^{(k)\mathsf{T}},\ldots,\mathbf{x}_N^{(k)\mathsf{T}}\right]^{\mathsf{T}}$$

and $\mathbf{x}_i^{(k)}$ $\forall i \in \mathcal{V}$ denotes the local estimate at agent i.

In Fig. 5.7, we plot the misalignment of the proposed algorithm versus its outer-loop iteration index for different values of P_i, i.e., $P_i = 2$, $P_i = 10$, $P_i = 20$, and $P_i = 50$ while $M = 800$, $M = 1000$, $M = 1100$, and $M = 1500$, respectively. Fig. 5.7 shows that the proposed algorithm converges faster as the number of local features P_i decreases. In Fig. 5.8, we set $P_i = 2$ and use the same topology as in Fig.5.7 but consider different values of M. Fig. 5.8 shows that the proposed algorithm achieves higher accuracy as the number of samples M increases. Note that we include the misalignment of the centralized optimal solution in all figures.

In Fig. 5.9, we consider different values of N while $P_i = 2$, $M = 500$, and the network topology is arbitrary but with an average node degree of three. Fig. 5.9 shows that the proposed algorithm converges faster as the number of agents N decreases. In Fig. 5.10, we evaluate the proposed algorithm by setting $N = 10$, $P_i = 2$, $M = 500$ and considering four different common simple topologies, i.e.,

- line: the agents are connected one after the other, hence, $|\mathcal{N}_i| = 2$ for $1 < i < N$ and $|\mathcal{N}_i| = 1$ for $i = 1$ and $i = N$
- ring: $|\mathcal{N}_i| = 2$ for each $i \in \mathcal{V}$
- star: $|\mathcal{N}_i| = N - 1$ for $i = 1$ and $|\mathcal{N}_i| = 1$ for $i = 2, \ldots, N$
- fully-connected: each agent in the network is connected to all the other agents.

In Fig. 5.10, we observe that the proposed algorithm converges faster as the average number of links per agent increases, i.e., the average connectivity of the network increases.

Distributed Ridge Regression

Considering a distributed ridge regression problem, in Figs. 5.11, 5.12, 5.13, and 5.14, we benchmark the proposed algorithm against some existing baseline algorithms,

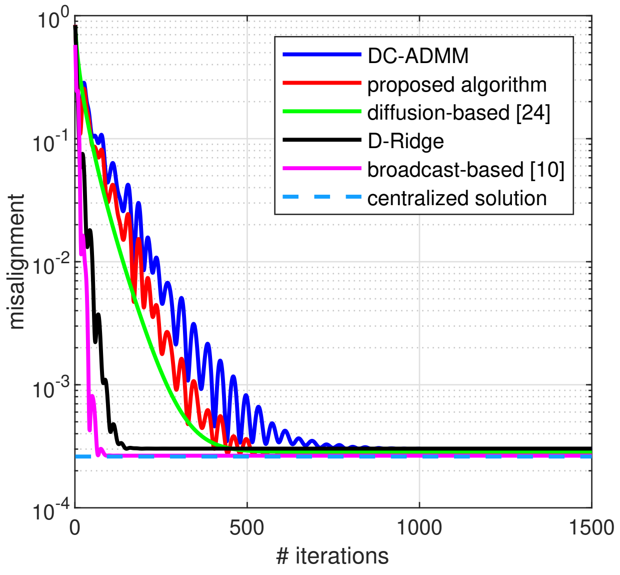


Figure 5.11: The misalignment of the proposed algorithm and other considered algorithms for the ridge regression problems with $N = 10$, $M = 50$, and $P_i = 2$.

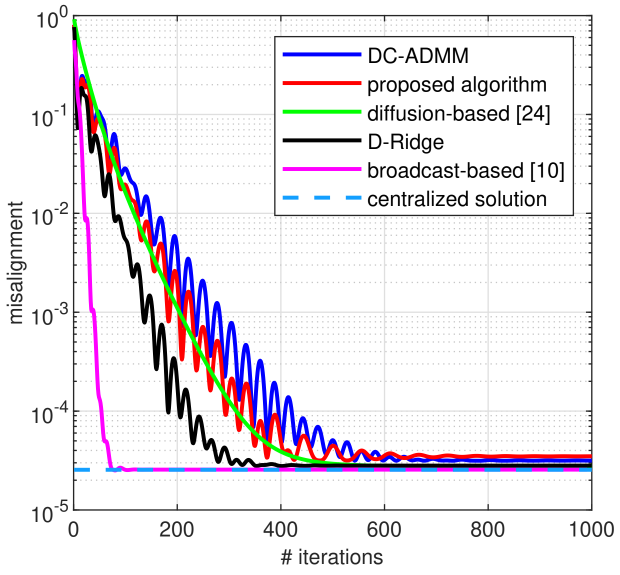


Figure 5.12: The misalignment of the proposed algorithm and other considered algorithms for the ridge regression problems with $N = 10$, $M = 200$, and $P_i = 2$.

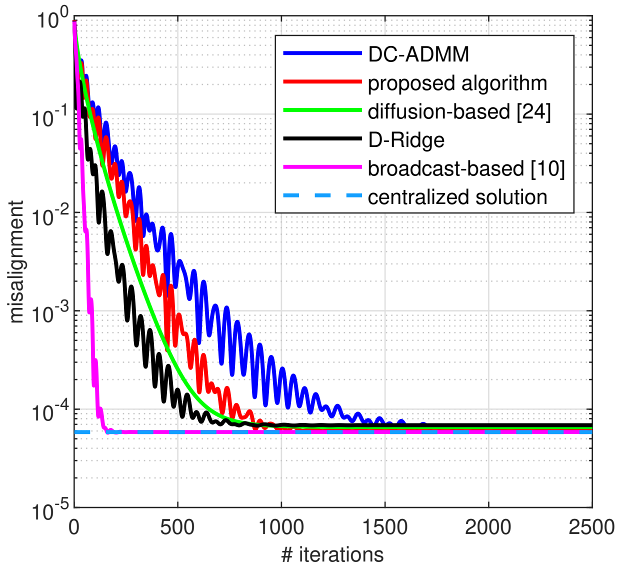


Figure 5.13: The misalignment of the proposed algorithm and other considered algorithms for the ridge regression problems with $N = 20$, $M = 200$, and $P_i = 2$.

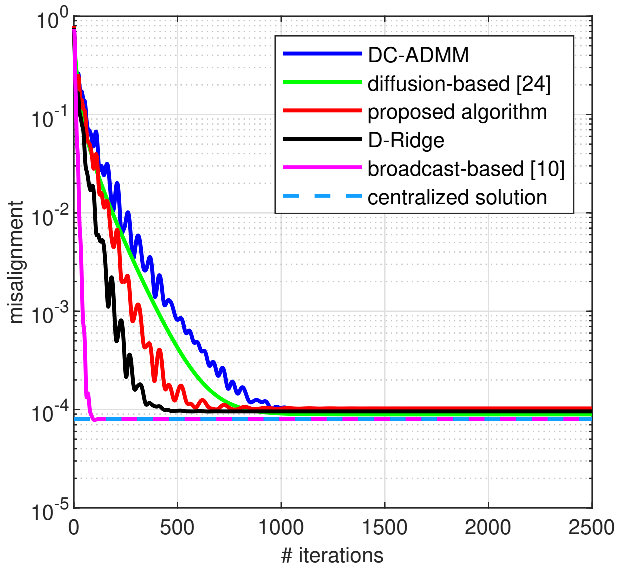


Figure 5.14: The misalignment of the proposed algorithm and other considered algorithms for the ridge regression problems with $N = 10$, $M = 200$, and $P_i = 10$.

namely, the broadcast-based algorithm for learning with distributed features proposed in [5], the dual consensus ADMM (DC-ADMM) algorithm of [49], the consensus-based algorithm for ridge regression (D-Ridge) introduced in [22], and the diffusion-based algorithm of [35]. The algorithms proposed in [22, 35] are only for solving the ridge regression problem. Here, we solve the problem (5.29) with the objective function (5.54) and set the ith agent's regularizer to

$$r_i(\mathbf{x}_i) = \eta \left\|\mathbf{x}_i\right\|^2$$

where $\eta \in \mathbb{R}^+$ is the regularization parameter.

We calculate the response vector $\mathbf{b}$ as in (5.55). As per [22, 35], we set the regularization parameter to $\eta = 0.001$. We also set the number of inner-loop BCD iterations of the proposed algorithm to 2 and obtain the results by averaging over 100 independent trials. In Fig. 5.11, we set $N = 10$, $M = 50$, and $P_i = 2$. In Fig. 5.12, the parameter setting is the same as Fig. 5.11 except for the number of samples M being larger, i.e., $M = 200$. In Fig. 5.13, we keep $M = 200$ and set the number of agents to $N = 20$. In Fig. 5.14, we set N and M to 10 and 200, respectively, while $P_i = 10 \, \forall i \in \mathcal{V}$.

We observe in Figs. 5.11, 5.12, 5.13, and 5.14, that the proposed algorithm outperforms the DC-ADMM algorithm. It also perform competitively in comparison with the algorithms of [22, 35], which are specifically tailored to the ridge regression problem. The superior performance of the broadcast-based algorithm of [5] is due to its centralized processing. We include it here only as a reference.

Distributed Lasso Regression

In Figs. 5.15, 5.16, 5.17, and 5.18, we compare the performance of the proposed algorithm with that of the broadcast-based algorithm for learning with distributed features proposed in [5] and the DC-ADMM algorithm of [49] considering a distributed lasso problem. Hence, we solve the problem (5.29) with the objective function (5.54) and set the ith agent's regularizer to

$$r_i(\mathbf{x}_i) = \eta \left\|\mathbf{x}_i\right\|_1$$

where $\eta \in \mathbb{R}^+$ is the regularization parameter.

We calculate the response vector $\mathbf{b}$ as in (5.55). As per [22, 35], we set the regularization parameter to $\eta = 0.001$. We also set the number of inner-loop BCD iterations of the proposed algorithm to 2 and obtain the results by averaging over 100 independent trials. In Fig. 5.15, we set $N = 10$, $M = 50$, and $P_i = 2$. In Fig. 5.16, the parameter setting is the same as Fig. 5.15 except for the number of samples M being larger, i.e., $M = 200$. In Fig. 5.17, we keep $M = 200$ and set

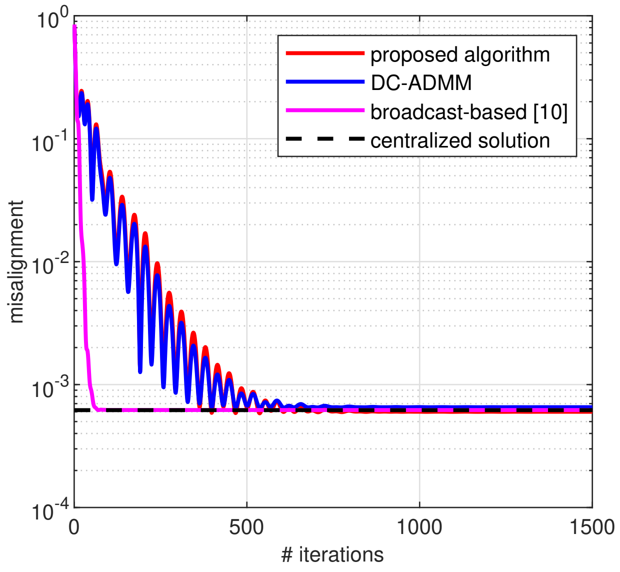


Figure 5.15: The misalignment of the proposed algorithm and other considered algorithms for the lasso regression problems with $N = 10$, $M = 50$, and $P_i = 2$.

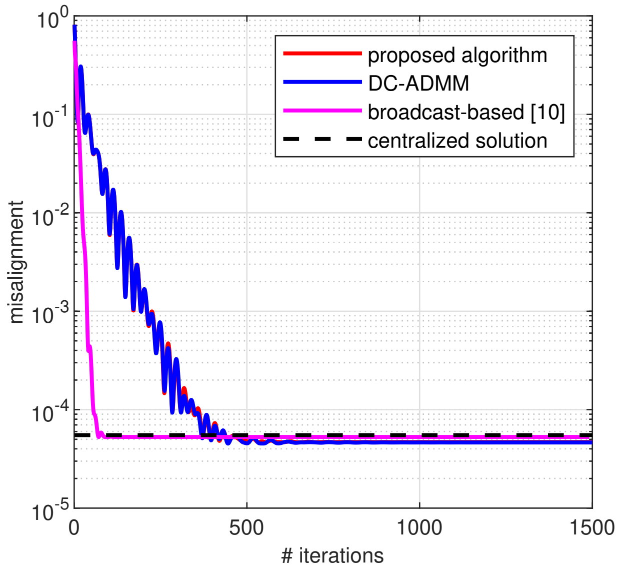


Figure 5.16: The misalignment of the proposed algorithm and other considered algorithms for the lasso regression problems with $N = 10$, $M = 200$, and $P_i = 2$.

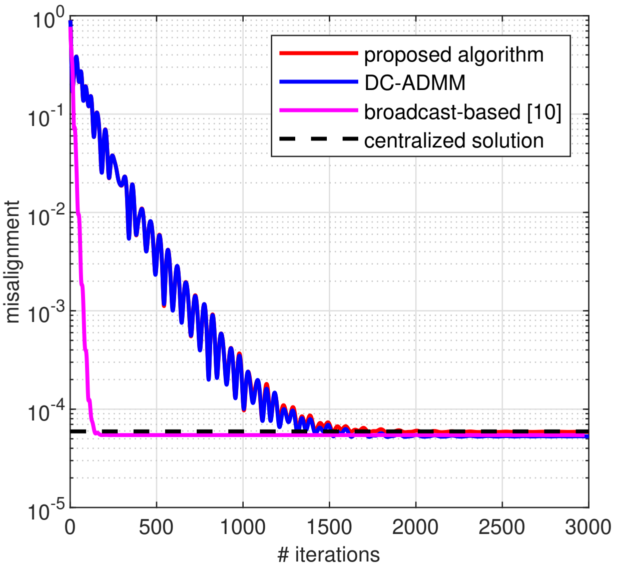


Figure 5.17: The misalignment of the proposed algorithm and other considered algorithms for the lasso regression problems with $N = 20$, $M = 200$, and $P_i = 2$.

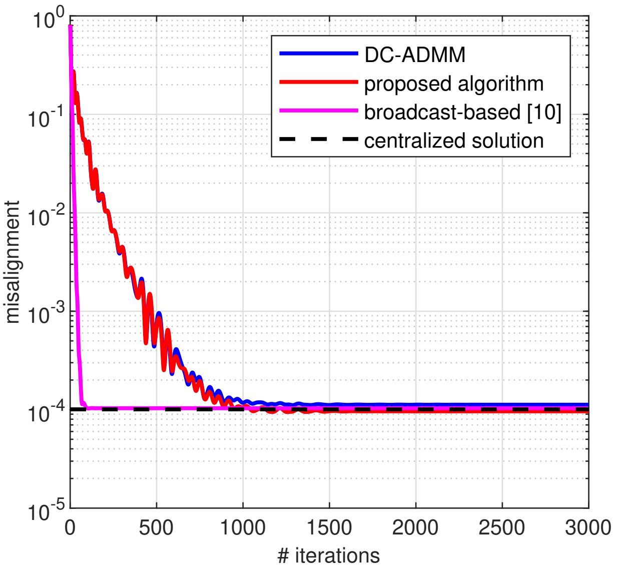


Figure 5.18: The misalignment of the proposed algorithm and other considered algorithms for the lasso regression problems with $N = 10$, $M = 200$, and $P_i = 10$.

the number of agents to $N = 20$. In Fig. 5.18, we set N and M to 10 and 200, respectively, while $P_i = 10 \, \forall i \in \mathcal{V}$.

We observe in Figs. 5.15, 5.16, 5.17, and 5.18, that the proposed algorithm performs very similar to the DC-ADMM algorithm as the learning curves of the two algorithms almost overlap. Again, the superior performance of the broadcast-based algorithm of [5] is due to its centralized processing.

Discussion

The main advantage of the proposed algorithm is in its ability to solve generic feature-partitioned distributed optimization problems without resorting to any conjugate function even when the objective function is non-smooth. This is unique to our proposed algorithm and, to the best of our knowledge, there is no existing algorithm with the same utility. That is why we do not compare the proposed algorithm with any other existing algorithm in Section 5.3.6 where the problem at hand is feature-distributed elastic-net regression. The existing algorithms for feature-partitioned distributed optimization such as DC-ADMM require the conjugate function of the objective or regularization function. In the case of elastic-net regression, calculating the conjugate function is impracticable.

The simulation results in Sections 5.3.6 are to provide a comparative study of the performance of the proposed algorithm with respect to the other most relevant existing algorithms. As evident by the results, the proposed algorithm's performance in solving the distributed ridge and lasso regression problems is on par with those of its state-of-the-art competitors, even those that have specifically been design to solve these problems.

As seen in the figures, in all simulations, the network-wide average estimate of the proposed algorithm converges to the corresponding optimal centralized solution. Although not shown here for conciseness, we have observed that the estimates at all agents also converge to the optimal solution in all the experiments corroborating our theoretical findings in Section 5.3.5.

In all simulations, we utilize only two BCD iterations with no extra inter-agent communication overhead. Therefore, the computational complexity and communication requirements of the proposed algorithm are of the same order as those of the related existing algorithms such as DC-ADMM. Indeed, we did not observe any significant difference in the per-iteration run time of the proposed and DC-ADMM algorithms.

5.3.7 Conclusion

In [1], we proposed a distributed algorithm for learning with non-smooth objective functions under distributed features. We reformulated the considered non-separable problem into a dual form that is separable and solved it via the ADMM. Subsequently, we devised an approach based on articulating the dual of the dual problem to overcome the challenge of computing the involved conjugate functions, which may be hard or even infeasible with some objective functions. We employed the BCD algorithm to solve the dual of the dual problem. Therefore, unlike most existing algorithms for solving learning problems with feature partitioning, the proposed algorithm does not require the explicit calculation of any conjugate of the objective function. We verified the convergence of the proposed algorithm to the optimal solution through both theoretical analysis and numerical simulations.

Chapter 6

Conclusions and Future Work

In the thesis, we developed machine learning algorithms for distributed optimization over networks of machines/agents both when the agents estimate the same common model (horizontal partitioning of data) and when every agent estimates a local model that is a part of the network-wide model (feature partitioning or vertical partitioning of data). Furthermore, we proposed a novel approach for fully-distributed optimization that is capable of exploiting the inherent randomness due to the use of a zeroth-order method to protect privacy and improve accuracy in comparison with the existing privacy-preserving distributed optimization approaches.

In Chapter 3, we presented two algorithms for distributed learning with horizontal partitioning of data. The former, named distributed ADMM TLS (DA-TLS), allows us to solve the distributed TLS problem without any careful tuning of the parameters involved in the algorithm. Moreover, DA-TLS converges to the centralized solution faster than the existing alternative algorithms. The latter, named distributed zeroth-order based ADMM (D-ZOA), solves a distributed learning problem with non-smooth convex objective functions. D-ZOA does not require subgradients or proximal operators to perform non-smooth optimization, it only requires the function values to approximate the gradient of the objective function. D-ZOA converges to a near-optimal solution and has comparable performance to related first-order algorithms.

In Chapter 4, we investigated and characterized the intrinsic privacy-preserving properties of the proposed D-ZOA algorithm. Unlike most existing related approaches where privacy mechanisms are based on the addition of noise with a given distribution, we only exploited the inherent randomness due to the use of a zeroth-order method and showed that this stochasticity was sufficient to ensure differential privacy. Furthermore, we showed that the total privacy leakage of the proposed D-ZOA algorithm grows sublinearly with the number of ADMM iterations. In addition, our proposed D-ZOA algorithm outperforms the existing differentially-private approaches in terms of accuracy while yielding similar privacy guarantee. A trade-off between privacy and accuracy is revealed by the convergence analysis of D-ZOA.

In Chapter 5, we presented three algorithms for distributed learning with feature partitioning of data. The first algorithm, named D-Ridge, solves the ridge regression problem with feature partitioning of the observation matrix and converges to the centralized solution faster than its diffusion-based contender does. The second algorithm is for learning tasks with feature partitioning of the observation matrix in a fully-distributed fashion when the objective is constituted by an ℓ_2-norm-square cost function and a non-smooth regularizer function. The proposed algorithm does not require any conjugate function of the non-smooth convex regularizer, which may be unfeasible or hard to obtain in some scenarios. The third algorithm extends the previously proposed algorithm by allowing us to solve distributed learning problems with feature partitioning and non-smooth convex objective functions. The proposed algorithm is fully-distributed and does not require the calculation of any conjugate of the possibly non-smooth cost or regularizer functions. The proposed algorithm converges to a near-optimal solution and has comparable performance to existing algorithms that are tailored to special objective functions.

The main drawbacks of the proposed algorithms are mainly due to the fact that some them are formed by two loops. This is the case of the DA-TLS algorithm where the outer loop is given by the Newton's method and the inner loop is given by the ADMM, the D-ZOA where the outer loop is given by the ADMM while the inner loop is given by the zeroth-order method and, also, the algorithm for feature-partitioned distributed learning with unknown conjugate functions where the outer loop is ADMM-based while the inner loop is given by the BCD method. Obviously, the existence of an inner loop for those cases brings about an increase of the computational complexity of the proposed algorithms. Consequently, this paves the way for future works.

Possible future research directions concern both distributed learning with horizontal and feature partitioning and the related privacy preservation concepts. In Chapter 3, we considered the distributed total least-squares problem with horizontal partitioning of data. The proposed DA-TLS algorithm is formed by two nested loops. A possible future work is to develop a reduced-complexity algorithm that is capable of solving the distributed total least-squares problem with horizontal partitioning of data using a single loop. Another possible direction is around developing solutions for the total least-squares problem with feature partitioning of data. Regarding distributed learning with horizontal partitioning of data and non-smooth objective functions, we proposed the D-ZOA algorithm. As we have seen in Chapter 3, this algorithm consists of two nested loops: an ADMM-based outer loop and a zeroth-order-based inner loop for solving the minimization problem in the ADMM primal update step. A possible future work consists in developing a computationally-efficient and fully-distributed algorithm for non-smooth distributed learning, which, unlike D-ZOA, does not require multiple iterations for the ADMM primal update step and thereby reduces the computational complexity. By employing zeroth-order information, the intrinsic privacy-preserving properties of this approach can be investigated similar to the privacy analysis performed for D-ZOA in Chapter 4.

Regarding distributed learning with feature partitioning, the distributed algorithm proposed in [1] and described in Chapter 5, which is designed for solving general regularized non-smooth learning problems with feature partitioning and non-smooth convex objective functions, consists of two nested loops: the inner-loop BCD and outer-loop ADMM. First, a possible future work extending the above-mentioned algorithm is to develop a solution that is capable of solving non-smooth distributed learning problems under distributed features within one loop and hence reducing the computational complexity. Another possible future work extending [1] can be around solving problems with feature partitioning of data where first-order information is not available and we only have access to zeroth-order information, i.e., function values. In this scenario, zeroth-order methods can be employed and, therefore, the inherent privacy-preserving properties can be investigated similar to the privacy analysis performed in Chapter 4.

Bibliography

[1] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Decentralized optimization with distributed features and non-smooth objective functions," *http://arxiv.org/abs/2208.11224*, 2022.

[2] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Privacy-preserved distributed learning with zeroth-order optimization," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 265–279, 2022.

[3] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: the power of two function evaluations," *IEEE Transactions on Information Theory*, vol. 61, pp. 2788–2806, May 2015.

[4] S. Liu, J. Chen, P.-Y. Chen, and A. Hero, "Zeroth-order online alternating direction method of multipliers: convergence analysis and applications," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, vol. 84 of *Proceedings of Machine Learning Research*, pp. 288–297, Apr. 2018.

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

[6] Z. Han, M. Hong, and D. Wang, *Signal processing and networking for big data applications*. Cambridge University Press, 2017.

[7] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, *Splitting Methods in Communication, Imaging, Science, and Engineering*. Scientific Computation, Cham: Springer International Publishing, 2016.

[8] D. Hajinezhad, M. Hong, and A. Garcia, "ZONE: Zeroth-order nonconvex multiagent optimization over networks," *IEEE Transactions on Automatic Control*, vol. 64, pp. 3995–4010, Oct. 2019.

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

[10] A. Nedic, A. Ozdaglar, and P. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, pp. 922–938, Apr. 2010.

[11] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, pp. 5262–5276, Oct. 2010.

[12] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Consensus-based distributed total least-squares estimation using parametric semidefinite programming," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5227–5231, May 2019.

[13] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed learning with non-smooth objective functions," in *Proc. 28th European Signal Processing Conference*, pp. 2180–2184, Jan. 2021.

[14] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, pp. 2320–2330, May 2011.

[15] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, pp. 977–992, Feb. 2019.

[16] R. Arablouei, K. Dogançay, and S. Werner, "Recursive total least-squares estimation of frequency in three-phase power systems," in *Proc. 22nd European Signal Processing Conference*, pp. 2330–2334, Sept. 2014.

[17] R. Arablouei, K. Doğançay, and S. Werner, "Adaptive frequency estimation of three-phase power systems," *Signal Processing*, vol. 109, pp. 290–300, Apr. 2015.

[18] R. Arablouei, S. Werner, and K. Doğançay, "Estimating frequency of three-phase power systems via widely-linear modeling and total least-squares," in *Proc. 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 464–467, Dec. 2013.

[19] E. Dall'Anese and G. B. Giannakis, "Distributed cognitive spectrum sensing via group sparse total least-squares," in *Proc. 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pp. 341–344, Dec. 2011.

[20] I. Markovsky, J. Willems, S. Van Huffel, Bart De Moor, and R. Pintelon, "Application of structured total least squares for system identification and model reduction," *IEEE Transactions on Automatic Control*, vol. 50, pp. 1490–1500, Oct. 2005.

[21] A. Bertrand and M. Moonen, "Low-complexity distributed total least squares estimation in ad hoc sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, pp. 4321–4333, Aug. 2012.

[22] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed ridge regression with feature partitioning," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018.

[23] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, pp. 1245–1260, Sep. 2018.

[24] A. Nedic and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 61, pp. 3936–3947, Dec. 2016.

[25] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1847–1862, 2010.

[26] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM*, vol. 58, Jun. 2011.

[27] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, pp. 572–596, 2011.

[28] M. Mardani, G. Mateos, and G. B. Giannakis, "Decentralized sparsity-regularized rank minimization: algorithms and applications," *IEEE Transactions on Signal Processing*, vol. 61, pp. 5374–5388, Nov. 2013.

[29] A. Agarwal, O. Dekel, and L. Xiao, "Optimal algorithms for online convex optimization with multi-point bandit feedback," in *Proc. 23rd Annual Conference on Learning Theory*, pp. 28–40, Jun. 2010.

[30] J. C. Spall, *Introduction to Stochastic Search and Optimization.* Wiley, 2003.

[31] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, Nov. 2017.

[32] F. Huang, S. Gao, S. Chen, and H. Huang, "Zeroth-order stochastic alternating direction method of multipliers for nonconvex nonsmooth optimization," in *Proc. 28th International Joint Conference on Artificial Intelligence* (S. Kraus, ed.), pp. 2549–2555, 2019.

[33] S. Liu, P. Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero III, and P. K. Varshney, "A primer on zeroth-order optimization in signal processing and machine learning: principals, recent advances, and applications," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 43–54, 2020.

[34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics, Springer, 2009.

[35] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Model-distributed solution of regularized least-squares problem over sensor networks," in *Proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3821–3825, Apr. 2015.

[36] C. Manss, D. Shutin, and G. Leus, "Distributed splitting-over-features sparse bayesian learning with alternating direction method of multipliers," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3654–3658, 2018.

[37] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Transactions on Signal Processing*, vol. 60, pp. 1942–1956, Apr. 2012.

[38] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "D-admm: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, pp. 2718–2723, May 2013.

[39] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed learning over networks with non-smooth regularizers and feature

partitioning," in *Proc. European Speech and Signal Processing Conference*, Aug. 2021.

[40] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "DP-ADMM: ADMM-based distributed learning with differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1002–1012, 2020.

[41] T. Zhang and Q. Zhu, "Dynamic differential privacy for ADMM-based distributed classification learning," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 172–187, Jan. 2017.

[42] N. Kashyap, S. Werner, Y.-F. Huang, and R. Arablouei, "Privacy preserving decentralized power system state estimation with phasor measurement units," in *Proc. 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop*, pp. 1–5, Jul. 2016.

[43] C. Heinze, B. McWilliams, and N. Meinshausen, "DUAL-LOCO: Distributing statistical estimation using random projections," in *Proc. 19th International Conference on Artificial Intelligence and Statistics*, vol. 51, pp. 875–883, May 2016.

[44] C. Heinze-Deml, B. McWilliams, N. Meinshausen, and G. Krummenacher, "LOCO: Distributing ridge regression with random projections," 2015.

[45] C. Heinze-Deml, B. McWilliams, and N. Meinshausen, "Preserving differential privacy between features in distributed estimation," 2017.

[46] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1001–1016, 2015.

[47] S. A. Alghunaim, M. Yan, and A. H. Sayed, "A multi-agent primal-dual strategy for composite optimization over distributed features," in *Proc. 28th European Signal Processing Conference*, pp. 2095–2099, Jan. 2021.

[48] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A general framework for communication-efficient distributed optimization," *J. Mach. Learn. Res.*, vol. 18, p. 8590–8638, Jan. 2017.

[49] T. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[50] B. Zhang, J. Geng, W. Xu, and L. Lai, “Communication efficient distributed learning with feature partitioned data,” in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, Mar. 2018.

[51] Y. Hu, D. Niu, J. Yang, and S. Zhou, “FDML: A collaborative machine learning framework for distributed features,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2232–2240, 2019.

[52] J. Szurley, A. Bertrand, and M. Moonen, “Topology-independent distributed adaptive node-specific signal estimation in wireless sensor networks,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 130–144, 2017.

[53] J. Chen, C. Richard, and A. H. Sayed, “Diffusion LMS for clustered multitask networks,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5487–5491, 2014.

[54] N. Bogdanović, J. Plata-Chaves, and K. Berberidis, “Distributed incremental-based LMS for node-specific adaptive parameter estimation,” *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5382–5397, 2014.

[55] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, “Distributed diffusion-based LMS for node-specific adaptive parameter estimation,” *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3448–3460, 2015.

[56] X. Zhang, M. M. Khalili, and M. Liu, “Recycled ADMM: Improve privacy and accuracy with less computation in distributed algorithms,” in *Proc. 56th Annual Allerton Conference on Communication, Control, and Computing*, pp. 959–965, Oct. 2018.

[57] X. Zhang, M. M. Khalili, and M. Liu, “Improving the privacy and accuracy of ADMM-based distributed algorithms,” in *Proc. 35th International Conference on Machine Learning*, vol. 80, pp. 5796–5805, Jul. 2018.

[58] J. Ding, Y. Gong, M. Pan, and Z. Han, “Optimal differentially private ADMM for distributed machine learning,” 2019.

[59] J. Ding, S. M. Errapotu, H. Zhang, Y. Gong, M. Pan, and Z. Han, “Stochastic ADMM based distributed machine learning with differential privacy,” in *Proc. 15th SecureComm*, pp. 257–277, Oct. 2019.

[60] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via functional perturbation," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 395–408, 2018.

[61] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2013.

[62] Y. Hu, P. Liu, L. Kong, and D. Niu, "Learning privately over distributed features: an ADMM sharing approach," 2019.

[63] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, "Robust federated learning using ADMM in the presence of data falsifying byzantines," 2017.

[64] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, "Robust decentralized learning using ADMM with unreliable agents," 2018.

[65] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: Models, limits, and algorithms," *IEEE Transactions on Signal Processing*, vol. 59, pp. 386–398, Jan. 2011.

[66] O. L. Mangasarian, E. W. Wild, and G. M. Fung, "Privacy-preserving classification of vertically partitioned data via random kernels," *ACM Transactions on Knowledge Discovery from Data*, vol. 2, Oct. 2008.

[67] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proc. 9th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 206–215, 2003.

[68] D. P. Bertsekas, *Parallel and distributed computation : numerical methods*. Englewood Cliffs, N.J: Prentice-Hall, 1989.

[69] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, Aug. 2010.

[70] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 2015 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, Oct. 2015.

[71] S. Huang and C. Li, "Distributed sparse total least-squares over networks," vol. 63, pp. 2986–2998, Jun. 2015.

[72] R. Lopez-Valcarce, S. S. Pereira, and A. Pages-Zamora, “Distributed Total Least Squares estimation over networks,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7580–7584, May 2014.

[73] R. Arablouei, S. Werner, and K. Doğançay, “Diffusion-based distributed adaptive estimation utilizing gradient-descent total least-squares,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5308–5312, May 2013.

[74] R. Abdolee and B. Champagne, “Diffusion LMS strategies in sensor networks with noisy input data,” *IEEE/ACM Transactions on Networking*, vol. 24, pp. 3–14, Feb. 2016.

[75] L. Lu, H. Zhao, and B. Champagne, “Diffusion total least-squares algorithm with multi-node feedback,” *Signal Processing*, Jul. 2018.

[76] S. Silva Pereira, A. Pages-Zamora, and R. Lopez-Valcarce, “Distributed TLS estimation under random data faults,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2949–2953, Apr. 2015.

[77] C. Li, S. Huang, Y. Liu, and Y. Liu, “Distributed TLS over multitask networks with adaptive intertask cooperation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, pp. 3036–3052, Dec. 2016.

[78] W. Dinkelbach, “On nonlinear fractional programming,” *Managemeny Science*, vol. 13, pp. 492–498, Mar. 1967.

[79] Z.-Q. Luo, W.-K. Ma, A. So, Y. Ye, and S. Zhang, “Semidefinite relaxation of quadratic optimization Problems,” *IEEE Signal Processing Magazine*, vol. 27, pp. 20–34, May 2010.

[80] R. Arablouei, S. Werner, and K. Doğançay, “Analysis of the gradient-descent total least-squares adaptive filtering algorithm,” *IEEE Transactions on Signal Processing*, vol. 62, pp. 1256–1264, Mar. 2014.

[81] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge University Press, 2004.

[82] J.-P. Crouzeix and J. A. Ferland, “Algorithms for generalized fractional programming,” *Mathematical Programming*, vol. 52, pp. 191–207, May 1991.

[83] G. Pataki, “On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues,” *Mathematics of Operations Research*, vol. 23, pp. 339–358, Jan. 1998.

[84] E. Dall’Anese, H. Zhu, and G. B. Giannakis, “Distributed optimal power flow for smart microgrids,” *IEEE Transactions on Smart Grid*, vol. 4, pp. 1464–1475, Sep. 2013.

[85] A. Nedic and A. Olshevsky, “Distributed optimization over time-varying directed graphs,” *IEEE Transactions on Automatic Control*, vol. 60, pp. 601–615, Mar. 2015.

[86] E. Ghadimi, I. Shames, and M. Johansson, “Multi-step gradient methods for networked optimization,” *IEEE Transactions on Signal Processing*, vol. 61, pp. 5417–5429, Nov. 2013.

[87] X. Wu and J. Lu, “Improved convergence rates of P-EXTRA for non-smooth distributed optimization,” in *IEEE International Conference on Control and Automation*, pp. 55–60, Jul. 2019.

[88] D. Yuan, D. W. C. Ho, and S. Xu, “Zeroth-order method for distributed optimization with approximate projections,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, pp. 284–294, Feb. 2016.

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

[90] Z. Huang, S. Mitra, and N. Vaidya, “Differentially private distributed optimization,” in *Proc. 2015 International Conference on Distributed Computing and Networking*, 2015.

[91] S. Han, U. Topcu, and G. J. Pappas, “Differentially private distributed constrained optimization,” *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 50–64, 2017.

[92] M. T. Hale and M. Egerstedty, “Differentially private cloud-based multi-agent optimization with constraints,” in *Proc. 2015 American Control Conference*, pp. 1235–1240, Jul. 2015.

[93] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

[94] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Foundations and Trends in Theoretical Computer Science*, vol. 9, pp. 211–407, Aug. 2014.

[95] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Proc. Third Conference on Theory of Cryptography*, pp. 265–284, Springer-Verlag, 2006.

[96] Y. Tang, J. Zhang, and N. Li, “Distributed zero-order algorithms for nonconvex multi-agent optimization,” 2020.

[97] A. H. Sayed, “Adaptive Networks,” *Proceedings of the IEEE*, vol. 102, pp. 460–497, Apr. 2014.

[98] T. Lin, S. Ma, and S. Zhang, “On the global linear convergence of the admm with multiblock variables,” *SIAM Journal on Optimization*, vol. 25, pp. 1478–1497, Jan. 2015.

[99] K. Eriksson, *Applied Mathematics: Body and Soul : Volume 1: Derivatives and Geometry in IR3*. 2004.

[100] D. Hallac, J. Leskovec, and S. Boyd, “Network lasso: Clustering and optimization in large graphs,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 387–396, 2015.

[101] T. Chang, A. Nedić, and A. Scaglione, “Distributed constrained optimization by consensus-based primal-dual perturbation method,” *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1524–1538, 2014.

[102] D. P. Palomar and Mung Chiang, “A tutorial on decomposition methods for network utility maximization,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[103] D. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[104] N. Parikh and S. Boyd, “Proximal algorithms,” *Foundations and Trends in Optimization*, vol. 1, p. 127–239, Jan. 2014.

[105] J. M. Borwein and A. S. Lewis, *Convex analysis and nonlinear optimization: theory and examples*. Springer, 2006.

[106] P. Tseng, “Convergence of a block coordinate descent method for nondifferentiable minimization,” *Journal of Optimization Theory and Applications*, vol. 109, pp. 475–494, Jan. 2001.

[107] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[108] M. Fukushima, “Application of the alternating direction method of multipliers to separable convex programming problems,” *Computational Optimization and Applications*, vol. 1, pp. 93–111, 1992.

[109] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization, Theory and Examples*. Springer, 2000.

Appendix A

Publications on Distributed Optimization with Horizontal Partitioning

- **P1**: [12] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Consensus-based distributed total least-squares estimation using parametric semidefinite programming,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5227–5231, May 2019.
- **P2**: [13] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Distributed learning with non-smooth objective functions,” in *Proc. 28th European Signal Processing Conference*, pp. 2180–2184, Jan. 2021.

Appendix B

Publications on Privacy-Preserved Distributed Learning with Zeroth-Order Optimization

- **P3**: [2] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Privacy-preserved distributed learning with zeroth-order optimization," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 265-279, 2022.

Appendix C

Publications on Distributed Optimization with Feature Partitioning

- **P4**: [22] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Distributed ridge regression with feature partitioning,” in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018.
- **P5**: [39] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Distributed learning over networks with non-smooth regularizers and feature partitioning,” in *Proc. European Speech and Signal Processing Conference*, Aug. 2021.
- **P6**: [1] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Decentralized optimization with distributed features and non-smooth objective functions,” 2022, *arXiv: 2208.11224*.

CONSENSUS-BASED DISTRIBUTED TOTAL LEAST-SQUARES ESTIMATION USING PARAMETRIC SEMIDEFINITE PROGRAMMING

Cristiano Gratton**, Naveen K. D. Venkategowda** *, Reza Arablouei*†*, Stefan Werner**

* Department of Electronic Systems, NTNU - Norwegian University of Science and Technology
† CSIRO's Data61, Pullenvale QLD 4069, Australia

ABSTRACT

We propose a new distributed algorithm to solve the total least-squares (TLS) problem when data are distributed over a multi-agent network. To develop the proposed algorithm, named distributed ADMM TLS (DA-TLS), we reformulate the TLS problem as a parametric semidefinite program and solve it using the alternating direction method of multipliers (ADMM). Unlike the existing consensus-based approaches to distributed TLS estimation, DA-TLS does not require careful tuning of any design parameter. Numerical experiments demonstrate that the DA-TLS converges to the centralized solution significantly faster than the existing consensus-based TLS algorithms.

***Index Terms*—** ADMM, consensus, distributed estimation, total least squares, semidefinite programming.

1. INTRODUCTION

With the recent advances in technology, large quantities of data are collected by numerous sensors, which are often geographically dispersed. Hence, performing data analysis tasks such as estimation and classification at a central processing unit is impractical due to transmission cost or privacy reasons. Furthermore, collecting all the data in a fusion center creates a single point of failure. Therefore, it is imperative to develop algorithms that are capable of processing data spread across multiple agents [1–7].

In the realm of linear estimation, the total least-squares (TLS) method has been introduced as an alternative to the ordinary least-squares method to deal with errors-in-variables models. In such models, both independent and dependent variables are corrupted by noise or perturbation. TLS has been successfully used in several signal processing applications, e.g., frequency estimation of power systems [8–10], cognitive spectrum sensing [11], system identification [12], and wireless sensor networks [13].

The distributed TLS problem has previously been considered in [13–21]. The works in [13, 14] are based on the consensus strategy and rely on the dual-based subgradient method. Their relatively high computational complexity has partially motivated the works in [16–20]. While the approach of [16] is based on the average consensus strategy, the algorithms in [17–21] are based on diffusion strategies and, therefore, suffer from relatively slow convergence [6]. The convergence speed of the algorithm proposed in [13] greatly depends on the network topology and dimensionality of the data. Although these shortcomings are mitigated in [14], the convergence rate of algorithms in [13] and [14] highly depends on the choice of the step-size, whose optimal tuning requires the global knowledge of the data and network topology.

In this paper, we solve the distributed TLS problem when each agent has access to parts of a set of linear equations, i.e., a subset of the rows of the observation matrix and the output vector. This is a common scenario in wireless sensor networks, e.g., distributed system identification [22]. Through a change of variable from a vector to a rank-one matrix and subsequent semidefinite relaxation (SDR), we transform the non-convex distributed TLS problem into a semidefinite program. We solve the modified problem using the alternating direction method of multipliers (ADMM) and a generalization of the algorithm proposed in [23] for fractional programming. Since the optimal solution is rank-one, the relaxation is tight and does not incur any loss of optimality [24]. In addition, as the objective function in the modified problem is the sum of fractions of linear functions, the convergence of the proposed algorithm to the globally optimal solution is guaranteed.

The proposed algorithm, called distributed ADMM TLS (DA-TLS), is fully distributed in the sense that it requires the agents to share data only with their immediate neighbors at each iteration. Furthermore, the performance of DA-TLS is not sensitive to the tuning of its parameters. This makes DA-TLS more flexible and suitable for distributed deployment in comparison with the algorithms of [13,14]. Simulation results show faster convergence of DA-TLS to the centralized solution at all agents in comparison with the existing algorithms.

2. SYSTEM MODEL

We consider a connected network of $K \in \mathbb{N}$ agents modeled as an undirected graph $\mathcal{G}(\mathcal{K}, \mathcal{E})$ where the set of vertices $\mathcal{K} = \{1, \ldots, K\}$ corresponds to the agents and the edge set

This work was partly supported by the Research Council of Noway.

$\mathcal{E}$ represents the communication links between the pairs of agents. Agent $k \in \mathcal{K}$ can communicate with its neighbors whose indexes are in the set $\mathcal{N}_k$ with cardinality $|\mathcal{N}_k|$. By convention, $\mathcal{N}_k$ does not include the agent k itself.

Let $\mathbf{X} \in \mathbb{R}^{N\times P}$ be the observation matrix, $\boldsymbol{\Delta} \in \mathbb{R}^{N\times P}$ the error in the observation matrix, $\mathbf{y} \in \mathbb{R}^{N\times 1}$ the response vector, $\boldsymbol{\delta} \in \mathbb{R}^{N\times 1}$ the error in the response vector, and $\mathbf{w} \in \mathbb{R}^{P\times 1}$ the sought-after parameter vector that relates $\mathbf{X}$ and $\mathbf{y}$ through $(\mathbf{X}-\boldsymbol{\Delta})\mathbf{w} = \mathbf{y}-\boldsymbol{\delta}$. The matrix $\mathbf{X}$ consists of K submatrices $\mathbf{X}_k$, i.e., $\mathbf{X} = [\mathbf{X}_1^{\mathrm{T}}, \mathbf{X}_2^{\mathrm{T}}, \dots, \mathbf{X}_K^{\mathrm{T}}]^{\mathrm{T}}$, and the vector $\mathbf{y} \in \mathbb{R}^{N\times 1}$ of K subvectors $\mathbf{y}_k$, i.e., $\mathbf{y} = \left[\mathbf{y}_1^{\mathrm{T}}, \mathbf{y}_2^{\mathrm{T}}, \dots, \mathbf{y}_K^{\mathrm{T}}\right]^{\mathrm{T}}$, as the data are distributed among the agents and each agent k holds its respective $\mathbf{X}_k \in \mathbb{R}^{N_k\times P}$ and $\mathbf{y}_k \in \mathbb{R}^{N_k\times 1}$ where $\sum_{k=1}^{K} N_k = N$ and $(\cdot)^{\mathrm{T}}$ denotes the matrix transpose.

The TLS estimate of the unknown parameter vector $\mathbf{w}$ can be found by solving the constrained optimization problem

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\Delta},\delta} \quad & \|\boldsymbol{\Delta}\|_{\mathrm{F}} + \|\boldsymbol{\delta}\| \\ \text{s.t.} \quad & (\mathbf{X}-\boldsymbol{\Delta})\mathbf{w} = \mathbf{y}-\boldsymbol{\delta} \end{aligned} \tag{1}$$

where $\|\cdot\|_{\mathrm{F}}$ and $\|\cdot\|$ denote the Frobenius norm and Euclidean norm, respectively. When the entries of $\boldsymbol{\Delta}$ and $\boldsymbol{\delta}$ are independent and identically distributed (i.i.d.), a centralized TLS solution $\mathbf{w}^c$ of (1) can be obtained as

$$\mathbf{w}^c = \frac{-1}{v_{P+1}}[v_1, v_2, \dots, v_P]^{\mathrm{T}} \tag{2}$$

where $\mathbf{v} = [v_1, v_2, \dots, v_{P+1}]^{\mathrm{T}}$ is the right singular vector corresponding to the smallest singular value of $[\mathbf{X}, \mathbf{y}]$ [25].

An equivalent but more practical solution can be obtained by minimizing the Rayleigh quotient cost function as [25]

$$\min_{\mathbf{w}} \frac{\|\mathbf{X}\mathbf{w}-\mathbf{y}\|^2}{\|\mathbf{w}\|^2+1} \quad \text{or} \quad \min_{\mathbf{w}} \sum_{k=1}^{K} \frac{\|\mathbf{X}_k\mathbf{w}-\mathbf{y}_k\|^2}{\|\mathbf{w}\|^2+1}. \tag{3}$$

Since finding a centralized solution of (3) over a network may be inefficient, we propose a distributed algorithm for this purpose in the following section.

3. DISTRIBUTED TLS

We first discuss the SDR technique that allows us to transform the TLS problem into a parametric semidefinite program, which we solve iteratively through two nested loops. Then, we describe the consensus-based reformulation of the resultant parametric semidefinite program that enables its distributed solution via the ADMM, which forms the inner loop. Finally, we describe the steps of the inner and outer loops of the algorithm.

3.1. Semidefinite Relaxation

Using the properties of the matrix trace operator, we rewrite the Rayleigh quotient cost function in (3) as

$$\sum_{k=1}^{K} \frac{\mathrm{tr}(\mathbf{w}\mathbf{w}^{\mathrm{T}}\mathbf{X}_k^{\mathrm{T}}\mathbf{X}_k) - 2\mathbf{y}_k^{\mathrm{T}}\mathbf{X}_k\mathbf{w} + \|\mathbf{y}_k\|^2}{\mathrm{tr}(\mathbf{w}\mathbf{w}^{\mathrm{T}})+1}. \tag{4}$$

Considering (4) and defining

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}\mathbf{w}^T & \mathbf{w} \\ \mathbf{w}^T & 1 \end{bmatrix} \text{ and } \mathbf{C}_k = \begin{bmatrix} \mathbf{X}_k^T\mathbf{X}_k & -\mathbf{X}_k^T\mathbf{y}_k \\ -\mathbf{y}_k^T\mathbf{X}_k & \|\mathbf{y}_k\|^2 \end{bmatrix}, \tag{5}$$

(3) can be recast as

$$\begin{aligned} \min_{\mathbf{W}\succeq\mathbf{0}} \quad & \sum_{k=1}^{K} \frac{\mathrm{tr}(\mathbf{C}_k\mathbf{W})}{\mathrm{tr}(\mathbf{W})} \\ \text{s.t.} \quad & \mathrm{rank}(\mathbf{W}) = 1. \end{aligned} \tag{6}$$

Relaxing the rank constraint in (6) turns it into the following aggregate linear-fractional program

$$\min_{\mathbf{W}\succeq\mathbf{0}} \sum_{k=1}^{K} \frac{\mathrm{tr}(\mathbf{C}_k\mathbf{W})}{\mathrm{tr}(\mathbf{W})}. \tag{7}$$

Both numerator and denominator of the summands in the objective function of (7) are linear functions of the matrix variable $\mathbf{W}$. Therefore, (7) can be converted to a parametric semidefinite program whose objective is in the subtractive form as per the following proposition.

Proposition 1. *Let $\mathbf{W}^*$ denote the optimal solution to (7). Then, there exists a vector $\boldsymbol{\beta}^* = [\beta_1^*, \dots, \beta_K^*]$ such that $\mathbf{W}^*$ is also the optimal solution of the following semidefinite program*

$$\mathbf{W}^* = \arg\min_{\mathbf{W}\succeq\mathbf{0}} \sum_{k=1}^{K} \mathrm{tr}(\mathbf{C}_k\mathbf{W}) - \beta_k^*\mathrm{tr}(\mathbf{W}). \tag{8}$$

In addition, $\mathbf{W}^*$ also satisfies the following system of equations:

$$\mathrm{tr}(\mathbf{C}_k\mathbf{W}^*) - \beta_k^*\mathrm{tr}(\mathbf{W}^*) = 0,\ k = 1, 2, \dots, K. \tag{9}$$

Proof. The Karush-Kuhn-Tucker (KKT) conditions of optimality [26] for problem (8) give the same solution set as the KKT conditions for the epigraph form of (7). Since the KKT conditions for both problems are sufficient for optimality, the two problems are equivalent. The system of equation (9) is due to the KKT conditions. □

In the next subsection, we describe a consensus-based reformulation of (8), which allows the application of the ADMM to solve (8) for any given $\boldsymbol{\beta}^*$.

3.2. Building Consensus

In order to tackle (8) in a distributed fashion, we introduce $\mathcal{W} := \{\mathbf{W}_k\}_{k=1}^K$ representing the local copies of $\mathbf{W}$ at the agents. Therefore, we rewrite (8) in the following equivalent form

$$\begin{aligned} \min_{\{\mathbf{W}_k \succeq \mathbf{0}\}} \quad & \sum_{k=1}^{K} \mathrm{tr}(\mathbf{C}_k \mathbf{W}_k) - \beta_k^* \mathrm{tr}(\mathbf{W}_k) \\ \text{s.t.} \quad & \mathbf{W}_k = \mathbf{W}_l, \quad l \in \mathcal{N}_k, \quad k \in \mathcal{K}. \end{aligned} \tag{10}$$

The equality constraints enforce consensus over $\mathbf{W}_k$, $k = 1, \ldots, K$, across each agent's neighborhood $\mathcal{N}_k$.

To solve (10) in a distributed fashion, we employ the ADMM [1]. Hence, we introduce the auxiliary local variables $\mathcal{Z} := \{\mathbf{Z}_k^l\}_{l \in \mathcal{N}_k}$ and rewrite (10) as

$$\begin{aligned} \min_{\{\mathbf{W}_k \succeq \mathbf{0}\}} \quad & \sum_{k=1}^{K} \mathrm{tr}(\mathbf{C}_k \mathbf{W}_k) - \beta_k^* \mathrm{tr}(\mathbf{W}_k) \\ \text{s.t.} \quad & \mathbf{W}_k = \mathbf{Z}_k^l, \mathbf{W}_l = \mathbf{Z}_k^l, \quad l \in \mathcal{N}_k, \quad k \in \mathcal{K}. \end{aligned} \tag{11}$$

Using the auxiliary variables $\mathcal{Z}$, we obtain an equivalent alternative representation of the constraints in (10). These variables are only used to derive the local recursions and are eventually eliminated. By associating the Lagrange multipliers $\mathcal{V} := \{\{\mathbf{\Gamma}_k^l\}_{l \in \mathcal{N}_k}, \{\mathbf{\Lambda}_k^l\}_{l \in \mathcal{N}_k}\}_{k=1}^K$ with the constraints in (11), we get the following augmented Lagrangian function:

$$\begin{aligned} \mathcal{L}_\rho(\mathcal{W}, \mathcal{Z}, \mathcal{V}) &= \sum_{k=1}^{K} \mathrm{tr}\left(\mathbf{C}_k \mathbf{W}_k\right) - \beta_k^* \mathrm{tr}(\mathbf{W}_k) \\ &+ \sum_{k=1}^{K} \sum_{l \in \mathcal{N}_k} \mathrm{tr}\left(\left(\mathbf{\Lambda}_k^l\right)^{\mathrm{T}} \left(\mathbf{W}_k - \mathbf{Z}_k^l\right) + \left(\mathbf{\Gamma}_k^l\right)^{\mathrm{T}} \left(\mathbf{W}_l - \mathbf{Z}_k^l\right)\right) \\ &+ \frac{\rho}{2} \sum_{k=1}^{K} \sum_{l \in \mathcal{N}_k} \left(\left\|\mathbf{W}_k - \mathbf{Z}_k^l\right\|_{\mathrm{F}}^2 + \left\|\mathbf{W}_l - \mathbf{Z}_k^l\right\|_{\mathrm{F}}^2\right), \end{aligned} \tag{12}$$

where the constant $\rho > 0$ is a penalty parameter.

Obtaining the solution through the ADMM entails an iterative process consisting of the following steps at each iteration: 1) $\mathcal{L}_\rho$ is minimized with respect to $\mathcal{W}$; 2) $\mathcal{L}_\rho$ is minimized with respect to $\mathcal{Z}$; and, 3) the Lagrange multipliers $\mathcal{V}$ are updated through gradient-ascent [1].

Thanks to the reformulation of (8) as (11), the Lagrangian function (12) can be decoupled with respect to variables in $\mathcal{W}$ and $\mathcal{Z}$ as well as across the network agents $\mathcal{K}$. It can be shown that, in the ADMM steps, the auxiliary variables $\mathcal{Z}$ and the Lagrange multipliers $\{\mathbf{\Gamma}_k^l\}_{l \in \mathcal{N}_k}$ are eliminated. Hence, we end up with the following iterative updates at the kth agent

$$\mathbf{W}_k(m+1) = \arg \min_{\mathbf{W}_k \succeq \mathbf{0}} \mathcal{L}_\rho(\mathbf{W}_k, \mathbf{\Lambda}_k(m)) \tag{13}$$

$$\mathbf{\Lambda}_k(m+1) = \mathbf{\Lambda}_k(m) + \rho \sum_{l \in \mathcal{N}_k} [\mathbf{W}_k(m+1) - \mathbf{W}_l(m+1)], \tag{14}$$

where $\mathbf{\Lambda}_k(m) = 2 \sum_{l \in \mathcal{N}_k} \mathbf{\Lambda}_k^l(m)$ and m is the iteration index.

The constrained minimization problem in (13) can be expressed as the following semidefinite least-squares problem

$$\min_{\mathbf{W}_k \succeq \mathbf{0}} \mathrm{tr}[\mathbf{W}_k^{\mathrm{T}}(\mathbf{W}_k - 2\mathbf{G}_k(m))], \tag{15}$$

where

$$\begin{aligned} \mathbf{G}_k(m) = \frac{1}{2\rho|\mathcal{N}_k|} \Big(& \rho|\mathcal{N}_k| \mathbf{W}_k(m) + \rho \sum_{l \in \mathcal{N}_k} \mathbf{W}_l(m) \\ & - \mathbf{C}_k + \beta_k \mathbf{I} - \mathbf{\Lambda}_k(m)\Big). \end{aligned} \tag{16}$$

The solution of (15) is given by

$$\mathbf{W}_k(m+1) = \mathbf{U}(m) \max\left(\mathbf{\Sigma}(m), \mathbf{0}\right) \mathbf{U}(m)^{\mathrm{T}}, \tag{17}$$

where $\mathbf{U}(m)$ and $\mathbf{\Sigma}(m)$ are the orthogonal and diagonal matrices coming from the eigen-decomposition (EVD) $\mathbf{G}(m) = \mathbf{U}(m)\mathbf{\Sigma}(m)\mathbf{U}^{\mathrm{T}}(m)$ and $\max(\mathbf{\Sigma}(m), \mathbf{0})$ denotes the diagonal matrix whose entries are the maxima of the diagonal entries of $\mathbf{\Sigma}(m)$, i.e., the eigenvalues of $\mathbf{G}_k(m)$, and zero. Note that the most computationally intensive operation is the EVD.

3.3. Algorithm

The DA-TLS algorithm consists of two loops. In the inner loop, the solution of (8) is obtained using the ADMM for a given $\boldsymbol{\beta}^*$. In the outer loop, we use a single iteration of the Newton's method [27] to find the solution of (9), i.e., $\beta_k(j+1) = \beta_k(j) - [\mathrm{tr}(\mathbf{W})]^{-1}[\beta_k(j)\mathrm{tr}(\mathbf{W}) - \mathrm{tr}(\mathbf{C}_k\mathbf{W})]$. The proposed algorithm is summarized in Algorithm 1.

Algorithm 1 DA-TLS

```
All agents $k \in \mathcal{K}$ initialize $\beta_k(1) = \mathrm{tr}(\mathbf{C}_k)/(p+1)$ and locally run
for $j = 1, 2, \ldots$ do
    Initialize $\mathbf{W}_k(0) = \mathbf{0}$ and $\mathbf{\Lambda}_k(0) = \mathbf{0}$
    for $m = 1, 2, \ldots$ do
        Receive $\mathbf{W}_k(m)$ from neighbors in $\mathcal{N}_k$
        Update $\mathbf{\Lambda}_k(m+1)$ as in (14)
        Compute $\mathbf{G}_k(m)$ as in (16)
        Compute EVD of $\mathbf{G}_k(m) = \mathbf{U}(m)\mathbf{\Sigma}(m)\mathbf{U}^{\mathrm{T}}(m)$
        Update $\mathbf{W}_k(m+1) = \mathbf{U}(m)\max\left(\mathbf{\Sigma}(m), \mathbf{0}\right)\mathbf{U}^{\mathrm{T}}(m)$
    end for
    Update $\beta_k(j+1) = \frac{\mathrm{tr}(\mathbf{C}_k\mathbf{W}_k(m+1))}{\mathrm{tr}(\mathbf{W}_k(m+1))}$
end for
```

After estimating $\mathbf{W}_k$, the vector estimate $\mathbf{w}_k$ is found as follows. Let $\breve{\mathbf{W}}_k = \mathbf{W}_k/\omega_k$ where ω_k is the $(P+1), (P+1)$ entry of $\mathbf{W}_k$. Then, $\mathbf{w}_k$ is the eigenvector corresponding to the smallest eigenvalue of the $P \times P$ upper-left submatrix of $\breve{\mathbf{W}}_k$.

Using the results in [24, 28], it can be observed that the solution of (7) and consequently (8) is rank-one. Hence, optimizing with respect to the matrix variable $\mathbf{W}$ and relaxing the rank constraint do not lead to any loss of optimality [24]. Therefore, the solutions to (3) and (10) coincide.

Convergence of the proposed DA-TLS algorithm to the global centralized solution can be proven by checking that both inner and outer loops converge. The convergence of the inner loop can be verified following [29, Proposition 3], i.e., for all $k \in \mathcal{K}$, the iterates $\{\mathbf{W}_k(m)\}$, $\{\mathbf{\Lambda}_k(m)\}$ produced by (13) and (14) are convergent and $\mathbf{W}_k(m) \to \mathbf{W}^*$ as $m \to \infty$. Moreover, the convergence of the outer loop follows setting $\bar{\mathbf{C}} = \sum_{k=1}^{K} \mathbf{C}_k$ and $\bar{\beta}^* = \sum_{k=1}^{K} \beta_k^*$ and observing that the optimization in (8) is equivalent to

$$\min_{\mathbf{W} \succeq \mathbf{0}} \operatorname{tr}(\bar{\mathbf{C}}\mathbf{W}) - \bar{\beta}^* \operatorname{tr}(\mathbf{W}). \tag{18}$$

Since the objective function in (18) is linear, (18) is a standard semidefinite program with a unique solution. Therefore, DA-TLS naturally inherits the theoretical properties of the algorithm proposed in [23] for fractional programming whose convergence is guaranteed.

4. SIMULATIONS

The simulated network is connected with a random topology and consists of $K = 20$ agents where each agent is linked to three other agents on average. We average results over 100 independent trials. In each trial, the scenario is generated according to the same procedure as described in the simulation sections of [13, 14]. For each agent $k \in \mathcal{K}$, we create a $2P \times P$ local observation matrix $\mathbf{X}_k$ whose entries are drawn from a standard normal distribution. The entries of the parameter vector $\mathbf{w}$ are also drawn from a standard normal distribution. The entries of the error matrix $\mathbf{\Delta}$ and error vector δ are i.i.d. zero-mean Gaussian with variance 0.25. To evaluate the performance of the proposed algorithm, we use the normalized error between the centralized TLS solution $\mathbf{w}^c$ as per (2) and the local estimates that is defined as $\sum_{k=1}^{K} \|\mathbf{w}_k - \mathbf{w}^c\|^2 / \|\mathbf{w}^c\|^2$ where $\mathbf{w}_k$ denotes the local estimate at agent k. In Figs. 1-2, we plot the normalized error versus the total number of iterations, which is given by the product between the number of iterations of the inner and the outer loop. The former is set to 80 for Fig. 1 and 40 for Fig. 2, while the latter is set to 5 for both the plots.

Fig. 1 shows that, for $P = 9$, DA-TLS with $\rho = 2$ and $\rho = 3$ converges significantly faster than the existing approaches, i.e., the distributed TLS (D-TLS) algorithm of [13] and the inverse-power-iteration-based distributed TLS (IPI-D-TLS) algorithm of [14]. Fig. 2 shows the superiority of DA-TLS with $\rho = 1$ over IPI-D-TLS with $\mu = 1$ for two different values of P. Although not further substantiated here due to the space constraints, we have observed that DA-TLS consistently outperforms its contenders in various scenarios.

5. CONCLUSION

In this paper, we developed a new distributed algorithm for solving the TLS problem. We recast the original optimization problem into an equivalent linear-fractional program. Then, employing semidefinite relaxation, we transformed the resultant problem into a parametric semidefinite program whose structure is suitable for distributed treatment via ADMM. Simulation results showed that the proposed algorithm converges faster than the existing alternative algorithms while being less sensitive to tuning of the parameters involved in the algorithm.

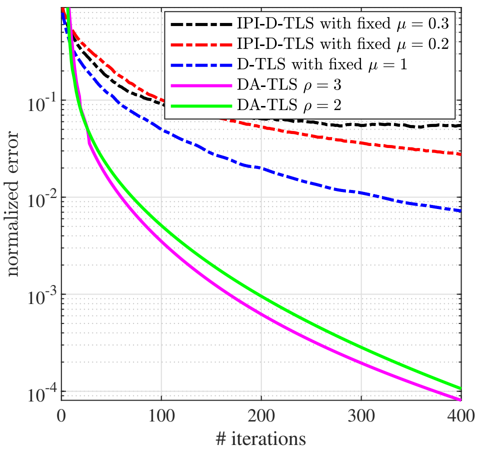


Fig. 1. Normalized error of the DA-TLS, D-TLS, and IPI-D-TLS algorithms with two values of penalty parameter ($\rho = 2$ and $\rho = 3$) for DA-TLS and two values of the step-size ($\mu = 0.2$ and $\mu = 0.3$) for IPI-D-TLS.

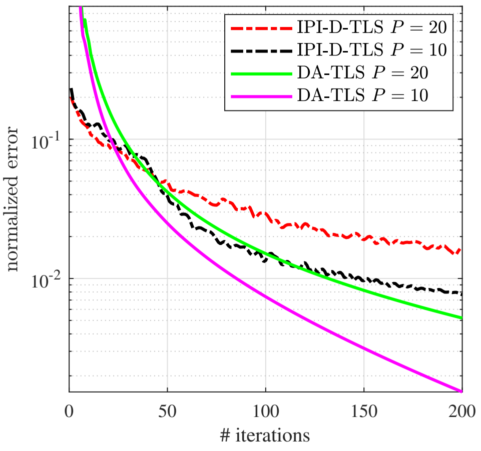


Fig. 2. Normalized error for different values of P. For DA-TLS, we set $\rho = 1$ and, for IPI-D-TLS, we set $\mu = 1$.

6. REFERENCES

[1] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, *Splitting Methods in Communication, Imaging, Science, and Engineering*, ser. Scientific Computation, R. Glowinski, S. J. Osher, and W. Yin, Eds. Cham: Springer International Publishing, 2016.

[2] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, Oct 2010.

[3] S. Kumar, R. Jain, and K. Rajawat, "Asynchronous optimization over heterogeneous networks via consensus admm," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 114–129, Mar 2017.

[4] J. Akhtar and K. Rajawat, "Distributed sequential estimation in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 86–100, Jan 2018.

[5] N. K. D. Venkategowda and S. Werner, "Privacy-preserving distributed precoder design for decentralized estimation," in *Proc. IEEE Global Conference on Signal and Information Processing*, Nov 2018.

[6] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed ridge regression with feature partitioning," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Oct 2018.

[7] S. P. Talebi, S. Werner, and D. P. Mandic, "Distributed adaptive filtering of α-stable signals," *IEEE Signal Processing Letters*, vol. 25, no. 10, pp. 1450–1454, Oct 2018.

[8] R. Arablouei, K. Doganay, and S. Werner, "Recursive total least-squares estimation of frequency in three-phase power systems," in *Proc. 22nd European Signal Processing Conference*, Sept 2014, pp. 2330–2334.

[9] R. Arablouei, K. Doğançay, and S. Werner, "Adaptive frequency estimation of three-phase power systems," *Signal Processing*, vol. 109, pp. 290–300, Apr 2015.

[10] R. Arablouei, S. Werner, and K. Doğançay, "Estimating frequency of three-phase power systems via widely-linear modeling and total least-squares," in *Proc. 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Dec 2013, pp. 464–467.

[11] E. Dall'Anese and G. B. Giannakis, "Distributed cognitive spectrum sensing via group sparse total least-squares," in *Proc. 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Dec 2011, pp. 341–344.

[12] I. Markovsky, J. Willems, S. Van Huffel, Bart De Moor, and R. Pintelon, "Application of structured total least squares for system identification and model reduction," *IEEE Transactions on Automatic Control*, vol. 50, no. 10, pp. 1490–1500, Oct 2005.

[13] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2320–2330, May 2011.

[14] ——, "Low-complexity distributed total least squares estimation in ad hoc sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4321–4333, Aug 2012.

[15] S. Huang and C. Li, "Distributed sparse total least-squares over networks," *IEEE Trans. Signal Process.*, vol. 63, no. 11, pp. 2986–2998, Jun 2015.

[16] R. Lopez-Valcarce, S. S. Pereira, and A. Pages-Zamora, "Distributed Total Least Squares estimation over networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2014, pp. 7580–7584.

[17] R. Arablouei, S. Werner, and K. Doğançay, "Diffusion-based distributed adaptive estimation utilizing gradient-descent total least-squares," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 5308–5312.

[18] R. Abdolee and B. Champagne, "Diffusion LMS strategies in sensor networks with noisy input data," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 3–14, Feb 2016.

[19] L. Lu, H. Zhao, and B. Champagne, "Diffusion total least-squares algorithm with multi-node feedback," *Signal Processing*, Jul 2018.

[20] S. Silva Pereira, A. Pages-Zamora, and R. Lopez-Valcarce, "Distributed TLS estimation under random data faults," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr 2015, pp. 2949–2953.

[21] C. Li, S. Huang, Y. Liu, and Y. Liu, "Distributed TLS over multitask networks with adaptive intertask cooperation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 6, pp. 3036–3052, Dec 2016.

[22] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, Jul 2008.

[23] W. Dinkelbach, "On nonlinear fractional programming," *Managemeny Science*, vol. 13, no. 7, pp. 492–498, Mar 1967.

[24] Z.-Q. Luo, W.-K. Ma, A. So, Y. Ye, and S. Zhang, "Semidefinite relaxation of quadratic optimization Problems," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, May 2010.

[25] R. Arablouei, S. Werner, and K. Doğançay, "Analysis of the gradient-descent total least-squares adaptive filtering algorithm," *IEEE Transactions on Signal Processing*, vol. 62, no. 5, pp. 1256–1264, Mar 2014.

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[27] J.-P. Crouzeix and J. A. Ferland, "Algorithms for generalized fractional programming," *Mathematical Programming*, vol. 52, no. 1, pp. 191–207, May 1991.

[28] G. Pataki, "On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues," *Mathematics of Operations Research*, vol. 23, no. 2, pp. 339–358, Jan. 1998.

[29] E. Dall'Anese, H. Zhu, and G. B. Giannakis, "Distributed optimal power flow for smart microgrids," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1464–1475, Sep 2013.

Distributed Learning with Non-Smooth Objective Functions

Cristiano Gratton*, Naveen K. D. Venkategowda*, Reza Arablouei†, Stefan Werner*
* Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway
† CSIRO's Data61, Pullenvale QLD 4069, Australia

***Abstract*—We develop a new distributed algorithm to solve a learning problem with non-smooth objective functions when data are distributed over a multi-agent network. We employ a zeroth-order method to minimize the associated augmented Lagrangian in the primal domain using the alternating direction method of multipliers (ADMM) to develop the proposed algorithm, named distributed zeroth-order based ADMM (D-ZOA). Unlike most existing algorithms for non-smooth optimization, which rely on calculating subgradients or proximal operators, D-ZOA only requires function values to approximate gradients of the objective function. Convergence of D-ZOA to the centralized solution is confirmed via theoretical analysis and simulation results.**

I. Introduction

Performing learning tasks at a central processing unit in a large distributed network can be prohibitive due to communication/computation costs or privacy issues. Therefore, it is important to develop algorithms that are able to distributedly process the data collected by agents scattered over a large geographical area [1]–[8]. In this context, each agent in the network only possesses information of a local cost function and the agents aim to collaboratively minimize the sum of the local objective functions. Such optimization problems are relevant to several applications in statistics [3]–[5], signal processing [6]–[8] and control [1], [2].

There have been several works developing algorithms for solving distributed convex optimization problems over ad-hoc networks. However, many existing algorithms only offer solutions for problems with smooth objective functions, see, e.g., [5], [9], [10]. Distributed optimization problems with non-smooth objectives have been considered in [1], [2], [4], [11]–[16]. The approaches taken in [2], [11], [12] are based on subgradient methods. The works of [13], [14] are based on dual decomposition techniques while the algorithms in [4], [15] are developed using soft-thresholding operations. However, all the aforementioned algorithms require either the computation of subgradients, which might be hard to achieve for some objectives, or derivation of proximal operators, which might not be feasible in some scenarios.

Moreover, there are some real-world problems where obtaining first-order information is impossible due to the lack of the complete loss function. For example, in bandit optimization [17], an adversary generates a sequence of loss functions and the goal is to minimize such sequence that is only available at some points. In addition, in simulation-based optimization, the objective is available only using repeated simulation [18], and in adversarial black-box machine learning models, only the function values are given [19]. This motivates the use of zeroth-order methods requiring only function values to approximate gradients.

The works in [1], [16] are based on zeroth-order methods within the distributed optimization setting. While the approach of [16] relies on approximate projections for dealing with constraints, the algorithm ZONE-S proposed in [1] is based on a primal-dual approach and deals with non-convex objectives. However, ZONE-S addresses only consensus problems with a non-smooth regularization that is handled by a central collector making the algorithm not fully distributed.

In this paper, we develop a fully-distributed algorithm to solve an optimization problem with a non-smooth convex objective function over an ad-hoc network. We utilize the alternating direction method of multipliers (ADMM) for distributed optimization. Furthermore, we employ the zeroth-order method called the two-point stochastic gradient algorithm [20] that is suitable for non-smooth objectives to obtain an approximate minimizer of the augmented Lagrangian in the ADMM's primal update step. The proposed algorithm, called distributed zeroth-order based ADMM (D-ZOA), is fully distributed in the sense that each agent in the network communicates only with its neighbors and no central coordinator is necessary. Furthermore, D-ZOA does not compute any subgradient and only requires the objective function values to approximate the gradient of the augmented Lagrangian. The simulations show that D-ZOA is competitive even on a problem that can be easily solved with a subgradient-based algorithm. Furthermore, the experiments show the usefulness of D-ZOA on a problem where calculating any subgradient is impractical. Convergence of D-ZOA to the centralized solution at all agents is verified through theoretical analysis and simulation results.

Mathematical Notations: The set of natural and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. Scalars, column vectors and matrices are respectively denoted by lowercase, bold lowercase, and bold uppercase letters. The operators $(\cdot)^{\mathsf{T}}$ and $\mathrm{tr}(\cdot)$ denote transpose and trace of a matrix, respectively. $\mathbf{I}_p$ denotes an identity matrix of size p, $\mathbf{0}_{q\times l}$ defines a matrix with all zero entries, and $\otimes$ stands for the Kronecker product. The statistical expectation and covariance operators are represented by $\mathbb{E}[\cdot]$ and $\mathrm{cov}[\cdot]$, respectively. For a vector $\mathbf{y}$ and a matrix $\mathbf{Y} \in \mathbb{R}^{r\times s}$, $\|\mathbf{y}\|_{\mathbf{Y}}$ denotes the quadratic form $\mathbf{y}^{\mathsf{T}}\mathbf{Y}\mathbf{y}$. The

This work was partly supported by the Research Council of Norway.

nuclear norm of $\mathbf{Y}$ is denoted by $\|\mathbf{Y}\|_*$ and is defined as

$$\|\mathbf{Y}\|_* = \sum_{i=1}^{\min\{r,s\}} \sigma_i(\mathbf{Y})$$

where $\sigma_i(\mathbf{Y})$ denotes the ith singular value of $\mathbf{Y}$. $\|\cdot\|$ and $\|\cdot\|_F$ represent the Euclidean norm and the Frobenius norm, respectively. The operators $\mathrm{vec}(\mathbf{Y})$ forms a column vector from the matrix $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_s]$ by stacking the column vectors $\mathbf{y}_i$. For a positive semidefinite matrix $\mathbf{X}$, $\lambda_{\min}(\mathbf{X})$ and $\lambda_{\max}(\mathbf{X})$ denote the nonzero smallest and largest eigenvalues of $\mathbf{X}$, respectively.

II. System Model

We consider a network with $K \in \mathbb{N}$ agents and $E \in \mathbb{N}$ edges that is modeled as an undirected graph $\mathcal{G}(\mathcal{K}, \mathcal{E})$, where the set of vertices $\mathcal{K} = \{1, \ldots, K\}$ corresponds to the agents and the set $\mathcal{E}$ represents the bidirectional communication links between the pairs of agents. Agent $k \in \mathcal{K}$ can communicate only with the agents in its neighborhood $\mathcal{N}_k$ whose cardinality is denoted by $|\mathcal{N}_k|$. By convention, the set $\mathcal{N}_k$ includes the agent k as well.

We consider the problem when the K agents of the network solve the following minimization problem collaboratively

$$\min_{\mathbf{x}} \sum_{k=1}^{K} f_k(\mathbf{x}; \mathcal{X}_k) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^P$ is the unknown model parameter, $\mathcal{X}_k$ represents the local information at agent k, and $f_k : \mathbb{R}^P \to \mathbb{R}$ is the local cost function that is convex but *non-smooth*. Let us denote the solution to (1) by $\mathbf{x}^c$.

III. Non-Smooth Distributed Learning

We first discuss the consensus-based reformulation of the problem that allows its distributed solution through an iterative process consisting of two nested loops. Then, we describe the ADMM procedure that forms the outer loop and the zeroth-order two-point stochastic gradient algorithm that constitutes the inner loop solving the ADMM primal update step. Finally, we establish the convergence of D-ZOA theoretically.

A. Consensus-Based Reformulation

To solve (1) in a distributed fashion, we introduce the primal variables $\mathcal{V} := \{\mathbf{x}_k\}_{k=1}^K$ that represent local copies of $\mathbf{x}$ at the agents. Then, we reformulate (1) as the following constrained minimization problem:

$$\begin{aligned} \min_{\{\mathbf{x}_k\}} \quad & \sum_{k=1}^{K} f_k(\mathbf{x}_k; \mathcal{X}_k) \\ \text{s.t.} \quad & \mathbf{x}_k = \mathbf{x}_l, \quad l \in \mathcal{N}_k, \quad \forall k \in \mathcal{K}. \end{aligned} \tag{2}$$

Since the network is connected, the equality constraints in (2) enforce consensus over $\{\mathbf{x}_k\}_{k=1}^K$ by imposing consensus across each agent's neighborhood $\mathcal{N}_k$. To solve (2) in a distributed fashion, we employ the ADMM [8]. Therefore, we introduce the auxiliary variables $\mathcal{Z} := \{\mathbf{z}_k^l\}_{l \in \mathcal{N}_k}$ and rewrite (2) as

$$\begin{aligned} \min_{\{\mathbf{x}_k\}} \quad & \sum_{k=1}^{K} f_k(\mathbf{x}_k; \mathcal{X}_k) \\ \text{s.t.} \quad & \mathbf{x}_k = \mathbf{z}_k^l,\ \mathbf{x}_l = \mathbf{z}_k^l, \quad l \in \mathcal{N}_k, \quad \forall k \in \mathcal{K}. \end{aligned} \tag{3}$$

The use of auxiliary variables $\mathcal{Z}$ renders an equivalent representation of the constraints in (2). These variables are only used to derive the local recursions and are eventually eliminated. The augmented Lagrangian function is given by

$$\begin{aligned} \mathcal{L}_\rho(\mathcal{V}, \mathcal{Z}, \mathcal{M}) &= \sum_{k=1}^{K} f_k(\mathbf{x}_k; \mathcal{X}_k) \\ &+ \sum_{k=1}^{K} \sum_{l \in \mathcal{N}_k} \left[\boldsymbol{\mu}_k^{l\mathsf{T}} \left(\mathbf{x}_k - \mathbf{z}_k^l\right) + \boldsymbol{\lambda}_k^{l\mathsf{T}} \left(\mathbf{x}_l - \mathbf{z}_k^l\right) \right] \\ &+ \frac{\rho}{2} \sum_{k=1}^{K} \sum_{l \in \mathcal{N}_k} \left(\left\|\mathbf{x}_k - \mathbf{z}_k^l\right\|^2 + \left\|\mathbf{x}_l - \mathbf{z}_k^l\right\|^2 \right) \end{aligned} \tag{4}$$

where $\mathcal{M} := \{\{\boldsymbol{\mu}_k^l\}_{l \in \mathcal{N}_k}, \{\boldsymbol{\lambda}_k^l\}_{l \in \mathcal{N}_k}\}_{k=1}^K$ are the Lagrange multipliers associated with (3), and $\rho > 0$ is a penalty parameter.

Solving (3) via the ADMM requires an iterative process that is described in the next subsection.

B. Distributed ADMM Algorithm

To solve the minimization problem (3) in a distributed fashion, the ADMM entails an iterative procedure consisting of three steps at each iteration. In the first step, $\mathcal{L}_\rho$ is minimized with respect to the primal variables $\mathcal{V}$. Then, $\mathcal{L}_\rho$ is minimized with respect to the auxiliary variables $\mathcal{Z}$. In the end, the Lagrange multipliers in $\mathcal{M}$ are updated via dual gradient-ascent iterations [8]. By using the Karush-Kuhn-Tucker conditions of optimality for (3) and setting $\boldsymbol{\lambda}_k(m) = 2\sum_{l \in \mathcal{N}_k} \boldsymbol{\lambda}_k^l(m)$, 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 [8]. Therefore, the distributed ADMM algorithm reduces to the following iterative updates at the kth agent

$$\mathbf{x}_k(m+1) = \arg\min_{\mathbf{x}_k} \mathcal{L}_\rho(\mathbf{x}_k, \boldsymbol{\lambda}_k(m)) \tag{5}$$

$$\boldsymbol{\lambda}_k(m+1) = \boldsymbol{\lambda}_k(m) + \rho \sum_{l \in \mathcal{N}_k} [\mathbf{x}_k(m+1) - \mathbf{x}_l(m+1)] \tag{6}$$

where m is the iteration index and all initial values $\{\mathbf{x}_k(0)\}_{k \in \mathcal{K}}$, $\{\boldsymbol{\lambda}_k(0)\}_{k \in \mathcal{K}}$ are set to zero. The iterations (5) and (6) can be implemented in a fully distributed manner as they only involve the parameters available within each node's neighborhood.

The objective function of the minimization problem in (5) is non-smooth, which makes it hard to obtain a solution using first-order information. To solve this problem, we employ a zeroth-order method described in the next subsection.

Algorithm 1 D-ZOA

```
At all agents $k \in \mathcal{K}$, initialize $\mathbf{x}_k(0) = \mathbf{0}$, $\boldsymbol{\lambda}_k(0) = \mathbf{0}$, and locally run
for $m = 1, 2, \dots$ do
    Receive $\mathbf{x}_k(m)$ from neighbors in $\mathcal{N}_k$
    Update $\boldsymbol{\lambda}_k(m+1)$ as in (6)
    Initialize $\mathbf{x}_k^0 = \mathbf{0}$
    for $t = 1, 2, \dots, T$ do
        Draw independent $\boldsymbol{\nu}_1$, $\boldsymbol{\nu}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_P)$
        Set $u_1^t = u_1/t$, $u_2^t = u_1/t^2$ and compute $\mathbf{g}^t$ as in (8)
        Update $\mathbf{x}_k^{t+1}$ as in (9)
    end for
    Update $\mathbf{x}_k(m+1) = \mathbf{x}_k^{T+1}$
end for
```

C. Zeroth-Order Method

In order to solve (5) utilizing a zeroth-order method, we assume that $\mathcal{L}_\rho(\cdot)$ is closed and Lipschitz-continuous with the Lipschitz constant G. These assumptions are common for zeroth-order optimization, see, e.g., [1], [20].

Subsequently, we employ the two-point stochastic gradient algorithm for general non-smooth functions proposed in [20]. More specifically, we use the stochastic mirror descent method with the proximal function $\|\cdot\|/2$ and the gradient estimator at point $\mathbf{x}_k$ given by

$$G_{\text{ns}}(\mathbf{x}_k; u_1, u_2, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2, \boldsymbol{\lambda}_k(m)) = u_2^{-1}[\mathcal{L}_\rho(\mathbf{x}_k + u_1\boldsymbol{\nu}_1 + u_2\boldsymbol{\nu}_2, \boldsymbol{\lambda}_k(m)) - \mathcal{L}_\rho(\mathbf{x}_k + u_1\boldsymbol{\nu}_1, \boldsymbol{\lambda}_k(m))]\boldsymbol{\nu}_2 \tag{7}$$

where $u_1 > 0$ and $u_2 > 0$ are smoothing constants and $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2$ are zero-mean Gaussian random vectors independent of each other with covariance matrix $\mathbf{I}_P$, i.e., $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_P)$.

The two-point stochastic gradient algorithm entails an iterative procedure that consists of three steps at each iteration t. First, independent random vectors $\boldsymbol{\nu}_1^t$ and $\boldsymbol{\nu}_2^t$ are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$. Second, a stochastic gradient $\mathbf{g}^t$ is formed as

$$\mathbf{g}^t = G_{\text{ns}}(\mathbf{x}_k^t; \boldsymbol{\lambda}_k(m), u_1^t, u_2^t, \boldsymbol{\nu}_1^t, \boldsymbol{\nu}_2^t) \tag{8}$$

where $\mathbf{x}_k^t$ is the tth iterate of the two-point stochastic gradient algorithm with the initial point $\mathbf{x}_k = \mathbf{0}$, $\{u_1^t\}_{t=1}^\infty$ and $\{u_2^t\}_{t=1}^\infty$ are two non-increasing sequences of positive parameters such that $u_2^t \le u_1^t/2$. Finally, $\mathbf{x}_k^{t+1}$ is updated as

$$\mathbf{x}_k^{t+1} = \mathbf{x}_k^t - \alpha(t)\mathbf{g}^t \tag{9}$$

where the time-dependent step-size $\alpha(t)$ is set as $\alpha(t) = (G\sqrt{tP\log(2P)})^{-1}\alpha_0 R$, α_0 is an appropriate initial step-size and R is an upper bound for the distance between a minimizer $\mathbf{x}_k^*$ to (5) and the first iterate $\mathbf{x}_k^1$ as per [20].

Note that no communication among agents is involved throughout the inner loop.

The proposed algorithm, D-ZOA, is summarized in Algorithm 1.

In the next subsection, we show that the D-ZOA produces sequences of local iterates $\mathbf{x}_k(m)$, $k \in \mathcal{K}$, that converge to the global centralized solution $\mathbf{x}^c$ as $m \to \infty$.

D. Convergence Analysis

The convergence of D-ZOA to the centralized solution is established by corroborating that both inner and outer loops of the algorithm converge.

The convergence of the inner loop can be proven following [20, Theorem 2], i.e., it can be shown that there exists a numerical constant c such that, for each T representing a fixed number of iterations of the inner loop, the following inequality holds:

$$\begin{aligned}&\mathbb{E}[\mathcal{L}_\rho(\hat{\mathbf{x}}_k(T)) - \mathcal{L}_\rho(\mathbf{x}_k^*)] \\ &\le c\frac{RG\sqrt{P}}{\sqrt{T}}\Big[\max\{\alpha_0, \alpha_0^{-1}\}\sqrt{\log(2P)} + \frac{u_1\log(2T)}{\sqrt{T}}\Big]\end{aligned} \tag{10}$$

where $\hat{\mathbf{x}}_k(T) = T^{-1}\sum_{t=1}^T \mathbf{x}_k^t$. In [20], it is shown that $c = 0.5$ whenever $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ are sampled from a normal distribution.

The convergence of the outer loop can be verified by proving the convergence of a fully distributed ADMM with inexact primal updates. For this purpose, the primal variable can be assumed to be a perturbed version of the exact primal update as per [21]. Therefore, $\mathbf{x}_k(m+1)$ can be written as

$$\mathbf{x}_k(m+1) = \bar{\mathbf{x}}_k(m+1) + \boldsymbol{\gamma}_k(m+1) \tag{11}$$

where $\bar{\mathbf{x}}_k(m+1)$ is the exact ADMM primal update and $\boldsymbol{\gamma}_k(m+1)$ is a random variable representing the perturbation of $\bar{\mathbf{x}}_k(m+1)$. Similar to [21], we assume the perturbation to have zero expectation, i.e., $\mathbb{E}[\boldsymbol{\gamma}_k(m+1)] = \mathbf{0}$, $\forall k \in \mathcal{K}$ and for all the ADMM iterations m, and have finite covariance matrix, i.e., $\text{cov}[\boldsymbol{\gamma}_k(m+1)]_{i,j} < \infty$, $\forall k \in \mathcal{K}$, $\forall i, j = 1, \dots, P$ and for all the ADMM iterations m.

For a clear presentation of the convergence results, we rewrite (3) in the matrix form. By defining $\tilde{\mathbf{x}} \in \mathbb{R}^{KP}$ as a vector concatenating all $\mathbf{x}_k$ and $\tilde{\mathbf{z}} \in \mathbb{R}^{2EP}$ concatenating all $\mathbf{z}_k^l$, (3) can be written as

$$\begin{aligned}\min_{\tilde{\mathbf{x}}, \tilde{\mathbf{z}}} \quad & f(\tilde{\mathbf{x}}) + g(\tilde{\mathbf{z}}) \\ \text{s.t.} \quad & \mathbf{A}\tilde{\mathbf{x}} + \mathbf{B}\tilde{\mathbf{z}} = \mathbf{0}\end{aligned} \tag{12}$$

where $f(\tilde{\mathbf{x}}) = \sum_{k=1}^K f_k(\mathbf{x}_k; \mathcal{X}_k)$, $g(\tilde{\mathbf{z}}) = 0$, $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2]$, and $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{2EP \times KP}$ are both composed of $2E \times K$ blocks of $P \times P$ matrices. If $(k, l) \in \mathcal{E}$ and $\mathbf{z}_k^l$ is the qth block of $\tilde{\mathbf{z}}$, then the (q, k)th block of $\mathbf{A}_1$ and the (q, l)th block of $\mathbf{A}_2$ are identity matrices $\mathbf{I}_P$. Otherwise, the corresponding blocks are $P \times P$ zero matrices $\mathbf{0}_P$. Furthermore, we have $\mathbf{B} = [-\mathbf{I}_{2EP}; -\mathbf{I}_{2EP}]$. We define the matrices $\mathbf{M}_+ = \mathbf{A}_1^\mathsf{T} + \mathbf{A}_2^\mathsf{T}$ and $\mathbf{M}_- = \mathbf{A}_1^\mathsf{T} - \mathbf{A}_2^\mathsf{T}$, $\mathbf{L}_+ = 0.5\mathbf{M}_+\mathbf{M}_+^\mathsf{T}$, $\mathbf{L}_- = 0.5\mathbf{M}_-\mathbf{M}_-^\mathsf{T}$, $\mathbf{Q} = \sqrt{0.5\mathbf{L}_-}$ and $\boldsymbol{\gamma}(m+1) \in \mathbb{R}^{KP}$ as the vector concatenating all $\boldsymbol{\gamma}_k(m+1)$.

We construct the auxiliary sequence $\mathbf{r}(m) = \sum_{s=0}^m \mathbf{Q}\tilde{\mathbf{x}}(s)$ and define the auxiliary vector $\mathbf{q}(m)$ and the auxiliary matrix $\mathbf{G}$ as

$$\mathbf{q}(m) = \begin{bmatrix}\mathbf{r}(m) \\ \tilde{\mathbf{x}}(m)\end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix}\rho\mathbf{I}_P & \mathbf{0}_{P\times P} \\ \mathbf{0}_{P\times P} & \rho\frac{\mathbf{L}_+}{2}\end{bmatrix}. \tag{13}$$

The convergence results of [21] can now be adapted to D-ZOA as per the following theorem.

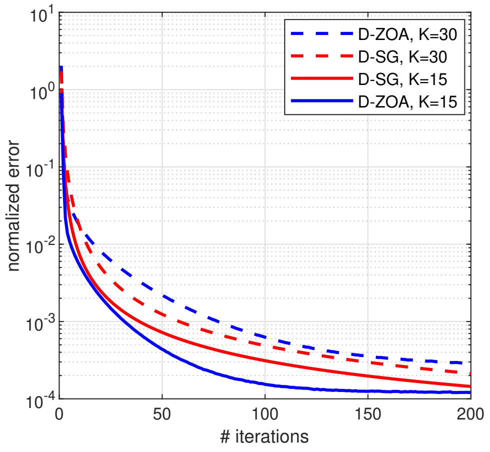


Fig. 1. Normalized error of D-ZOA and D-SG for generalized lasso with $P = 10$, $\rho = 3$ and two different values of K.

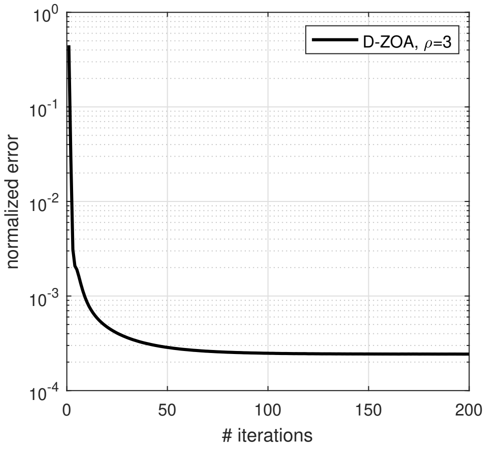


Fig. 2. Normalized error of D-ZOA for RRR with $P = 5$, $S = 4$, $\rho = 3$ and $K = 10$.

Theorem 1. *If $f(\cdot)$ is convex, then, for any fixed number of iterations N of the outer loop, we have*

$$\begin{aligned}&\mathbb{E}[f(\hat{\mathbf{x}}^N) - f(\tilde{\mathbf{x}}^*)]\\ \leq&\frac{\|\mathbf{q}(0) - \mathbf{q}\|^2_{\mathbf{G}}}{N} + \frac{\rho\lambda^2_{max}(\mathbf{L}_+)\sum_{m=0}^{N-1} tr\,(cov[\boldsymbol{\gamma}(m)])}{2N\lambda_{min}(\mathbf{L}_-)}\end{aligned} \tag{14}$$

where the expectation is taken with respect to the perturbation, $\tilde{\mathbf{x}}^*$ is the optimal solution of (12) *and $\hat{\mathbf{x}}^N = \frac{1}{N}\sum_{m=0}^{N-1}\tilde{\mathbf{x}}(m+1)$.*

Proof. Since $\mathbb{E}[\boldsymbol{\gamma}_k(m)] = \mathbf{0}$ and $\text{cov}[\boldsymbol{\gamma}_k(m)]_{i,j} < \infty, \forall k \in \mathcal{K}$, $\forall i, j = 1, \dots, P$ and for all the ADMM iterations m, proof follows from [21, Lemma 6] and [21, Theorem 5]. □

IV. Simulations

The D-ZOA algorithm is tested on a multi-agent network with a random topology, where each agent is linked to three other agents on average. To benchmark D-ZOA with existing solutions, we consider a distributed version of the generalized lasso [15] that can be solved with subgradient methods [2]. Furthermore, we consider a distributed version of the reduced-rank regression (RRR) problem where the objective function is least squares with nuclear norm regularization [8]. Nuclear norm is a non-smooth function that is used as a convex surrogate for the rank. Calculating any subgradient of the nuclear norm function is impractical. RRR has applications in robust PCA [22], low-rank matrix decomposition [23], matrix completion [24], etc.

The network-wide observations are represented as an observation matrix $\mathbf{D} \in \mathbb{R}^{M\times P}$ and a response matrix $\mathbf{H} \in \mathbb{R}^{M\times S}$, where M is the number of data samples and P is the number of features in each sample. The matrix $\mathbf{D}$ consists of K submatrices $\mathbf{D}_k$, i.e., $\mathbf{D} = [\mathbf{D}_1^\mathsf{T}, \mathbf{D}_2^\mathsf{T}, \dots, \mathbf{D}_K^\mathsf{T}]^\mathsf{T}$, and the matrix $\mathbf{H}$ of K submatrices $\mathbf{H}_k$, i.e., $\mathbf{H} = \left[\mathbf{H}_1^\mathsf{T}, \mathbf{H}_2^\mathsf{T}, \dots, \mathbf{H}_K^\mathsf{T}\right]^\mathsf{T}$, as the data are distributed among the agents and each agent k holds its respective $\mathbf{D}_k \in \mathbb{R}^{M_k\times P}$ and $\mathbf{H}_k \in \mathbb{R}^{M_k\times S}$ where $\sum_{k=1}^K M_k = M$. The parameter matrix that establishes a linear regression between $\mathbf{D}$ and $\mathbf{H}$ is $\mathbf{X} \in \mathbb{R}^{P\times S}$. In the generalized lasso, $S = 1$ and, hence, $\mathbf{H}$ is the vector $\mathbf{h} \in \mathbb{R}^M$ and $\mathbf{X}$ becomes $\mathbf{x} \in \mathbb{R}^P$. In the centralized approach, a generalized lasso estimate of $\mathbf{x}$ is given by

$$\mathbf{x}^c = \arg\min_{\mathbf{x}}\{\|\mathbf{D}\mathbf{x} - \mathbf{b}\|^2 + \eta\,\|\mathbf{F}\mathbf{x}\|_1\} \tag{15}$$

where $\eta > 0$ is a regularization parameter and $\mathbf{F}$ is an arbitrary matrix. An RRR estimate of $\mathbf{X}$ is also given by

$$\mathbf{X}^c = \arg\min_{\mathbf{X}}\{\|\mathbf{D}\mathbf{X} - \mathbf{H}\|^2 + \eta_*\,\|\mathbf{X}\|_*\} \tag{16}$$

where $\eta_* > 0$ is a rank-controlling parameter. In the distributed setting, we solve problem (2) with

$$f_k(\mathbf{x}_k; \mathcal{X}_k) = \|\mathbf{D}_k\mathbf{x}_k - \mathbf{h}_k\|^2 + \frac{\eta}{K}\,\|\mathbf{F}\mathbf{x}_k\|_1 \tag{17}$$

for the generalized lasso case and with

$$f_k(\mathbf{X}_k; \mathcal{X}_k) = \|\mathbf{D}_k\mathbf{X}_k - \mathbf{H}_k\|^2 + \frac{\eta}{K}\,\|\mathbf{X}_k\|_* \tag{18}$$

for the RRR case. For each agent $k \in \mathcal{K}$, we create a $10P \times P$ local observation matrix $\mathbf{D}_k$ whose entries are independent identically distributed zero-mean unit-variance Gaussian random variables. The response vector $\mathbf{h}$ is obtained as

$$\mathbf{h} = \mathbf{D}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\beta} \in \mathbb{R}^P$ and $\boldsymbol{\epsilon} \in \mathbb{R}^M$ are chosen as random vector with distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$. The response matrix $\mathbf{H}$ is obtained as

$$\mathbf{H} = \mathbf{D}\boldsymbol{\Phi} + \boldsymbol{\Psi}$$

where $\mathbf{\Phi} \in \mathbb{R}^{P\times S}$ and $\mathbf{\Psi} \in \mathbb{R}^{M\times S}$ are random matrices with matrix normal distributions $\mathcal{MN}(\mathbf{0}_{P\times S}, \mathbf{I}_P, \mathbf{I}_S)$ and $\mathcal{MN}(\mathbf{0}_{M\times S}, 0.1\mathbf{I}_M, 0.1\mathbf{I}_S)$, respectively. The regularization parameter η is set to $0.01\left\|\mathbf{D}^{\mathsf{T}}\mathbf{b}\right\|_\infty$ and η_* is set to $0.01\left\|(\mathbf{I}_S\otimes\mathbf{D})^{\mathsf{T}}\text{vec}(\mathbf{H})\right\|_\infty$ as in [15]. The number of iterations of the ADMM outer loop is set to 200. For the inner loop, the number of iterations is set to 1000, the smoothing constant u_1 is set to 1 and the convergence in mean is achieved by averaging the outputs of 10 inner loops. Performance of D-ZOA is evaluated using the normalized error between the centralized solutions $\mathbf{x}^c$ as per (15) or $\mathbf{X}^c$ as per (16) and the local estimates. It is defined as $\sum_{k=1}^{K}\|\mathbf{x}_k-\mathbf{x}^c\|^2/\|\mathbf{x}^c\|^2$ for generalized lasso and as $\sum_{k=1}^{K}\|\mathbf{X}_k-\mathbf{X}^c\|_F^2/\|\mathbf{X}^c\|_F^2$ for RRR, where $\mathbf{x}_k$ and $\mathbf{X}_k$ denote the local estimates at agent k. The centralized solutions $\mathbf{x}^c$ and $\mathbf{X}^c$ are computed using the convex optimization toolbox CVX [25]. Results are obtained by averaging over 100 independent trials.

Figs. 1-2 show the performance of D-ZOA for the generalized lasso and the RRR scenarios, respectively. In Fig. 1, we plot the normalized error versus the outer loop iteration index for D-ZOA and a subgradient-based distributed algorithm, called D-SG and proposed in [2]. We observe that, for $P = 10$ and $\rho = 3$, D-ZOA has similar performance to D-SG both when the network consists of 15 and 30 agents. Fig. 2 shows that D-ZOA converges to the centralized solution of the considered RRR problem for $P = 5$, $S = 4$, $K = 10$ and $\rho = 3$.

V. Conclusion

We developed a new consensus-based algorithm for solving a distributed optimization problem with a non-smooth convex objective. We recast the original problem into an equivalent constrained optimization problem whose structure is suitable for distributed implementation via ADMM. We employed a zeroth-order method, known as the two-point stochastic gradient algorithm, to minimize the augmented Lagrangian in the primal update step. Compared to existing algorithms for non-smooth optimization, D-ZOA is fully-distributed and does not require the computation of subgradients, nor proximal operators which may be difficult to derive in some scenarios. D-ZOA only requires the computation of objective function values. The convergence of D-ZOA to the centralized solution was verified through theoretical analysis and simulations.

References

[1] D. Hajinezhad, M. Hong, and A. Garcia, "ZONE: Zeroth-order nonconvex multiagent optimization over networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 3995–4010, Oct. 2019.

[2] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[3] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Consensus-based distributed total least-squares estimation using parametric semidefinite programming," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2019, pp. 5227–5231.

[4] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[5] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed ridge regression with feature partitioning," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018.

[6] J. Akhtar and K. Rajawat, "Distributed sequential estimation in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 86–100, Jan. 2018.

[7] N. K. D. Venkategowda and S. Werner, "Privacy-preserving distributed precoder design for decentralized estimation," in *Proc. IEEE Global Conference on Signal and Information Processing*, Nov. 2018.

[8] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, *Splitting Methods in Communication, Imaging, Science, and Engineering*, ser. Scientific Computation, R. Glowinski, S. J. Osher, and W. Yin, Eds. Cham: Springer International Publishing, 2016.

[9] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, Sep. 2018.

[10] A. Nedic and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3936–3947, Dec. 2016.

[11] ——, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, Mar. 2015.

[12] A. Nedic, A. Ozdaglar, and P. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, Apr. 2010.

[13] E. Ghadimi, I. Shames, and M. Johansson, "Multi-step gradient methods for networked optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5417–5429, Nov. 2013.

[14] X. Wu and J. Lu, "Improved convergence rates of P-EXTRA for non-smooth distributed optimization," in *IEEE International Conference on Control and Automation*, Jul. 2019, pp. 55–60.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2010.

[16] D. Yuan, D. W. C. Ho, and S. Xu, "Zeroth-order method for distributed optimization with approximate projections," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 284–294, Feb. 2016.

[17] A. Agarwal, O. Dekel, and L. Xiao, "Optimal algorithms for online convex optimization with multi-point bandit feedback," in *Proc. 23rd Annual Conference on Learning Theory*, Jun. 2010, pp. 28–40.

[18] J. C. Spall, *Introduction to Stochastic Search and Optimization*. Wiley, 2003.

[19] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop on Artificial Intelligence and Security*, Nov. 2017, pp. 15–26.

[20] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: the power of two function evaluations," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2788–2806, May 2015.

[21] J. Ding, Y. Gong, M. Pan, and Z. Han, "Optimal differentially private ADMM for distributed machine learning," 2019. [Online]. Available: http://arxiv.org/abs/1901.02094

[22] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM*, vol. 58, no. 3, Jun. 2011.

[23] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, pp. 572–596, 2011.

[24] M. Mardani, G. Mateos, and G. B. Giannakis, "Decentralized sparsity-regularized rank minimization: algorithms and applications," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5374–5388, Nov. 2013.

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

Privacy-Preserved Distributed Learning with Zeroth-Order Optimization

Cristiano Gratton, *Member, IEEE,* Naveen K. D. Venkategowda, *Member, IEEE,* Reza Arablouei, and Stefan Werner, *Senior Member, IEEE*

***Abstract*—We develop a privacy-preserving distributed algorithm to minimize a regularized empirical risk function when the first-order information is not available and data is distributed over a multi-agent network. We employ a zeroth-order method to minimize the associated augmented Lagrangian function in the primal domain using the alternating direction method of multipliers (ADMM). We show that the proposed algorithm, named distributed zeroth-order ADMM (D-ZOA), has intrinsic privacy-preserving properties. Most existing privacy-preserving distributed optimization/estimation algorithms exploit some perturbation mechanism to preserve privacy, which comes at the cost of reduced accuracy. Contrarily, by analyzing the inherent randomness due to the use of a zeroth-order method, we show that D-ZOA is intrinsically endowed with $(\epsilon,\delta)-$differential privacy. In addition, we employ the moments accountant method to show that the total privacy leakage of D-ZOA grows sublinearly with the number of ADMM iterations. D-ZOA outperforms the existing differentially-private approaches in terms of accuracy while yielding similar privacy guarantee. We prove that D-ZOA reaches a neighborhood of the optimal solution whose size depends on the privacy parameter. The convergence analysis also reveals a practically important trade-off between privacy and accuracy. Simulation results verify the desirable privacy-preserving properties of D-ZOA and its superiority over the state-of-the-art algorithms as well as its network-wide convergence.**

***Index Terms*—Alternating direction method of multipliers, differential privacy, distributed optimization, zeroth-order optimization methods.**

I. Introduction

PERFORMING learning tasks at a central processing hub in a large distributed network may be prohibitive due to computation/communication costs. Collecting all data at a central hub may also create a single point of failure. Therefore, it is important to develop algorithms that are capable of processing the data gathered by agents dispersed over a distributed network [2]–[10]. Such distributed solutions are highly demanded in many of today's optimization problems pertaining to statistics [2]–[4], signal processing [5]–[7], and control [8]–[10].

This work was partly supported by the Research Council of Noway. A conference precursor of this work appears in the *Proceedings of the European Signal Processing Conference*, Amsterdam, NL, January 2021 [1].

C. Gratton and S. Werner are with the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway (email:cristiano.gratton@ntnu.no; stefan.werner@ntnu.no).

N. K. D. Venkategowda is with the Department of Science and Technology, Linköping University, 601 74, Norrköping, Sweden (email: naveen.venkategowda@liu.se).

R. Arablouei is with the Commonwealth Scientific and Industrial Research Organisation, Pullenvale QLD 4069, Australia (email:reza.arablouei@csiro.au).

Moreover, in some real-world problems, obtaining first-order information is hard due to non-smooth objectives [2], [8], [9] or lack of any complete objective function, e.g., in bandit optimization [11], in simulation-based optimization [12], or in adversarial black-box machine learning [13]. This motivates the use of zeroth-order methods, which only use the values of the objective functions to approximate their gradients [14]–[16].

However, the communications between neighboring agents in a distributed network may lead to privacy violation issues. An adversary may infer sensitive data of one or more agents by sniffing the communicated information. The adversary can be either a curious member of the network or an eavesdropper. Therefore, it is important to develop privacy-preserving methods that allow distributed processing of data without revealing private information. Differential privacy provides privacy protection against adversarial attacks by ensuring minimal change in the outcome of the algorithm regardless of whether or not a single individual's data is taken into account.

There have been several works developing privacy-preserving algorithms for distributed convex optimization [17]–[28]. The work in [17] proposes two differentially private distributed algorithms that are based on the alternating direction method of multipliers (ADMM). The algorithms in [17] are obtained by perturbing the dual and the primal variable, respectively. However, in both algorithms, the privacy leakage of an agent is bounded only at a single iteration and an adversary might exploit knowledge available from all iterations to infer sensitive information. This shortcoming is mitigated in [18]–[21]. The works in [18], [19] develop ADMM-based differentially private algorithms with improved accuracy. The work in [20] employs the ADMM to develop a distributed algorithm where the primal variable is perturbed by adding a Gaussian noise with diminishing variance to ensure zero-concentrated differential privacy enabling higher accuracy compared to the common (ϵ,δ)-differential privacy. The work in [21] develops a stochastic ADMM-based distributed algorithm that further enhances the accuracy while ensuring differential privacy. The authors of [22]–[24] propose differentially-private distributed algorithms that utilize the projected-gradient-descent method for handling constraints. The differentially private distributed algorithm proposed in [25] is based on perturbing the local objective functions. However, the algorithms in [17]–[25], [27] offer distributed solutions only for problems with smooth objective functions.

The work in [26] addresses problems with non-smooth objective functions by employing a first-order approximation of the augmented Lagrangian with a scalar l_2-norm proximity operator. However, this algorithm is not fully distributed since it requires a central coordinator to average all the perturbed primal variable updates over the network at every iteration. All the above-mentioned algorithms in [17]–[26] require some modifications through deliberately perturbing either the local estimates or the objective functions. This compromises the performance of the algorithm by degrading its accuracy especially when large amount of noise is required to provide high privacy levels. The work in [28] considers privacy-preserving properties that are intrinsic, i.e., they do not require any change in the algorithm but are associated with the algorithm's inherent properties. However, the approach taken in [28] considers a privacy metric based on the topology of the communication graph. Therefore, none of the existing algorithms are able to offer fully-distributed solutions that are intrinsically capable of ensuring differential privacy.

A. Contributions

In this paper, we develop a fully-distributed differentially-private algorithm to solve a class of regularized empirical risk minimization (ERM) problems when first-order information is unavailable or hard to obtain. We utilize the ADMM for distributed optimization and a zeroth-order method, called the two-point stochastic gradient algorithm [29], to minimize the augmented Lagrangian function in the ADMM's primal update step. The proposed algorithm, called distributed zeroth-order ADMM (D-ZOA), is fully distributed in the sense that each agent of the network communicates only with its immediate neighbors and no central coordination is necessary. No communication among agents is required throughout the inner loop.

The privacy-preserving properties of the proposed D-ZOA algorithm are intrinsic. To substantiate this novel finding, we model the primal variable at each agent as the sum of an exact (unperturbed) value and a random perturbation. This enables us to address the challenging problem of approximating the distribution of the primal variable and verify that the stochasticity inherent to the employed zeroth-order method can adequately make D-ZOA differentially private. To this end, we find a suitable approximation for the probability distribution of the primal variable. Subsequently, we show that the inherent randomness in D-ZOA enables it to preserve (ϵ, δ)-differential privacy. Utilizing the moments accountant method [30], we also show that the total privacy leakage over all iterations grows sublinearly with the number of ADMM iterations. This is particularly important as we observe that, with any similar level of privacy, the optimization accuracy of D-ZOA is higher compared to the existing privacy-preserving approaches, which perturb the variables exchanged among the network agents by adding noise.

We prove that D-ZOA reaches a neighborhood of the optimal solution, i.e., a near-optimal solution, and the size of the neighborhood is determined by the privacy parameter. This gives an explicit privacy-accuracy trade-off where a stronger privacy guarantee corresponds to a lower accuracy. Through numerical simulations, we show that D-ZOA is competitive with the state-of-the-art zeroth-order-based optimization algorithms even though they are designed for centralized processing. We also verify numerically that the entries of the zeroth-order stochastic gradient are normally distributed by illustrating the associated histograms and (quantile-quantile) QQ plots. Simulation results also demonstrate that, with any given level of required privacy guarantee, D-ZOA outperforms existing privacy-preserving algorithms in terms of accuracy. To the best of our knowledge, this is the first work on distributed non-smooth optimization that is capable of exploiting the inherent randomness due to the use of a zeroth-order method and enjoy the ensuing intrinsic (ϵ, δ)-differential privacy.

B. Paper Organization

The rest of the paper is organized as follows. In Section II, we describe the system model and formulate the distributed ERM problem when first-order information is not available. In Section III, we describe our proposed D-ZOA algorithm. In Section IV, we explain the privacy issues associated with distributed learning and we propose a solution to the very difficult problem of characterizing the distribution of the inherent randomness. Hence, we present the intrinsic privacy-preserving properties of the proposed D-ZOA algorithm by showing that the privacy leakage of each agent at any iteration is bounded and the total privacy leakage grows sublinearly with the number of ADMM iterations. In Section V, we prove the convergence of D-ZOA by confirming that both inner and outer loops of the algorithm converge. We provide some simulation results in Section VI and draw conclusions in Section VII.

C. Mathematical Notations

The set of natural and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. The set of positive real numbers is denoted by $\mathbb{R}_+$. Scalars, column vectors, and matrices are respectively denoted by lowercase, bold lowercase, and bold uppercase letters. The operators $(\cdot)^\mathsf{T}$, det$(\cdot)$, and tr$(\cdot)$ denote transpose, determinant, and trace of a matrix, respectively. $\|\cdot\|$ represents the Euclidean norm of its vector argument. $\mathbf{I}_n$ is an identity matrix of size n, $\mathbf{0}_n$ is an $n \times 1$ vector with all zeros entries, $\mathbf{0}_{n\times p} = \mathbf{0}_n\mathbf{0}_p^\mathsf{T}$, and $|\cdot|$ denotes the cardinality if its argument is a set. The statistical expectation and covariance operators are represented by $\mathbb{E}[\cdot]$ and cov$[\cdot]$, respectively. The notation $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. For a positive semidefinite matrix $\mathbf{X}$, $\lambda_{\min}(\mathbf{X})$ and $\lambda_{\max}(\mathbf{X})$ denote the nonzero smallest and largest eigenvalues of $\mathbf{X}$, respectively. For a vector $\mathbf{x} \in \mathbb{R}^n$ and a matrix $\mathbf{A}$, $\|\mathbf{x}\|_\mathbf{A}^2$ denotes the quadratic form $\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}$. For a function f, its subgradient and subdifferential are denoted by f' and ∂f, respectively.

II. System Model

We consider a network with $K \in \mathbb{N}$ agents and $E \in \mathbb{N}$ edges modeled as an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V} = \{1, \ldots, K\}$ corresponds to the agents and the set $\mathcal{E}$ represents the bidirectional communication links between the pairs of neighboring agents. Edge $e_{kl} = (k, l) \in \mathcal{E}$

indicates that agent k and l are neighbors. Agent $k \in \mathcal{V}$ can communicate only with the agents in its neighborhood $\mathcal{V}_k = \{l \in \mathcal{V} \,|\, (k,l) \in \mathcal{E}\}$.

Each agent $k \in \mathcal{V}$ has a private dataset

$$\mathcal{D}_k = \Big\{(\mathbf{X}_k, \mathbf{y}_k) : \mathbf{X}_k = [\mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \ldots, \mathbf{x}_{k,N_k}]^\mathsf{T} \in \mathbb{R}^{N_k \times P}, \\ \mathbf{y}_k = [y_{k,1}, y_{k,2}, \ldots, y_{k,N_k}]^\mathsf{T} \in \mathbb{R}^{N_k}\Big\}$$

where N_k is the number of data samples collected at agent k and P is the number of features in each sample.

We consider the problem of estimating a parameter of interest $\boldsymbol{\beta} \in \mathbb{R}^P$ that relates the value of an output measurement stored in the response vector $\mathbf{y}_k$ to input measurements collected in the corresponding row of the local matrix $\mathbf{X}_k$. The associated supervised learning problem can be cast as a regularized ERM expressed by

$$\min_{\boldsymbol{\beta}} \sum_{k=1}^{K} \frac{1}{N_k} \sum_{j=1}^{N_k} \ell(\mathbf{x}_{k,j}, y_{k,j}; \boldsymbol{\beta}) + \eta R(\boldsymbol{\beta}) \tag{1}$$

where $\ell : \mathbb{R}^P \to \mathbb{R}$ is the loss function, $R : \mathbb{R}^P \to \mathbb{R}$ is the regularizer function, and $\eta > 0$ is the regularization parameter. The ERM problem pertains to several applications in machine learning, e.g., linear regression [2], support vector machine [31], and logistic regression [20], [26]. We assume that the loss function $\ell(\cdot)$ and the regularizer function $R(\cdot)$ are both closed and convex but at least one of them is *non-smooth*. Let us denote the optimal solution of (1) by $\boldsymbol{\beta}^c$.

III. Non-Smooth Distributed Learning

We first discuss the consensus-based reformulation of the problem that allows its distributed solution through an iterative process consisting of two nested loops. Then, we describe the ADMM procedure that forms the outer loop and the zeroth-order two-point stochastic gradient algorithm that constitutes the inner loop solving the ADMM primal update step. Finally, we discuss the related computational complexity.

A. Consensus-Based Reformulation

To solve (1) in a distributed manner, we reformulate it as the following constrained minimization problem

$$\begin{aligned} \min_{\{\boldsymbol{\beta}_k\}} \quad & \sum_{k=1}^{K} \Big(\frac{1}{N_k} \sum_{j=1}^{N_k} \ell(\mathbf{x}_{k,j}, y_{k,j}; \boldsymbol{\beta}_k) + \frac{\eta}{K} R(\boldsymbol{\beta}_k) \Big) \\ \text{s.t.} \quad & \boldsymbol{\beta}_k = \boldsymbol{\beta}_l, \quad l \in \mathcal{V}_k, \quad \forall k \in \mathcal{V} \end{aligned} \tag{2}$$

where $\{\boldsymbol{\beta}_k\}_{k=1}^K$ are the primal variables representing local copies of $\boldsymbol{\beta}$ at the agents. The equality constraints impose consensus across each agent's neighborhood $\mathcal{V}_k$. To solve (2) collaboratively and in a fully-distributed manner, we utilize the ADMM [7]. For this purpose, we rewrite (2) as

$$\begin{aligned} \min_{\{\boldsymbol{\beta}_k\}} \quad & \sum_{k=1}^{K} \Big(\frac{1}{N_k} \sum_{j=1}^{N_k} \ell(\mathbf{x}_{k,j}, y_{k,j}; \boldsymbol{\beta}_k) + \frac{\eta}{K} R(\boldsymbol{\beta}_k) \Big) \\ \text{s.t.} \quad & \boldsymbol{\beta}_k = \mathbf{z}_k^l, \;\; \boldsymbol{\beta}_l = \mathbf{z}_k^l, \quad l \in \mathcal{V}_k, \quad \forall k \in \mathcal{V} \end{aligned} \tag{3}$$

where $\{\mathbf{z}_k^l\}_{k \in \mathcal{V}, l \in \mathcal{V}_k}$ are the auxiliary variables yielding an alternative but equivalent representation of the constraints in (2). They help decouple $\boldsymbol{\beta}_k$ in the constraints and facilitate the derivation of the local recursions before being eventually eliminated. Solving (3) via the ADMM requires an iterative process that is described in the next subsection.

B. Zeroth-Order-Based Distributed ADMM Algorithm

To solve (3) by employing the ADMM, we generate the augmented Lagrangian function by associating the Lagrange multipliers $\{\bar{\gamma}_k^l\}_{l \in \mathcal{V}_k}$, $\{\tilde{\gamma}_k^l\}_{l \in \mathcal{V}_k}$ with the constraints in (3). Following the steps outlined in [7], the ADMM iterations to solve (3) in a distributed manner are given by

$$\boldsymbol{\beta}_k^{(m)} = \arg\min_{\boldsymbol{\beta}_k} \mathcal{F}_k(\boldsymbol{\beta}_k) \tag{4}$$

$$\boldsymbol{\gamma}_k^{(m)} = \boldsymbol{\gamma}_k^{(m-1)} + \rho \sum_{l \in \mathcal{V}_k} \Big(\boldsymbol{\beta}_k^{(m)} - \boldsymbol{\beta}_l^{(m)} \Big). \tag{5}$$

where

$$\begin{aligned} \mathcal{F}_k(\boldsymbol{\beta}_k) =& f_k(\boldsymbol{\beta}_k) + h_k(\boldsymbol{\beta}_k), \\ f_k(\boldsymbol{\beta}_k) =& \frac{1}{N_k} \sum_{j=1}^{N_k} \ell(\mathbf{x}_{k,j}, y_{k,j}; \boldsymbol{\beta}_k) + \frac{\eta}{K} R(\boldsymbol{\beta}_k), \\ h_k(\boldsymbol{\beta}_k) =& \boldsymbol{\beta}_k^\mathsf{T} \boldsymbol{\gamma}_k^{(m-1)} + \rho \sum_{l \in \mathcal{V}_k} \Big\| \boldsymbol{\beta}_k - \frac{\boldsymbol{\beta}_k^{(m-1)} + \boldsymbol{\beta}_l^{(m-1)}}{2} \Big\|^2, \\ \boldsymbol{\gamma}_k^{(m)} =& 2 \sum_{l \in \mathcal{V}_k} \bar{\gamma}_k^{l(m)}, \end{aligned} \tag{6}$$

m is the iteration index, and all initial values $\{\boldsymbol{\beta}_k^{(0)}\}_{k \in \mathcal{V}}$, $\{\boldsymbol{\gamma}_k^{(0)}\}_{k \in \mathcal{V}}$ are set to zero. Note that the update equations in (4) and (5) can be implemented in a fully-distributed fashion since they involve only the variables available within every agent's neighborhood.

Since the objective function in (4) is assumed to be non-smooth, the corresponding minimization problem cannot be solved using any first-order method. To overcome this, we use a zeroth-order method as in [1]. We utilize the two-point stochastic-gradient algorithm that has been proposed in [29] for optimizing general non-smooth functions. More specifically, we use the stochastic mirror descent method with the proximal function $\frac{1}{2}\|\cdot\|$ and the gradient estimator at point $\boldsymbol{\beta}_k$ given by

$$\begin{aligned} \Gamma(\boldsymbol{\beta}_k, \boldsymbol{\gamma}_k^{(m-1)}, u_1, u_2, \boldsymbol{\nu}_1, \boldsymbol{\nu}_2) = u_2^{-1} [\mathcal{F}_k(\boldsymbol{\beta}_k + u_1 \boldsymbol{\nu}_1 \\ + u_2 \boldsymbol{\nu}_2, \boldsymbol{\gamma}_k^{(m-1)}) - \mathcal{F}_k(\boldsymbol{\beta}_k + u_1 \boldsymbol{\nu}_1, \boldsymbol{\gamma}_k^{(m-1)})] \boldsymbol{\nu}_2 \end{aligned} \tag{7}$$

where $u_1 > 0$ and $u_2 > 0$ are smoothing constants and $\boldsymbol{\nu}_1$, $\boldsymbol{\nu}_2$ are independent zero-mean Gaussian random vectors with the covariance matrix $\mathbf{I}_P$, i.e., $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \sim \mathcal{N}(\mathbf{0}_P, \mathbf{I}_P)$.

The two-point stochastic-gradient algorithm consists of two randomization steps where the second step is aimed at preventing the perturbation vector $\boldsymbol{\nu}_2$ from being close to a point of non-smoothness [29]. This algorithm entails an iterative procedure that consists of three steps at each iteration t. First, $J \in \mathbb{N}$ independent random vectors $\{\boldsymbol{\nu}_{1,t}^{j,k}\}_{j=1}^J$ and $\{\boldsymbol{\nu}_{2,t}^{j,k}\}_{j=1}^J$

are sampled from $\mathcal{N}(\mathbf{0}_P, \mathbf{I}_P)$. Second, a k-local stochastic gradient $\mathbf{g}_k^{(t)}$ is computed as

$$\mathbf{g}_k^{(t)} = \frac{1}{J}\sum_{j=1}^{J} \mathbf{g}_{j,k}^{(t)} \tag{8}$$

where

$$\mathbf{g}_{j,k}^{(t)} = \Gamma(\tilde{\boldsymbol{\beta}}_k^{(t)}, \boldsymbol{\gamma}_k^{(m-1)}, u_{1,t}, u_{2,t}, \boldsymbol{\nu}_{1,t}^{j,k}, \boldsymbol{\nu}_{2,t}^{j,k}),$$

$\tilde{\boldsymbol{\beta}}_k^{(t)}$ is the tth iterate of the two-point stochastic-gradient algorithm with the initial value $\tilde{\boldsymbol{\beta}}_k^{(0)} = \mathbf{0}$ and $\{u_{1,t}\}_{t=1}^{\infty}$ and $\{u_{2,t}\}_{t=1}^{\infty}$ are two non-increasing sequences of positive parameters such that $u_{2,t} \leq u_{1,t}/2$. Finally, $\tilde{\boldsymbol{\beta}}_k^{(t)}$ is updated as

$$\tilde{\boldsymbol{\beta}}_k^{(t)} = \tilde{\boldsymbol{\beta}}_k^{(t-1)} - \alpha_t \mathbf{g}_k^{(t)} \tag{9}$$

where α_t is a time-varying step-size. The step-size is computed as

$$\alpha_t = \left(L\sqrt{tP\log(2P)}\right)^{-1} \alpha_0 R$$

where α_0 is an appropriate initial step-size and R is an upper bound on the distance between the minimizer $\boldsymbol{\beta}_k^*$ to (4) and the first iterate $\tilde{\boldsymbol{\beta}}_k^{(1)}$ as per [29].

We use multiple independent random samples $\{\boldsymbol{\nu}_{1,t}^{j,k}\}_{j=1}^{J}$ and $\{\boldsymbol{\nu}_{2,t}^{j,k}\}_{j=1}^{J}$ to obtain a more accurate estimate of the gradient $\mathbf{g}_k^{(t)}$ as remarked in [29]. Furthermore, no communication among agents is needed in the inner loop.

The proposed algorithm, D-ZOA, is summarized in Algorithm 1.

Algorithm 1 Distributed Zeroth-Order ADMM (D-ZOA)

```
At all agents $k \in \mathcal{V}$, initialize $\boldsymbol{\beta}_k^{(0)} = \mathbf{0}$, $\boldsymbol{\gamma}_k^{(0)} = \mathbf{0}$, and locally run
for $m = 1, 2, \ldots, M$ do
    Share $\boldsymbol{\beta}_k^{(m-1)}$ with neighbors in $\mathcal{V}_k$
    Update $\boldsymbol{\gamma}_k^{(m)}$ as in (5)
    Initialize $\tilde{\boldsymbol{\beta}}_k^{(0)} = \mathbf{0}$
    for $t = 1, 2, \ldots, T$ do
        Draw independent $\{\boldsymbol{\nu}_{1,t}^{j,k}\}_{j=1}^{J}, \{\boldsymbol{\nu}_{2,t}^{j,k}\}_{j=1}^{J} \sim \mathcal{N}(\mathbf{0}_P, \mathbf{I}_P)$
        Set $u_{1,t} = u_1/t$, $u_{2,t} = u_1/(Pt)^2$
        Compute $\mathbf{g}_k^{(t)}$ as in (8) and (7)
        Update $\tilde{\boldsymbol{\beta}}_k^{(t)} = \tilde{\boldsymbol{\beta}}_k^{(t-1)} - \alpha_t \mathbf{g}_k^{(t)}$
    end for
    Update $\boldsymbol{\beta}_k^{(m)} = \tilde{\boldsymbol{\beta}}_k^{(T)}$
end for
```

C. Computational Complexity

Solving D-ZOA's inner loop, i.e., the minimization in (4), requires multiple evaluations of the function $\mathcal{F}_k(\cdot)$. The computational requirement at each agent and each ADMM outer loop iteration depends on the local objective function f_k. Let us indicate the number of computations required by D-ZOA to carry out one iteration of the inner loop at agent k and the number of iterations of the inner loop by m_k and T, respectively. Hence, the total number of computations required by D-ZOA at agent k and each ADMM outer loop iteration is $\mathcal{O}(Tm_k)$. However, the cost of transmission/communication among the neighboring agents does not depend on T or m_k since the inner loop does not require any communication among agents.

IV. Intrinsic Differential Privacy Guarantee

In this section, we consider the privacy concerns associated with distributed learning and reveal that the inherent randomness due to the use of a zeroth-order method is sufficient for the proposed D-ZOA algorithm to preserve (ϵ, δ)-differential privacy. First, we present the attack model along with the definition of the attacker. Second, we propose our solution to the challenging problem of characterizing the randomness inherent to the algorithm. Subsequently, we assess the l_2-norm sensitivity of the primal variable and compute the covariance that the primal variable is required to have so that the privacy leakage of a single iteration of D-ZOA is bounded at each agent. Finally, we prove that the total privacy leakage over all iterations grows sublinearly with the number of ADMM iterations.

A. Attack Model and Privacy Concerns

In Algorithm 1, the data stored at each agent, $\mathbf{X}_k$ and $\mathbf{y}_k$, is not shared with any other agent. However, the local estimates $\{\boldsymbol{\beta}_k^{(m)}\}_{k\in\mathcal{V}}$ are exchanged within the local neighborhoods. Therefore, the risk of privacy breach still exists as it has been shown by the model inversion attacks [32].

In this paper, we consider the following attack model. We assume that the adversary is able to access the local estimates $\{\boldsymbol{\beta}_k^{(m)}\}_{k\in\mathcal{V}}$ that are exchanged throughout the intermediate ADMM iterations as well as the final output. The adversary can be either a honest-but-curious member of the network or an external eavesdropper. The adversary's goal is to infer sensitive data of one or more agents by sniffing the communicated information $\{\boldsymbol{\beta}_k^{(m)}\}_{k\in\mathcal{V}}$.

We show that D-ZOA guarantees (ϵ, δ)-differential privacy as per the below definition since it is intrinsically resistant to such inference attacks.

Definition 1. A randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if for any two neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$ differing in only one data sample and for any subset of outputs $\mathcal{O} \subseteq \text{range}(\mathcal{M})$, we have

$$\Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{O}] \leq e^{\epsilon}\Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{O}] + \delta. \tag{10}$$

This means the ratio of the probability distributions of $\mathcal{M}(\mathcal{D})$ and $\mathcal{M}(\mathcal{D}')$ is bounded by e^{ϵ}.

In Definition 1, ϵ and δ are privacy parameters indicating the level of privacy preservation ensured by a differentially private algorithm. A better privacy preservation is achieved with smaller ϵ or δ. On the other hand, low privacy guarantee corresponds to higher values of ϵ, i.e., close to 1. Therefore, it is reasonable to assume that $\epsilon \in (0, 1]$ as in [20], [33].

In the next subsection, we find an approximate distribution of the primal variable, which is needed to prove that (10) holds for our proposed D-ZOA.

B. Primal Variable Distribution

Due to the stochasticity inherent to the zeroth-order method, its employment for the ADMM primal update produces a perturbed estimate. Therefore, the solution in the primal update step in (4) at agent k using D-ZOA can be modeled as

$$\boldsymbol{\beta}_k^{(m)} = \breve{\boldsymbol{\beta}}_k^{(m)} + \boldsymbol{\xi}_k^{(m)} \tag{11}$$

where $\breve{\boldsymbol{\beta}}_k^{(m)} \in \mathbb{R}^P$ is the exact ADMM primal update and $\boldsymbol{\xi}_k^{(m)} \in \mathbb{R}^P$ is a random variable representing the perturbation. As in [34], the optimality condition for $\breve{\boldsymbol{\beta}}_k^{(m)}$ is given by

$$\begin{aligned}\mathbf{0} \in &\partial f_k(\breve{\beta}_k^{(m)}) + \boldsymbol{\gamma}_k^{(m-1)} + 2\rho|\mathcal{V}_k|\breve{\boldsymbol{\beta}}^{(m)} \\ &- \rho|\mathcal{V}_k|\boldsymbol{\beta}_k^{(m-1)} - \rho\sum_{l\in\mathcal{V}_k}\boldsymbol{\beta}_l^{(m-1)}.\end{aligned} \tag{12}$$

Hence, for any subgradient $f_k'(\breve{\boldsymbol{\beta}}_k^{(m)}) \in \partial f_k(\breve{\boldsymbol{\beta}}_k^{(m)})$, we have

$$\begin{aligned}f_k'(\breve{\beta}_k^{(m)}) = &- \boldsymbol{\gamma}_k^{(m-1)} - 2\rho|\mathcal{V}_k|\breve{\boldsymbol{\beta}}^{(m)} \\ &+ \rho|\mathcal{V}_k|\boldsymbol{\beta}_k^{(m-1)} + \rho\sum_{l\in\mathcal{V}_k}\boldsymbol{\beta}_l^{(m-1)}.\end{aligned} \tag{13}$$

The model (11) represents an implicit primal variable perturbation that can be contrasted with the explicit primal variable perturbation used in [17], [20]. Using (11) and the primal update equation in (13), the ADMM primal update step in (4) can be expressed as

$$\begin{aligned}\breve{\boldsymbol{\beta}}_k^{(m)} = &- \frac{1}{2\rho|\mathcal{V}_k|}f_k'(\breve{\boldsymbol{\beta}}_k^{(m)}) + \frac{1}{2|\mathcal{V}_k|}\Big(|\mathcal{V}_k|\boldsymbol{\beta}_k^{(m-1)} \\ &+ \sum_{l\in\mathcal{V}_k}\boldsymbol{\beta}_l^{(m-1)}\Big) - \frac{1}{2\rho|\mathcal{V}_k|}\boldsymbol{\gamma}_k^{(m-1)}\end{aligned} \tag{14}$$

$$\boldsymbol{\beta}_k^{(m)} = \breve{\boldsymbol{\beta}}_k^{(m)} + \boldsymbol{\xi}_k^{(m)} \tag{15}$$

where $\breve{\boldsymbol{\beta}}_k$ is the local exact primal update at agent k and $\boldsymbol{\xi}_k$ is the local perturbation of $\breve{\boldsymbol{\beta}}_k$ at agent k.

To prove that the inherent randomness due to employing a zeroth-order method makes D-ZOA differentially private, we need the knowledge of the probability distribution of the primal variable $\boldsymbol{\beta}_k^{(m)}$. To approximate the probability distribution, in view of (9) and the fact that $\boldsymbol{\beta}_k^{(0)} = \mathbf{0}$, we unfold $\boldsymbol{\beta}_k^{(m)}$ as $\boldsymbol{\beta}_k^{(m)} = -\sum_{t=1}^{T}\alpha_t\mathbf{g}_k^{(t)}$. The stochastic gradient $\mathbf{g}_k^{(t)}$ is the average of J independent random samples $\{\mathbf{g}_{j,k}^{(t)}\}_{j=1}^{J}$ that are functions of the random values $\{\boldsymbol{\nu}_{1,t}^{j,k}\}_{j=1}^{J}$ and $\{\boldsymbol{\nu}_{2,t}^{j,k}\}_{j=1}^{J}$ drawn from the same normal distribution. Therefore, we assume that $\mathbf{g}_k^{(t)}$ is normally distributed with the mean $\boldsymbol{\mu}_k^{(t)}$ and the finite covariance matrix $\boldsymbol{\Psi}_k^{(t)}$, i.e., $\mathbf{g}_k^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Psi}_k^{(t)})$. Thus, the probability distribution of $\boldsymbol{\beta}_k^{(m)}$ is given by the following lemma.

Lemma 1. *Given $\mathbf{g}_k^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Psi}_k^{(t)})$, the distribution of $\boldsymbol{\beta}_k^{(m)}$ is*

$$\boldsymbol{\beta}_k^{(m)} \sim \mathcal{N}\Big(\breve{\boldsymbol{\beta}}_k^{(m)}, \frac{1}{J}\sum_{t=1}^{T}\alpha_t^2\boldsymbol{\Psi}_k^{(t)}\Big). \tag{16}$$

Proof. See Appendix A.

In the next subsection, we find an explicit expression for the covariance of $\boldsymbol{\beta}_k^{(m)}$.

The assumption of normal distribution for $\mathbf{g}_k^{(t)}$ is a natural one as $\mathbf{g}_k^{(t)}$ is the average of stochastic variable vectors $\{\mathbf{g}_{j,k}^{(t)}\}_{j=1}^{J}$, which are themselves functions of normally-distributed random variable vectors $\{\boldsymbol{\nu}_{1,t}^{j,k}\}_{j=1}^{J}$ and $\{\boldsymbol{\nu}_{2,t}^{j,k}\}_{j=1}^{J}$. We provide some numerical experiments to explicitly verify this assumption in Section VI.

The assumption is necessary to make the problem of deriving theoretical differential privacy guarantees for the proposed algorithm tractable. Note that our analysis based on this and other assumptions does not result in any deterministic guarantee but yields a probabilistic statement for privacy guarantee by setting a bound on a ratio of probabilities relevant in the concept of differential privacy. Therefore, we do not require perfectly accurate evaluations of the parameters, variables, or statistical models involved in the analysis. Nonetheless, we are cognizant that the reliability of the results highly depends on the accuracy of the underlying assumptions and approximations. Our simulation results in Section VI implicitly corroborate the veracity of our assumptions.

C. Covariance of the Primal Variable

In this subsection, we derive an explicit expression for the covariance of the primal variable $\boldsymbol{\beta}_k^{(m)}$. This is needed to show that the privacy leakage of any iteration of D-ZOA is bounded at all agents.

To make the problem more tractable, we assume that the entries of the random vector $\boldsymbol{\beta}_k^{(m)}$ are independent of each other and have the same variance [17], [26], [33]. Let us denote the variance of every entry of $\boldsymbol{\xi}_k^{(m)}$ by σ_k^2. Therefore, in view of Lemma 1, we have

$$\sigma_k^2 = \frac{1}{JP}\sum_{t=1}^{T}\alpha_t^2\text{tr}\left(\boldsymbol{\Psi}_k^{(t)}\right),$$

which can be computed as per the following lemma.

Lemma 2. *There exists a constant c such that*

$$\sigma_k^2 = \frac{c\alpha_0^2R^2}{JP\log(2P)}\Big(s_1(1+\log P) + s_2\Big) - \frac{4\left\|\boldsymbol{\beta}^c\right\|^2}{TJP}. \tag{17}$$

where $s_1 = \sum_{t=1}^{T}t^{-1}$, $s_2 = \sum_{t=1}^{T}t^{-1.5}$, and $\boldsymbol{\beta}^c$ is the optimal solution.

Proof. See Appendix B.

In [29], it is shown that $c = 0.5$ is suitable when $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ are sampled from a multivariate normal distribution. Note that s_1 and s_2 grow slowly with T. Hence, even for a very large T, s_1 and s_2 have reasonable values. For example, with $T = 2.5 \times 10^8$, we have $\sum_{t=1}^{T}t^{-1} < 20$. A large T will increase the computational complexity according to the discussions of Section III.D.

D. l_2-norm Sensitivity

In this subsection, we estimate the l_2-norm sensitivity of $\breve{\boldsymbol{\beta}}_k^{(m)}$. The l_2-norm sensitivity calibrates the magnitude of the noise by which $\breve{\boldsymbol{\beta}}_k^{(m)}$ has to be perturbed to preserve privacy. Unlike the existing privacy-preserving methods where the noise is added to the output of the algorithm [17], [20], [26], [33], [35], [36], in D-ZOA, the noise is inherent.

We introduce the following assumption that is widely used in the literature, see, e.g., [17], [26], [33].

Assumption 1: There exists a constant c_1 such that $\|\ell'(\cdot)\| \leq c_1$ where $\ell(\cdot)$ is the loss function defined in Section II.

Similar to the classical methods of differential privacy analysis, e.g., [26], [33], we first define the l_2 norm sensitivity. Subsequently, we estimate the l_2-norm sensitivity of $\boldsymbol{\beta}_k^{(m)}$.

Definition 2. The l_2-norm sensitivity of $\breve{\boldsymbol{\beta}}_k^{(m)}$ is defined as

$$\Delta_{k,2} = \max_{\mathcal{D}_k, \mathcal{D}'_k} \left\| \breve{\boldsymbol{\beta}}_{k,\mathcal{D}_k}^{(m)} - \breve{\boldsymbol{\beta}}_{k,\mathcal{D}'_k}^{(m)} \right\| \tag{18}$$

where $\breve{\boldsymbol{\beta}}_{k,\mathcal{D}_k}^{(m)}$ and $\breve{\boldsymbol{\beta}}_{k,\mathcal{D}'_k}^{(m)}$ denote the local primal variables for two neighboring datasets $\mathcal{D}_k$ and $\mathcal{D}'_k$ differing in only one data sample, i.e., one row of $\mathbf{X}_k$ and the corresponding entry of $\mathbf{y}_k$.

The l_2-norm sensitivity of $\breve{\boldsymbol{\beta}}_k^{(m)}$ is an upper bound on $\Delta_{k,2}$ and is computed as in the following lemma.

Lemma 3. *Under Assumption 1, the l_2-norm sensitivity of $\breve{\boldsymbol{\beta}}_k^{(m)}$ is given by*

$$\Delta_{k,2} = \frac{c_1}{\rho|\mathcal{V}_k|N_k}. \tag{19}$$

Proof. See Appendix C.

E. Intrinsic (ϵ, δ)-Differential Privacy Guarantee

In this subsection, we reveal that the immanent stochasticity imparted by the embedded zeroth-order method makes D-ZOA (ϵ, δ)-differentially private. We provide an expression relating the primal variable variance, σ_k, to the privacy parameters ϵ and δ as well as an expression for ϵ relating it to the relevant algorithmic parameters.

We first prove that Algorithm 1 is (ϵ, δ)-differentially private at each iteration providing a relationship between σ_k and ϵ, δ.

Theorem 1. *Let $\epsilon \in (0, 1]$ and*

$$\sigma_k = \frac{c_1\sqrt{2.1\log(1.25/\delta)}}{\rho|\mathcal{V}_k|N_k\epsilon}. \tag{20}$$

Under Assumption 1, at each iteration of D-ZOA, (ϵ, δ)-differential privacy is guaranteed. Specifically, for any neighboring datasets $\mathcal{D}_k$ and $\mathcal{D}'_k$ and any output $\boldsymbol{\beta}_k^{(m)}$, the following inequality holds:

$$Pr[\boldsymbol{\beta}_{k,\mathcal{D}_k}^{(m)}] \leq e^{\epsilon} Pr[\boldsymbol{\beta}_{k,\mathcal{D}'_k}^{(m)}] + \delta. \tag{21}$$

Proof. See Appendix D.

Theorem 1 shows that the primal variable variance is inversely proportional to the privacy parameter ϵ. This implies that a higher variance leads to a smaller ϵ and higher privacy guarantee. A smaller ϵ means that the ratio of the probability distributions of $\boldsymbol{\beta}_{k,\mathcal{D}_k}^{(m)}$ and $\boldsymbol{\beta}_{k,\mathcal{D}'_k}^{(m)}$ is smaller and consequently less information is available to any sniffing/spoofing adversary through $\boldsymbol{\beta}_k$ hence the improved privacy [17].

We then introduce the following corollary.

Corollary 1. *If $\{\mathbf{g}_{j,k}^{(t)}\}_{j=1}^{J}$ are i.i.d. with $\mathbf{g}_{j,k}^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_k^{(t)}, \Psi_k^{(t)})$, and Assumption 1 holds, we have*

$$\epsilon = \frac{c_1}{\rho|\mathcal{V}_k|N_k}\sqrt{\frac{2.1JP\log(\frac{1.25}{\delta})}{\frac{cR^2\alpha_0^2}{\log(2P)}\left(s_1(1+\log P)+s_2\right) - \frac{4\|\boldsymbol{\beta}^c\|^2}{T}}}. \tag{22}$$

Proof. The proof follows from equating the expressions for σ_k in Lemma 2, (17), and Theorem 1, (20), and solving for ϵ. □

The equation (22) shows how the intrinsic privacy preserving property of D-ZOA is affected by various involved parameters. For example, a smaller J results in a smaller ϵ. This is consistent with the fact that a smaller J leads to a higher variance, which yields a higher privacy guarantee due to the inherent randomness brought about by using a zeroth-order method in the inner loop.

The denominator of the second factor in (22) is required to be positive for the factor to be real. We ensure this by setting

$$T > \frac{4\|\boldsymbol{\beta}^c\|^2\log(2P)}{cR^2\alpha_0^2(s_1(1+\log(P))+s_2)}. \tag{23}$$

F. Total Privacy Leakage

In this subsection, we consider the total privacy leakage of the proposed D-ZOA algorithm. Since D-ZOA is an M-fold adaptive algorithm, we utilize the results of [30] together with the moments accountant method to evaluate its total privacy leakage. The main result is summarized in the following theorem.

Theorem 2. *Let $\epsilon \in (0, 1]$ and*

$$\sigma_k = \frac{c_1\sqrt{2.1\log(1.25/\delta)}}{\rho|\mathcal{V}_k|N_k\epsilon}. \tag{24}$$

Under Assumption 1, Algorithm 1 guarantees $(\bar{\epsilon}, \delta)$-differential privacy where

$$\bar{\epsilon} = \epsilon\sqrt{\frac{M\log(1/\delta)}{1.05\log(1.25/\delta)}}. \tag{25}$$

Proof. The proof is obtained by using the log moments of the privacy loss and their linear composability in the same way as in [26, Theorem 2]. □

V. Convergence Analysis and Privacy-Accuracy Trade-off

We establish the convergence of D-ZOA to a near-optimal solution by corroborating that both inner and outer loops of the algorithm converge.

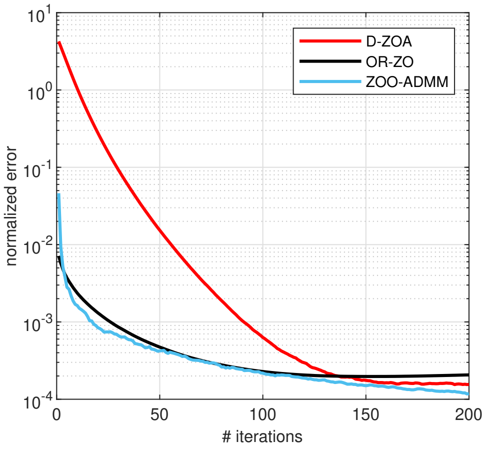


Fig. 1. The normalized errors of D-ZOA, OR-ZO [29], and ZOO-ADMM [14] versus the iteration number.

A. Convergence of the Inner Loop

The convergence of the inner loop can be verified following [29, Theorem 2], i.e., it can be shown that, if $\mathcal{F}_k(\cdot)$ is Lipschitz-continuous with the Lipschitz constant L, there exists a constant c such that, for each T representing a fixed number of inner-loop iterations, the following inequality holds:

$$\begin{aligned}&\mathbb{E}[\mathcal{F}_k(\hat{\boldsymbol{\beta}}_k^{(T)}) - \mathcal{F}_k(\boldsymbol{\beta}_k^*)]\\ \leq& c\frac{RL\sqrt{P}}{\sqrt{T}}\left(\max\{\alpha_0, \alpha_0^{-1}\}\sqrt{\log(2P)} + \frac{u_1\log(2T)}{\sqrt{T}}\right)\end{aligned} \tag{26}$$

where $\boldsymbol{\beta}_k^*$ is the minimizer to (4) and

$$\hat{\boldsymbol{\beta}}_k^{(T)} = \frac{1}{T}\sum_{t=1}^{T}\tilde{\boldsymbol{\beta}}_k^{(t)}.$$

The inequality in (26) implies that the two-point stochastic gradient algorithm constituting the inner loop converges at a rate of $\mathcal{O}(\sqrt{P/T})$. In [29], it is shown that $c = 0.5$ is suitable when $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ are sampled from a normal distribution. The function $\mathcal{F}_k(\cdot)$ is the sum of $f_k(\cdot)$, which is assumed to be closed and convex, and $h_k(\cdot)$ that is also both closed and convex since it is a positive definite quadratic function. Hence, the function $\mathcal{F}_k(\cdot)$ is closed and convex in addition to being Lipschitz-continuous [37]. Therefore the convergence result in (26) follow from [29].

B. Convergence of the Outer Loop

The convergence of the outer loop can be proven by verifying the convergence of a fully-distributed ADMM with perturbed primal updates. To present the convergence result, we rewrite (3) in the matrix form. By defining $\mathbf{w} \in \mathbb{R}^{KP}$ concatenating all $\boldsymbol{\beta}_k$ and $\mathbf{z} \in \mathbb{R}^{2EP}$ concatenating all $\mathbf{z}_k^l$, (3) can be written as

$$\begin{aligned}\min_{\mathbf{w},\mathbf{z}} \quad & f(\mathbf{w})\\ \text{s.t.} \quad & \mathbf{A}\mathbf{w} + \mathbf{B}\mathbf{z} = \mathbf{0}\end{aligned} \tag{27}$$

where

$$f(\mathbf{w}) = \sum_{k=1}^{K} f_k(\boldsymbol{\beta}_k),$$

$\mathbf{A} = [\mathbf{A}_1^\intercal, \mathbf{A}_2^\intercal]^\intercal$, and $\mathbf{A}_1, \mathbf{A}_2 \in \mathbb{R}^{2EP\times KP}$ are both composed of $2E \times K$ blocks of $P \times P$ matrices. If $(k,l) \in \mathcal{E}$ and $\mathbf{z}_k^l$ is the qth block of $\mathbf{z}$, then the (q,k)th block of $\mathbf{A}_1$ and the (q,l)th block of $\mathbf{A}_2$ are the identity matrix $\mathbf{I}_P$. Otherwise, the corresponding blocks are $\mathbf{0}_{P\times P}$. Furthermore, we have

$$\mathbf{B} = [-\mathbf{I}_{2EP}, -\mathbf{I}_{2EP}]^\intercal.$$

To facilitate the representation, we also define the following matrices

$$\begin{aligned}\mathbf{M}_+ &= \mathbf{B}_1^\intercal + \mathbf{B}_2^\intercal\\ \mathbf{M}_- &= \mathbf{B}_1^\intercal - \mathbf{B}_2^\intercal\\ \mathbf{L}_+ &= 0.5\mathbf{M}_+\mathbf{M}_+^\intercal\\ \mathbf{L}_- &= 0.5\mathbf{M}_-\mathbf{M}_-^\intercal\\ \mathbf{H} &= 0.5(\mathbf{L}_+ + \mathbf{L}_-)\\ \mathbf{Q} &= \sqrt{0.5\mathbf{L}_-}.\end{aligned}$$

We construct the auxiliary sequence

$$\mathbf{r}^{(m)} = \sum_{s=0}^{m}\mathbf{Q}\mathbf{w}^{(s)}$$

and define the auxiliary vector $\mathbf{q}^{(m)}$ and the auxiliary matrix $\mathbf{G}$ as

$$\mathbf{q}^{(m)} = \begin{bmatrix}\mathbf{r}^{(m)}\\ \mathbf{w}^{(m)}\end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix}\rho\mathbf{I}_P & \mathbf{0}_{P\times P}\\ \mathbf{0}_{P\times P} & \rho\frac{\mathbf{L}_+}{2}\end{bmatrix}. \tag{28}$$

The convergence results of [38], [20], and [39] can now be adapted to D-ZOA as per the following theorem that also provides an explicit privacy-accuracy trade-off.

Theorem 3. *For any $M > 0$, we have*

$$\begin{aligned}&\mathbb{E}[f(\hat{\mathbf{w}}^{(M)}) - f(\mathbf{w}^*)]\\ \leq& \frac{\left\|\mathbf{q}^{(0)} - \mathbf{q}\right\|_{\mathbf{G}}^2}{M} + \frac{2.1c_1^2P\log(1.25/\delta)\lambda_{max}^2(\mathbf{L}_+)}{2\rho|\mathcal{V}_k|^2N_k^2\epsilon^2\lambda_{min}(\mathbf{L}_-)}\end{aligned} \tag{29}$$

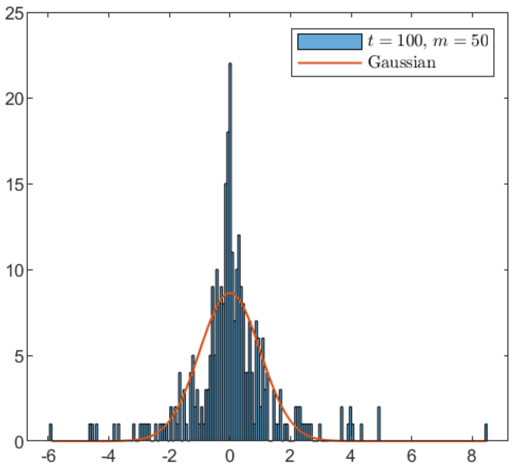
where $\mathbf{q} = [\mathbf{r}^\intercal, (\mathbf{w}^*)^\intercal]^\intercal$ and

$$\hat{\mathbf{w}}^{(M)} = \frac{1}{M}\sum_{m=1}^{M}\breve{\mathbf{w}}^{(m)}.$$

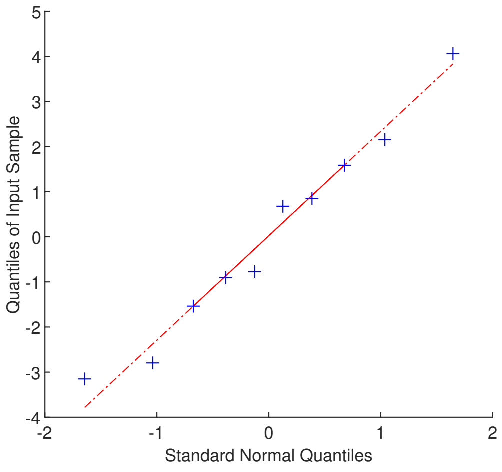
Proof. See Appendix E. □

Theorem 3 shows that D-ZOA reaches a neighborhood of the optimal (centralized) solution with the size of the neighborhood determined by the privacy-parameter ϵ. This discloses a privacy-accuracy trade-off offered by D-ZOA. When the privacy guarantee is stronger (smaller ϵ and δ), the accuracy is lower.

Note that we do not need to solve the minimization problem in the ADMM primal update step of the outer loop with high accuracy [26], [34]. We perform the ADMM primal update step in the outer loop after obtaining an inexact solution to (4). Therefore, we select the number of inner loop iterations T as the minimum number of iterations satisfying (23) and

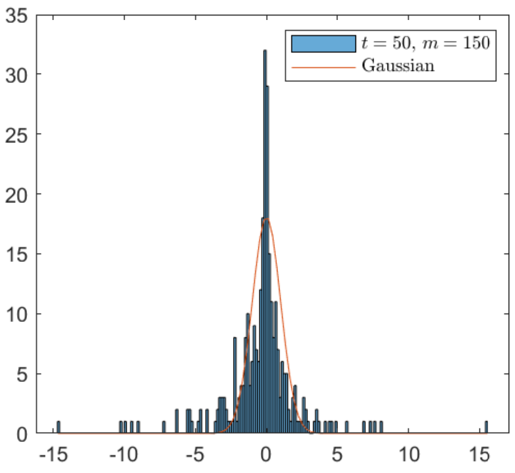


(a) Histogram

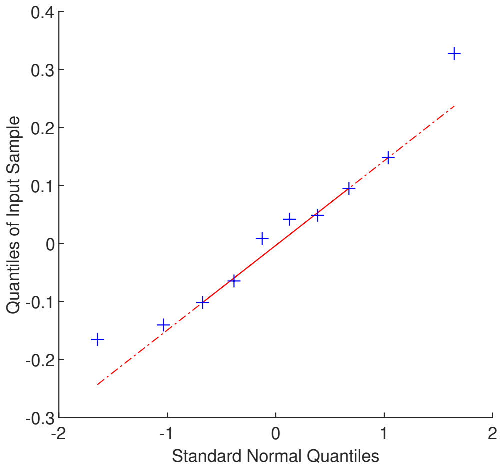


(b) QQ plot

Fig. 2. The histogram and the QQ plot of $\mathbf{g}_k^{(t)}$ at agent 2, the inner loop iteration $t = 100$, and the outer loop iteration $m = 50$.



(a) Histogram



(b) QQ plot

Fig. 3. The histogram and the QQ plot of $\mathbf{g}_k^{(t)}$ at agent 2, the inner loop iteration $t = 50$, and the outer loop iteration $m = 150$.

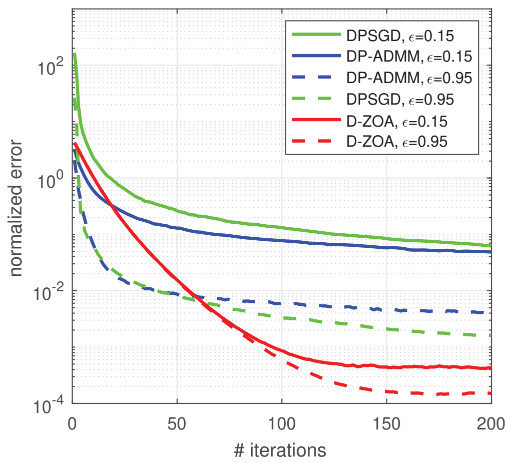
entailing an accuracy that is sufficient to ensure convergence of ADMM according to [29]. In fact, T should be chosen as low as possible within the above-mentioned constraints to minimize the computational complexity according to the discussions of Section III.D.

VI. Simulations

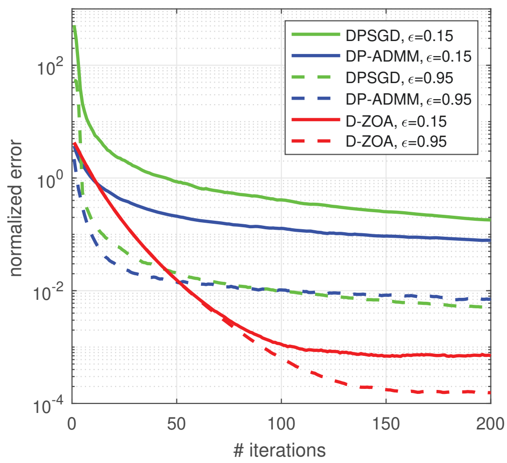
In this section, we present some simulation examples to evaluate the performance of the proposed D-ZOA algorithm in comparison with the most relevant state-of-the-art algorithms as well as the privacy-accuracy trade-off offered by the proposed algorithm.

We first benchmark D-ZOA against two popular existing algorithms for zeroth-order-based optimization, which have originally been designed for centralized settings, i.e., those proposed in [14] and [29] and called zeroth-order online ADMM (ZOO-ADMM) and optimal-rate zeroth-order (OR-ZO) algorithm, respectively. Then, we illustrate two sets of example histograms and QQ plots to verify that the entries of the gradient vector $\mathbf{g}_k^{(t)}$ are normally distributed.

Next, we benchmark D-ZOA against some existing baseline differentially-private algorithms: the ADMM algorithm with primal variable perturbation (PVP) proposed in [17], [26], the ADMM with dual variable perturbation (DVP) proposed

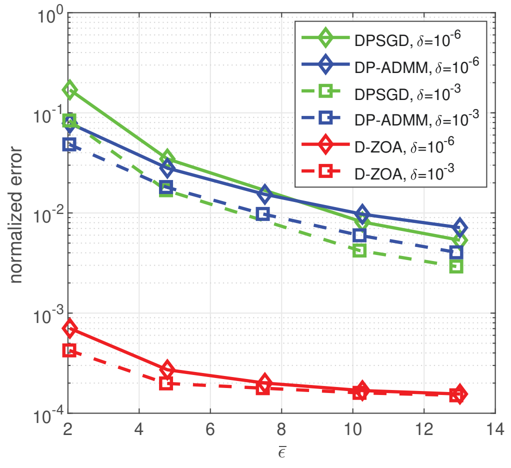


(a) $\delta = 10^{-}$

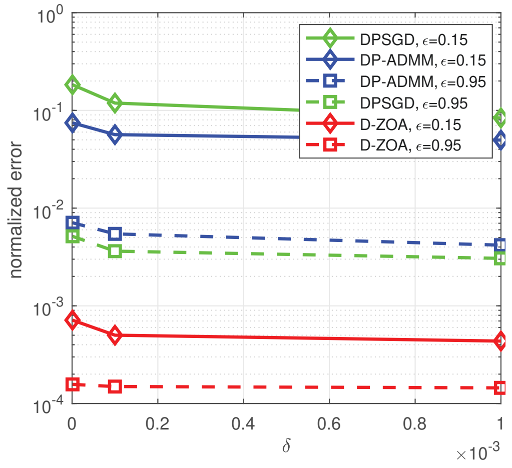


(b) $\delta = 10^{-6}$

g. 4. Normalized error of DPSGD, DP-ADMM, and D-ZOA for two values of ϵ and fixed δ for ERM with ℓ_1-norm regularization.



(a) $\delta = 10^{-6}$ and $\delta = 10^{-}$



(b) $\epsilon = 0.15$ and $\epsilon = 0.95$

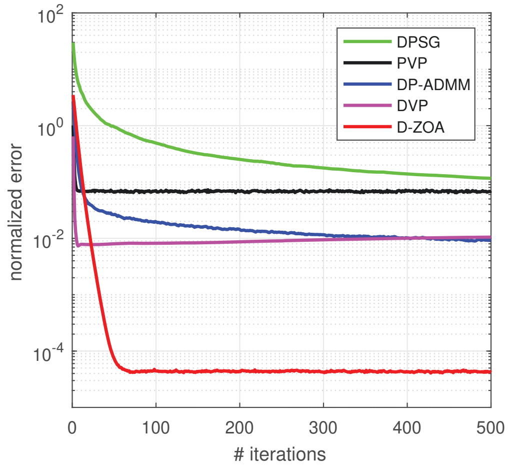
g. 5. Privacy-accuracy trade-off of DPSGD, DP-ADMM, and D-ZOA for ERM with ℓ_1-norm regularization.

[17], the ADMM-based differentially private distributed gorithm called DP-ADMM and proposed in [26], and the distributed subgradient method proposed in [9] that is customized include differential privacy (DPSG). Note that DP-ADMM the only existing privacy-preserving distributed algorithm r non-smooth objectives.

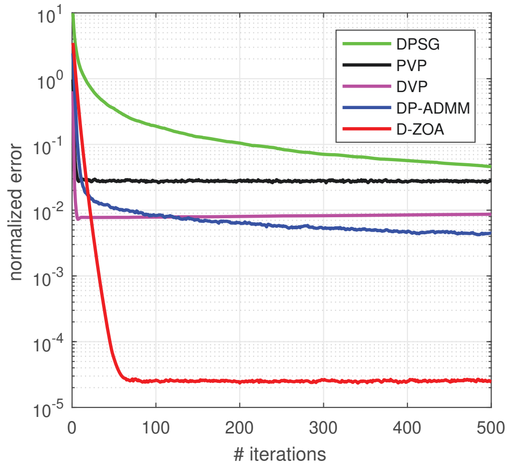
As for the applications, we consider a distributed version the empirical risk minimization problem with an ℓ_1-norm gularization (lasso penalty) and an ℓ_2-norm regularization idge penalty) [40].

of features in each sample. The matrix $\mathbf{X}$ consists of K submatrices $\mathbf{X}_k$, i.e., $\mathbf{X} = [\mathbf{X}_1^\mathsf{T}, \mathbf{X}_2^\mathsf{T}, \ldots, \mathbf{X}_K^\mathsf{T}]^\mathsf{T}$, and the vector $\mathbf{y}$ consists of K subvectors $\mathbf{y}_k$, i.e., $\mathbf{y} = \left[\mathbf{y}_1^\mathsf{T}, \mathbf{y}_2^\mathsf{T}, \ldots, \mathbf{y}_K^\mathsf{T}\right]^\mathsf{T}$, as the data is distributed among the agents and each agent k holds its respective $\mathbf{X}_k \in \mathbb{R}^{N_k \times P}$ and $\mathbf{y}_k \in \mathbb{R}^{N_k \times 1}$ where $N = \sum_{k=1}^{K} N_k$. The parameter vector that establishes a linear regression between $\mathbf{X}$ and $\mathbf{y}$ is $\boldsymbol{\beta} \in \mathbb{R}^{P \times 1}$. In the centralized approach, a lasso estimate of $\boldsymbol{\beta}$ is given by

$$\boldsymbol{\beta}^c = \arg\min_{\boldsymbol{\beta}} \{\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|^2 + \eta \|\boldsymbol{\beta}\|_1\} \qquad (30)$$

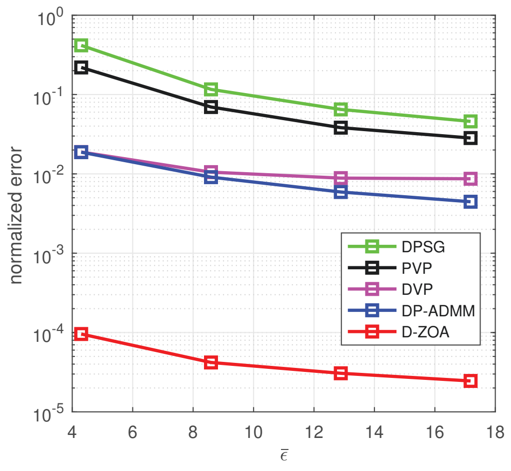


(a) $\epsilon = 0.40$ and $\delta = 10^{-}$

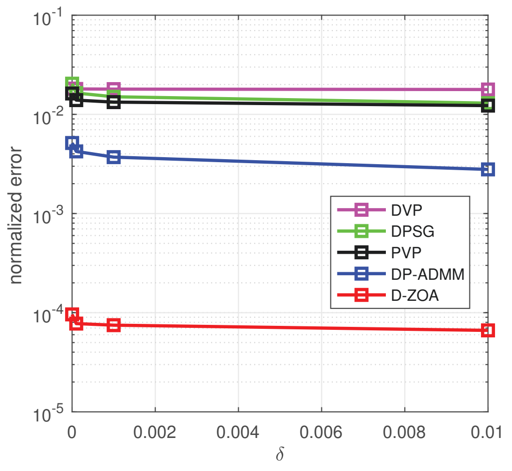


(b) $\epsilon = 0.80$ and $\delta = 10^{-}$

g. 6. Normalized error of DPSGD, PVP, DP-ADMM, DVP, and D-ZOA for two values of ϵ and fixed δ for ERM with ℓ_2-norm regularization.



(a) $\delta = 10^{-}$



(b) $\epsilon = 0.95$

g. 7. Privacy-accuracy trade-off of DPSGD, PVP, DP-ADMM, DVP, and D-ZOA for ERM with ℓ_2-norm regularization.

In the distributed setting, we solve problem (2) with

$$\sum_{j=1}^{N_k} \ell(\mathbf{x}_{k,j}, y_{k,j}; \boldsymbol{\beta}_k) = \|\mathbf{X}_k\boldsymbol{\beta}_k - \mathbf{y}_k\|^2 \qquad (32)$$

nd $R(\boldsymbol{\beta}_k) = \|\boldsymbol{\beta}_k\|_1$ for the lasso penalty. For the ridge enalty, we have $R(\boldsymbol{\beta}_k) = \|\boldsymbol{\beta}_k\|^2$.

We assess the performance of the D-ZOA algorithm ver a network of $K = 5$ agents with edge set $\mathcal{E} =$ $e_{12}, e_{14}, e_{23}, e_{34}, e_{45}\}$. The number of samples at each agent set to $N_k = 20\ \forall k \in \mathcal{V}$ and the total number of samples

variance Gaussian random variables. The response vector $\mathbf{y}$ is synthesized as $\mathbf{y} = \mathbf{X}\boldsymbol{\omega} + \boldsymbol{\psi}$ where $\boldsymbol{\omega} \in \mathbb{R}^P$ and $\boldsymbol{\psi} \in \mathbb{R}^M$ are random vectors with distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_N)$, respectively. The data are preprocessed by normalizing the columns of $\mathbf{X}$ to guarantee that the maximum value of each column is 1 and by normalizing the rows to enforce their l_2-norm to be less than 1 as in [26]. This is motivated by the need for homogeneous scaling of the features. Therefore, we have $c_1 = 1$. The regularization parameter is set to $\eta = 1$ and the penalty parameter is set to $\rho = 4$. The number of calculate J from equation (22) by fixing ϵ, solving for J and rounding the solution to the nearest integer. Performance of D-ZOA is evaluated using the normalized error between the centralized solutions $\boldsymbol{\beta}^c$ as per (31) and the local estimates. It is defined as $\sum_{k=1}^{K} \|\boldsymbol{\beta}_k - \boldsymbol{\beta}^c\|^2/\|\boldsymbol{\beta}^c\|^2$ where $\boldsymbol{\beta}_k$ denotes the local estimate at agent k. The centralized solution $\boldsymbol{\beta}^c$ is computed using the convex optimization toolbox CVX [42]. Results are obtained by averaging over 100 independent trials.

In Fig. 1, we compare the performance of the proposed D-ZOA algorithm with that of two existing zeroth-order-based algorithms, i.e., those proposed in [14] and [29]. The simulation results show that the steady-state normalized error of D-ZOA is comparable to those of these algorithms, even though they are designed for centralized processing. The centralized algorithms converge faster than Z-DOA since, unlike our fully-distributed D-ZOA, they have all data concentrated at a central processing hub and do not rely on diffusing information across the network by sharing the local estimates within each agent's neighborhood. Note that the notion of iteration is essentially different for each algorithm whose learning curve is shown in Fig. 1. Thus, we provide the learning curve plots in Fig. 1 only to examine how D-ZOA performs in comparison with the existing zeroth-order optimization algorithms notwithstanding the underlying fundamental differences.

In Figs. 2 and 3, we provide two sets of histograms and QQ plots for an arbitrary entry of the stochastic gradient vector $\mathbf{g}_k^{(t)}$, i.e., the one corresponding to agent 2, and different inner and outer loop iterations, i.e., $t = 100$, $t = 50$, and $m = 50$, $m = 150$. The plots help us verify that the entries of $\mathbf{g}_k^{(t)}$ are normally distributed hence attest to the validity of our related assumption in Section IV-B. Meanwhile, we made similar observations with other entries of $\mathbf{g}_k^{(t)}$ and iteration numbers. However, due to space limitation, we only provide Figs. 2 and 3 as examples.

We first benchmark our D-ZOA in the case of the ERM with the lasso penalty. Since PVP and DVP cannot be employed when the objective function is non-smooth, we benchmark our algorithm only with DP-ADMM and DPSGD similar to [26]. In Fig. 4, we plot the normalized error versus the outer loop iteration index for D-ZOA, DP-ADMM, and DPSGD. The plots show that all algorithms converge for two different values of ϵ and δ. In all plots, accuracy improves as ϵ increases. This is consistent with both Theorem 3 and [26, Theorem 3]. The hyper-parameters in DP-ADMM and DPSGD are tuned to achieve the best accuracy and convergence rate. However, D-ZOA has higher accuracy than DP-ADMM and DPSGD.

In Fig. 5, we illustrate the privacy-accuracy trade-off for D-ZOA, DP-ADMM, and DPSGD. The figures show that D-ZOA, DP-ADMM, and DPSGD achieve higher accuracy with larger ϵ and δ. In Fig. 5(a), we show the normalized error versus the privacy parameter $\bar{\epsilon}$ as given in (25) for $\delta = 10^{-6}$ and $\delta = 10^{-3}$. We observe that D-ZOA outperforms both DP-ADMM and DPSGD in terms of accuracy likely due to its intrinsic privacy-preserving properties. Fig. 5(b) also attests to the superiority of D-ZOA over DP-ADMM and DPSGD when $\epsilon = 0.15$ and $\epsilon = 0.95$ and δ varies between 10^{-6} and 10^{-2}.

We also evaluate the performance of the D-ZOA algorithm in comparison with the considered benchmark algorithms for the ERM with the ridge penalty. In Fig. 6, we plot the normalized error versus the outer loop iteration index for D-ZOA, DP-ADMM, DPSGD, PVP, and DVP. The plots show that D-ZOA outperforms all other considered algorithms in terms of accuracy for different values of ϵ.

In Fig. 7, we demonstrate the privacy-accuracy trade-off for D-ZOA, DP-ADMM, DPSGD, PVP, and DVP. As expected, smaller values of the privacy parameters ϵ and δ lead to lower accuracy. However, D-ZOA outperforms all the other approaches in terms of accuracy due to its intrinsic privacy-preserving properties.

In the considered applications of ERM with lasso and ridge penalty, we make the following observations regarding the complexity-accuracy trade-off of D-ZOA and the baseline algorithms. D-ZOA outperforms all the baseline algorithms in terms of accuracy by roughly two orders of magnitude. However, D-ZOA has a relatively high computational complexity due to its inner loop that is run at every agent k and every ADMM iteration. Since the number of arithmetic operations required to evaluate the objective function is $\mathcal{O}(PN_k)$, calculation of (8) needs $\mathcal{O}(JPN_k)$ operations and, therefore, D-ZOA requires $\mathcal{O}(TJPN_k)$ operations to perform (4). The baseline algorithms have the following computational complexities: $\mathcal{O}(P^2N_k)$ for DP-ADMM and DPSGD, and $\mathcal{O}(P^2N_k + P^3)$ for PVP and DVP.

VII. Conclusion

We proposed an intrinsically privacy-preserving consensus-based algorithm for solving a class of distributed regularized ERM problems where first-order information is hard or even impossible to obtain. We recast the original problem into an equivalent constrained optimization problem whose structure is suitable for distributed implementation via ADMM. We employed a zeroth-order method, known as the two-point stochastic-gradient algorithm, to minimize the augmented Lagrangian in the primal update step. We proved that the inherent randomness due to employing the zeroth-order method can adequately make the D-ZOA algorithm intrinsically privacy-preserving. In addition, we used the moments accountant method to show that the total privacy leakage of D-ZOA grows sublinearly with the number of ADMM iterations. We verified the convergence of D-ZOA to a near-optimal solution as well as studying its privacy-preserving properties through both theoretical analysis and numerical simulations.

Appendix A
Proof of Lemma 1

Proof. We prove this lemma in two steps. First, we prove that $\mathbb{E}[\boldsymbol{\beta}_k^{(m)}] = \breve{\boldsymbol{\beta}}_k^{(m)}$. Then, we calculate the covariance of $\boldsymbol{\beta}_k^{(m)}$.

We prove that $\mathbb{E}[\boldsymbol{\beta}_k^{(m)}] = \breve{\boldsymbol{\beta}}_k^{(m)}$ by induction over m.
Base case: Since $\boldsymbol{\beta}_k^{(0)} = \breve{\boldsymbol{\beta}}_k^{(0)} = \mathbf{0}$, we have $\mathbb{E}[\boldsymbol{\beta}_k^{(0)}] = \breve{\boldsymbol{\beta}}_k^{(0)}$.

Induction step: We assume that $\mathbb{E}[\boldsymbol{\beta}_k^{(m-1)}] = \breve{\boldsymbol{\beta}}_k^{(m-1)}$ as the induction hypothesis. Considering (14) and (11), we have

$$\begin{aligned}\mathbb{E}[\boldsymbol{\beta}_k^{(m)}] &= \mathbb{E}[\breve{\boldsymbol{\beta}}_k^{(m)}] + \mathbb{E}[\boldsymbol{\xi}_k^{(m)}] \\ &= -\frac{1}{2\rho|\mathcal{V}_k|} f_k'(\breve{\boldsymbol{\beta}}_k^{(m)}) + \frac{1}{2|\mathcal{V}_k|}\Big(|\mathcal{V}_k|\breve{\boldsymbol{\beta}}_k^{(m-1)} \\ &\quad + \sum_{l\in\mathcal{V}_k} \breve{\boldsymbol{\beta}}_l^{(m-1)}\Big) - \frac{1}{2\rho|\mathcal{V}_k|}\boldsymbol{\gamma}_k^{(m-1)} + \mathbb{E}[\boldsymbol{\xi}_k^{(m)}] \\ &= -\frac{1}{2\rho|\mathcal{V}_k|}\mathbb{E}[f_k'(\breve{\boldsymbol{\beta}}_k^{(m)})] \\ &\quad + \frac{1}{2|\mathcal{V}_k|}\Big(|\mathcal{V}_k|\mathbb{E}[\boldsymbol{\beta}_k^{(m-1)}] + \sum_{l\in\mathcal{V}_k}\mathbb{E}[\boldsymbol{\beta}_l^{(m-1)}]\Big) \\ &\quad - \frac{1}{2\rho|\mathcal{V}_k|}\mathbb{E}[\boldsymbol{\gamma}_k^{(m-1)}] + \mathbb{E}[\boldsymbol{\xi}_k^{(m)}] \\ &= \mathbb{E}[\boldsymbol{\beta}_k^{(m)}] + \mathbb{E}[\boldsymbol{\xi}_k^{(m)}],\end{aligned}$$

which implies that $\mathbb{E}[\boldsymbol{\xi}_k^{(m)}] = \mathbf{0}$. Therefore, $\mathbb{E}[\boldsymbol{\beta}_k^{(m)}] = \breve{\boldsymbol{\beta}}_k^{(m)}$.

Since we have $\mathbf{g}_k^{(t)} \sim \mathcal{N}(\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Psi}_k^{(t)})$ and $\boldsymbol{\beta}_k^{(m)} = -\sum_{t=1}^{T}\alpha_t\mathbf{g}_k^{(t)}$, in view of the additive property of the normal distribution, $\boldsymbol{\beta}_k^{(m)}$ is normally distributed with the mean $\breve{\boldsymbol{\beta}}_k^{(m)}$ and the covariance

$$\mathrm{cov}[\boldsymbol{\beta}_k^{(m)}] = \frac{1}{J}\sum_{t=1}^{T}\alpha_t^2\boldsymbol{\Psi}_k^{(t)}. \tag{33}$$

□

Appendix B
Proof of Lemma 2

Proof. It is easy to verify that

$$\mathrm{tr}(\boldsymbol{\Psi}_k^{(t)}) = \mathbb{E}\left[\left\|\mathbf{g}_{j,k}^{(t)}\right\|^2\right] - \left\|\boldsymbol{\mu}_k^{(t)}\right\|^2. \tag{34}$$

By [29, Lemma 2], there exists a constant c such that

$$\mathbb{E}\left[\left\|\mathbf{g}_{j,k}^{(t)}\right\|^2\right] \le cL^2P\Big(\sqrt{\frac{u_{2,t}}{u_{1,t}}}P + 1 + \log(P)\Big). \tag{35}$$

Since $u_{2,t}/u_{1,t} = P^{-2}t^{-1}$, we have

$$\mathbb{E}\left[\left\|\mathbf{g}_{j,k}^{(t)}\right\|^2\right] \le cL^2P\Big(\frac{1}{\sqrt{t}} + 1 + \log(P)\Big). \tag{36}$$

In addition, from $\boldsymbol{\beta}_k^{(m)} = -\sum_{t=1}^{T}\alpha_t\mathbf{g}_k^{(t)}$ and (11), we have

$$\breve{\boldsymbol{\beta}}_k^{(m)} = -\sum_{t=1}^{T}\alpha_t\boldsymbol{\mu}_k^{(t)}. \tag{37}$$

Taking the Euclidean norm of both sides in (37) and using the triangle inequality, we have

$$\left\|\breve{\boldsymbol{\beta}}_k^{(m)}\right\| = \left\|-\sum_{t=1}^{T}\alpha_t\boldsymbol{\mu}_k^{(t)}\right\| \le \sum_{t=1}^{T}|\alpha_t|\left\|\boldsymbol{\mu}_k^{(t)}\right\|. \tag{38}$$

Squaring both sides of (38) and using the Cauchy-Schwarz inequality, we get

$$\left\|\breve{\boldsymbol{\beta}}_k^{(m)}\right\|^2 \le \Big(\sum_{t=1}^{T}|\alpha_t|\left\|\boldsymbol{\mu}_k^{(t)}\right\|\Big)^2 \le T\sum_{t=1}^{T}|\alpha_t|^2\left\|\boldsymbol{\mu}_k^{(t)}\right\|^2 \tag{39}$$

and consequently

$$-\frac{1}{JP}\sum_{t=1}^{T}\alpha_t^2\left\|\boldsymbol{\mu}_k^{(t)}\right\|^2 \le -\frac{1}{TJP}\left\|\breve{\boldsymbol{\beta}}_k^{(m)}\right\|^2. \tag{40}$$

Using (36), (40), and the definition of α_t after (9), we have

$$\begin{aligned}&\frac{1}{JP}\sum_{t=1}^{T-1}\alpha_t^2\mathrm{tr}(\boldsymbol{\Psi}_k^{(t)}) \\ \le&\frac{1}{JP}\sum_{t=1}^{T-1}\alpha_t^2cL^2P\Big(\frac{1}{\sqrt{t}} + 1 + \log(P)\Big) \\ &- \frac{1}{JP}\sum_{t=1}^{T-1}\alpha_t^2\left\|\boldsymbol{\mu}_k^{(t)}\right\|^2 \\ =&\frac{1}{JP}\frac{c\alpha_0^2R^2}{\log(2P)}\Big(\sum_{t=1}^{T}\frac{1}{t\sqrt{t}} + (1+\log(P))\sum_{t=1}^{T}\frac{1}{t}\Big) \\ &- \frac{1}{TJP}\left\|\breve{\boldsymbol{\beta}}_k^{(m)}\right\|^2.\end{aligned} \tag{41}$$

Defining $s_1 = \sum_{t=1}^{T-1}t^{-1}$ and $s_2 = \sum_{t=1}^{T-1}t^{-1.5}$, (41) simplifies to

$$\begin{aligned}&\frac{1}{JP}\sum_{t=1}^{T}\alpha_t^2\mathrm{tr}(\boldsymbol{\Psi}_k^{(t)}) \\ &\le \frac{c\alpha_0^2R^2}{JP\log(2P)}\Big(s_1(1+\log(P)) + s_2\Big) - \frac{\left\|\breve{\boldsymbol{\beta}}_k^{(m)}\right\|^2}{TJP}.\end{aligned} \tag{42}$$

Considering that the algorithm converges as proven in Section V, i.e., $\breve{\boldsymbol{\beta}}_k^{(m)} \to \boldsymbol{\beta}^c$ as $m \to \infty$, $\breve{\boldsymbol{\beta}}_k^{(0)} = \mathbf{0}$, and the triangle inequality, for $m > 0$ we have

$$\left|\left\|\breve{\boldsymbol{\beta}}_k^{(m)}\right\| - \|\boldsymbol{\beta}^c\|\right| \le \left\|\breve{\boldsymbol{\beta}}_k^{(m)} - \boldsymbol{\beta}^c\right\| \le \|\boldsymbol{\beta}^c\|, \tag{43}$$

which implies $\left\|\breve{\boldsymbol{\beta}}_k^{(m)}\right\| \le 2\|\boldsymbol{\beta}^c\|$. Therefore, we obtain

$$\begin{aligned}&\frac{1}{JP}\sum_{t=1}^{T}\alpha_t^2\mathrm{tr}(\boldsymbol{\Psi}_k^{(t)}) \\ &= \frac{c\alpha_0^2R^2}{JP\log(2P)}\Big(s_1(1+\log(P)) + s_2\Big) - \frac{4\|\boldsymbol{\beta}^c\|^2}{TJP}.\end{aligned} \tag{44}$$

□

and consequently (17).

Appendix C
Proof of Lemma 3

Proof. From the adopted exact primal update equation (14), we obtain

$$\begin{aligned}\breve{\boldsymbol{\beta}}_{k,\mathcal{D}_k}^{(m)} =& -\frac{0.5}{\rho|\mathcal{V}_k|}\Big(\frac{1}{N_k}\sum_{j=1}^{N_k}\ell'(\mathbf{x}_{k,j},y_{k,j};\breve{\boldsymbol{\beta}}_k)+\boldsymbol{\gamma}_k^{(m-1)}\Big)\\ &+\frac{0.5}{|\mathcal{V}_k|}\Big(\breve{\boldsymbol{\beta}}_k^{(m-1)}+\sum_{l\in\mathcal{V}_k}\breve{\boldsymbol{\beta}}_l^{(m-1)}+\frac{\eta R'(\breve{\boldsymbol{\beta}}_k)}{\rho K}\Big)\\ \breve{\boldsymbol{\beta}}_{k,\mathcal{D}'_k}^{(m)} =& -\frac{0.5}{\rho|\mathcal{V}_k|}\Big(\frac{1}{N_k}\sum_{j=1}^{N_k-1}\ell'(\mathbf{x}_{k,j},y_{k,j};\breve{\boldsymbol{\beta}}_k)+\boldsymbol{\gamma}_k^{(m-1)}\\ &+\frac{1}{N_k}\ell'(\mathbf{x}'_{k,N_k},y'_{k,N_k};\breve{\boldsymbol{\beta}}_k)\Big)\\ &+\frac{0.5}{|\mathcal{V}_k|}\Big(\breve{\boldsymbol{\beta}}_k^{(m-1)}+\sum_{l\in\mathcal{V}_k}\breve{\boldsymbol{\beta}}_l^{(m-1)}+\frac{\eta R'(\breve{\boldsymbol{\beta}}_k)}{\rho K}\Big).\end{aligned} \tag{45}$$

Using Assumption 1, the quantity $\left\|\breve{\boldsymbol{\beta}}_{k,\mathcal{D}_k}^{(m)}-\breve{\boldsymbol{\beta}}_{k,\mathcal{D}'_k}^{(m)}\right\|$ is upper bounded as follows

$$\begin{aligned}&\left\|\breve{\boldsymbol{\beta}}_{k,\mathcal{D}_k}^{(m)}-\breve{\boldsymbol{\beta}}_{k,\mathcal{D}'_k}^{(m)}\right\|\\ &=\frac{\left\|\ell'(\mathbf{x}'_{k,N_k},y'_{k,N_k};\breve{\boldsymbol{\beta}}_k)-\ell'(\mathbf{x}_{k,N_k},y_{k,N_k};\breve{\boldsymbol{\beta}}_k)\right\|}{2\rho|\mathcal{V}_k|N_k}\\ &\le\frac{c_1}{\rho|\mathcal{V}_k|N_k}.\end{aligned} \tag{46}$$

□

Appendix D
Proof of Theorem 1

Proof. The privacy loss due to sharing $\boldsymbol{\beta}_k^{(m)}$ is calculated as

$$\left|\log\frac{\Pr[\boldsymbol{\beta}_{k,\mathcal{D}_k}^{(m)}]}{\Pr[\boldsymbol{\beta}_{k,\mathcal{D}'_k}^{(m)}]}\right|=\left|\log\frac{\Pr[\boldsymbol{\xi}_{k,\mathcal{D}_k}^{(m)}]}{\Pr[\boldsymbol{\xi}_{k,\mathcal{D}'_k}^{(m)}]}\right|=\left|\log\frac{\Pr[\xi_{s,k,\mathcal{D}_k}^{(m)}]}{\Pr[\xi_{s,k,\mathcal{D}'_k}^{(m)}]}\right| \tag{47}$$

where the first equality holds since the Jacobian matrix of the linear transformation from $\boldsymbol{\beta}_k^{(m)}$ to $\boldsymbol{\xi}_k^{(m)}$ is the identity matrix and the second equality holds as the entries of $\boldsymbol{\xi}_k^{(m)}$, denoted by $\xi_{s,k}^{(m)}$, are independent of each other, for any entry s.

Using the triangle inequality, Lemma 3, and substituting σ_k in the resulting expression, we obtain

$$\left|\log\frac{\Pr[\boldsymbol{\beta}_{k,\mathcal{D}_k}^{(m)}]}{\Pr[\boldsymbol{\beta}_{k,\mathcal{D}'_k}^{(m)}]}\right|\le\frac{\rho|\mathcal{V}_k|N_k\epsilon^2}{2.1c_1\log(1.25/\delta)}\left|\xi_{s,k}^{(m)}+\frac{c_1}{2\rho|\mathcal{V}_k|N_k}\right|. \tag{48}$$

When

$$|\xi_{s,k}^{(m)}|\le\frac{c_1}{\rho|\mathcal{V}_k|N_k}\left(2.1\epsilon^{-1}\log(1.25/\delta)-0.5\right),$$

the privacy loss is bounded by ϵ. Hence, let us define

$$r=\frac{c_1}{\rho|\mathcal{V}_k|N_k}\left(2.1\epsilon^{-1}\log(1.25/\delta)-0.5\right).$$

Subsequently, we need to prove that

$$\Pr[|\xi_{s,k}^{(m)}|>r]\le\delta$$

or equivalently

$$\Pr[\xi_{s,k}^{(m)}>r]\le0.5\delta.$$

Using the tail bound of the normal distribution $\mathcal{N}(0,\sigma_k^2)$ [35], we obtain

$$\Pr[\xi_{s,k}^{(m)}>r]\le\frac{\sigma_k}{r\sqrt{2\pi}}\exp\Big(-\frac{r^2}{2\sigma_k^2}\Big). \tag{49}$$

Since δ is assumed to be small (≤0.01) and $\epsilon\le1$, we have $\sigma_k<r$ and $-r^2<2\sigma_k^2\log(0.5\sqrt{2\pi}\delta)$. Therefore, $\Pr[\xi_{s,k}^{(m)}>r]<0.5\delta$, which implies $\Pr[|\xi_{s,k}^{(m)}|>r]\le\delta$. By defining

$$\mathbb{A}_1=\{\xi_{s,k}^{(m)}:|\xi_{s,k}^{(m)}|\le r\},\quad\mathbb{A}_2=\{\xi_{s,k}^{(m)}:|\xi_{s,k}^{(m)}|>r\},$$

we have

$$\begin{aligned}\Pr[\boldsymbol{\beta}_{k,\mathcal{D}_k}^{(m)}]&=\Pr[\breve{\beta}_{s,k,\mathcal{D}_k}^{(m)}+\xi_{s,k}^{(m)}:\xi_{s,k}^{(m)}\in\mathbb{A}_1]\\ &\quad+\Pr[\breve{\beta}_{s,k,\mathcal{D}_k}^{(m)}+\xi_{s,k}^{(m)}:\xi_{s,k}^{(m)}\in\mathbb{A}_2]\\ &<e^{\epsilon}\Pr[\boldsymbol{\beta}_{k,\mathcal{D}'_k}^{(m)}]+\delta,\end{aligned} \tag{50}$$

which concludes the proof by showing that, at each iteration of D-ZOA, (ϵ,δ)-differential privacy is guaranteed. □

Appendix E
Proof of Theorem 3

Proof. In virtue of [38, Lemma 1 and Lemma 2], $\breve{\mathbf{w}}^{(m)}$ satisfies the following equation

$$\frac{f'(\breve{\mathbf{w}}^{(m)})}{\rho}=2\mathbf{H}\boldsymbol{\xi}^{(m)}-2\mathbf{Q}\mathbf{r}^{(m)}-\mathbf{L}_+(\mathbf{w}^{(m)}-\mathbf{w}^{(m-1)}). \tag{51}$$

Therefore, by using (51), [38, Lemma 3, Lemma 4 and Lemma 5], and the steps in the proof of [39, Theorem 1], we can show that, for any $\mathbf{r}\in\mathbb{R}^{KP}$ and $m>0$, we have

$$\begin{aligned}&\frac{f(\breve{\mathbf{w}}^{(m)})-f(\mathbf{w}^*)}{\rho}+2\mathbf{r}^{\mathsf{T}}\mathbf{Q}\breve{\mathbf{w}}^{(m)}\\ &\le\frac{\left\|\mathbf{q}^{(m-1)}-\mathbf{q}\right\|_{\mathbf{G}}^2}{\rho}-\frac{\left\|\mathbf{q}^{(m)}-\mathbf{q}\right\|_{\mathbf{G}}^2}{\rho}\\ &\quad-\left\|\mathbf{Q}\breve{\mathbf{w}}^{(m)}\right\|^2-\left\|\mathbf{Q}\boldsymbol{\xi}^{(m)}\right\|^2+2(\boldsymbol{\xi}^{(m)})^{\mathsf{T}}\mathbf{Q}(\mathbf{r}^{(m)}-\mathbf{r})\\ &\quad+2\Big(\frac{\mathbf{L}_+}{2}(\breve{\mathbf{w}}^{(m)}-\mathbf{w}^*)\Big)^{\mathsf{T}}(\mathbf{w}^{(m-1)}-\breve{\mathbf{w}}^{(m-1)})\end{aligned} \tag{52}$$

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

For any symmetric matrix $\mathbf{X}\in\mathbb{R}^{P\times P}$ and vector $\mathbf{y}\in\mathbb{R}^P$, we have

$$\|\mathbf{y}\|^2\lambda_{\text{min}}(\mathbf{X})\le\mathbf{y}^{\mathsf{T}}\mathbf{X}\mathbf{y}\le\|\mathbf{y}\|^2\lambda_{\text{max}}(\mathbf{X})$$

and, for any $\mathbf{a},\mathbf{b}\in\mathbb{R}^P$ and $\tau\in\mathbb{R}_+$, we have

$$2\mathbf{a}^{\mathsf{T}}\mathbf{b}\le\tau^{-1}\|\mathbf{a}\|^2+\tau\|\mathbf{b}\|^2.$$

Therefore, (52) yields

$$\begin{aligned}&\frac{f(\breve{\mathbf{w}}^{(m)})-f(\mathbf{w}^*)}{\rho}+2\mathbf{r}^\mathsf{T}\mathbf{Q}\breve{\mathbf{w}}^{(m)}\\ \le&\frac{\left\|\mathbf{q}^{(m-1)}-\mathbf{q}\right\|_\mathbf{G}^2}{\rho}-\frac{\left\|\mathbf{q}^{(m)}-\mathbf{q}\right\|_\mathbf{G}^2}{\rho}-\left\|\mathbf{Q}\boldsymbol{\xi}^{(m)}\right\|^2\\ &-\frac{\lambda_{\text{min}}(\mathbf{L}_-)}{2}\left\|\breve{\mathbf{w}}^{(m)}-\mathbf{w}^*\right\|^2+\frac{1}{\tau}\left\|\frac{\mathbf{L}_+}{2}(\breve{\mathbf{w}}^{(m)}-\mathbf{w}^*)\right\|^2\\ &+\tau\left\|\mathbf{w}^{(m-1)}-\breve{\mathbf{w}}^{(m-1)}\right\|^2+2(\boldsymbol{\xi}^{(m)})^\mathsf{T}\mathbf{Q}(\mathbf{r}^{(m)}-\mathbf{r}).\end{aligned} \tag{53}$$

By setting

$$\tau=\frac{\lambda_{\text{max}}^2(\mathbf{L}_+)}{2\lambda_{\text{min}}(\mathbf{L}_-)},$$

(53) leads to

$$\begin{aligned}&\frac{f(\breve{\mathbf{w}}^{(m)})-f(\mathbf{w}^*)}{\rho}+2\mathbf{r}^\mathsf{T}\mathbf{Q}\breve{\mathbf{w}}^{(m)}\\ \le&\frac{\left\|\mathbf{q}^{(m-1)}-\mathbf{q}\right\|_\mathbf{G}^2}{\rho}-\frac{\left\|\mathbf{q}^{(m)}-\mathbf{q}\right\|_\mathbf{G}^2}{\rho}\\ &+\frac{\lambda_{\text{max}}^2(\mathbf{L}_+)}{2\lambda_{\text{min}}(\mathbf{L}_-)}\left\|\boldsymbol{\xi}^{(m-1)}\right\|^2+2(\boldsymbol{\xi}^{(m)})^\mathsf{T}\mathbf{Q}(\mathbf{r}^{(m)}-\mathbf{r}).\end{aligned} \tag{54}$$

Setting $\mathbf{r}=\mathbf{0}_P$ and summing both sides of (54) over $m=1$ to M gives

$$\begin{aligned}&\frac{1}{\rho}\sum_{m=1}^{M}(f(\breve{\mathbf{w}}^{(m)})-f(\mathbf{w}^*))\le\frac{1}{\rho}\left\|\mathbf{q}^{(0)}-\mathbf{q}\right\|_\mathbf{G}^2\\ &+\sum_{m=1}^{M}\frac{\lambda_{\text{max}}^2(\mathbf{L}_+)}{2\lambda_{\text{min}}(\mathbf{L}_-)}\left\|\boldsymbol{\xi}^{(m-1)}\right\|^2+2(\boldsymbol{\xi}^{(m)})^\mathsf{T}\mathbf{Q}\mathbf{r}^{(m)}.\end{aligned} \tag{55}$$

Using Jensen's inequality [43], (17), (20), and applying the expectation operator to both sides of (55), we obtain

$$\begin{aligned}&\mathbb{E}[f(\hat{\mathbf{w}}^{(M)})-f(\mathbf{w}^*)]\\ \le&\frac{1}{M}\left\|\mathbf{q}^{(0)}-\mathbf{q}\right\|_\mathbf{G}^2+\frac{\rho\lambda_{\text{max}}^2(\mathbf{L}_+)}{2M\lambda_{\text{min}}(\mathbf{L}_-)}\sum_{m=1}^{M}\mathbb{E}\left[\left\|\boldsymbol{\xi}^{(m-1)}\right\|^2\right]\\ \le&\frac{\left\|\mathbf{q}^{(0)}-\mathbf{q}\right\|_\mathbf{G}^2}{M}+\frac{2.1c_1^2P\log(1.25/\delta)\lambda_{\text{max}}^2(\mathbf{L}_+)}{2\rho|\mathcal{V}_k|^2N_k^2\epsilon^2\lambda_{\text{min}}(\mathbf{L}_-)}\end{aligned} \tag{56}$$

where

$$\hat{\mathbf{w}}^{(M)}=\frac{1}{M}\sum_{m=1}^{M}\breve{\mathbf{w}}^{(m)}.$$

□

References

[1] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed learning with non-smooth objective functions," in *Proc. European Speech and Signal Processing Conference*, Jan. 2021.

[2] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[3] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Consensus-based distributed total least-squares estimation using parametric semidefinite programming," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2019, pp. 5227–5231.

[4] ——, "Distributed ridge regression with feature partitioning," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018.

[5] J. Akhtar and K. Rajawat, "Distributed sequential estimation in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 86–100, Jan. 2018.

[6] N. K. D. Venkategowda and S. Werner, "Privacy-preserving distributed precoder design for decentralized estimation," in *Proc. IEEE Global Conference on Signal and Information Processing*, Nov. 2018.

[7] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, *Splitting Methods in Communication, Imaging, Science, and Engineering*, ser. Scientific Computation, R. Glowinski, S. J. Osher, and W. Yin, Eds. Cham: Springer International Publishing, 2016.

[8] D. Hajinezhad, M. Hong, and A. Garcia, "ZONE: Zeroth-order nonconvex multiagent optimization over networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 3995–4010, Oct. 2019.

[9] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[10] S. P. Talebi and S. Werner, "Distributed Kalman filtering and control through embedded average consensus information fusion," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4396–4403, Mar. 2019.

[11] A. Agarwal, O. Dekel, and L. Xiao, "Optimal algorithms for online convex optimization with multi-point bandit feedback," in *Proc. 23rd Annual Conference on Learning Theory*, Jun. 2010, pp. 28–40.

[12] J. C. Spall, *Introduction to Stochastic Search and Optimization*. Wiley, 2003.

[13] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop on Artificial Intelligence and Security*, Nov. 2017, pp. 15–26.

[14] S. Liu, J. Chen, P.-Y. Chen, and A. Hero, "Zeroth-order online alternating direction method of multipliers: convergence analysis and applications," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 84, Apr. 2018, pp. 288–297.

[15] F. Huang, S. Gao, S. Chen, and H. Huang, "Zeroth-order stochastic alternating direction method of multipliers for nonconvex nonsmooth optimization," in *Proc. 28th International Joint Conference on Artificial Intelligence*, S. Kraus, Ed., 2019, pp. 2549–2555.

[16] S. Liu, P. Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero III, and P. K. Varshney, "A primer on zeroth-order optimization in signal processing and machine learning: principals, recent advances, and applications," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 43–54, 2020.

[17] T. Zhang and Q. Zhu, "Dynamic differential privacy for ADMM-based distributed classification learning," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 172–187, Jan. 2017.

[18] X. Zhang, M. M. Khalili, and M. Liu, "Recycled ADMM: Improve privacy and accuracy with less computation in distributed algorithms," in *Proc. 56th Annual Allerton Conference on Communication, Control, and Computing*, Oct. 2018, pp. 959–965.

[19] X. Zhang, M. M. Khalili, and M. Liu, "Improving the privacy and accuracy of ADMM-based distributed algorithms," in *Proc. 35th International Conference on Machine Learning*, vol. 80, Jul. 2018, pp. 5796–5805.

[20] J. Ding, Y. Gong, M. Pan, and Z. Han, "Optimal differentially private ADMM for distributed machine learning," 2019. [Online]. Available: http://arxiv.org/abs/1901.02094

[21] J. Ding, S. M. Errapotu, H. Zhang, Y. Gong, M. Pan, and Z. Han, "Stochastic ADMM based distributed machine learning with differential privacy," in *Proc. 15th SecureComm*, Oct. 2019, pp. 257–277.

[22] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proc. 2015 International Conference on Distributed Computing and Networking*, 2015.

[23] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 50–64, 2017.

[24] M. T. Hale and M. Egerstedty, "Differentially private cloud-based multi-agent optimization with constraints," in *Proc. 2015 American Control Conference*, Jul. 2015, pp. 1235–1240.

[25] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via functional perturbation," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 395–408, 2018.

[26] Z. Huang, R. Hu, Y. Guo, E. Chan-Tin, and Y. Gong, "DP-ADMM: ADMM-based distributed learning with differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1002–1012, 2020.

[27] F. Farokhi, N. Wu, D. Smith, and M. A. Kaafar, "The cost of privacy in asynchronous differentially-private machine learning," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2118–2129, 2021.

[28] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2013.

[29] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: the power of two function evaluations," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2788–2806, May 2015.

[30] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[31] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, pp. 1663–1707, Aug. 2010.

[32] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 2015 ACM SIGSAC Conference on Computer and Communications Security*, Oct. 2015, pp. 1322–1333.

[33] Y. Hu, P. Liu, L. Kong, and D. Niu, "Learning privately over distributed features: an ADMM sharing approach," 2019. [Online]. Available: http://arxiv.org/abs/1907.07735

[34] Z. Han, M. Hong, and D. Wang, *Signal processing and networking for big data applications*. Cambridge University Press, 2017.

[35] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, Aug. 2014.

[36] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Third Conference on Theory of Cryptography*. Springer-Verlag, 2006, pp. 265–284.

[37] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[38] Q. Li, B. Kailkhura, R. Goldhahn, P. Ray, and P. K. Varshney, "Robust federated learning using ADMM in the presence of data falsifying byzantines," 2017. [Online]. Available: http://arxiv.org/abs/1710.05241

[39] ——, "Robust decentralized learning using ADMM with unreliable agents," 2018. [Online]. Available: http://arxiv.org/abs/1710.05241

[40] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2010.

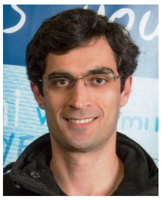
[41] Y. Tang, J. Zhang, and N. Li, "Distributed zero-order algorithms for nonconvex multi-agent optimization," 2020. [Online]. Available: https://arxiv.org/abs/1908.11444

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

[43] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*, 4th ed. McGraw Hill, 2002.

Cristiano Gratton received both the B.Sc. and M.Sc. degree in Mathematics from the University of Udine (Italy) in 2014 and 2017, respectively. He is pursuing the Ph.D. degree at the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway. His research interests are around distributed optimization and private data analysis.

Naveen K. D. Venkategowda Naveen K. D. Venkategowda (S'12–M'17) received the B.E. degree in electronics and communication engineering from Bangalore University, Bengaluru, India, in 2008, and the Ph.D. degree in electrical engineering from Indian Institute of Technology, Kanpur, India, in 2016. He is currently an Universitetslektor at the Department of Science and Technology, Linköping University, Sweden. From Oct. 2017 to Feb. 2021, he was postdoctoral researcher at the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway. He was a Research Professor at the School of Electrical Engineering, Korea University, South Korea from Aug. 2016 to Sep. 2017. He was a recipient of the TCS Research Fellowship (2011-15) from TCS for graduate studies in computing sciences and the ERCIM Alain Bensoussan Fellowship in 2017.



Reza Arablouei received the Ph.D. degree in telecommunications engineering from the Institute for Telecommunications Research, University of South Australia, Mawson Lakes, SA, Australia, in 2013. He was a Research Fellow with the University of South Australia from 2013 to 2015. He is currently a Senior Research Scientist with the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Pullenvale, QLD, Australia. His research interests include signal processing and machine learning on embedded systems.

Stefan Werner (SM'07) received the M.Sc. degree in electrical engineering from the Royal Institute of Technology, Stockholm, Sweden, in 1998, and the D.Sc. degree (Hons.) in electrical engineering from the Signal Processing Laboratory, Helsinki University of Technology, Espoo, Finland, in 2002. He is currently a Professor at the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), and Director of IoT@NTNU. He is also an Adjunct Professor with Aalto University in Finland and an Adjunct Senior Research Fellow with the Institute for Telecommunications Research, University of South Australia. He was a visiting Melchor Professor with the University of Notre Dame during summer 2019 and held an Academy Research Fellowship, funded by the Academy of Finland, from 2009 to 2014. His research interests include adaptive and statistical signal processing, signal processing for communications, and security and privacy in cyberphysical 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.

Distributed Ridge Regression with Feature Partitioning

Cristiano Gratton*, Naveen K. D. Venkategowda*, Reza Arablouei†, Stefan Werner*
* Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway
† CSIRO's Data61, Pullenvale QLD 4069, Australia

***Abstract*—We develop a new distributed algorithm to solve the ridge regression problem with feature partitioning of the observation matrix. The proposed algorithm, named D-Ridge, is based on the alternating direction method of multipliers (ADMM) and estimates the parameters when the observation matrix is distributed among different agents with feature (or vertical) partitioning. We formulate the associated ridge regression problem as a distributed convex optimization problem and utilize the ADMM to obtain an iterative solution. Numerical results demonstrate that D-Ridge converges faster than its diffusion-based contender does.**

I. Introduction

With the recent advances in technology, ever-growing amounts of data are constantly collected and stored on electronic devices, which are often geographically dispersed. Transporting the entire data to a central processing unit is often unfeasible due to energy constraints or privacy concerns. In addition, concentrating the data in a central hub can create a single point of failure. Hence, we need algorithms that are capable of processing data spread across multiple agents. They ought to operate in a distributed fashion relying only on the available local information [1]–[10].

Distributed solutions for learning, inference, or prediction using sensor data are highly demanded in many of today's data analysis tasks pertaining to statistics, signal processing and machine learning. In this context, an important data analysis tool is the distributed multivariate linear regression.

In recent years, there have been several works describing algorithms to distribute regression problems, i.e., [5]–[19]. In particular, shrinkage methods such as ridge regression and lasso have attracted a lot of attention since they play an important role in preventing the problem from being ill-posed due to possible rank deficiency of the observation matrix. Moreover, such methods regularize the regression parameters by imposing a penalty on their size or density to avoid overfitting [6], [8], [19]–[21]. Example applications are in wireless sensor networks operating under strict power budget constraints where agents collecting and processing data are distributed over a large geographical area [8].

A central issue in distributed regression is how the data are distributed among agents. Horizontal partitioning of data refers to the case when the data samples containing all features are distributed over the network. On the other hand, when subsets of features of all data samples are distributed over the agents, we have feature (vertical) partitioning of data [22]. Regression problems with horizontally partitioned data have been considered for example in [7], [8], [23]. In the framework of vertically partitioned data, some applications related to clustering and classification have been considered in [24], [25]. The regression problem with feature partitioning has also previously been considered in [6], [19]–[21]. However, works of [20], [21] assume a proper coloring scheme of the network and cannot be extended to a general graph labeling. The algorithm proposed in [19] is not truly distributed since its consensus constraints involve the entire network instead of each agent's local neighborhood. The algorithm in [6] is fully distributed and based on the diffusion strategy [26]. However, as we will show later on, it converges relatively slowly.

In this paper, we solve the ridge regression problem with feature partitioning of the observation matrix in a distributed fashion using the alternating direction method of multipliers (ADMM). The proposed algorithm, called D-Ridge, is fully distributed and requires communications only among neighboring agents. It also converges faster than the diffusion-based algorithm of [6] and has a per-iteration per-agent computational complexity order that is linear in the sample size. In addition, D-Ridge does not require the agents to share their local data or dual variables with the other agents but only the primal variables, which are the estimate solutions of the corresponding local optimization subproblems. Hence, D-Ridge respects the possible data privacy of the agents. We verify the convergence of D-Ridge to the centralized solution at all agents through both theoretical analysis and simulations. Our experiments with a verity of network topologies show that D-Ridge outperforms its diffusion-based contender in terms of convergence rate.

II. System Model

We consider a network with $K \in \mathbb{N}$ agents modeled as an undirected graph $\mathcal{G}(\mathcal{K}, \mathcal{E})$ where the set of vertices $\mathcal{K} := \{1, \ldots, K\}$ corresponds to the agents and the edge set $\mathcal{E}$ represents the bidirectional communication links between the pairs of agents. Agent $k \in \mathcal{K}$ can communicate with the agents in its neighborhood set $\mathcal{N}_k$ whose cardinality is denoted by $|\mathcal{N}_k|$. The set $\mathcal{N}_k$ includes the agent k as well.

Let us denote the network-wide observations as an observation matrix $\mathbf{X} \in \mathbb{R}^{N \times P}$ and a response vector $\mathbf{y} \in \mathbb{R}^{N \times 1}$

This work was carried out during the tenure of an ERCIM 'Alain Bensoussan' Fellowship Programme and was partly supported by the Research Council of Noway.

where N is the number of data samples and P is the number of features in each sample. The data collected at each agent k are stored in the matrix $\mathbf{X}_k \in \mathbb{R}^{N \times P_k}$ where $\sum_{k=1}^{K} P_k = P$. Due to feature partitioning, the observation matrix $\mathbf{X} \in \mathbb{R}^{N \times P}$ consists of K submatrices $\mathbf{X}_k$, i.e., $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_K]$. Accordingly, the parameter vector $\boldsymbol{\beta} \in \mathbb{R}^{P \times 1}$ that establishes a linear regression between $\mathbf{X}$ and $\mathbf{y}$ is a stack of K subvectors $\boldsymbol{\beta}_k \in \mathbb{R}^{P_k \times 1}$, i.e., $\boldsymbol{\beta} = \left[\boldsymbol{\beta}_1^{\mathrm{T}}, \boldsymbol{\beta}_2^{\mathrm{T}}, \ldots, \boldsymbol{\beta}_K^{\mathrm{T}}\right]^{\mathrm{T}}$.

In the centralized approach, a ridge regression estimator of $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}}^o = \arg\min_{\boldsymbol{\beta}} \{\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \eta \|\boldsymbol{\beta}\|_2^2\} \tag{1}$$

where $\eta > 0$ is the regularization parameter. From the normal equation associated with (1), the centralized estimate is given by

$$\hat{\boldsymbol{\beta}}^o = \mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}} + \eta \mathbf{I}_N)^{-1}\mathbf{y} \tag{2}$$

where $\mathbf{I}_N$ indicates the $N \times N$ identity matrix.

Since computing a centralized solution of (1) over a network may be inefficient, we propose a distributed algorithm for this purpose in the following section.

III. Distributed Ridge Regression via ADMM

We first discuss the consensus-based reformulation of the ridge regression problem whose solution allows us to find a distributed solution to (1) via the ADMM. Then, we describe the construction steps and main properties of the proposed algorithm for solving the resulting constrained minimization problem. Finally, we establish the global convergence of D-Ridge theoretically.

A. Consensus-Based Reformulation of Ridge Regression

Let us define a vector $\mathbf{f}^o \in \mathbb{R}^{N \times 1}$ as

$$\mathbf{f}^o = (\mathbf{X}\mathbf{X}^{\mathrm{T}} + \eta \mathbf{I}_N)^{-1}\mathbf{y}.$$

From (2), the part of β^o corresponding to agent k can be calculated as

$$\hat{\boldsymbol{\beta}}_k^o = \mathbf{X}_k^{\mathrm{T}} \mathbf{f}^o. \tag{3}$$

For computing $\mathbf{f}^o$ at all agents using only in-network processing of the locally available data, we propose a consensus-based distributed algorithm. Note that $\mathbf{f}^o$ is the unique minimizer of the quadratic global cost function $\mathcal{J}(\mathbf{f})$ defined as

$$\mathcal{J}(\mathbf{f}) = \frac{1}{2}\mathbf{f}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}} + \eta \mathbf{I}_N)\mathbf{f} - \mathbf{f}^{\mathrm{T}}\mathbf{y}. \tag{4}$$

Since $\mathbf{X}\mathbf{X}^{\mathrm{T}} = \sum_{k=1}^{K} \mathbf{X}_k \mathbf{X}_k^{\mathrm{T}}$, $\mathbf{f}^o$ is given by

$$\mathbf{f}^o = \arg\min_{\mathbf{f}} \sum_{k=1}^{K} \mathcal{J}_k(\mathbf{f}) \tag{5}$$

where

$$\mathcal{J}_k(\mathbf{f}) = \frac{1}{2}\mathbf{f}^{\mathrm{T}}\Big(\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \frac{\eta}{K}\mathbf{I}_N\Big)\mathbf{f} - \frac{\delta_k}{B}\mathbf{f}^{\mathrm{T}}\mathbf{y}, \tag{6}$$

$B \in \mathbb{N}$ is the number of agents having access to $\mathbf{y}$, and $\delta_k = 1$ if $\mathbf{y}$ is available at agent k and $\delta_k = 0$ otherwise.

We introduce the local variables $\mathcal{F} := \{\mathbf{f}_k\}_{k=1}^{K}$ representing the local copies of $\mathbf{f}^o$ at the agents. Then, we reformulate the unconstrained optimization problem (5) as the following convex *constrained* minimization problem:

$$\{\mathbf{f}_k^o\}_{k=1}^{K} = \arg\min_{\{\mathbf{f}_k\}} \sum_{k=1}^{K} \frac{1}{2}\mathbf{f}_k^{\mathrm{T}}\Big(\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \frac{\eta}{K}\mathbf{I}_N\Big)\mathbf{f}_k - \frac{\delta_k}{B}\mathbf{f}_k^{\mathrm{T}}\mathbf{y}$$
$$\text{s.t. } \mathbf{f}_k = \mathbf{f}_l, \quad l \in \mathcal{N}_k, \quad k \in \mathcal{K}. \tag{7}$$

The equality constraints enforce local consensus over $\{\mathbf{f}_k\}$ across each agent's neighborhood.

To solve (7) in a distributed fashion, we use the ADMM [5]. Hence, we introduce the auxiliary local variables $\mathcal{A} := \{\mathbf{g}_k^l\}_{l \in \mathcal{N}_k}$ and rewrite the problem (7) as

$$\arg\min_{\{\mathbf{f}_k\}} \sum_{k=1}^{K} \frac{1}{2}\mathbf{f}_k^{\mathrm{T}}\Big(\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \frac{\eta}{K}\mathbf{I}_N\Big)\mathbf{f}_k - \frac{\delta_k}{B}\mathbf{f}_k^{\mathrm{T}}\mathbf{y}$$
$$\text{s.t. } \mathbf{f}_k = \mathbf{g}_k^l, \mathbf{f}_l = \mathbf{g}_k^l, \quad l \in \mathcal{N}_k, \quad k \in \mathcal{K}, \quad k \neq l. \tag{8}$$

Using the auxiliary variables $\mathcal{A}$ yields an equivalent alternative representation of the constraints in (7). These variables are only used to derive the local recursions and are eventually eliminated. Associating the Lagrange multipliers $\mathcal{V} := \{\{\boldsymbol{\mu}_k^l\}_{l \in \mathcal{N}_k}, \{\boldsymbol{\lambda}_k^l\}_{l \in \mathcal{N}_k}\}_{k=1}^{K}$ with the constraints in (8), we have the following augmented Lagrangian function:

$$\begin{aligned}\mathcal{L}_\rho(\mathcal{F}, \mathcal{A}, \mathcal{V}) = &\sum_{k=1}^{K}\Big(\frac{1}{2}\mathbf{f}_k^{\mathrm{T}}\Big(\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \frac{\eta}{K}\mathbf{I}_N\Big)\mathbf{f}_k - \frac{\delta_k}{B}\mathbf{f}_k^{\mathrm{T}}\mathbf{y}\Big) \\ &+ \sum_{k=1}^{K}\sum_{l \in \mathcal{N}_k}\Big((\boldsymbol{\mu}_k^l)^{\mathrm{T}}(\mathbf{f}_k - \mathbf{g}_k^l) + (\boldsymbol{\lambda}_k^l)^{\mathrm{T}}(\mathbf{f}_l - \mathbf{g}_k^l)\Big) \\ &+ \frac{\rho}{2}\sum_{k=1}^{K}\sum_{l \in \mathcal{N}_k}\Big(\big\|\mathbf{f}_k - \mathbf{g}_k^l\big\|_2^2 + \big\|\mathbf{f}_l - \mathbf{g}_k^l\big\|_2^2\Big)\end{aligned} \tag{9}$$

where the constant $\rho > 0$ is the penalty parameter.

Minimizing (7) through ADMM entails an iterative process that is described in the next section.

B. Algorithm Description

The D-Ridge algorithm consists of three steps at each iteration. First, the augmented Lagrangian function $\mathcal{L}_\rho$ is minimized with respect to $\mathcal{F}$. Second, $\mathcal{L}_\rho$ is minimized with respect to $\mathcal{A}$. Finally, the Lagrange multipliers in $\mathcal{V}$ are updated through gradient-ascent [27].

Thanks to the reformulation of the original problem (5) as (8), the augmented Lagrangian in (9) is decomposable both with respect to variables in $\mathcal{F}$, $\mathcal{A}$ and across agents.

Setting $\boldsymbol{\mu}_k(m) = 2\sum_{l \in \mathcal{N}_k} \boldsymbol{\mu}_k^l(m)$, and using the Karush-Kuhn-Tucker conditions of optimality [28] for (8), the auxiliary local variables $\mathcal{A}$ and multipliers $\mathcal{V}$ can be eliminated.

```
Algorithm 1 D-Ridge
  At all agents $k \in \mathcal{K}$, initialize $\mathbf{f}_k(0)$, $\boldsymbol{\mu}_k(0)$ to zero vectors,
  and run locally
  for $m = 0, 1, \ldots, M$ do
      Receive $\mathbf{f}_k(m)$ from neighbors in $\mathcal{N}_k$.
      Update $\boldsymbol{\mu}_k(m)$ as in (10).
      Update $\mathbf{f}_k(m+1)$ as in (11).
  end for
  Estimate $\hat{\boldsymbol{\beta}}_k = \mathbf{X}_k^{\mathrm{T}}\mathbf{f}_k(M+1)$.
```

Hence, the D-Ridge algorithm reduces to the following iterative updates that are carried out locally at every agent:

$$\boldsymbol{\mu}_k(m) = \boldsymbol{\mu}_k(m-1) + \rho \sum_{l\in\mathcal{N}_k} [\mathbf{f}_k(m) - \mathbf{f}_l(m)] \tag{10}$$

$$\begin{aligned}\mathbf{f}_k(m+1) &= \arg\min_{\{\mathbf{f}_k\}} \Big\{ \frac{1}{2}\mathbf{f}_k^{\mathrm{T}}\Big(\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \frac{\eta}{K}\mathbf{I}_N\Big)\mathbf{f}_k - \frac{\delta_k}{B}\mathbf{f}_k^{\mathrm{T}}\mathbf{y} \\ &\quad + \boldsymbol{\mu}_k^{\mathrm{T}}(m)\mathbf{f}_k + \rho \sum_{l\in\mathcal{N}_k} \Big\| \mathbf{f}_k - \frac{\mathbf{f}_k(m)+\mathbf{f}_l(m)}{2} \Big\|_2^2 \Big\} \\ &= \Big[\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \Big(\frac{\eta}{K} + 2\rho|\mathcal{N}_k|\Big)\mathbf{I}_N\Big]^{-1} \\ &\quad \Big(\frac{\delta_k}{B}\mathbf{y} - \boldsymbol{\mu}_k(m) + \rho|\mathcal{N}_k|\mathbf{f}_k(m) + \rho\sum_{l\in\mathcal{N}_k}\mathbf{f}_l(m)\Big)\end{aligned} \tag{11}$$

where m is the iteration index and all initial values $\{\mathbf{f}_k(0)\}_{k\in\mathcal{K}}$, $\{\boldsymbol{\mu}_k(0)\}_{k\in\mathcal{K}}$ are set to zero. The proposed approach is summarized in Algorithm 1.

Note that $\mathbf{f}_k(m)$ is the only vector that is shared between the agents at every iteration. The computation of (11) has a per-iteration per-agent complexity of $\mathcal{O}(N^3 + N^2P_k)$. It involves the inversion of the $N \times N$ matrix $\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \Big(\frac{\eta}{K} + 2\rho|\mathcal{N}_k|\Big)\mathbf{I}_N$ that may be computationally demanding for $N \gg P_k$. However, this operation can be carried out off-line before running the algorithm. We can also use the matrix inversion lemma to obtain $(\mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + c\mathbf{I}_N)^{-1} = c^{-1}[\mathbf{I}_N - \mathbf{X}_k(c\mathbf{I}_{P_k} + \mathbf{X}_k^{\mathrm{T}}\mathbf{X}_k)^{-1}\mathbf{X}_k^{\mathrm{T}}]$ where $c = \frac{\eta}{K} + 2\rho|\mathcal{N}_k|$. Hence, the dimensions of the matrix to be inverted become $P_k \times P_k$ entailing a per-iteration per-agent computational complexity of $\mathcal{O}(NP_k^2 + P_k^3)$.

In the next subsection, we show that D-Ridge generates sequences of local iterates $\mathbf{f}_k(m)$, $k = 1, \ldots, K$, that, at each agent k, converge to the global centralized solution $\mathbf{f}^o$ as $m \to \infty$.

C. Convergence Analysis

Convergence of the proposed algorithm is established by verifying that both conditions for the ADMM to converge are fulfilled, namely, for each agent $k \in \mathcal{K}$, the cost function $\mathcal{J}_k(\mathbf{f})$ is strongly convex and its gradient $\nabla_{\mathbf{f}}\mathcal{J}_k(\mathbf{f})$ is Lipschitz continuous [29].

The function $\mathcal{J}_k(\mathbf{f})$ is strongly convex since it is twice continuously differentiable and has a positive-definite Hessian matrix:

$$\nabla_{\mathbf{f}}^2\mathcal{J}_k(\mathbf{f}) = \mathbf{X}_k\mathbf{X}_k^{\mathrm{T}} + \frac{\eta}{K}\mathbf{I}_N \succ 0.$$

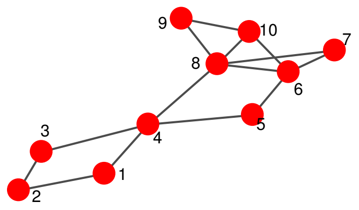


Fig. 1. Topology of the considered multi-agent network.

Moreover, $\nabla_{\mathbf{f}}\mathcal{J}_k(\mathbf{f})$ is a linear function of $\mathbf{f}$. Therefore, it is Lipschitz continuous [30] with a Lipschitz constant being the operator norm of $\nabla_{\mathbf{f}}^2\mathcal{J}_k(\mathbf{f})$.

IV. Simulations

The D-Ridge algorithm is tested here on a network of $K = 10$ agents with the topology as shown in Fig. 1. Each agent holds the data for two features. Therefore, $P_k = 2$, $k = 1, ..., K$, and $P = 20$. The observation data matrix $\mathbf{X}$ has $N = 50$ regressor vectors with independent zero-mean multivariate Gaussian distribution as its rows. The relationship between the entries of $\mathbf{y}$, denoted by $y_n \in \mathbb{R}$, and the rows of $\mathbf{X}$, denoted by $\mathbf{x}_n \in \mathbb{R}^{1\times P}$, with $n = 1, \ldots, N$, is governed by

$$y_n = \sum_{k=1}^{K} \mathbf{x}_{n,k}\boldsymbol{\beta}_k + \epsilon_n$$

where $\mathbf{x}_{n,k} \in \mathbb{R}^{1\times P_k}$ is the part of $\mathbf{x}_n$ that is available at agent k and $\epsilon_n \in \mathbb{R}$ is the zero-mean Gaussian noise with variance $\sigma_\epsilon^2 = 0.1$. The penalty parameter is set to $\rho = 4$ and, as in [6], the regularization parameter is set to $\eta = 10^{-3}$.

In Figs. 2-4, we plot the normalized mean squared error (MSE) versus the iteration index for D-Ridge and the diffusion-based algorithm of [6] with different values of the step-size μ.

The normalized MSE is defined as

$$n_{\mathrm{MSE}}(m) = \frac{\sum_{k=1}^{K} \|\boldsymbol{\beta}_k(m) - \boldsymbol{\beta}_k\|_2^2}{\|\boldsymbol{\beta}\|_2^2}$$

where $\boldsymbol{\beta}_k$ is given by (3) and $\boldsymbol{\beta}_k(m) = \mathbf{X}_k^{\mathrm{T}}\mathbf{f}_k(m)$.

The results in Figs. 2-4 are obtained by averaging over 100 independent trials. The number of agents having access to $\mathbf{y}$, i.e., B affects the convergence speed of D-Ridge, while it does not have any significant effect on the performance of the diffusion-based algorithm [6]. In Figs. 2-4, the regression vector is placed in the agent k with the greatest $|\mathcal{N}_k|$ if $B = 1$, while it is randomly placed over the network if $B > 1$.

Fig. 2 shows that D-Ridge converges significantly faster than the diffusion-based algorithm, especially when all agents have access to $\mathbf{y}$, i.e., $B = 10$. Fig. 3 shows that the D-Ridge algorithm converges faster as the number of agents that have access to $\mathbf{y}$ increases. Fig. 4 shows that D-Ridge converges faster than the diffusion-based algorithm irrespective of the

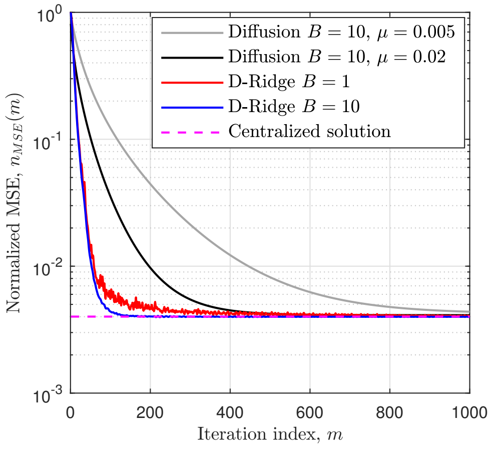


Fig. 2. Normalized MSE of D-Ridge and the diffusion-based algorithm with different values of the step-size $\boldsymbol{\mu}$ when one or all agents have access to $\mathbf{y}$.

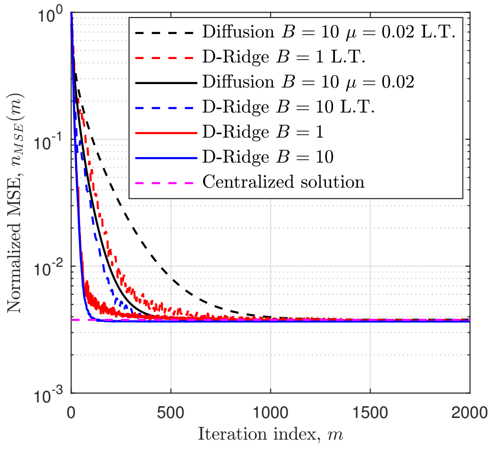


Fig. 4. Normalized MSE of D-Ridge and the diffusion-based algorithm for the considered network topology and for the linear topology (L.T.).

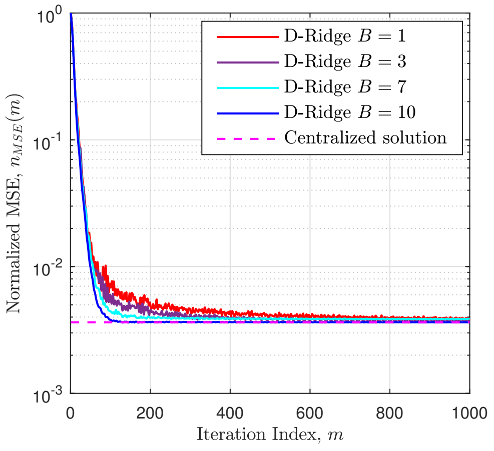


Fig. 3. Normalized MSE of D-Ridge for different values of B.

network topology. The performance of the algorithm with the topology shown in Fig. 1 is compared to a linear topology with the same number of agents where the agents are connected one after the other, hence $|\mathcal{N}_k| = 3$ for $1 < k < K$ and $|\mathcal{N}_k| = 2$ for $k = 1$ and $k = K$.

V. Conclusion

In this paper, we developed a new consensus-based algorithm for distributed solution of the ridge regression problem with feature partitioning of the observation matrix. To this end, we recast the ridge regression problem into an equivalent constrained separable form, whose structure is suitable for distributed implementation through ADMM. In the proposed algorithm, D-Ridge, the agents exchange messages only within their neighborhoods. Simulation results showed that the sequences of local iterates generated by D-Ridge converge to the centralized solution faster than the diffusion-based algorithm does.

References

[1] N. K. D. Venkategowda and S. Werner, "Privacy-preserving distributed precoder design for decentralized estimation," in *Proc. IEEE Global Conference on Signal and Information Processing*, Nov. 2018.

[2] C. Li, S. Huang, Y. Liu, and Z. Zhang, "Distributed jointly sparse multitask learning over networks," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 151–164, Jan. 2018.

[3] J. Akhtar and K. Rajawat, "Distributed sequential estimation in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 86–100, Jan. 2018.

[4] S. P. Talebi, S. Werner, and D. P. Mandic, "Distributed adaptive filtering of α-stable signals," *IEEE Signal Processing Letters*, vol. 25, no. 10, pp. 1450–1454, Oct. 2018.

[5] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, *Splitting Methods in Communication, Imaging, Science, and Engineering*, ser. Scientific Computation, R. Glowinski, S. J. Osher, and W. Yin, Eds. Cham: Springer International Publishing, 2016.

[6] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Model-distributed solution of regularized least-squares problem over sensor networks," in *Proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2015, pp. 3821–3825.

[7] R. Arablouei, S. Werner, and K. Doğançay, "Diffusion-based distributed adaptive estimation utilizing gradient-descent total least-squares," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 5308–5312.

[8] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[9] I. Schizas, G. Mateos, and G. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365–2382, Jun. 2009.

[10] G. Mateos, I. Schizas, and G. Giannakis, "Distributed recursive least-squares for consensus-based in-network adaptive estimation," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4583–4588, Nov. 2009.

[11] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2320–2330, May 2011.
[12] ——, "Low-complexity distributed total least squares estimation in ad hoc sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4321–4333, Aug. 2012.
[13] R. Abdolee and B. Champagne, "Diffusion LMS strategies in sensor networks with noisy input data," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 3–14, Feb. 2016.
[14] L. Lu, H. Zhao, and B. Champagne, "Diffusion total least-squares algorithm with multi-node feedback," *Signal Processing*, Jul. 2018.
[15] R. Arablouei, S. Werner, Y.-F. Huang, and K. Doğançay, "Distributed least mean-square estimation with partial diffusion," *IEEE Transactions on Signal Processing*, vol. 62, no. 2, pp. 472–484, Jan. 2014.
[16] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Adaptive distributed estimation based on recursive least-squares and partial diffusion," *IEEE Transactions on Signal Processing*, vol. 62, no. 14, pp. 3510–3522, Jul. 2014.
[17] R. Arablouei, S. Werner, and K. Doğançay, "Partial-diffusion recursive least-squares estimation over adaptive networks," in *Proc. 2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Dec. 2013, pp. 89–92.
[18] R. Arablouei, K. Doğançay, and S. Werner, "Reduced-complexity distributed least-squares estimation over adaptive networks," in *Proc. 2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications*, Jun. 2013, pp. 150–154.
[19] N. Kashyap, S. Werner, Y.-F. Huang, and R. Arablouei, "Privacy preserving decentralized power system state estimation with phasor measurement units," in *Proc. 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop*, Jul. 2016, pp. 1–5.
[20] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1942–1956, Apr. 2012.
[21] ——, "D-admm: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, May 2013.
[22] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: Models, limits, and algorithms," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 386–398, Jan. 2011.
[23] J. Predd, S. Kulkarni, and H. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, Jul. 2006.
[24] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proc. 9th ACM International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 206–215.
[25] O. L. Mangasarian, E. W. Wild, and G. M. Fung, "Privacy-preserving classification of vertically partitioned data via random kernels," *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 3, Oct. 2008.
[26] A. H. Sayed, "Adaptive Networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.
[27] D. P. Bertsekas, *Parallel and distributed computation : numerical methods*. Englewood Cliffs, N.J: Prentice-Hall, 1989.
[28] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[29] T. Lin, S. Ma, and S. Zhang, "On the global linear convergence of the admm with multiblock variables," *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1478–1497, Jan. 2015.
[30] K. Eriksson, *Applied Mathematics: Body and Soul : Volume 1: Derivatives and Geometry in IR3*, 2004.

Distributed Learning over Networks with Non-Smooth Regularizers and Feature Partitioning

Cristiano Gratton*, Naveen K. D. Venkategowda†, Reza Arablouei‡, Stefan Werner*
* Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway
† Department of Science and Technology, Linköping University, Norrköping, Sweden
‡ CSIRO's Data61, Pullenvale QLD 4069, Australia

***Abstract*—We develop a new algorithm for distributed learning with non-smooth regularizers and feature partitioning. To this end, we transform the underlying optimization problem into a suitable dual form and solve it using the alternating direction method of multipliers. The proposed algorithm is fully-distributed and does not require the conjugate function of any non-smooth regularizer function, which may be unfeasible or computationally inefficient to acquire. Numerical experiments demonstrate the effectiveness of the proposed algorithm.**

I. Introduction

An important issue associated with distributed learning is how the data is distributed among the agents. Horizontal partitioning of data refers to when subsets of data samples with a common set of features are distributed over the network. Examples of learning with horizontal partitioning of data can be found in [1]–[4]. However, many regression or classification problems encountered in machine learning deal with heterogeneous data that do not contain common features. These problems lead to the so-called feature (column) partitioning of the data where subsets of features of all data samples are distributed over the network agents. Distributed learning problems with feature partitioning also arise in several signal processing applications, e.g., bioinformatics, multi-view learning, and dictionary learning, as mentioned in [5], [6].

There have been several attempts to solve learning problems with feature partitioning of data, e.g., in [5]–[19]. However, the algorithms in [8], [9] can only be used to solve the basis pursuit and lasso problems, respectively, while the work in [10] is based on assuming an appropriate coloring scheme of the network and cannot be extended to a general graph labeling. The algorithms developed in [6], [11], [12] are based on the diffusion strategy. In contrast, the approaches in [5], [13] are based on the consensus strategy. However, [5] is not fully distributed since the consensus constraints are imposed globally across the entire network rather than being applied locally within each agent's neighborhood. Although the algorithm in [13] is fully distributed, it assumes a specific structure for the objective function and is only suitable for ridge regression. The works of [14]–[17] consider distributed agent-specific estimation. However, the objective functions considered in these works are smooth. The authors of [18] propose a coordinate-descent-based algorithm with an inexact update to reduce communication costs for feature-partitioned distributed learning. In [19], an asynchronous stochastic gradient-descent algorithm was developed for distributed learning with feature partitioning of data. However, none of the above-mentioned algorithms consider distributed problems with general non-smooth regularization and arbitrary graphs.

In this paper, we develop a new fully-distributed algorithm for distributed learning with non-smooth regularizers and feature partitioning of data. We consider a general regularized learning problem whose cost function cannot be written as the sum of the local agent-specific cost functions, i.e., it is not separable. To achieve separability, we formulate the dual problem associated with the underlying convex optimization problem and exploit its favorable structure that, unlike the original problem, allows us to solve it by utilizing the alternating direction method of multipliers (ADMM). By utilizing the dual of the optimization problem associated with the ADMM primal variable update step, we devise a new strategy that does not require any conjugate function of the non-smooth regularizers, which may be infeasible or hard to obtain in some scenarios. The proposed algorithm is fully-distributed as every agent communicates only with its neighboring agents and no central coordinator is needed. Our simulation results show that the proposed algorithm converges in various scenarios.

Notations: The operators $(\cdot)^\mathsf{T}$ and $\mathrm{tr}(\cdot)$ denote transpose and trace of a matrix, respectively. $\|\cdot\|$ represents the Euclidean norm of its vector argument. $\mathbf{I}_n$ is an identity matrix of size n, $\mathbf{0}_n$ is an $n \times 1$ vector with all zeros, $\mathbf{0}_{n\times p} = \mathbf{0}_n\mathbf{0}_p^\mathsf{T}$, and $|\cdot|$ denotes the cardinality if its argument is a set. For a function f, f^* denotes the conjugate function of f.

II. System Model

We consider a network with $N \in \mathbb{N}$ agents and $E \in \mathbb{N}$ edges that is modeled as an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with the set of vertices $\mathcal{V} = \{1, \ldots, N\}$ corresponding to the agents and the set of edges $\mathcal{E}$ representing the bidirectional communication links between the pairs of agents. Agent $i \in \mathcal{V}$ communicates only with its neighbors specified by the set $\mathcal{V}_i$.

Due to feature partitioning, the data of each agent i resides in the matrix $\mathbf{A}_i \in \mathbb{R}^{M\times P_i}$ and the response vector $\mathbf{b} \in \mathbb{R}^{M\times 1}$ where M is the number of data samples and P_i the number of features in each sample at agent i. The feature vector at agent i that relates $\mathbf{A}_i$ and $\mathbf{b}$ is denoted by $\mathbf{x}_i \in \mathbb{R}^{P_i\times 1}$.

We consider a regularized learning problem of form

$$\min_{\{\mathbf{x}_i\}} \quad f\left(\sum_{i=1}^{N}\mathbf{A}_i\mathbf{x}_i - \mathbf{b}\right) + \sum_{i=1}^{N} r_i(\mathbf{x}_i) \tag{1}$$

where $f(\cdot)$ is the global cost function and $r_i(\cdot)$, $i = 1, \dots, N$, are the agent-specific regularizer functions. The learning problem (1) pertains to several applications in machine learning, e.g., regression over distributed features [5], clustering in graphs [20], smart grid control [21], dictionary learning [22], and network utility maximization [23]. In this work, we consider learning problems where functions $r_i(\cdot)$, $i = 1, \dots, N$, are convex, proper, and lower semi-continuous but not necessarily smooth and $f(\cdot) = \|\cdot\|^2$. In the next section, we solve (1) in a distributed manner, where each agent communicates only with its neighbors.

III. Distributed Algorithm for Learning with Feature Partitioning

First, we present the reformulation of the considered non-separable problem into a dual form that is separable and can be solved in a fully-distributed fashion via the ADMM. Then, we describe the new strategy that allows us to employ the ADMM without computing any conjugate function of the non-smooth regularizers explicitly.

A. Distributed ADMM for the Dual Problem

To develop a distributed solution, we introduce the auxiliary variables $\{\mathbf{z}_i\}_{i=1}^N$ and recast (1) as

$$\begin{aligned}\min_{\{\mathbf{x}_i,\mathbf{z}_i\}} \quad & f\left(\sum_{i=1}^{N}\mathbf{z}_i - \mathbf{b}\right) + \sum_{i=1}^{N} r_i(\mathbf{x}_i) \\ \text{s. t.} \quad & \mathbf{A}_i\mathbf{x}_i = \mathbf{z}_i, \quad i = 1, \dots, N.\end{aligned} \tag{2}$$

The objective function in (2) is not separable among the agents. Therefore, we consider the dual problem of (2). For this purpose, we associate the Lagrange multipliers $\{\boldsymbol{\mu}_i\}_{i=1}^N$ with the equality constraints in (2) and form the Lagrangian function $\mathcal{L}(\{\mathbf{x}_i\}, \{\mathbf{z}_i\}, \{\boldsymbol{\mu}_i\})$. The dual function for problem (2) is given by

$$\begin{aligned} d(\{\boldsymbol{\mu}_i\}) &= \inf_{\{\mathbf{x}_i,\mathbf{z}_i\}} \mathcal{L}(\{\mathbf{x}_i\}, \{\mathbf{z}_i\}, \{\boldsymbol{\mu}_i\}) \\ &= -\sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) + \inf_{\mathbf{z}_i}\Big\{ f(\sum_{i=1}^{N}\mathbf{z}_i - \mathbf{b}) - \sum_{i=1}^{N}\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{z}_i\Big\}\end{aligned} \tag{3}$$

where r_i^* is the conjugate function of r defined as

$$r_i^*(\mathbf{y}) = \sup_{\mathbf{x}} \ \mathbf{y}^T\mathbf{x} - r_i(\mathbf{x}).$$

Next, for the second infimum in (3), introducing

$$\mathbf{z} = \sum_{i=1}^{N}\mathbf{z}_i$$

and its corresponding dual variable $\boldsymbol{\lambda}$, and using the duality theory, an alternate form of the dual function (3) is obtained as

$$\tilde{d}(\{\boldsymbol{\mu}_i\}, \boldsymbol{\lambda}) = \begin{cases} -\tilde{f}^*(\boldsymbol{\lambda}) - \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i), & \boldsymbol{\lambda} = \boldsymbol{\mu}_i, \forall i \in \mathcal{V} \\ -\infty, & \text{otherwise} \end{cases} \tag{4}$$

where

$$\tilde{f}^*(\boldsymbol{\lambda}) = f^*(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{b}.$$

Eliminating the redundant variable $\boldsymbol{\lambda}$, the dual problem for (2) can be expressed as

$$\begin{aligned}\max_{\{\boldsymbol{\mu}_i\}} \quad & -\frac{1}{N}\sum_{i=1}^{N}\tilde{f}^*(\boldsymbol{\mu}_i) - \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ \text{s. t.} \quad & \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \cdots = \boldsymbol{\mu}_N.\end{aligned} \tag{5}$$

To solve (5) in a distributed fashion, we employ the ADMM [24]. First, we recast (5) as a constrained minimization problem by imposing consensus constraints across each agent's neighborhood $\mathcal{V}_i$ as follows

$$\begin{aligned}\min_{\{\boldsymbol{\mu}_i\},\{\mathbf{u}_i^j\}} \quad & \frac{1}{N}\sum_{i=1}^{N}\tilde{f}^*(\boldsymbol{\mu}_i) + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ \text{s. t.} \quad & \boldsymbol{\mu}_i = \mathbf{u}_i^j, \quad \boldsymbol{\mu}_j = \mathbf{u}_i^j, \ j \in \mathcal{V}_i, \ i = 1, \dots, N.\end{aligned} \tag{6}$$

To facilitate a fully-distributed solution, we decouple the constraints in (5) by introducing the auxiliary variables $\{\mathbf{u}_i^j\}_{j\in\mathcal{V}_i}$. Then, we generate the relevant augmented Lagrangian function by associating the Lagrange multipliers $\{\bar{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$, $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ with the consensus constraints. In [24], it is shown that, by setting

$$\mathbf{v}_i^{(k)} = 2\sum_{j\in\mathcal{V}_i}(\bar{\mathbf{v}}_i^j)^{(k)},$$

the Lagrange multipliers $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ and the auxiliary variables $\{\mathbf{u}_i^j\}_{j\in\mathcal{V}_i}$ are eliminated and the ADMM reduces to an iterative procedure with two steps at each iteration as

$$\begin{aligned}\boldsymbol{\mu}_i^{(k)} =& \arg\min_{\boldsymbol{\mu}_i}\Big\{\frac{1}{N}f^*(\boldsymbol{\mu}_i) + \frac{1}{N}\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b} + r_i^*(-\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i) \\ & + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{v}_i^{(k-1)} + \rho\sum_{j\in\mathcal{V}_i}\Big\|\boldsymbol{\mu}_i - \frac{\boldsymbol{\mu}_i^{(k-1)} + \boldsymbol{\mu}_j^{(k-1)}}{2}\Big\|^2\Big\},\end{aligned} \tag{7}$$

$$\mathbf{v}_i^{(k)} = \mathbf{v}_i^{(k-1)} + \rho\sum_{j\in\mathcal{V}_i}(\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}). \tag{8}$$

where $\rho > 0$ is the penalty parameter.

Since $r_i(\cdot)$, $i = 1, \dots, N$, are non-smooth, the minimization problem in (7) can be solved by employing appropriate subgradients or proximal operators [25], [26]. However, computing the conjugate function of the regularizers in (7) may be hard. To overcome this challenge, in the next subsection, we describe a new procedure that does not require the explicit calculation of any conjugate function.

B. ADMM with no Conjugate Function

In order to solve the problem in (7), we need to calculate the conjugate function of r_i^*. This can be difficult, especially for non-smooth functions. We exploit the Fenchel-Moreau theorem to eliminate the computation of conjugate function.

To that end, the problem in (7) can be restated as

$$\begin{aligned} \min_{\{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i\}} \quad & \frac{f^*(\boldsymbol{\mu}_i)+\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{c}_i^{(k-1)} + \bar{\rho}_i \left\|\boldsymbol{\mu}_i\right\|_2^2 \\ \text{s. t.} \quad & \mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i = \mathbf{0} \end{aligned} \tag{9}$$

where

$$\mathbf{c}_i^{(k-1)} = \mathbf{v}_i^{(k-1)} - \rho|\mathcal{V}_i|\boldsymbol{\mu}_i^{(k-1)} - \rho\sum_{j\in\mathcal{V}_i}\boldsymbol{\mu}_j^{(k-1)}$$

and $\bar{\rho}_i = \rho|\mathcal{V}_i|$. The Lagrangian function for (9) is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}_i,\boldsymbol{\nu}_i,\boldsymbol{\theta}_i) =& \frac{f^*(\boldsymbol{\mu}_i)+\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{c}_i^{(k-1)} \\ &+ \bar{\rho}_i\left\|\boldsymbol{\mu}_i\right\|_2^2 + \boldsymbol{\theta}_i^{\mathsf{T}}(\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i) \end{aligned} \tag{10}$$

where $\boldsymbol{\theta}_i$ is the Lagrange multiplier vector associated with the constraint in (9). Hence, the dual function for the objective in (9) can be expressed as

$$\begin{aligned} &\delta(\boldsymbol{\theta}_i) = \inf_{\{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i\}} \mathcal{L}(\boldsymbol{\mu}_i,\boldsymbol{\nu}_i,\boldsymbol{\theta}_i) \\ &= \inf_{\boldsymbol{\nu}_i}\{r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\theta}_i^{\mathsf{T}}\boldsymbol{\nu}_i\} \\ &+ \inf_{\boldsymbol{\mu}_i}\left\{\frac{f^*(\boldsymbol{\mu}_i)+\boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{b}}{N} + (\mathbf{c}_i^{(k-1)} + \mathbf{A}_i\boldsymbol{\theta}_i)^{\mathsf{T}}\boldsymbol{\mu}_i + \bar{\rho}_i\left\|\boldsymbol{\mu}_i\right\|^2\right\} \\ &= -r_i^{**}(-\boldsymbol{\theta}_i) \\ &+ \inf_{\boldsymbol{\mu}_i}\left\{\frac{f^*(\boldsymbol{\mu}_i)}{N} + \left(\mathbf{c}_i^{(k-1)} + \mathbf{A}_i\boldsymbol{\theta}_i + \frac{\mathbf{b}}{N}\right)^{\mathsf{T}}\boldsymbol{\mu}_i + \bar{\rho}_i\left\|\boldsymbol{\mu}_i\right\|^2\right\} \end{aligned} \tag{11}$$

where the last equality follows from the definition of conjugate function.

For $f(\cdot) = \left\|\cdot\right\|^2$, the conjugate function is given by $f^*(\boldsymbol{\mu}_i) = \left\|\boldsymbol{\mu}_i\right\|^2/4$. Thus, the optimal value of second infimum of the dual function in (11) is

$$\frac{-1}{4\rho|\mathcal{V}_i| + \frac{1}{N}}\left\|\mathbf{A}_i\boldsymbol{\theta}_i + \mathbf{c}_i^{(k-1)} + \frac{\mathbf{b}}{N}\right\|^2$$

and the infimum is attained at the optimal point

$$\boldsymbol{\mu}_i^o = \frac{-2}{4\rho|\mathcal{V}_i| + \frac{1}{N}}\left(\mathbf{A}_i\boldsymbol{\theta}_i^o + \mathbf{c}_i^{(k-1)} + \frac{\mathbf{b}}{N}\right) \tag{12}$$

where $\boldsymbol{\theta}_i^o = \arg\max_{\boldsymbol{\theta}_i}\delta(\boldsymbol{\theta}_i)$. Since $r_i(\cdot)$ is convex, proper, and lower semi-continuous, we have $r_i^{**} = r_i$ due to the Fenchel-Moreau theorem [27]. Therefore, the dual function is given by

$$\delta(\boldsymbol{\theta}_i) = -r_i(-\boldsymbol{\theta}_i) - \frac{1}{4\rho|\mathcal{V}_i| + \frac{1}{N}}\left\|\mathbf{A}_i\boldsymbol{\theta}_i + \mathbf{c}_i^{(k-1)} + \frac{\mathbf{b}}{N}\right\|^2. \tag{13}$$

Algorithm 1 Proposed algorithm for feature-partitioned distributed learning

```
At all agents $i \in \mathcal{V}$, initialize $\boldsymbol{\mu}_i^{(0)} = \mathbf{0}$, $\mathbf{v}_i^{(0)} = \mathbf{0}$, and locally run:
for $k = 1, 2, \ldots, K$ do
    Update $\boldsymbol{\theta}_i^{(k)}$ via (14).
    Update the dual variables $\boldsymbol{\mu}_i^{(k)}$ via (15).
    Share $\boldsymbol{\mu}_i^{(k)}$ with the neighbors in $\mathcal{V}_i$.
    Update the Lagrange multipliers $\mathbf{v}_i^{(k)}$ via (16).
    Update the auxiliary variables $\mathbf{c}_i^{(k)}$ via (17).
end for
```

Using (12) and (13), the ADMM steps in (7) and (8) can be equivalently expressed as

$$\begin{aligned} \boldsymbol{\theta}_i^{(k)} &= \arg\min_{\boldsymbol{\theta}_i}\Big\{r_i(-\boldsymbol{\theta}_i) \\ &\quad + \frac{1}{4\rho|\mathcal{V}_i| + \frac{1}{N}}\left\|\mathbf{A}_i\boldsymbol{\theta}_i + \mathbf{c}_i^{(k-1)} + \frac{\mathbf{b}}{N}\right\|^2\Big\} \quad (14) \\ \boldsymbol{\mu}_i^{(k)} &= \frac{-2}{4\rho|\mathcal{V}_i| + \frac{1}{N}}\left(\mathbf{A}_i\boldsymbol{\theta}_i^{(k)} + \mathbf{c}_i^{(k-1)} + \frac{\mathbf{b}}{N}\right) \quad (15) \\ \mathbf{v}_i^{(k)} &= \mathbf{v}_i^{(k-1)} + \rho\sum_{j\in\mathcal{V}_i}(\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}) \quad (16) \\ \mathbf{c}_i^{(k)} &= \mathbf{v}_i^{(k)} - \rho|\mathcal{V}_i|\boldsymbol{\mu}_i^{(k)} - \rho\sum_{j\in\mathcal{V}_i}\boldsymbol{\mu}_j^{(k)}. \quad (17) \end{aligned}$$

The proposed algorithm is summarized in Algorithm 1. Note that the minimization problem in (14) can be solved using standard optimization techniques, or alternatively, subgradient-based algorithms [28]. Regardless of the technique used to solve (14), the proposed algorithm converges according to [28, Section 3.6.2]. Convergence of Algorithm 1 follows from [1, Proposition 2] and [29]. Moreover, due to the strong duality theorem, we have $\boldsymbol{\theta}_i^o = \mathbf{x}_i^o$, i.e., the optimal dual variable $\boldsymbol{\theta}_i^o$ at agent i is the optimal estimate $\mathbf{x}_i^o$ [30].

IV. SIMULATIONS

To illustrate the performance of the proposed algorithm, we consider the elastic-net regression problem [31] and benchmark the proposed algorithm against a broadcast-based algorithm for learning with distributed features [5]. The only existing work considering non-smooth distributed learning with feature partitioning over general graphs is [5]. Therefore, we compared our algorithm only with this algorithm to provide a comparison that is as fair as possible. In a centralized setting, the optimal solution $\mathbf{x}^c$ is obtained as

$$\mathbf{x}^c = \arg\min_{\mathbf{x}} \left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2 + \eta_1\left\|\mathbf{x}\right\|_1 + \eta_2\left\|\mathbf{x}\right\|_2^2 \tag{18}$$

where

$$\begin{aligned} \mathbf{A} &= [\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N] \\ \mathbf{x} &= \left[\mathbf{x}_1^{\mathsf{T}}, \mathrm{x}_2^{\mathsf{T}}, \ldots, \mathrm{x}_N^{\mathsf{T}}\right]^{\mathsf{T}}, \end{aligned}$$

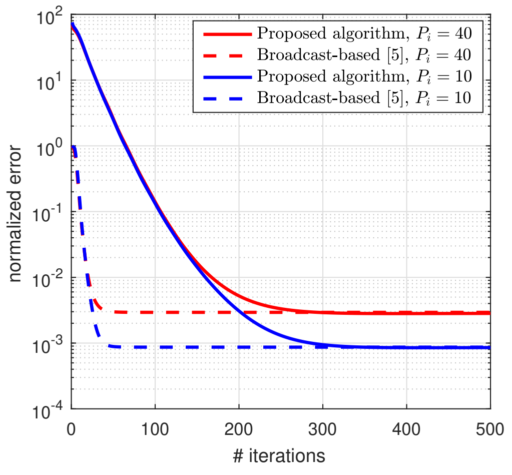


Fig. 1. Normalized error of the proposed algorithm and the broadcast-based algorithm of [5] with $N = 20$ agents and different values of P_i.

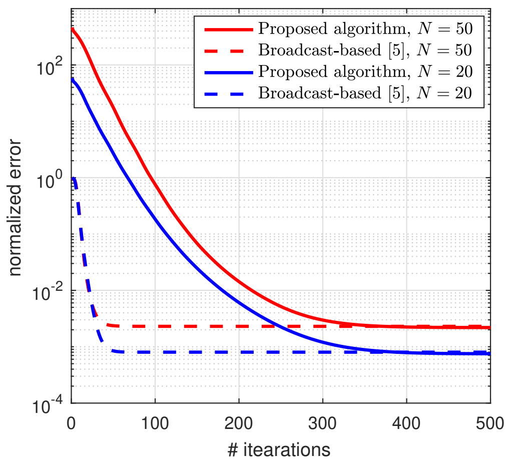


Fig. 2. Normalized error of the proposed algorithm and the broadcast-based algorithm of [5] with $P_i = 10$ and different values of N.

and $\eta_1 \in \mathbb{R}^+$ and $\eta_2 \in \mathbb{R}^+$ are the regularization parameters. In the distributed setting, we solve the problem (1) with

$$f(\mathbf{x}_i) = \left\| \sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_i - \mathbf{b} \right\|^2,$$
$$r_i(\mathbf{x}_i) = \eta_1 \left\| \mathbf{x}_i \right\|_1 + \eta_2 \left\| \mathbf{x}_i \right\|^2.$$

We test the proposed algorithm on a multi-agent network with a random topology, where each agent links to three other agents on average. For each agent $i \in \mathcal{V}$, we create a $2P_i \times P_i$ local observation matrix $\mathbf{A}_i$ whose entries are independent identically distributed Gaussian random variables with zero mean and unit variance. The response vector $\mathbf{b}$ is obtained as

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\psi}$$

where $\boldsymbol{\omega} \in \mathbb{R}^P$, $P = \sum_{i=1}^{N} P_i$, and $\boldsymbol{\psi} \in \mathbb{R}^M$ are drawn from the distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$, respectively. The regularization parameters are set to $\eta_1 = \eta_2 = 1$ and penalty parameter to $\rho = 1$. The performance of the proposed algorithm is evaluated using the normalized error $\epsilon(k)$ between the centralized solution $\mathbf{x}^c$ as per (18) and the solution from Algorithm 1 at iteration k denoted by

$$\mathbf{x}^d(k) = \left[(\mathbf{x}_1^{(k)})^\mathsf{T}, \ldots, (\mathbf{x}_N^{(k)})^\mathsf{T} \right]^\mathsf{T}.$$

The normalized error is defined as

$$\epsilon(k) = \frac{\left\| \mathbf{x}^d(k) - \mathbf{x}^c \right\|^2}{\left\| \mathbf{x}^c \right\|^2}.$$

The centralized solution $\mathbf{x}^c$ is computed using the optimization toolbox CVX [32]. Results are obtained by averaging over 100 independent trials.

Fig. 1 shows that, for $N = 20$ agents, the proposed algorithm converges when the number of parameters at the ith agent is $P_i = 10$ and $P_i = 40$, $\forall i \in \mathcal{V}$. Fig. 2 shows that the proposed algorithm converges when $P_i = 10$ and the network consists of 20 or 50 agents. The faster convergence of the broadcast-based algorithm of [5] is due to its centralized processing.

V. Conclusion

We developed a fully-distributed algorithm for learning with non-smooth regularization functions under distributed features. We reformulated the underlying problem into an equivalent dual form and used the ADMM to solve it in a distributed fashion without using any conjugate function. To the best of our knowledge, the proposed algorithm is the first of its kind that solves the feature-distributed learning problems with non-smooth regularizer functions over arbitrary graphs while not relying on any conjugate function. We verified the convergence of the proposed algorithm at all agents via simulation results.

References

[1] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5262–5276, Oct. 2010.

[2] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Consensus-based distributed total least-squares estimation using parametric semidefinite programming," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2019, pp. 5227–5231.

[3] ——, "Distributed learning with non-smooth objective functions," in *Proc. 28th European Signal Processing Conference*, Jan. 2021, pp. 2180–2184.

[4] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2320–2330, May 2011.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2010.

[6] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, Feb. 2019.

[7] H. Zheng, S. R. Kulkarni, and H. V. Poor, "Attribute-distributed learning: Models, limits, and algorithms," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 386–398, Jan. 2011.

[8] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "Distributed basis pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1942–1956, Apr. 2012.

[9] C. Manss, D. Shutin, and G. Leus, "Distributed splitting-over-features sparse bayesian learning with alternating direction method of multipliers," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 3654–3658.

[10] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, May 2013.

[11] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Model-distributed solution of regularized least-squares problem over sensor networks," in *Proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2015, pp. 3821–3825.

[12] S. A. Alghunaim, M. Yan, and A. H. Sayed, "A multi-agent primal-dual strategy for composite optimization over distributed features," in *Proc. 28th European Signal Processing Conference*, Jan. 2021, pp. 2095–2099.

[13] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, "Distributed ridge regression with feature partitioning," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018.

[14] J. Szurley, A. Bertrand, and M. Moonen, "Topology-independent distributed adaptive node-specific signal estimation in wireless sensor networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 130–144, 2017.

[15] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS for clustered multitask networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 5487–5491.

[16] N. Bogdanović, J. Plata-Chaves, and K. Berberidis, "Distributed incremental-based LMS for node-specific adaptive parameter estimation," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5382–5397, 2014.

[17] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3448–3460, 2015.

[18] B. Zhang, J. Geng, W. Xu, and L. Lai, "Communication efficient distributed learning with feature partitioned data," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2018, pp. 1–6.

[19] Y. Hu, D. Niu, J. Yang, and S. Zhou, "FDML: A collaborative machine learning framework for distributed features," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2232–2240.

[20] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, p. 387–396.

[21] T. Chang, A. Nedić, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1524–1538, 2014.

[22] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1001–1016, 2015.

[23] D. P. Palomar and Mung Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[24] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, *Splitting Methods in Communication, Imaging, Science, and Engineering*, ser. Scientific Computation, R. Glowinski, S. J. Osher, and W. Yin, Eds. Cham: Springer International Publishing, 2016.

[25] D. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[26] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, p. 127–239, Jan. 2014.

[27] J. M. Borwein and A. S. Lewis, *Convex analysis and nonlinear optimization: theory and examples*. Springer, 2006.

[28] Z. Han, M. Hong, and D. Wang, *Signal processing and networking for big data applications*. Cambridge University Press, 2017.

[29] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, pp. 475–494, Jan. 2001.

[30] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[31] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

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

Decentralized Optimization with Distributed Features and Non-Smooth Objective Functions

Cristiano Gratton, Naveen K. D. Venkategowda, *Member, IEEE,* Reza Arablouei, and Stefan Werner, *Senior Member, IEEE*

***Abstract*—We develop a new consensus-based distributed algorithm for solving learning problems with feature partitioning and non-smooth convex objective functions. Such learning problems are not separable, i.e., the associated objective functions cannot be directly written as a summation of agent-specific objective functions. To overcome this challenge, we redefine the underlying optimization problem as a dual convex problem whose structure is suitable for distributed optimization using the alternating direction method of multipliers (ADMM). Next, we propose a new method to solve the minimization problem associated with the ADMM update step that does not rely on any conjugate function. Calculating the relevant conjugate functions may be hard or even unfeasible, especially when the objective function is non-smooth. To obviate computing any conjugate function, we solve the optimization problem associated with each ADMM iteration in the dual domain utilizing the block coordinate descent algorithm. Unlike the existing related algorithms, the proposed algorithm is fully distributed and does away with the conjugate of the objective function. We prove theoretically that the proposed algorithm attains the optimal centralized solution. We also confirm its network-wide convergence via simulations.**

***Index Terms*—Alternating direction method of multipliers, distributed optimization, learning with feature partitioning.**

I. Introduction

PERFORMING data analytic tasks at a central processing unit in a distributed network can be infeasible due to the associated computing/communication costs or privacy issues. In addition, collecting all the data in a central hub creates a single point of failure. Therefore, it is necessary to develop algorithms that facilitate in-network processing and model learning using data collected by nodes/agents that are dispersed over a network [2]–[7]. Distributed optimization problems pertain to several applications in statistics [2], [3], signal processing [4], [5], machine learning, and control [6], [7].

An essential aspect of distributed learning is how the data is distributed among the agents that determines what each agent intends to or is able to learn. Horizontal partitioning of data refers to the case when the data samples containing all features are distributed over the network. That is, all the agents estimate the same common model. Examples of learning with horizontal partitioning of data can be found in [2], [4], [8], [9]. On the other hand, when subsets of the features of all data samples are distributed over the network agents, we have feature partitioning of the data and every agent estimates a local model that is a part of the network-wide model. In the machine learning terminology, features are the descriptors or measurable characteristics of the data samples. In regression analysis, they may be called predictors or independent explanatory variables.

Several machine learning problems deal with heterogeneous distributed data that common features cannot describe. For example, in multi-agent systems, each agent may acquire data to learn a local model and refrains from sharing the data with other agents due to resource constraints or privacy concerns. However, the aggregate data can be exploited to enhance accuracy or augment inference due to the correlation of the data across agents. In the Internet of things, a device may only be interested in estimating its own local model parameters. However, multiple devices distributed over an ad hoc network may be able to collectively process the network-wide data and enhance the estimation/inference quality. Distributed learning problems with feature partitioning arise in several signal processing applications, e.g., bioinformatics, multi-view learning, and dictionary learning, as mentioned in [10], [11]. Data with feature partitioning can also be referred to as attribute-distributed data [12], vertically-partitioned data [13], [14], data with column-partitioning [15], or heterogeneous data [12].

A. Related Works

Learning problems with feature partitioning of data have been considered in [1], [3], [10], [11], [15]–[28]. The algorithms proposed in [15], [16] solve the basis pursuit and lasso problems, respectively. The work of [17] assumes an appropriate coloring scheme of the network and cannot be extended to a general graph labeling.

The algorithms proposed in [10], [18]–[21] are not fully distributed since their consensus constraints involve the entire network instead of each agent's local neighborhood. Furthermore, the algorithms proposed in [18], [19] only solve the ridge regression problem, while the works of [20], [21] assume the cost function to be convex and smooth with Lipschitz-continuous gradient. Both algorithms proposed in [20], [21] can only be used for minimization problems with ℓ_2-norm regularization (ridge penalty) and rely on the computation of the conjugate of the cost function.

This work was partly supported by the Research Council of Noway. A conference precursor of this work appears in the *Proceedings of the European Signal Processing Conference*, Dublin, Ireland, August 2021 [1].

C. Gratton and S. Werner are with the Department of Electronic Systems, Norwegian University of Science and Technology, Trondheim, Norway (email:cristiano.gratton@ntnu.no; stefan.werner@ntnu.no).

N. K. D. Venkategowda is with the Department of Science and Technology, Linköping University, Norrköping, Sweden (email: naveen.venkategowda@liu.se)

R. Arablouei is with the Commonwealth Scientific and Industrial Research Organisation, Pullenvale QLD 4069, Australia (email:reza.arablouei@csiro.au).

TABLE I
COMPARATIVE SUMMARY

	fully distributed	non-smooth cost function	non-smooth regularizer	no conjugate function
[1]	✓		✓	
[3]	✓			
[10]		✓	✓	
[11]	✓			✓
[15]	✓	✓		
[16]			✓	✓
[17]	✓		✓	
[18]				✓
[20]				
[19]				✓
[21]				
[22]		✓	✓	✓
[23]	✓	✓		
[24]	✓			✓
[25]			✓	
[26]	✓		✓	
[27]			✓	✓
[28]				✓
[29]	✓			✓
[30]	✓			✓
[31]	✓			✓
[32]	✓			✓
proposed	✓	✓	✓	✓

The algorithms in [11], [22]–[24] are based on the diffusion strategy, which is suitable when stochastic gradients are available. Furthermore, the work in [11] assumes that the cost function is convex and smooth with Lipschitz-continuous gradient. The algorithm developed in [22] relies on the calculation of the relevant conjugate functions. In addition, it assumes that the cost function is convex and smooth, and the regularization functions are strongly convex. The work of [23] also assumes that the regularizer functions are smooth and strongly convex. Moreover, it relies on the computation of conjugate functions similar to the algorithms proposed in [25], [26]. The diffusion-based algorithm proposed in [24] only solves the ridge regression problem. The consensus-based algorithm of [3] is also designed for ridge regression with feature partitioning. It outperforms the algorithm proposed in [24] in terms of convergence speed.

The algorithm proposed in [1] is designed for an ℓ_2-norm-square cost function and hence cannot be extended to general objective functions. The works of [29]–[32] consider distributed agent-specific parameter estimation problem. However, the objective functions considered in these works are smooth. The authors of [27] propose a distributed coordinate-descent algorithm to reduce the communication cost in distributed learning with feature partitioning. However, the cost function in [27] is assumed to be strongly convex and smooth. The work of [28] considers an asynchronous stochastic gradient-descent algorithm for learning with distributed features. However, the objective function in [28] is assumed to be smooth.

None of the above-mentioned existing algorithms for distributed learning with feature partitioning is designed for optimizing generic non-smooth objective functions over arbitrary graphs without using or computing any conjugate function.

B. Contributions

In this paper, we develop a new fully-distributed algorithm for solving learning problems when the data is distributed among agents in feature partitions and computing the conjugate of the possibly non-smooth cost or regularizer functions is challenging or unfeasible. We consider a general regularized non-smooth learning problem whose cost function cannot be written as the sum of local agent-specific cost functions, i.e., it is not separable as in (2) ahead.

To tackle the problem, we articulate the associated dual optimization problem and utilize the alternating direction method of multipliers (ADMM) to solve it as, unlike the original problem, its structure is suitable for distributed treatment via the ADMM. We then consider the dual of the optimization problem associated with the ADMM update step and solve it via the block coordinate-descent (BCD) algorithm. In that manner, we devise an approach that enables us to avoid the explicit computation of any conjugate function, which may be hard or infeasible for some objective functions. The proposed algorithm is fully distributed, i.e., it only relies on single-hop communications among neighboring agents and does not need any central coordinator or processing hub. We demonstrate that the proposed algorithm approaches the optimal centralized solution at all agents. Our experiments show that the proposed algorithm converges to the optimal solution in various scenarios and is competitive with the relevant existing algorithms even when dealing with problems that, unlike its contenders, it is not tailored for.

In Table I, we provide a comparative summary of the proposed algorithm with respect to the most relevant existing ones in terms of the key features of being fully distributed, ability to handle non-smooth cost or regularization functions, and non-reliance on any conjugate function.

C. Paper Organization

The rest of the paper is organized as follows. In Section II, we describe the system model and formulate the distributed learning problem with feature partitioning when both the cost and regularizer functions are convex but not necessarily smooth. In Section III, we describe our proposed algorithm for solving the considered regularized learning problem in a distributed fashion without computing any conjugate function. Subsequently, we prove the convergence of the proposed algorithm by confirming that both its inner and outer loops converge in Section IV. We provide some simulation results in Section V and draw conclusions in Section VI.

D. Mathematical Notations

The set of natural and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. The set of positive real numbers is denoted by $\mathbb{R}_+$. Scalars, column vectors, and matrices are respectively denoted by lowercase, bold lowercase, and bold uppercase letters. The operators $(\cdot)^\mathsf{T}$, det$(\cdot)$, and tr$(\cdot)$ denote transpose, determinant, and trace of a matrix, respectively. The symbol $\|\cdot\|$ represents the Euclidean norm of its vector argument and $\otimes$ stands for the Kronecker product. $\mathbf{I}_n$ is an identity

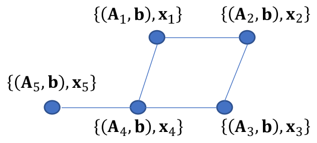

$\{(\mathbf{A}_1,\mathbf{b}),\mathbf{x}_1\}$
$\{(\mathbf{A}_2,\mathbf{b}),\mathbf{x}_2\}$
$\{(\mathbf{A}_5,\mathbf{b}),\mathbf{x}_5\}$
$\{(\mathbf{A}_4,\mathbf{b}),\mathbf{x}_4\}$
$\{(\mathbf{A}_3,\mathbf{b}),\mathbf{x}_3\}$

Fig. 1. Distributed features over a network with five agents.

matrix of size n, $\mathbf{0}_n$ is an $n \times 1$ vector with all zeros entries, $\mathbf{0}_{n\times p} = \mathbf{0}_n\mathbf{0}_p^{\mathsf{T}}$, and $|\cdot|$ denotes the cardinality operator if its argument is a set. The statistical expectation and covariance operators are represented by $\mathbb{E}[\cdot]$ and $\operatorname{cov}[\cdot]$, respectively. For any positive semidefinite matrix $\mathbf{X}$, $\lambda_{\min}(\mathbf{X})$ and $\lambda_{\max}(\mathbf{X})$ denote the nonzero smallest and largest eigenvalues of $\mathbf{X}$, respectively. For a vector $\mathbf{x} \in \mathbb{R}^n$ and a positive semi-definite matrix $\mathbf{A}$, $\|\mathbf{x}\|_{\mathbf{A}}^2$ denotes the quadratic form $\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x}$. The conjugate function of any function f is denoted by f^*.

II. System Model

We model a network with $N \in \mathbb{N}$ agents and $E \in \mathbb{N}$ edges as an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with the set of vertices $\mathcal{V} = \{1,\dots,N\}$ corresponding to the agents and the set of edges $\mathcal{E}$ standing for the bidirectional communication links between the pairs of agents. Agent $i \in \mathcal{V}$ communicates only with its neighbors specified by the set $\mathcal{V}_i$.

Let us denote the network-wide data as an observation matrix $\mathbf{A} \in \mathbb{R}^{M\times P}$ and a response vector $\mathbf{b} \in \mathbb{R}^{M\times 1}$ where M is the number of data samples and P is the total number of features across the network. As we consider the feature partitioning of the data, we denote the observation matrix of the ith agent by $\mathbf{A}_i \in \mathbb{R}^{M\times P_i}$ and its local model vector by $\mathbf{x}_i \in \mathbb{R}^{P_i\times 1}$ where P_i is the number of features specific to agent i. Accordingly, we have $P = \sum_{i=1}^N P_i$ and $\mathbf{A}$ consists of N submatrices $\mathbf{A}_i$ as

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N].$$

The network-wide model vector $\mathbf{x} \in \mathbb{R}^{P\times 1}$ that relates $\mathbf{A}$ and $\mathbf{b}$ is also a stack of N subvectors $\mathbf{x}_i$ as

$$\mathbf{x} = \left[\mathbf{x}_1^{\mathsf{T}}, \mathbf{x}_2^{\mathsf{T}}, \dots, \mathbf{x}_N^{\mathsf{T}}\right]^{\mathsf{T}}.$$

We give an example for feature partitioning of data in Fig. 1 where features are distributed over a network of five agents.

We consider a regularized learning problem consisting in minimizing a global cost function $f(\cdot)$ that is a function of the error $\mathbf{A}\mathbf{x} - \mathbf{b}$ and is added by a regularization function $r(\cdot)$. In the centralized approach, the optimal solution is given by

$$\mathbf{x}^o = \arg\min_{\mathbf{x}} \Big\{ f\left(\mathbf{A}\mathbf{x} - \mathbf{b}\right) + r(\mathbf{x}) \Big\}. \tag{1}$$

Considering feature partitioning of the data, $\mathbf{A}\mathbf{x}$ can be written as

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^N \mathbf{A}_i\mathbf{x}_i$$

and assuming that the regularizer function $r(\cdot)$ can be written as a sum of agent-specific regularizer functions as

$$r(\mathbf{x}) = \sum_{i=1}^N r_i(\mathbf{x}_i),$$

the regularized learning problem (1) is of the following form

$$\min_{\{\mathbf{x}_i\}} \quad f\left(\textstyle\sum_{i=1}^N \mathbf{A}_i\mathbf{x}_i - \mathbf{b}\right) + \sum_{i=1}^N r_i(\mathbf{x}_i). \tag{2}$$

The learning problem (2) pertains to several applications in machine learning, e.g., regression over distributed features [10], clustering in graphs [33], smart grid control [34], dictionary learning [22], and network utility maximization [35]. Similar to most existing works, e.g., [10], [11], [36], we consider learning problems where functions $f(\cdot)$ and $r_i(\cdot)$, $i = 1,\dots,N$, are convex, proper, and lower semi-continuous. However, in this work, the objective functions are not necessarily smooth or their conjugate functions known. Therefore, we propose a novel algorithm that solves (2) in a fully distributed fashion wherein each agent communicates only with its neighbors without requiring the computation of any conjugate function. In the next section, we describe our proposed algorithm.

III. Algorithm

We first present the reformulation of the considered non-separable problem into a dual form that can be solved in a fully-distributed fashion via the ADMM. Subsequently, we describe a new approach to perform the ADMM primal update step without explicitly computing any conjugate function of the cost or regularizer functions.

A. Distributed ADMM for the Dual Problem

To develop a distributed solution, we introduce the auxiliary variables $\{\mathbf{z}_i\}_{i=1}^N$ and recast (2) as

$$\begin{aligned} \min_{\{\mathbf{x}_i,\mathbf{z}_i\}} \quad & f\left(\textstyle\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}\right) + \sum_{i=1}^N r_i(\mathbf{x}_i) \\ \text{s. t.} \quad & \mathbf{A}_i\mathbf{x}_i = \mathbf{z}_i, \quad i = 1,\dots,N. \end{aligned} \tag{3}$$

The cost function $f(\cdot)$ in (3) is not separable among the agents. We consider the dual problem of (3) and exploit its separability property, which is lacking in the primal domain, to solve it by employing the ADMM. For this purpose, we associate the Lagrange multipliers $\{\boldsymbol{\mu}_i\}_{i=1}^N$ with the equality constraints in (3) and state the related Lagrangian function as

$$\begin{aligned} &\mathcal{L}(\{\mathbf{x}_i\},\{\mathbf{z}_i\},\{\boldsymbol{\mu}_i\}) \\ &= f\left(\textstyle\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}\right) + \sum_{i=1}^N r_i(\mathbf{x}_i) + \sum_{i=1}^N \boldsymbol{\mu}_i^{\mathsf{T}}(\mathbf{A}_i\mathbf{x}_i - \mathbf{z}_i) \\ &= \sum_{i=1}^N \left(r_i(\mathbf{x}_i) + (\mathbf{A}_i^{\mathsf{T}}\boldsymbol{\mu}_i)^{\mathsf{T}}\mathbf{x}_i\right) \\ &\quad + f\left(\sum_{i=1}^N \mathbf{z}_i - \mathbf{b}\right) - \sum_{i=1}^N \boldsymbol{\mu}_i^{\mathsf{T}}\mathbf{z}_i. \end{aligned} \tag{4}$$

The dual function for problem (3) can be computed as

$$\begin{aligned} d(\{\boldsymbol{\mu}_i\}) &= \inf_{\{\mathbf{x}_i,\mathbf{z}_i\}} \mathcal{L}(\{\mathbf{x}_i\},\{\mathbf{z}_i\},\{\boldsymbol{\mu}_i\}) \\ &= -\sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i) + \inf_{\mathbf{z}_i} f(\sum_{i=1}^{N}\mathbf{z}_i - \mathbf{b}) - \sum_{i=1}^{N}\boldsymbol{\mu}_i^\mathsf{T}\mathbf{z}_i \end{aligned} \quad (5)$$

where r_i^* is the conjugate function of r defined as

$$r_i^*(\mathbf{y}) = \sup_{\mathbf{x}} \ \mathbf{y}^T\mathbf{x} - r_i(\mathbf{x}).$$

Introducing auxiliary variable $\mathbf{z}$ that is defined as

$$\mathbf{z} = \sum_{i=1}^{N}\mathbf{z}_i$$

and using the duality theory, an alternate form of the dual function (5) is given by

$$\tilde{d}(\{\boldsymbol{\mu}_i\},\boldsymbol{\lambda}) = -f^*(\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\mathsf{T}\mathbf{b} - \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i) \quad (6)$$

when $\boldsymbol{\lambda} = \boldsymbol{\mu}_i \ \forall i \in \mathcal{V}$ with $\boldsymbol{\lambda}$ being the dual variable corresponding to $\mathbf{z} = \sum_{i=1}^{N}\mathbf{z}_i$. Otherwise, we have $\tilde{d}(\{\boldsymbol{\mu}_i\},\boldsymbol{\lambda}) = -\infty$.

By eliminating $\boldsymbol{\lambda}$, the dual problem for (3) can be expressed as

$$\begin{aligned} \min_{\{\boldsymbol{\mu}_i\}} \quad & \frac{1}{N}\sum_{i=1}^{N}\left(f^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{b}\right) + \sum_{i=1}^{N} r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i) \\ \text{s. t.} \quad & \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \cdots = \boldsymbol{\mu}_N. \end{aligned} \quad (7)$$

To facilitate a fully-distributed solution, we decouple the constraints in (7) as

$$\boldsymbol{\mu}_i = \mathbf{u}_i^j, \quad \boldsymbol{\mu}_j = \mathbf{u}_i^j, \ j \in \mathcal{V}_i, \ i = 1,\ldots,N \quad (8)$$

where $\{\mathbf{u}_i^j\}_{i\in\mathcal{V},j\in\mathcal{V}_i}$ are auxiliary variables that will eventually be eliminated. We generate a new augmented Lagrangian function by associating the new Lagrange multipliers $\{\bar{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ and $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ with the consensus constraints in (8). By using the Karush-Kuhn-Tucker conditions of optimality for (8) and setting

$$\mathbf{v}_i^{(k)} = 2\sum_{j\in\mathcal{V}_i}(\bar{\mathbf{v}}_i^j)^{(k)},$$

it can be shown that the Lagrange multipliers $\{\tilde{\mathbf{v}}_i^j\}_{j\in\mathcal{V}_i}$ and the auxiliary variables $\{\mathbf{u}_i^j\}_{j\in\mathcal{V}_i}$ are eliminated [5]. Hence, the ADMM to solve (7) reduces to the following iterative updates at the ith agent

$$\begin{aligned} \boldsymbol{\mu}_i^{(k)} = \arg\min_{\boldsymbol{\mu}_i} \Big\{ & \frac{1}{N}f^*(\boldsymbol{\mu}_i) + \frac{1}{N}\boldsymbol{\mu}_i^\mathsf{T}\mathbf{b} + r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i) \\ & + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{v}_i^{(k-1)} + \rho\sum_{j\in\mathcal{V}_i}\Big\|\boldsymbol{\mu}_i - \frac{\boldsymbol{\mu}_i^{(k-1)} + \boldsymbol{\mu}_j^{(k-1)}}{2}\Big\|^2\Big\} \end{aligned} \quad (9)$$

$$\mathbf{v}_i^{(k)} = \mathbf{v}_i^{(k-1)} + \rho\sum_{j\in\mathcal{V}_i}(\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}) \quad (10)$$

where $\rho > 0$ is the penalty parameter and k is the iteration index.

The objective function in (9) may be non-smooth as the global cost function $f(\cdot)$ or the agent-specific regularizer functions $r_i(\cdot)$, $i = 1,\ldots,N$, and consequently their conjugate functions may be non-smooth. Thus, the minimization problem in (9) can be solved by employing suitable subgradient methods or proximal operators [37], [38]. However, computing the conjugate functions of the cost or the regularizer functions in (9) may be hard or even unfeasible. To overcome this challenge, in the next subsection, we describe a new approach that does not require the explicit calculation of any conjugate function.

B. ADMM without Conjugate Function

We rewrite the minimization problem in the ADMM primal update (9) as

$$\begin{aligned} \boldsymbol{\mu}_i^{(k)} = \arg\min_{\{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i,\boldsymbol{\alpha}_i\}} & \Big\{\frac{f^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) \\ & + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{c}_i^{(k-1)} + \bar{\rho}_i\|\boldsymbol{\alpha}_i\|^2\Big\} \\ \text{s.t.} \quad & \mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i = \mathbf{0} \\ & \boldsymbol{\mu}_i = \boldsymbol{\alpha}_i \end{aligned} \quad (11)$$

where $\bar{\rho}_i = \rho|\mathcal{V}_i|$ and

$$\mathbf{c}_i^{(k-1)} = \mathbf{v}_i^{(k-1)} - \rho|\mathcal{V}_i|\boldsymbol{\mu}_i^{(k-1)} - \rho\sum_{j\in\mathcal{V}_i}\boldsymbol{\mu}_j^{(k-1)}. \quad (12)$$

The Lagrangian function related to (11) is stated as

$$\begin{aligned} \mathcal{L}_k(\boldsymbol{\mu}_i,\boldsymbol{\nu}_i,\boldsymbol{\alpha}_i,\boldsymbol{\theta}_i^{(k)},\boldsymbol{\beta}_i^{(k)}) = & \frac{f^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{b}}{N} + r_i^*(\boldsymbol{\nu}_i) + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{c}_i^{(k-1)} \\ & + \bar{\rho}_i\|\boldsymbol{\alpha}_i\|^2 + (\boldsymbol{\theta}_i^{(k)})^\mathsf{T}(\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i + \boldsymbol{\nu}_i) \\ & + (\boldsymbol{\beta}_i^{(k)})^\mathsf{T}(\boldsymbol{\mu}_i - \boldsymbol{\alpha}_i) \end{aligned} \quad (13)$$

where $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$ are the Lagrange multipliers associated with the first and the second constraints in (11), respectively, at iteration k.

Motivated by the close connection between a function and its double conjugate (conjugate of conjugate), we express the dual for the objective in (11) as

$$\begin{aligned} \delta_k(\boldsymbol{\theta}_i^{(k)},\boldsymbol{\beta}_i^{(k)}) &= \inf_{\{\boldsymbol{\mu}_i,\boldsymbol{\nu}_i,\boldsymbol{\alpha}_i\}} \mathcal{L}_k(\boldsymbol{\mu}_i,\boldsymbol{\nu}_i,\boldsymbol{\alpha}_i,\boldsymbol{\theta}_i^{(k)},\boldsymbol{\beta}_i^{(k)}) \\ &= \inf_{\boldsymbol{\nu}_i}\{r_i^*(\boldsymbol{\nu}_i) + (\boldsymbol{\theta}_i^{(k)})^\mathsf{T}\boldsymbol{\nu}_i\} + \inf_{\boldsymbol{\alpha}_i}\{\bar{\rho}_i\|\boldsymbol{\alpha}_i\|^2 - (\boldsymbol{\beta}_i^{(k)})^\mathsf{T}\boldsymbol{\alpha}_i\} \\ &\quad + \inf_{\boldsymbol{\mu}_i}\Big\{\frac{f^*(\boldsymbol{\mu}_i)}{N} + \Big(\mathbf{c}_i^{(k-1)} + \mathbf{A}_i\boldsymbol{\theta}_i^{(k)} + \frac{\mathbf{b}}{N} + \boldsymbol{\beta}_i^{(k)}\Big)^\mathsf{T}\boldsymbol{\mu}_i\Big\}. \end{aligned} \quad (14)$$

By employing the definition of conjugate function, the first infimum in (14) is equal to $-r_i^{**}(-\boldsymbol{\theta}_i^{(k)})$. The second infimum in (14) can be easily obtained by noting that the function

$$l_k(\boldsymbol{\alpha}_i) \coloneqq \bar{\rho}_i\|\boldsymbol{\alpha}_i\|^2 - (\boldsymbol{\beta}_i^{(k)})^\mathsf{T}\boldsymbol{\alpha}_i$$

is quadratic in $\boldsymbol{\alpha}_i$. Hence, this infimum can be calculated by computing the gradient of $l_k(\cdot)$ and equating it to zero, i.e.,

$$\bar{\rho}_i\|\boldsymbol{\alpha}_i\|^2 - (\boldsymbol{\beta}_i^{(k)})^\mathsf{T}\boldsymbol{\alpha}_i = 0.$$

Solving this equation for $\boldsymbol{\alpha}_i$ gives

$$\boldsymbol{\alpha}_i^o = \frac{\boldsymbol{\beta}_i^{(k)}}{2\bar{\rho}_i}. \tag{15}$$

This implies that the second infimum in (14) is attained at the optimal value $\boldsymbol{\alpha}_i^o$, which in turn means that it is equal to

$$l_k(\boldsymbol{\alpha}_i^o) = -\frac{\left\|\boldsymbol{\beta}_i^{(k)}\right\|^2}{4\bar{\rho}_i}.$$

In view of the definition and properties of the conjugate function [39], the third infimum in (14) is given by

$$-Nf^{**}\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(k)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)}\right).$$

Therefore, we have

$$\begin{aligned}\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}) = &- r_i^{**}(-\boldsymbol{\theta}_i^{(k)}) - \frac{\left\|\boldsymbol{\beta}_i^{(k)}\right\|^2}{4\bar{\rho}_i} \\ &- Nf^{**}\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(k)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)}\right).\end{aligned} \tag{16}$$

Since $f(\cdot)$ and $r_i(\cdot)$ are convex, proper, and lower semi-continuous, we know $f^{**} = f$ and $r_i^{**} = r_i$ due to the Fenchel Moreau Theorem [40]. Therefore, we have

$$\begin{aligned}\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)}) = &- r_i(-\boldsymbol{\theta}_i^{(k)}) - \frac{\left\|\boldsymbol{\beta}_i^{(k)}\right\|^2}{4\bar{\rho}_i} \\ &- Nf\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(k)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)}\right).\end{aligned} \tag{17}$$

To find the optimal $(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$, we need to maximize $\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$ or, equivalently, to minimize $-\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$. Since this is a function of two variables $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$, we employ the block coordinate descent algorithm (BCD) to minimize $-\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$ and find the optimal values for $(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$. The BCD steps are obtained by alternatively minimizing $-\delta_k(\boldsymbol{\theta}_i^{(k)}, \boldsymbol{\beta}_i^{(k)})$ with respect to $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$ as follows

$$\begin{aligned}\boldsymbol{\theta}_i^{(k,t)} = \arg\min_{\boldsymbol{\theta}_i^{(k)}} \Big\{ & r_i(-\boldsymbol{\theta}_i^{(k)}) \\ &+ Nf\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(k,t)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k,t-1)}\right)\Big\}\end{aligned} \tag{18}$$

$$\begin{aligned}\boldsymbol{\beta}_i^{(k,t)} = \arg\min_{\boldsymbol{\beta}_i^{(k)}} \Big\{ & \frac{1}{4\bar{\rho}_i}\left\|\boldsymbol{\beta}_i^{(k)}\right\|^2 \\ &+ Nf\left(-\mathbf{c}_i^{(k-1)} - \mathbf{A}_i\boldsymbol{\theta}_i^{(t)} - \frac{\mathbf{b}}{N} - \boldsymbol{\beta}_i^{(k)}\right)\Big\}\end{aligned} \tag{19}$$

where t is the BCD iteration index. If we assume that the BCD algorithm converges after T iterations. The optimal values of $\boldsymbol{\theta}_i^{(k)}$ and $\boldsymbol{\beta}_i^{(k)}$ can be denoted by $\boldsymbol{\theta}_i^{(k,T)}$ and $\boldsymbol{\beta}_i^{(k,T)}$, respectively.

To update the Lagrange multipliers $\boldsymbol{\mu}_i^{(k)}$ we employ the complementary slackness conditions, i.e.,

$$\boldsymbol{\beta}_i^{(k,T)}(\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\alpha}_i^o) = \mathbf{0} \quad \forall i \in \mathcal{V}.$$

Algorithm 1 The proposed algorithm for feature-partitioned distributed learning with unknown conjugate functions

```
At all agents $i \in \mathcal{V}$, initialize $\boldsymbol{\mu}_i^{(0)} = \mathbf{0}$, $\mathbf{v}_i^{(0)} = \mathbf{0}$, and locally run:
for $k = 1, 2, \ldots, K$ do
    Run BCD loop
    for $t = 1, 2, \ldots, T$ do
        Update $\boldsymbol{\theta}_i^{(k,t)}$ via (18).
        Update $\boldsymbol{\beta}_i^{(k,t)}$ via (19).
    end for
    Update the dual variables $\boldsymbol{\mu}_i^{(k)} = \boldsymbol{\beta}_i^{(k,T)}/(2\bar{\rho}_i)$.
    Share $\boldsymbol{\mu}_i^{(k)}$ with the neighbors in $\mathcal{V}_i$.
    Update the Lagrange multipliers $\mathbf{v}_i^{(k)}$ via (22).
    Update the auxiliary variables $\mathbf{c}_i^{(k)}$ via (23).
end for
```

Since $\boldsymbol{\beta}_i^{(k,T)} \neq \mathbf{0}$, $\forall i \in \mathcal{V}$, we have

$$\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\alpha}_i^o = \mathbf{0} \quad \forall i \in \mathcal{V}.$$

Using (15), we can update $\boldsymbol{\mu}_i^{(k)}$ as

$$\boldsymbol{\mu}_i^{(k)} = \frac{\boldsymbol{\beta}_i^{(k,T)}}{2\bar{\rho}_i}. \tag{20}$$

Collating the expressions in (20), (12), (10), the ADMM steps in (9) and (10) can be equivalently expressed as

$$\boldsymbol{\mu}_i^{(k)} = \frac{\boldsymbol{\beta}_i^{(k,T)}}{2\bar{\rho}_i} \tag{21}$$

$$\mathbf{v}_i^{(k)} = \mathbf{v}_i^{(k-1)} + \rho\sum_{j\in\mathcal{V}_i}(\boldsymbol{\mu}_i^{(k)} - \boldsymbol{\mu}_j^{(k)}) \tag{22}$$

$$\mathbf{c}_i^{(k-1)} = \mathbf{v}_i^{(k-1)} - \rho|\mathcal{V}_i|\boldsymbol{\mu}_i^{(k-1)} - \rho\sum_{j\in\mathcal{V}_i}\boldsymbol{\mu}_j^{(k-1)} \tag{23}$$

where k is the ADMM iteration index. We summarize the proposed algorithm in Algorithm 1.

Assuming that the ADMM outer loop converges after K iterations, we denote the optimal dual variable $\boldsymbol{\theta}_i^{(K,T)}$ by $\boldsymbol{\theta}_i^o$. The estimate $\boldsymbol{\theta}_i^o$ at agent i is indeed the optimal solution to the original problem (2), i.e., $\mathbf{x}_i^o$, as per the following theorem.

Theorem 1. *For all agents $i \in \mathcal{V}$, the optimal dual variable $\boldsymbol{\theta}_i^o$ at agent i is equal to the optimal estimate $\mathbf{x}_i^o$, i.e., $\boldsymbol{\theta}_i^o = \mathbf{x}_i^o$.*

Proof. Since the optimization problem in (3) has a convex objective and is feasible, the Slater's condition is satisfied. Therefore, due to the Slater's theorem, strong duality holds and $\boldsymbol{\theta}_i^o = \mathbf{x}_i^o$, $\forall i \in \mathcal{V}$ [39]. □

IV. Convergence Analysis

The convergence of the proposed algorithm can be proven by corroborating that both the inner-loop BCD and outer-loop ADMM iterations converge. First, the convergence of the inner loop can be verified from results in [2] since all the assumptions required for the convergence are satisfied, i.e., the function $\delta(\cdot)$ is convex and the feasible sets $\mathbb{R}^M$ and $\mathbb{R}^{P_i}$, $\forall i \in \mathcal{V}$, are all convex. Assuming that the optimal solution $\boldsymbol{\beta}_i^o$

of the inner-loop BCD algorithm is attained for each $i \in \mathcal{V}$, the dual variable $\boldsymbol{\mu}_i^{(k)}$ in the outer loop is updated accordingly.

Next, we prove that the estimates produced by the fully-distributed ADMM outer loop, i.e., (9) and (10), approach the optimal centralized solution at all agents. To present the convergence result, we rewrite the constraints in (7) as follows

$$\boldsymbol{\mu}_i = \bar{\mathbf{u}}_i^j, \quad \boldsymbol{\mu}_j = \breve{\mathbf{u}}_i^j, \quad \bar{\mathbf{u}}_i^j = \breve{\mathbf{u}}_i^j, \quad j \in \mathcal{V}_i,\ i = 1, \ldots, N. \tag{24}$$

Note that the constraints $\mathbf{u} \in \mathcal{C}_{\mathbf{u}} := \{\mathbf{u} : \bar{\mathbf{u}}_i^j = \breve{\mathbf{u}}_i^j,\ i \in \mathcal{V},\ j \in \mathcal{N}_i\}$ are not dualized and are introduced only to present the convergence result. Let us define the following vectors

$$\begin{aligned}\boldsymbol{\mu} =& [\boldsymbol{\mu}_1^\mathsf{T}, \ldots, \boldsymbol{\mu}_N^\mathsf{T}]^\mathsf{T}\\ \mathbf{u} =& [(\mathbf{u}_1^{a_1(1)})^\mathsf{T}, \ldots, (\mathbf{u}_1^{a_N(|\mathcal{V}_N|)})^\mathsf{T},\\ & \ldots, (\mathbf{u}_N^{a_N(1)})^\mathsf{T}, \ldots, (\mathbf{u}_N^{a_N(|\mathcal{V}_N|)})]^\mathsf{T}\end{aligned}$$

where $a_i(j)$ is the index of the jth neighbor of agent i.

The problem (8) with the constraints in (24) can be written as

$$\begin{aligned}\min_{\boldsymbol{\mu},\mathbf{u}} \quad & G_1(\boldsymbol{\mu}) + G_2(\mathbf{u})\\ \text{s.t.} \quad & \boldsymbol{\mu} \in \mathcal{C}_1,\ \mathbf{u} \in \mathcal{C}_2,\ \mathbf{C}\boldsymbol{\mu} = \mathbf{u}\end{aligned} \tag{25}$$

where $\mathbf{C} = [\mathbf{C}_1^\mathsf{T}, \mathbf{C}_2^\mathsf{T}]^\mathsf{T}$, $G_2(\mathbf{u}) = 0$, $\mathcal{C}_1 := \mathbb{R}^M$, $\mathcal{C}_2 := \mathcal{C}_{\mathbf{u}}$,

$$\mathbf{C}_1 = \begin{bmatrix}\mathbf{C}_{11}\\ \vdots \\ \mathbf{C}_{1N}\end{bmatrix}, \quad \mathbf{C}_{1i} := (\mathbf{1}_{|\mathcal{N}_i|}\mathbf{e}_i^\mathsf{T}) \otimes \mathbf{I}_M,\ i \in \mathcal{V}$$

$$\mathbf{C}_2 = \begin{bmatrix}\mathbf{C}_{21}\\ \vdots \\ \mathbf{C}_{2N}\end{bmatrix}, \quad \mathbf{C}_{2i} := \begin{bmatrix}\mathbf{e}_{i_i(1)}^\mathsf{T}\\ \vdots \\ \mathbf{e}_{i_i(|\mathcal{N}_i|)}^\mathsf{T}\end{bmatrix} \otimes \mathbf{I}_M,\ i \in \mathcal{V}$$

$$G_1(\boldsymbol{\mu}) = \frac{1}{N}\sum_{i=1}^N \left(f^*(\boldsymbol{\mu}_i) + \boldsymbol{\mu}_i^\mathsf{T}\mathbf{b}\right) + \sum_{i=1}^N r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}_i),$$

and $\mathbf{e}_i$ is the ith vector of the canonical basis of $\mathbb{R}^M$. The convergence result relies on the following lemma.

Lemma 1. *If $\mathcal{G}$ is a connected graph, then the local optimal solution $\boldsymbol{\mu}_i^o$ at agent i is equal to the optimal centralized solution of* (8)*, i.e., $\boldsymbol{\mu}_i^o = \boldsymbol{\mu}^o$, $\forall i \in \mathcal{V}$, where*

$$\boldsymbol{\mu}^o = \arg\min_{\boldsymbol{\mu}} \Big\{ f^*(\boldsymbol{\mu}) + \boldsymbol{\mu}^\mathsf{T}\mathbf{b} + \sum_{i=1}^N r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}) \Big\}.$$

Proof. Let i and i' be arbitrary agents in $\mathcal{G}$ and $p(i,i') : i, i_1, i_2, \ldots, i_n, i'$ an arbitrary path on $\mathcal{G}$ that connects i and i'. Since the adjacent agents in $p(i,i')$ are neighbors, we have

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i_1} = \boldsymbol{\mu}_{i_2} = \ldots = \boldsymbol{\mu}_{i_n} = \boldsymbol{\mu}_{i'},$$

which imply

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i'}.$$

Since $\mathcal{G}$ is connected and the path is arbitrary, the local constraints $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i'}$ can be removed and replaced by the common constraint $\boldsymbol{\mu}_i = \boldsymbol{\mu}$. Hence, $\boldsymbol{\mu}_i^o = \boldsymbol{\mu}^o$ $\forall i \in \mathcal{V}$ where

$$\boldsymbol{\mu}^o = \arg\min_{\boldsymbol{\mu}} \Big\{ f^*(\boldsymbol{\mu}) + \boldsymbol{\mu}^\mathsf{T}\mathbf{b} + \sum_{i=1}^N r_i^*(-\mathbf{A}_i^\mathsf{T}\boldsymbol{\mu}) \Big\}.$$

□

We can now prove the convergence of the proposed algorithm as per the following theorem.

Theorem 2. *If $\mathcal{G}$ is a connected graph, then the proposed algorithm converges to the optimal centralized solution, i.e.,*

$$\lim_{k\to\infty} \boldsymbol{\mu}_i^{(k)} = \boldsymbol{\mu}^o, \forall i \in \mathcal{V}. \tag{26}$$

Proof. Thanks to Lemma 1, we only need to prove that

$$\lim_{k\to\infty} \boldsymbol{\mu}_i^{(k)} = \boldsymbol{\mu}_i^o.$$

For this purpose, we observe that (25) is in the same form as [37, eq. 4.77, p. 255]. Furthermore, the following assumptions are satisfied:

- $G_1(\cdot)$ and $G_2(\cdot)$ are convex functions;
- $\mathcal{C}_1$ and $\mathcal{C}_2$ are nonempty polyhedral sets;
- $\mathbf{C}$ is full column rank, hence, $\mathbf{C}^\mathsf{T}\mathbf{C}$ is invertible.

Therefore, due to [37, Proposition 4.2, p. 256], we have

$$\lim_{k\to\infty} \boldsymbol{\mu}_i^{(k)} = \boldsymbol{\mu}^o, \forall i \in \mathcal{V}.$$

□

V. Simulations

In this section, we present some simulation results to evaluate the performance of the proposed algorithm. We first assess the proposed algorithm considering a distributed elastic-net regression problem with different numbers of local features, samples, and agents as well as different network topologies. Subsequently, we benchmark the proposed algorithm against the most relevant existing algorithms considering distributed ridge and lasso regression problems.

A. Distributed Elastic-Net Regression

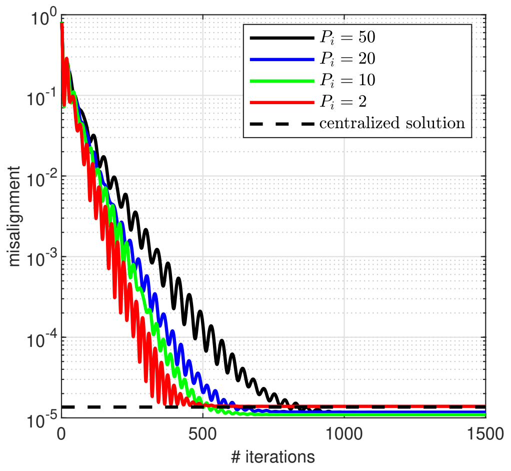
To evaluate the performance of the proposed algorithm in different scenarios, we consider the elastic-net regression problem. The calculation of the conjugate function for the objective function corresponding to this problem is practically infeasible. In the distributed setting, we solve the elastic-net regression problem by considering

$$\begin{aligned}f(\mathbf{x}_i) &= \left\|\sum_{i=1}^N \mathbf{A}_i\mathbf{x}_i - \mathbf{b}\right\|^2\\ r_i(\mathbf{x}_i) &= \eta_1 \|\mathbf{x}_i\|_1 + \eta_2 \|\mathbf{x}_i\|^2\end{aligned} \tag{27}$$

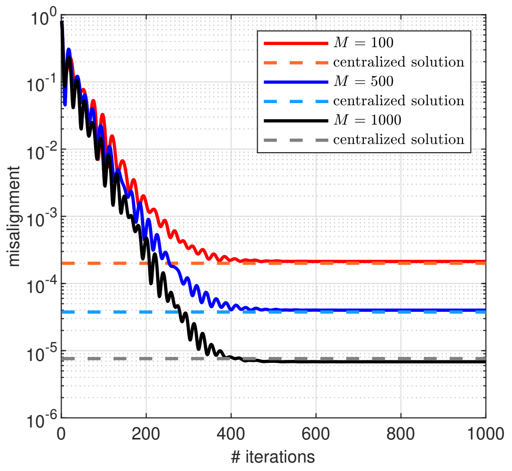
where $\eta_1 \in \mathbb{R}^+$ and $\eta_2 \in \mathbb{R}^+$ are the regularization parameters. We calculate the response vector $\mathbf{b}$ as

$$\mathbf{b} = \mathbf{A}\boldsymbol{\omega} + \boldsymbol{\psi} \tag{28}$$

where $\boldsymbol{\omega} \in \mathbb{R}^P$ and $\boldsymbol{\psi} \in \mathbb{R}^M$ are independently drawn from the multivariate normal distributions $\mathcal{N}(\mathbf{0}, \mathbf{I}_P)$ and $\mathcal{N}(\mathbf{0}, 0.1\mathbf{I}_M)$, respectively. We set the regularization parameters to $\eta_1 = 1$, $\eta_2 = 1$ and the penalty parameter to $\rho = 2$. We use two iterations in the inner-loop BCD algorithm. We obtain the results by averaging over 100 independent trials while considering a multi-agent network with a random topology

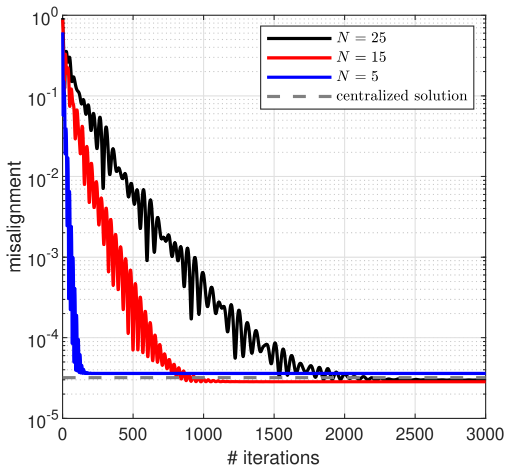


(a) different values of P_i

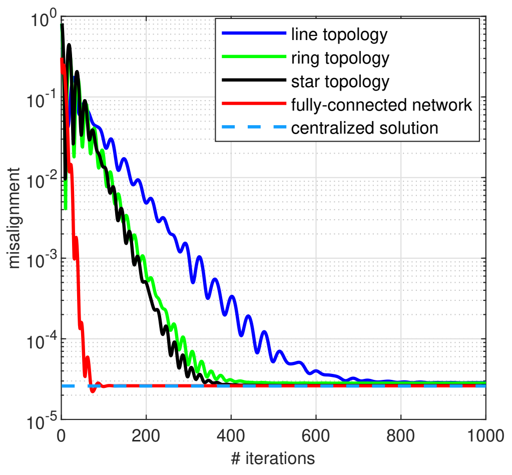


(b) different values of M

Fig. 2. The misalignment of the proposed algorithm solving the distributed elastic-net regression problem with different values of P_i and M.



(a) different values of N



(b) different topologies

Fig. 3. The misalignment of the proposed algorithm solving the distributed elastic-net regression problem with different values of N and different topologies.

where each agent links to three other agents on average. We evaluate the performance of the proposed algorithm using the misalignment metric that is defined as

$$\frac{\left\|\mathbf{x}^d(k)-\boldsymbol{\omega}\right\|^2}{\left\|\boldsymbol{\omega}\right\|^2}$$

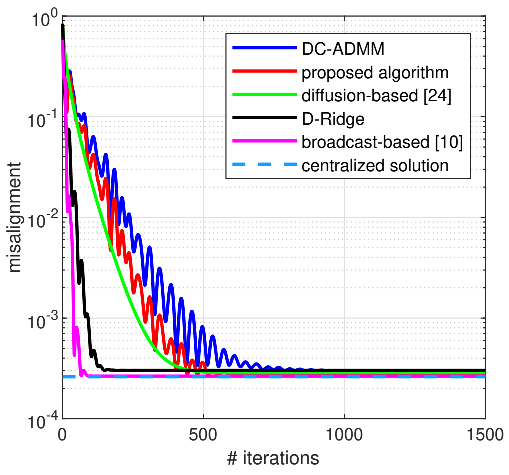
where

$$\mathbf{x}^d(k)=\left[\mathbf{x}_1^{(k)\mathsf{T}},\ldots,\mathbf{x}_N^{(k)\mathsf{T}}\right]^{\mathsf{T}}$$

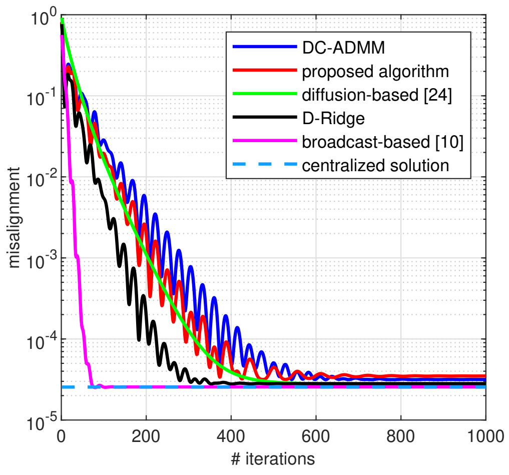
and $\mathbf{x}_i^{(k)}$ $\forall i \in \mathcal{V}$ denotes the local estimate at agent i.

In Fig. 2(a), we plot the misalignment of the proposed algorithm versus its outer-loop iteration index for different values of P_i, i.e., $P_i = 2$, $P_i = 10$, $P_i = 20$, and $P_i = 50$ while $M = 800$, $M = 1000$, $M = 1100$, and $M = 1500$, respectively. Fig. 2(a) shows that the proposed algorithm converges faster as the number of local features P_i decreases. In Fig. 2(b), we set $P_i = 2$ and use the same topology as in Fig.2(a) but consider different values of M. Fig. 2(b) shows that the proposed algorithm achieves higher accuracy as the number of samples M increases. Note that we include the misalignment of the centralized optimal solution in all figures.

In Fig. 3(a), we consider different values of N while $P_i = 2$, $M = 500$, and the network topology is arbitrary but with an average node degree of three. Fig. 3(a) shows that the proposed algorithm converges faster as the number of agents N decreases. In Fig. 3(b), we evaluate the proposed algorithm

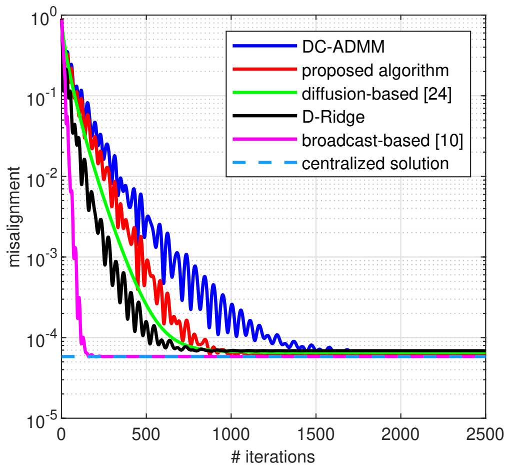


(a) Ridge regression with $N = 10$, $M = 50$, and $P_i = 2$.

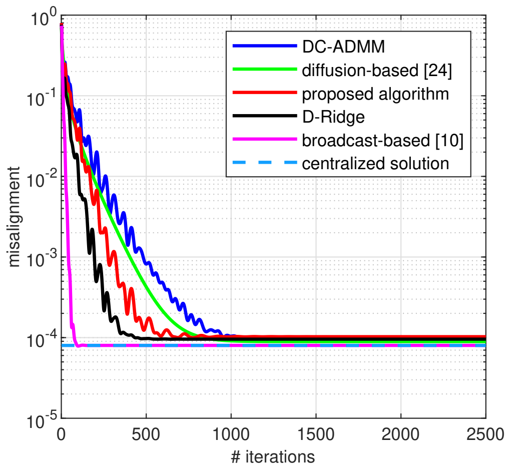


(b) Ridge regression with $N = 10$, $M = 200$, and $P_i = 2$.

Fig. 4. The misalignment of the proposed algorithm and other considered algorithms for the ridge regression problems in different scenarios.



(a) Ridge regression with $N = 20$, $M = 200$, and $P_i = 2$



(b) Ridge regression with $N = 10$, $M = 200$, and $P_i = 10$

Fig. 5. The misalignment of the proposed algorithm and other considered algorithms for the ridge regression problems in different scenarios.

by setting $N = 10$, $P_i = 2$, $M = 500$ and considering four different common simple topologies, i.e.,

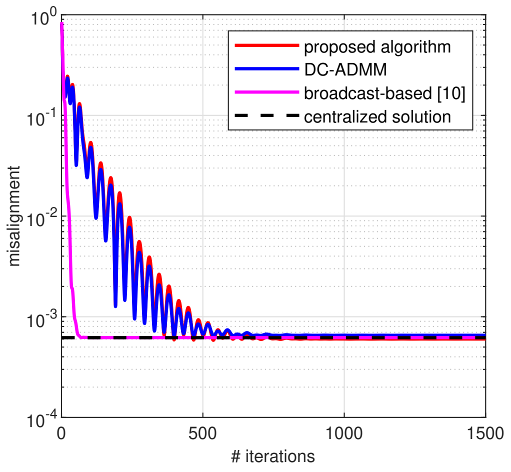
- line: the agents are connected one after the other, hence, $|\mathcal{N}_i| = 2$ for $1 < i < N$ and $|\mathcal{N}_i| = 1$ for $i = 1$ and $i = N$
- ring: $|\mathcal{N}_i| = 2$ for each $i \in \mathcal{V}$
- star: $|\mathcal{N}_i| = N - 1$ for $i = 1$ and $|\mathcal{N}_i| = 1$ for $i = 2, \ldots, N$
- fully-connected: each agent in the network is connected to all the other agents.

In Fig. 3(b), we observe that the proposed algorithm converges faster as the average number of links per agent increases, i.e., the average connectivity of the network increases.

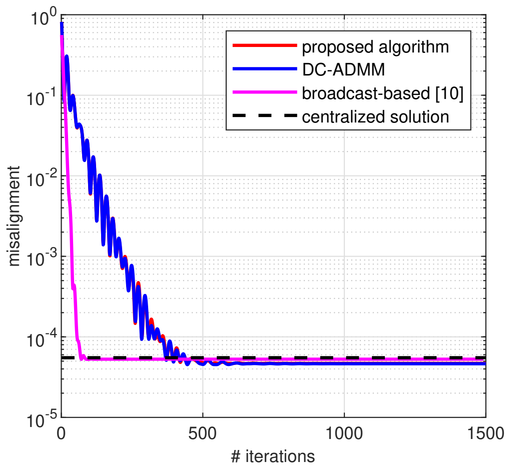
B. *Distributed Ridge Regression*

Considering a distributed ridge regression problem, in Figs. 4 and 5, we benchmark the proposed algorithm against some existing baseline algorithms, namely, the broadcast-based algorithm for learning with distributed features proposed in [10], the dual consensus ADMM (DC-ADMM) algorithm of [26], the consensus-based algorithm for ridge regression (D-Ridge) introduced in [3], and the diffusion-based algorithm of [24]. The algorithms proposed in [3], [24] are only for solving the ridge regression problem. Here, we solve the problem (2) with the objective function (27) and set the ith agent's regularizer to

$$r_i(\mathbf{x}_i) = \eta \|\mathbf{x}_i\|^2$$

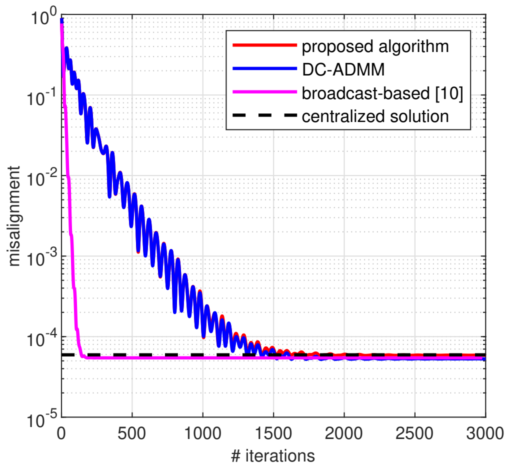


(a) Lasso regression with $N = 10$, $M = 50$, and $P_i = 2$

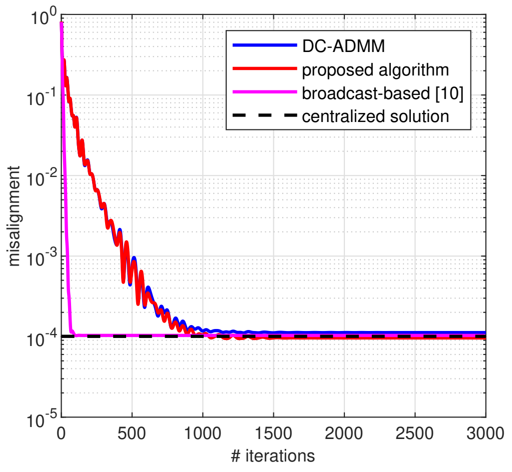


(b) Lasso regression with $N = 10$, $M = 200$, and $P_i = 2$

Fig. 6. The misalignment of the proposed algorithm and other considered algorithms for the lasso regression problems in different scenarios.



(a) Lasso regression with $N = 20$, $M = 200$, and $P_i = 2$



(b) Lasso regression with $N = 10$, $M = 200$, and $P_i = 10$

Fig. 7. The misalignment of the proposed algorithm and other considered algorithms for the lasso regression problems in different scenarios.

where $\eta \in \mathbb{R}^+$ is the regularization parameter.

We calculate the response vector $\mathbf{b}$ as in (28). As per [3], [24], we set the regularization parameter to $\eta = 0.001$. We also set the number of inner-loop BCD iterations of the proposed algorithm to 2 and obtain the results by averaging over 100 independent trials. In Fig. 4(a), we set $N = 10$, $M = 50$, and $P_i = 2$. In Fig. 4(b), the parameter setting is the same as Fig. 4(a) except for the number of samples M being larger, i.e., $M = 200$. In Fig. 5(a), we keep $M = 200$ and set the number of agents to $N = 20$. In Fig. 5(b), we set N and M to 10 and 200, respectively, while $P_i = 10$ $\forall i \in \mathcal{V}$.

We observe in Figs. 4 and 5 that the proposed algorithm outperforms the DC-ADMM algorithm. It also perform competitively in comparison with the algorithms of [3], [24], which are specifically tailored to the ridge regression problem. The superior performance of the broadcast-based algorithm of [10] is due to its centralized processing. We include it here only as a reference.

C. Distributed Lasso Regression

In Figs. 6 and 7, we compare the performance of the proposed algorithm with that of the broadcast-based algorithm for learning with distributed features proposed in [10] and the DC-ADMM algorithm of [26] considering a distributed lasso problem. Hence, we solve the problem (2) with the objective function (27) and set the ith agent's regularizer to

$$r_i(\mathbf{x}_i) = \eta \left\|\mathbf{x}_i\right\|_1$$

where $\eta \in \mathbb{R}^+$ is the regularization parameter.

We calculate the response vector $\mathbf{b}$ as in (28). As per [3], [24], we set the regularization parameter to $\eta = 0.001$. We also set the number of inner-loop BCD iterations of the proposed algorithm to 2 and obtain the results by averaging over 100 independent trials. In Fig. 6(a), we set $N = 10$, $M = 50$, and $P_i = 2$. In Fig. 6(b), the parameter setting is the same as Fig. 6(a) except for the number of samples M being larger, i.e., $M = 200$. In Fig. 7(a), we keep $M = 200$ and set the number of agents to $N = 20$. In Fig. 7(b), we set N and M to 10 and 200, respectively, while $P_i = 10\ \forall i \in \mathcal{V}$.

We observe in Figs. 6 and 7 that the proposed algorithm performs very similar to the DC-ADMM algorithm as the learning curves of the two algorithms almost overlap. Again, the superior performance of the broadcast-based algorithm of [10] is due to its centralized processing.

D. Discussion

The main advantage of the proposed algorithm is in its ability to solve generic feature-partitioned distributed optimization problems without resorting to any conjugate function even when the objective function is non-smooth. This is unique to our proposed algorithm and, to the best of our knowledge, there is no existing algorithm with the same utility. That is why we do not compare the proposed algorithm with any other existing algorithm in Section V-A where the problem at hand is feature-distributed elastic-net regression. The existing algorithms for feature-partitioned distributed optimization such as DC-ADMM require the conjugate function of the objective or regularization function. In the case of elastic-net regression, calculating the conjugate function is impracticable.

The simulation results in Sections V-B and V-C are to provide a comparative study of the performance of the proposed algorithm with respect to the other most relevant existing algorithms. As evident by the results, the proposed algorithm's performance in solving the distributed ridge and lasso regression problems is on par with those of its state-of-the-art competitors, even those that have specifically been design to solve these problems.

As seen in the figures, in all simulations, the network-wide average estimate of the proposed algorithm converges to the corresponding optimal centralized solution. Although not shown here for conciseness, we have observed that the estimates at all agents also converge to the optimal solution in all the experiments corroborating our theoretical findings in Section IV.

In all simulations, we utilize only two BCD iterations with no extra inter-agent communication overhead. Therefore, the computational complexity and communication requirements of the proposed algorithm are of the same order as those of the related existing algorithms such as DC-ADMM. Indeed, we did not observe any significant difference in the per-iteration run time of the proposed and DC-ADMM algorithms.

VI. Conclusion

We proposed a distributed algorithm for learning with non-smooth objective functions under distributed features. We reformulated the considered non-separable problem into a dual form that is separable and solved it via the ADMM. Subsequently, we devised an approach based on articulating the dual of the dual problem to overcome the challenge of computing the involved conjugate functions, which may be hard or even infeasible with some objective functions. We employed the BCD algorithm to solve the dual of the dual problem. Therefore, unlike most existing algorithms for solving learning problems with feature partitioning, the proposed algorithm does not require the explicit calculation of any conjugate of the objective function. We verified the convergence of the proposed algorithm to the optimal solution through both theoretical analysis and numerical simulations.

References

[1] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Distributed learning over networks with non-smooth regularizers and feature partitioning,” in *Proc. European Speech and Signal Processing Conference*, Aug. 2021.

[2] Z. Han, M. Hong, and D. Wang, *Signal processing and networking for big data applications*. Cambridge University Press, 2017.

[3] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Distributed ridge regression with feature partitioning,” in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018.

[4] ——, “Consensus-based distributed total least-squares estimation using parametric semidefinite programming,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2019, pp. 5227–5231.

[5] G. B. Giannakis, Q. Ling, G. Mateos, and I. D. Schizas, *Splitting Methods in Communication, Imaging, Science, and Engineering*, ser. Scientific Computation, R. Glowinski, S. J. Osher, and W. Yin, Eds. Cham: Springer International Publishing, 2016.

[6] D. Hajinezhad, M. Hong, and A. Garcia, “ZONE: Zeroth-order non-convex multiagent optimization over networks,” *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 3995–4010, Oct. 2019.

[7] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[8] C. Gratton, N. K. D. Venkategowda, R. Arablouei, and S. Werner, “Distributed learning with non-smooth objective functions,” in *Proc. 28th European Signal Processing Conference*, Jan. 2021, pp. 2180–2184.

[9] A. Bertrand and M. Moonen, “Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks,” *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2320–2330, May 2011.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jan. 2010.

[11] B. Ying, K. Yuan, and A. H. Sayed, “Supervised learning under distributed features,” *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, Feb. 2019.

[12] H. Zheng, S. R. Kulkarni, and H. V. Poor, “Attribute-distributed learning: Models, limits, and algorithms,” *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 386–398, Jan. 2011.

[13] O. L. Mangasarian, E. W. Wild, and G. M. Fung, “Privacy-preserving classification of vertically partitioned data via random kernels,” *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 3, Oct. 2008.

[14] J. Vaidya and C. Clifton, “Privacy-preserving k-means clustering over vertically partitioned data,” in *Proc. 9th ACM International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 206–215.

[15] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, “Distributed basis pursuit,” *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1942–1956, Apr. 2012.

[16] C. Manss, D. Shutin, and G. Leus, “Distributed splitting-over-features sparse bayesian learning with alternating direction method of multipliers,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018, pp. 3654–3658.

[17] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Puschel, "D-admm: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, May 2013.

[18] N. Kashyap, S. Werner, Y.-F. Huang, and R. Arablouei, "Privacy preserving decentralized power system state estimation with phasor measurement units," in *Proc. 2016 IEEE Sensor Array and Multichannel Signal Processing Workshop*, Jul. 2016, pp. 1–5.

[19] C. Heinze-Deml, B. McWilliams, N. Meinshausen, and G. Krummenacher, "LOCO: Distributing ridge regression with random projections," 2015. [Online]. Available: http://arxiv.org/pdf/1406.3469

[20] C. Heinze, B. McWilliams, and N. Meinshausen, "DUAL-LOCO: Distributing statistical estimation using random projections," in *Proc. 19th International Conference on Artificial Intelligence and Statistics*, vol. 51, May 2016, pp. 875–883.

[21] C. Heinze-Deml, B. McWilliams, and N. Meinshausen, "Preserving differential privacy between features in distributed estimation," 2017. [Online]. Available: http://arxiv.org/abs/1703.00403

[22] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1001–1016, 2015.

[23] S. A. Alghunaim, M. Yan, and A. H. Sayed, "A multi-agent primal-dual strategy for composite optimization over distributed features," in *Proc. 28th European Signal Processing Conference*, Jan. 2021, pp. 2095–2099.

[24] R. Arablouei, K. Doğançay, S. Werner, and Y.-F. Huang, "Model-distributed solution of regularized least-squares problem over sensor networks," in *Proc. 2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2015, pp. 3821–3825.

[25] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, "CoCoA: A general framework for communication-efficient distributed optimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 8590–8638, Jan. 2017.

[26] T. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2015.

[27] B. Zhang, J. Geng, W. Xu, and L. Lai, "Communication efficient distributed learning with feature partitioned data," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2018, pp. 1–6.

[28] Y. Hu, D. Niu, J. Yang, and S. Zhou, "FDML: A collaborative machine learning framework for distributed features," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2232–2240.

[29] J. Szurley, A. Bertrand, and M. Moonen, "Topology-independent distributed adaptive node-specific signal estimation in wireless sensor networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 1, pp. 130–144, 2017.

[30] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS for clustered multitask networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 5487–5491.

[31] N. Bogdanović, J. Plata-Chaves, and K. Berberidis, "Distributed incremental-based LMS for node-specific adaptive parameter estimation," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5382–5397, 2014.

[32] J. Plata-Chaves, N. Bogdanović, and K. Berberidis, "Distributed diffusion-based LMS for node-specific adaptive parameter estimation," *IEEE Transactions on Signal Processing*, vol. 63, no. 13, pp. 3448–3460, 2015.

[33] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, p. 387–396.

[34] T. Chang, A. Nedić, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1524–1538, 2014.

[35] D. P. Palomar and Mung Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[36] M. Fukushima, "Application of the alternating direction method of multipliers to separable convex programming problems," *Computational Optimization and Applications*, vol. 1, pp. 93–111, 1992.

[37] D. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[38] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, p. 127–239, Jan. 2014.

[39] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[40] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization, Theory and Examples*. Springer, 2000.

ISBN 978-82-326-5563-2 (printed ver.)
ISBN 978-82-326-6584-6 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU

Norwegian University of
Science and Technology